Technology, Property Rights and Organizational Equilibria: An Explanation of the Co-existence of Open and Closed-source Productions

Fabio Landini*
e-mail: landini4@unisi.it

Version 1.1†

January 24, 2011

Abstract

This paper presents a model on the viability and persistence of open and closed-source productions in the software industry. Starting from some empirical evidences on the organizational diversity that characterizes this sector of the economy, the paper answers one main question: why do open and closed-source productions co-exist? Drawing on a two-ways causality between technology and property rights the model shows that: (a) for a sufficiently high degree of technical mellaebility (i.e. high substitutability between production inputs) there exist two asymptotically stable organizational equilibria, namely an open and a closed-source production; (b) the convergence toward one equilibrium rather than the other is affected by the cost advantages of the two productive solutions, including rents and design costs; and (c) in the presence of conflicting interests among developers, there exist a wide range of parameters for which production inefficiency is persistent. The paper adds to the previous literature on free/open-source software in three ways: first, it presents open and closed-source productions as two distinct production systems characterized by a specific combination of technology and property rights; second, it explains the emergence of open-source productions considering also a causality that runs from property rights to technology; and third, it suggests that institutional complementarities can motivate the sustained co-existence of open and closed-source productions despite of their relative (in)efficiency.

Key words: Open-source software, organizational equilibria, institutional complementarities, transaction costs, evolutionary game theory

JEL Classification Numbers: C73 (Evolutionary Games); D23 (Transaction Costs, Property Rights); L17 (Open-source Products and Markets); O34 (Intellectual Property Rights).

*Department of Economics, University of Siena.

†This paper was completed while the author was visiting the Berkman Center for Internet and Society at Harvard University. The author is grateful to Ugo Pagano, Sam Bowles and Yochai Benkler for their useful comments. The usual caveat applies.

1 Introduction

Economists, since long time, have been interested in the existence of different ways of organizing production (Coase, 1937). In particular, using as a benchmark the view according to which 'in a competitive economy it really does not matter who hires whom' (Samuelson, 1957, p. 894), several authors have gradually relaxed the standard assumptions about market completness and discussed the viability and persistence of organizations based on different property rights regimes (Alchian and Demsetz, 1972; Demsetz, 1966; Grossman and Hart, 1986; Jensen and Meckling, 1976; Craig and Pencavel, 1992; Dow, 1993; Dow and Putterman, 2000; Pagano and Rowthorn, 1994a; Gintis, 1989; Bowles and Gintis, 1993). Although such a debate may seem to some extent an out-of-date residual of the economy of "grain and steal", it is not necessarily so. In fact, it turns out that the existence of organizational diversity is an issue that plays a central role in one of the key sectors of the present knowledge economy, namely the software industry.

As it is well known to both scholars and practitioners there exist nowadays two main ways of profitably organizing the production of software: to rely on flat communities of self-selected peer producers that develop software components through an exchange based on non-exclusive copyrights claims; or, to rely on a conventional firm-based production in which one (or few) subject owns exclusive copyrights claims on the software and the workforce is hired and coordinated through managerial hierarchies. The former is addressed at the production of free/open-source software (FOSS) and I will define it as open-source production. The latter is addressed at the production of closed-source software and I will define it as closed-source production. Despite the fact that both forms of production have existed since long time[1], none of them seems to effectively prevail in the software industry. Why, then, both open and closed-source productions exist?

In the literature on FOSS (for a detailed review see Rossi, 2006) most of the theoretical models dealing with the viability of open-source production have placed particular importance on the specific features of the technology, i.e. the set of tools, processes and procedures used in production. The general approach to the issue, in particular, has been to take as given the technological framework in which open-source productions take place (i.e. the widespread diffusion of the Internet, the low cost of IT and the modularity of code architecture), and

[1]The free sharing of software among programmers (mainly working for public institutions such as universities) was an established practice in the early 70's, and it has survived side by side to the production of proprietary software (although with relative upside down) all the way through the present days. On this point see Lerner and Tirole (2002, 2005) and McGowan (2001)

to compare the cost advantages that open-source productions can boast with respect to closed-source productions. Examples in this stream of literature include Benkler (2002, 2006) Johnson (2006), Baldwin and Clark (2006) and Bessen (2006) among the others.

Looking at these contributions from the perspective of the theory of economic organizations it is clear that, in a more or less explicit way, they represent an adaptation to FOSS production of the standard New Institutional argument for the distribution of organizational rights in the economy. According to this view - see for instance Williamson (1985) - in a world of positive transaction costs and contractual incompleteness, technologies are indeed non-neutral in regard to the nature of property rights. Changes in the characteristics of the production technology alter the nature of the information asymmetries among agents, thereby influencing the nature and attribution of property rights. The latter, in particular, go to the agents that, in because of their intrinsic features, can best economize on transaction costs. When applied to FOSS this line of reasoning marks a point in favor of the viability of open-source productions for two main reasons: first of all, programming is a knowledge-intensive and difficult-to-monitor activity that can be better motivated if agents are residual claimants over the output rather than hired workforce; secondly, software development is a production process that heavily depends on the recombination of others' ideas and the latter, having a marginal cost equal to zero, can be more efficiently exchanged through social sharing than through markets (Benkler, 2002, 2006). Bringing this interpretation to its extreme consequences one should conclude that, if in the present technological paradigm open-source production is relatively more efficient than closed-source production - i.e. the same output can be produced at a lower cost per unit of transaction, the former is inevitably going to displace the latter as the standard way of developing software.

When compared with the actual trends of the software industry, however, this interpretation fails. In particular, it cannot explain three stylized facts.

Fact 1. *In the software industry the world's top organizations produce both open and closed-source software.*

According to the Canonical Chief Operating Officier Matt Asay[2], for instance, both Oracle-Sun and IBM, despite being well established producers of proprietary software, have been investing substantial resources in open source projects. Oracle-Sun is the primary developer behind Java (more than 6.5 million lines of code), Solaris (over 2 million lines of code) and Open Office (approximately

[2]Matt Asay (2009). World's biggest open-source company? Google, 2009. URL `http://news.cnet.com/8301-13505_3-10354530-16.html`. Last time checked: 29th of July, 2010.

10 million lines of code). IBM contributed more than 12.5 million lines of code to Eclipse, not to mention Linux (6,3 % of total contributions), Geronimo and other open-source projects. A similar support to open-source initiatives can be observed on the side of SAP AG, the world leader in business management software. As reported in the company's website, the German firm has been deeply involved in the Eclipse ecosystem since 2004 moving to the membership level of strategic developer in 2009. In 2003, SAP AG, has also opened the source code of its business database MaxDB to the MySQL community, except reverting it back to closed-source in 2007. According to their official website, moreover, among the active contributors to open-source projects appear HP (63), Adobe (20), Microsoft (53) and Apple, which has developed an apposite repository of open-source applications for Mac products named MacOS Forge. Despite of the huge amount of resources invested to maintain this active support to open-source projects, however, none of these companies has actually abandoned the development of proprietary software. Then, the following question arises: **Q1** - *If, given the technology, one type of production (let's say open-source) is more efficient than the other, why do the same organizations develop software using both open and closed-source productions?*

Fact 2. *In each market segment of the software industry there often exist both open and closed-source software.*

Data on the trends of market shares over the last two decades are instructive in this sense. According to the last Netcraft's survey[3], for instance, the market for web servers has been steadily dominated by two software packages: the open-source Apache with an average market share well above 50%, followed directly by the closed-source Microsoft Web Server with a share constantly lower than 40%. The market for web browsers[4] presents a similar although reverse trend: the closed-source Internet Explorer has constantly been the market leader with more than the 55% of the market, followed by the open-source Mozilla Firefox which has gained shares going up until the 23% and lately by the open-source Linux-based Google Chrome with the 8%. In the market for database the actual presence of open and closed-source software is again undeniable, with the open-source MySQL contending the lead to the closed-source Microsoft SQL Server and Oracle. A similar situation can be observed in the market for operating systems (OS), where the various distributions of the open-source Unix/Linux OS have progressively extended their usage share and are nowadays well accepted as

[3]Netcraft, web server survey, July 2010. URL `http://news.netcraft.com/archives/2010/`. Last time checked: 29th of July, 2010.

[4]NetApplications, Top browser share trend, 2010 `http://marketshare.hitslink.com/browser-market-share.aspx?qprid=1`. Last time checked: 29th of July, 2010.

direct competitors of the closed-source Microsoft Windows[5]. In general, the picture that one can get by looking at these as well as other segments of the software industry (e.g. office suites, finance and accountability packages, mail servers) is that open and closed-source software packages simply co-exist. Moreover, such co-existence tends to reproduce also in newly emerging market segments. Recent data collected by Gartner, Inc.[6] on the market for mobile operating systems (the ones that one can find on smartphones), for instance, highlight the presence of five main competitors: Nokia Symbian (44%) and Google Android (10%) on the side of open-source, and Apple iPhone (15%), RIM Blackbarry (19%) and Microsoft Windows Mobile (7%) on the side of closed-source. In the light of these data, then, another question arises: **Q2** - *If, given the technology, organizational rights flow so as to increase production efficiency, why do (almost) technologically equivalent packages get to be steadily produced both as open and closed-source software?*

Fact 3. *In the software industry, software that is born as closed-source (open-source) is highly likely to remain closed-source (open-source).*

Figure 1 has been constructed on the basis of a sample of 164 software packages from six different market segments: web browsers, database, office suites, finance and accountability packages, operative systems and web servers (see the Appendix for a complete list). These software packages were divided in four categories according to the type of license used for their 1.0 and present versions. "Closed-closed" stands for software that has been licensed as closed-source both in its 1.0 and present version. "Closed-open" stands for software that has been licensed as close-source in its 1.0 version and as free/open-source in its present version. And so on. Data refer to a time-span that goes from 1980 until 2010. Figure 1 shows the frequency distribution for these four different categories. As it is easy to see most of the software that is distributed as either free/open-source or closed-source in its inception, continues to be distributed under the same licensing terms even today (almost 90% of the total). Such a result is obviously affected by the fact that the most popular free/open-source licenses expressly limit the possibility of "closing" the source code once it is freed. However, even limiting the attention at closed-source software alone, the data is impressive: more than the 90% of the considered software packages exhibits

[5]For data on both database and OS market shares see David A. Wheeler (2007), Why Open Source Software/Free Software (OSS/FS, FLOSS, or FOSS)? Look at the Numbers! URL `http://www.dwheeler.com/oss_fs_why.html`. Last time checked: 16th of August, 2010.

[6]Gartner Inc., Competitive landscape: Mobile devices, 1q10, 2010. URL `http://www.gartner.com/DisplayDocument?doc_cd=2009461\ref=g_rss`. Last time checked: 29th of July, 2010.

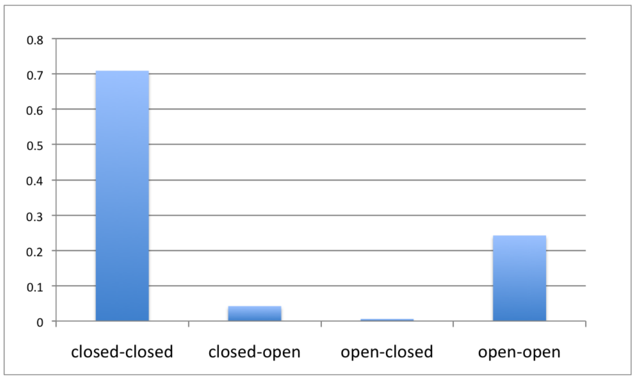

Figure 1: Frequency distributions of four different paths of software development.

path dependency in the way in which it is developed. Once again this result is inconsistent with the above described arguments concerning the viability of open-source production: **Q3 -** *If, given the technology, open-source production is more efficient than closed-source production, why so few software packages change from the latter to the former?*

The aim of this paper is to offer an explanation for the viability and persistence of both open and closed-source productions which is able to account for Facts 1, 2 and 3 above, and that, as a consequence, provides an answer to questions Q1, Q2 and Q3. In particular, it will be argued that the co-existence of these two different organizations of production can be explained by integrating the arguments based on a causality that runs from technology to property rights, with a simultaneous causality that runs from property rights to technology. This, following Pagano (1993) and Pagano and Rowthorn (1994b), amounts at qualifying open and closed-source productions as two distinct organizational equilibira. From this result, two main implications follow: first, the viability of open and closed-source productions is strongly affected by the degree of technical malleability; second, it can well happen that production inefficiency may be asymptotically persistent.

The structure of the paper is the following. Section 2 introduces the concept of organizational equilibria and some definitions that will be used in the remaining parts of the paper. Section 3 presents a formalization of the arguments introduced in Section 2 by framing the model presented by Pagano and Rowthorn (1994b) in a two-players game and finds the conditions under which multiple organizational equilibria exist. Such conditions are then used in order to interpret some of the empirical facts outlined above. Section 4 studies the asymptotic stability of these equilibria. This is done, in particular, by assuming that the conditions for the existence of multiple organizational equilibria are satisfied and by inserting the two-players game developed in Section 3 into an evolutionary dynamics with intentional idiosyncratic plays. Section 5 finally concludes.

2 Orgnanizational equilibria

Ugo Pagano (1993) defines organizational equilibria as situations in which two conditions are simultaneously met: (a) the technological characteristics of the resources used in production bring about a set of rights which is consistent with this technology; and (b) the set of rights brings about technological characteristics of the resources which are consistent with these rights. This concept is developed by integrating two distinct arguments. The first is the standard New Institutional argument according to which technology causes property rights, i.e. in an organization of production property rights optimally adjust to the characteristics of the resources employed. The second is the so-called "reversed" argument according to which property rights causes technology, i.e. in an organization of production the characteristics of the resources employed optimally adjust to the initial distribution of property rights[7]. When both causalities hold, property rights self-reinforce via technology and vice-versa, and organizational equilibria emerge.[8]

A more formal definition of organizational equilibrium can be worked out as follows. Define an organization of production as a production system characterized by a specific combination of two domains: the property rights domain R and the technology domain T. Write $\Pi(R,T)$ as the profit obtained under a particular organization of production. Then,

Definition 1 *An organization of production is a an organizational equilibrium if **(a)** R maximizes $\Pi(R,T)$ given T; and **(b)** T maximizes $\Pi(R,T)$ given R.*

In order to apply this definition to the study of open and closed-source productions two main aspects need to be clarified. Firstly it is important to notice that, in the domain of property rights, the distinction between open and closed-source production goes beyond the simple licensing terms on the software, so far as to include employment contracts and hardware ownership. In particular, while open-source productions rely on non-exclusive copyrights claims coupled

[7]The reversed causality running from property rights to technology as been suggested also by Marglin (1974), Rowthorn (1974), Putterman (1982), Bowles and Gintis (1983) and Bowles (1985).

[8]For a formal discussion of organizational equilibria see Pagano and Rowthorn (1994b).

with limited use of employment contracts and decentralized hardware ownership, closed-source productions present the opposite characteristics with exclusive copyrights claims combined with extended use of employment contract and centralized hardware ownership (on this point see Benkler, 2006). This distinction is important because it affects the costs faced by these two ways of organizing software development. On this basis, the following definitions can be introduced

Definition 2. *A set of property rights $R^O \in R$ is an open-source regime if it combines a marginal (or absent) use of employment contracts with non-exclusive copyrights claims and decentralized hardware ownership.*

Definition 3. *A set of property rights $R^C \in R$ is a closed-source regime if it combines a wide use of employment contracts with exclusive copyrights claims and centralized hardware ownership.*

Secondly it is important to point out that, in the domain of technology, there also exist a significant difference between open and closed-source productions. In this sense the two main technological dimensions that need to be considered are the design of the code architecture and the organization of production inputs. While open-source productions adopt a relatively modular technology characterized by a finely-grained code architecture combined with a decentralized decision making process, closed-source productions employ a relatively non-modular technology characterized by a coarsely-grained code architecture coupled with a fairly centralized decision making process.[9] On this basis, we can once again introduce the following definitions

Definition 4. *A technology $T^M \in T$ is modular if it combines a finely-grained code architecture with a polyarchical governance structure*

Definition 5. *A technology $T^L \in T$ is non-modular if it combines a coarsely-grained code architecture with a hierarchical governance structure*

Given these definitions, open and closed-source productions can be represented as organizations of production characterized by the combinations $\{R^O, T^M\}$ and $\{R^C, T^L\}$ respectively. The question, at this point, is to understand whether they also represent two distinct organizational equilibria. Before moving to for-

[9]Such differences in the design of the technology has been made clear by Raymond (1999) and his evocative image of the "Cathedral and the Bazaar". For a discussion on the effects of modularity in open-source collaboration see also Langlois and Garzarelli (2008). For a more general treatment of modularity in *decomposable* systems see Simon (1962)

mal modeling, we can try to address this point by considering the two arguments on the basis of which Definition 1 has been worked out and evaluating how they would explain these two different organizations of production.

Let's start from the New Institutional argument according to which *technology causes property rights*, and let's consider a relatively modular technology first. In this case one could argue that an open-source regime is likely to emerge because it is more efficient than a closed-source regime, where efficiency depends on two main factors. First, as suggested by Benkler (2002, 2004), the presence of commons-based proprietary claims gives way to a system of production that has a lower cost per transaction compared to market or firm-based productions, and has therefore a cost advantage when, as in the case of a modular technology, the number of software modules is high. The main reason for this is that, being based on extended social sharing, self-selection and usage of local knowledge, the former requires a lower degree of formalism and less information exchange in the communication among software developers. Second, as argued by Raymond (1999), in the presence of sufficient technological modularity there is much to gain in terms of software quality by adopting non-exclusive copyrights claims and thus opening the access to the source code at a wide and differentiated community of developers. In relation to the latter point, Hong and Page (2004) have also offered a formal proof of the productivity of collective diversity in modular problem solving.

A different conclusion, however, could be obtained if a relatively non-modular technology is available instead. In this case one could argue that a closed-source regime is more appropriate than an open-source regime in because of one main factor, that is the possibility of motivating programmers through monetary compensations and threat of dismissal. The presence of a coarsely-grained code architecture, in fact, makes each software module rather complex and labor demanding. When employment contracts are only marginally used - as in the case of an open-source regime - such modules are subject to individual free-riding and are likely not to be developed (Benkler, 2002) (on this point see also den Besten et al., 2008 for empirical evidence). On the contrary, when the labor force is hired - as in the case of a closed-source regime, the contract itself and the associated wage can be used in order to extract individual effort and avoid free-riding. Under this regime software modules are likely to be developed and production to take place. On this basis, and differently from before, one could conclude that, once a relatively non-modular technology is given, a closed-source regime instead of an open-source one is likely to emerge.

This way of reasoning, however, can be inverted. Following the so-called "reversed" argument, in fact, it can well be that it is *property rights that cause technology*, and not the reverse. Let's assume, for instance, that an open-source regime is given in the first place and that technology needs to be designed accordingly. In this case one could argue that the design of a modular technology becomes a key requirement in order to ensure that the software is effectively developed on the basis of two main factors. First, as suggested by Benkler (2002) and Baldwin and Clark (2006), the presence of a finely-grained code architecture is essential in order to ensure that work is performed on the basis of motivations different from the monetary compensations included in a standard employment contract. Second, as pointed out by Giuri et al. (2008), the evolution of a polyarchical governance structure is a necessary condition for informal authority to be respected in virtual teams. When both these factors are considered the conclusion goes in the same direction as the one observed above, although in this case the causality is inverted: once an open-source regime is given, a relatively modular technology is likely to emerge.[10]

But, once again, a similar line of reasoning could be employed also under the assumption that a closed-source regime is given in the first place. In this case the convenience of a relatively non-modular technology can be motivated on two main grounds. First, as argued above, a reduction in the degree of modularity decreases the number of software modules, and therefore it allows the organization to economize on discrete transactions. Second, the presence of a relatively coarsely-grained code architecture combined with a hierarchical governance structure gives to the owner of the exclusive copyrights claim a much stronger control over the software, which can become extremely important especially if the latter has to be placed on the market. On this basis we obtain again the same result observed under the New Institutional approach but a different causal mechanism: once an closed-source regime is given, a relatively non-modular technology is likely to emerge.

Which one of these two approaches to the problem is the correct? Discriminating between them is not an easy task. Most likely, both the New Institutional and the "reversed" arguments are correct. When this is the case, there are reasons to believe that open and closed-source productions qualify as two distinct organizational equilibria. In order to give a formal representation of this result and to find the conditions under which multiple organizational equilibria in software development exist we turn now to a simple model. The results of the model will then be used in order to obtain some insights on the empirical facts outlined in the Introduction.

[10]One example of the way in which this causality applies comes from the common practice of refactoring the source code of a software before going open-source. Code refactoring, i.e. the process of changing the non-functional attributes of a software in order to foster the source code's maintainability and extensibility, indeed entails an improved readability and a modularization of the code architecture. The author is grateful to Yochai Benkler for pointing out the analogy between code refactoring and the "reversed" argument described above.

3 The model

Let's consider two agents r and t who project to develop a software. In order to do so they need to adopt an organization of production, namely they need to make a choice within two domains of the choice set S: the property rights domain $R \in S$, in which they choose whether to adopt a closed-source regime (R^C) or an open-source regime (R^O), i.e. $R = \{R^O, R^C\}$; and the technology domain $T \in S$, in which they choose the design of the production technology. For the sake of simplicity I restrict the model to a technology with only two design variables: the degree of modularity in the code architecture $M \in (0,1)$, which is measured by the ratio between the number of independent tasks and the total number of tasks performed by the software; and the degree of labor commitment necessary to develop the software modules $L \in (0,1)$, which is measured by the portion of daily time devoted to programming each module (i.e. collection of interdependent tasks). In this framework M and L can be interpreted as two factors of production in the standard economic sense. M and L, in fact, can to a certain extent be considered substitute under the condition that, for a given size of the software, an increase in modularity decreases the size of each software module and therefore reduces the amount of work needed to develop *that* module. Under this interpretation the nature of a generic technology T^j is defined by the factors proportion (or intensity) $T^j = M^j/L^j$. A technology T^j is thus defined as relatively modular when compared to a technology T^k if and only if $T^j > T^k$. The idea is that while in a modular technology highly modular software (high M^j) is developed by programmers devoting relatively small portions of time to programming (low L^j), in a non-modular technology weakly modular software (low M^k) is developed by programmers devoting big portion of their time to programming (high L^k). In this framework the nature of the choice that needs to be made in domain T is to set the factors proportion M/L.

Both M and L have positive costs. Under the assumption that complete contracts cannot be written, one typology of costs can be classified as transaction costs. In this category, I will call m the cost of modularity, i.e. the cost associated to the transfer of information among developers working at different software modules, and l the cost of labor, i.e. the cost associated with the extraction of effort from the labor force. In line with the discussion presented in Section 2, an open-source regime R^O has a cost advantage x in terms of information transfers, while a closed-source regime R^C has a cost advantage y in terms of labor extraction. Notice that the latter condition does not mean that labor is cheaper in closed-source production than in open-source production. Rather, it means that work effort is more cheaply extracted in a closed-source production system than in a open-source production system, where the cost is measured both in terms of monetary expenditure (e.g. the wages paid to the employees hired to work on FOSS projects) and decrease in profit due to unproductive work (e.g. free-riding). The cost function, thus, assume the following form:

$$C(M, L, R) = \begin{cases} (m-x)M + lL, & \text{if } R = R^O \\ \\ mM + (l-y)L, & \text{if } R = R^C \end{cases} \tag{1}$$

where $m > l - y$ and $l > m - x$. The latter conditions mean that there exist cost advantages in terms of production factors not only between but also within production systems. In addition, I will consider also a second type of costs which are associated with the design of the code architecture. In particular, I will call d the design cost of modularity which is paid by the agent involved in the design of the production technology, namely (as we will see) agent t.[11]

The return obtainable from production is modeled in the following way. Software gives rise to two types of returns: returns from the services sold as complement to the software, which are captured by a function $Q(M, L)$ and are the same independently of the licensing system adopted; and a rent obtainable from the sale of proprietary software which is captured by a function $z(R, L)$ taking the following form:

$$z(R, L) = \begin{cases} 0, & \text{if } R = R^O \\ \\ zL, & \text{if } R = R^C \end{cases} \tag{2}$$

where $z > 0$ captures the positive effects of labor commitment on the rent from software sales. The idea in the latter case is that, by increasing the control over the software of the agents in possess of the exclusive copyrights claims, a greater labor commitment increases the software's marketability. While returns from services are equally shared, the rent from sales is appropriated only by the agent owning the copyrights, in our case agent r. Agents are assumed to be risk neutral and software price is equal one.

Under these assumptions, the payoffs to agent r and t can be respectively written as follows:

$$\pi_r(R, T(M, L)) = z(R, L) + \frac{[Q(M, L) - C(M, L, R)]}{2} \tag{3}$$

$$\pi_t(R, T(M, L)) = \frac{[Q(M, L) - C(M, L, R)]}{2} - dM \tag{4}$$

[11]A similar distinction between transaction and design costs of production technologies is presented in Baldwin and von Hippel (2009).

Given equations (3) and (4), I model the selection process leading to the adoption of an organization of production as follows. Agents r and t make an independent choice in the property rights and technology domain respectively. To lend some concreteness to the model we can imagine r as being the agent having the original idea about the software and thus deciding how to copyright it, and t as being the engineer designing the technology through which the software is to be produced. In both domains, choices are made in order to maximize payoff, i.e. r will choose the property rights regime that maximizes π_r for a given technology (*Ass1*), while t will choose the technology that maximizes π_t for a given property rights regime (*Ass2*). Notice that, abstracting from the problem at stake, r stands as representative of the causality mechanism that runs from technology to property rights (the New Institutional argument), while t stands as representative of the causality mechanism that runs from property rights to technology (the "reversed" argument).

In the property rights domain R, given (3), (4), *Ass1* and a generic technology T^j, r will choose to adopt an open-source regime as long as $\pi_r(R^O, T^j) \geq \pi_r(R^C, T^j)$, which is the case if and only if:

$$T^j = \frac{M^j}{L^j} \geq \frac{y + 2z}{x} \tag{5}$$

Similarly, r will choose to adopt a closed-source regime as long as $\pi_r(R^C, T^j) \geq \pi_r(R^O, T^j)$, which is the case if and only if:

$$T^j = \frac{M^j}{L^j} \leq \frac{y + 2z}{x} \tag{6}$$

From equation (5) and (6) the following proposition can be derived:

Proposition 1. *In the domain of property rights R, the incremental benefit from choosing an open-source regime R^O (instead of choosing R^C) is greater when a technology characterized by a relatively higher intensity of modularity M is selected in the domain T, i.e. when T^M is selected instead of T^L.*

Proof. For a given value of x, y and z, consider two technologies T^M and T^L such that conditions (5) and (6) are simultaneously satisfied, i.e. $T^M \geq (y + 2z)/x \geq T^L$. Then, it follows directly from (5) and (6) that:

$$\pi_r(R^O, T^M) \geq \pi_r(R^C, T^M) \tag{7}$$

$$\pi_r(R^C, T^L) \geq \pi_r(R^O, T^L) \tag{8}$$

Adding equations (7) and (6) side by side and rearranging, we obtain the following relation:

$$\pi_r(R^O, T^M) - \pi_r(R^C, T^M) \geq \pi_r(R^O, T^L) - \pi_r(R^C, T^L) \tag{9}$$

which proves the proposition $\square$.

Let's now consider the domain of technology. Under *Ass2*, t will set M and L so as to maximize $\pi_t(R^C, T(M, L))$ and $\pi_t(R^O, T(M, L))$. Let:

$$(M^C, L^C) = \arg\max \pi_t(R^C, T(M, L)) \tag{10}$$

$$(M^O, L^O) = \arg\max \pi_t(R^O, T(M, L)) \tag{11}$$

Then, from equation (3) and (4) above and under standard assumption about the shape of the marginal product, i.e. $\partial^2 Q/\partial M^2 < 0$ and $\partial^2 Q/\partial L^2 < 0$, it follows that $M^C \leq M^O$ and $L^C \geq L^O$. From the latter conditions it is straightforward to derive the following relation:

$$T^O = \frac{M^O}{L^O} \geq \frac{M^C}{L^C} = T^C \tag{12}$$

Relation (12) in turn implies that:

Proposition 2. *In the domain of technology T, the incremental benefit from choosing an M-intensive technology T^M (instead of choosing an L-intensive technology T^L), is greater when an open-source regime is selected in the domain R, i.e. when R^O is selected instead of R^C.*

Proof. Consider two technologies T^M and T^L such that conditions (12) is satisfied, i.e. $T^M \geq T^L$. Then, it follows directly from (12) that:

$$\pi_t(R^O, T^M) \geq \pi_t(R^O, T^L) \tag{13}$$

$$\pi_t(R^C, T^L) \geq \pi_t(R^C, T^M) \tag{14}$$

Adding equations (14) and (13) side by side and rearranging, we obtain the following relation:

$$\pi_t(R^O, T^M) - \pi_t(R^O, T^L) \geq \pi_t(R^C, T^M) - \pi_r(R^C, T^L) \tag{15}$$

which proves the proposition $\square$.

Under some continuity conditions of function $\pi(.)$ and assuming that strategy choices $S_r = \{R^O, R^C\}$ and $S_t = \{T^M, T^L\}$ have a partial order $\geq$ (see Milgrom and Roberts, 1990), Propositions (1) and (2) imply that the game $G = \{2, (S_i, \pi_i, i = r, t), \geq\}$ is supermodular. Furthermore, it can be proved[12] that in G there exist two pure strategy Nash equilibria, namely $\{R^O, T^M\}$ and $\{R^C, T^L\}$. The first, $\{R^O, T^M\}$, is characterized by an open-source regime and a relatively modular technology; I will call this equilibrium an *open-source organizational equilibrium.* The second, $\{R^C, T^L\}$, is characterized by a closed-source regime and a relatively non-modular technology; I will call this equilibrium a *closed-source organizational equilibrium.* When these two equilibria exist, using Aoki (2001)'s terminology, we would say that R^O and T^M as well as R^C and T^L are institutional complements.

The technological conditions supporting the existence of multiple organizational equilibria in software development can be summarized in the following proposition:

Proposition 3. **(a)** *Suppose* $T^O = T^M \geq (y + 2z)/x \geq T^L = T^C$. *Then in game G there exist two pure strategy Nash equilibria* $\{R^O, T^M\}$ *and* $\{R^C, T^L\}$, *i.e. multiple organizational equilibria exist.* **(b)** *Suppose* $T^O = T^M \geq T^L = T^C \geq (y + 2z)/x$. *Then in game G there exist only one pure strategy Nash equilibria* $\{R^O, T^M\}$, *i.e. only an open-source organizational equilibria exists.* **(c)** *Suppose* $(y + 2z)/x \geq T^O = T^M \geq T^L = T^C$. *Then in game G there exist only one pure strategy Nash equilibria* $\{R^C, T^L\}$, *i.e. only a closed-source organizational equilibria exists.* **(d)** *For any ratio* $(y + 2z)/x$ *in game G there exists at least one pure strategy Nash equilibrium, i.e. there always exist at least one organizational equilibria.*

Proof. Points **(a)**, **(b)** and **(c)** follow directly from conditions (5), (6) and (12) above. Point **(d)** is a direct consequence of points **(a)**, **(b)** and **(c)** $\square$.

Proposition 3 suggests that if the ratio between the cost advantages $(y + 2z)/x$ falls into the closed intervals defined by the factors proportions employed under the two different property rights regimes, two distinct ways of organizing the production of software exist. The key question, then, becomes to understand how likely it is that such condition obtains. Intuition suggests that the 'malleability' of the technology plays an important role in this respect because it ensures that, for any given property rights regime, factors proportion can be adjusted so as to improve efficiency. Under the standard assumption of decreasing marginal product, in particular, it can be proved that:

[12]See Theorem 5 in Milgrom and Roberts (1990)

Proposition 4. **(a)** *For any standard production function $Q(M, L)$ and for any set of costs (m, l, d), there exists at least one triple (x, y, z) such that multiple organizational equilibria exist.* **(b)** *If the elasticity of substitution is equal zero, i.e. if M and L are perfect complements, then there exist only one triple (x, y, z) such that multiple organizational equilibria exist.* **(c)** *If the elasticity of substitution is infinite, i.e. if M and L are perfect substitutes, then any positive triple (x, y, z) will imply that multiple organizational equilibria exist.*

Proof. See Pagano and Rowthorn (1994b), Propositions (2), (3) and (4). □

The results contained in Propositions 3 and 4 tell us three main things regarding the co-existence of open and closed-source production. First, multiple organizational equilibria in software development exist for a wide range of parameters' value. Without assuming the extreme conditions of zero or infinite elasticity of substitution, the presence of a fairly malleable technology is in fact sufficient in order to ensure that both open and closed-source productions exist. Such a result, at least in principle, could explain the high degree of organizational variety that we observe in the software industry. Second, the degree of technological rigidity has important implications for the way in which the production of software is organized. In particular, we have found that the more malleable the technology, the more likely multiple organizational equilibria exist. This could explain why the same organization often produces software under different property rights regimes (Fact 1). Specific technological rigidities, in fact, may impose constraints on the way in which a particular software can be developed and thus oblige organizations to adopt different organizations of production for different software. Finally, the results of the model highlight the existence of an institutional complementarity between property rights and technology in software development. When this is the case, the emergence of both open and closed-source productions is likely to follow a pattern that is closely related to the biological theory of "punctuated equilibria" according to which change cannot be approached by individual, gradual modifications, but it requires simultaneous, complementary shocks (Eldredge and Gould, 1972). In relation to the present discussion these characteristics imply that: first, to plan a shift from a closed to an open-source production (and vice-versa) is rather complex, because it requires simultaneous changes in different organizational domains; second, in analogy with speciation, the effective emergence of open and closed-source productions is likely to be driven by major institutional shocks followed by period of gradual adjustment. These two points could partially explain the path dependency that we observe in the organization of software development (Fact 3). Moreover, they suggest that credible explanations for the emergence of open-source production should rely on simultaneous shocks occurred both on the technology and property rights domain, e.g. the development of the General Public License (GPL) hand-in-hand with the commercial diffusion of the Internet in the late 80's.

4 Dynamics and evolutionary stability

In this section I study the asymptotic stability of organizational equilibria. In order to do so I model the dynamics behind the co-evolution of property rights and technology starting from the assumption that multiple organizational equilibria exist. Such a dynamics, in particular, could be useful in order to address two main points: first, why technologically equivalent software get to be steadily produced under different property rights regimes (Fact 2); second, why and how in the presence of multiple organizational equilibria production efficiency may no longer be achieved.

Instead of assuming an exogenous law of motion for both domains R and T, I have chosen to model the co-evolution of property rights and technology by adopting the now-standard evolutionary game theoretic approach (Bowles, 2006; Gintis, 2009). Besides being rather simple, this approach has the advantage of micro-founding the evolution of institutional forms on a decentralized decision making process based on credible behavioral rules. This is obviously done at the cost of introducing some degree of abstraction, in my case the substitution of a causality mechanism between technology and copyrights with a participatory decision process based on voting.

The structure of the evolutionary game is the following. I consider a large group (or population) of agents of size n, which is divided in two sub-populations r and t of size n_r and n_t respectively ($n = n_r + n_t$).[13] Once again agents have to choose the organization of production to be adopted in software development, i.e. they play a version of the game described in Section 3. This time, however, the choice does not concern only two agents but the whole group and is based on a voting mechanism. Agents in sub-population r (r-agents) vote in the property rights domain and have two options: an *open-source* regime or a *closed-source* regime. Agents in sub-population t (t-agents) vote instead in the technology domain. For the sake of simplicity and without loss of generality, I assume that there exist perfect substitutability between M and L and therefore only two technologies are available to t-agents: a modular technology T^M (with $M^M = 1$ and $L^M = 0$) and a non-modular technology T^L (with $M^L = 0$ and $L^L = 1$). In

[13]For the easiness of notation I have decided not to introduce new variables. Notice that, differently from he previous section, r and t refer to the two sub-populations and not to individual agent.

Technology (t) ($\rightarrow$) **Copyright (r)** ($\downarrow$)	**Modular** ($T = T^M$)	**Non-modular** ($T = T^L$)
Open-source ($R = R^O$)	$\frac{Q-(m-x)}{n}, \frac{Q-(m-x)}{n} - \frac{d}{n_t}$	$\frac{Q-l}{n}, \frac{Q-l}{n}$
Closed-source ($R = R^C$)	$\frac{Q-m}{n}, \frac{Q-m}{n} - \frac{d}{n_t}$	$\frac{Q-(l-y)}{n} + \frac{z}{n_r}, \frac{Q-(l-y)}{n}$

Table 1: Stage game matrix of payoffs. $\{R^O, T^M\}$ and $\{R^C, T^L\}$ are Nash equilibria in pure strategies. Note: each cell of the matrix identifies a different organization of production; the payoffs reported are the ones that the agents in the two sub-populations would get if production were to take place under each alternative organization of production.

standard game theoretic terms each voting option represents a strategy available to agents in their own domain of choice.

Individual votes are expressed on the basis of the payoffs defined by equations (3) and (4) with the exception that this time the net revenue from services is divided over n, while the rent and the design cost are divided over n_r and n_t respectively. By substituting into equations (3) and (4) the alternatives available in both the property rights and technology domains, we obtain the payoffs matrix reported in Table 1 which defines the stage game. In order to make the problem interesting I assume $d \leq x/\xi$ where $\xi = n/n_t$. Moreover, I assume $m = l$.

Under these conditions, the stage game has two Nash equilibria in pure strategies, namely $\{R^O, T^M\}$, that is, an *open-source organizational equilibrium* and $\{R^C, T^L\}$, that is, a *closed-source organizational equilibrium.* These two equilibria qualify as organizational conventions, meaning that conforming to it is a mutual best response as long as most members of each sub-population (r and t) expect most members of the other to conform to it.

The dynamics of voting is modeled as follows. At the beginning of every time period τ agents express an individual and autonomous vote in their own domain of choice. At τ_0 such votes are exogenously determined by causes not expressly modeled (it can be due to preferences or previous experience). Then, for any $\tau > \tau_0$, each agent updates her vote following the updating process described below.

Once votes are expressed agents are paired across the two sub-populations to compare their individual decisions. To lend some concreteness to the model we can imagine such pairings as being periodic meetings during which "inventors" and engineers discuss the costs and benefits of software production. In the course of such meetings each agent receives two pieces of information. The first one is the distribution of votes in each sub-population. The second one consists of the payoffs reported in Table 1 - i.e. the payoff that each agent would receive if the software were to be developed under each organization of production. Such payoffs, however, do not immediately translate in individual earnings since production takes place only when "near" unanimity obtains. Writing ρ and ω the fractions of agents voting R^O and T^M respectively, "near" unanimity obtains when the following conditions hold: $\rho < \varepsilon$ or $\rho > 1 - \varepsilon$ and $\omega < \varepsilon$ or $\omega > 1 - \varepsilon$ with $\varepsilon > 0$ being arbitrarily small, i.e. when only a negligible portion of agents disagrees with the majority in both domains. Once production takes place, agents earn the payoffs associated with the organization of production chosen by the majority, i.e. version 1.0 of the software comes out. In this framework the "near" unanimity condition can be simply interpreted as a managerial rule which ensures that consistent practices are maintained within the group.

The process leading to a change in vote is then modeled as a standard payoff monotonic updating. At the beginning of every time period $\tau > \tau_0$ each agent in both sub-populations can use the two pieces of information obtained in the previous period in order to compute her expected final payoff, where expectations are formed on the basis of the distributions of votes in the other sub-population. Using the payoffs in Table 1 the expected payoff to agents voting T^M and T^L can be written as

$$\pi_M(\rho) = \rho \left[\frac{Q - (m - x)}{n} - \frac{d}{n_t} \right] + (1 - \rho) \left[\frac{(Q - m)}{n} - \frac{d}{n_t} \right] \tag{16}$$

$$\pi_L(\rho) = \rho \frac{(Q - l)}{n} + (1 - \rho) \left[\frac{Q - (l - y)}{n} \right] \tag{17}$$

Similarly, the expected payoffs to agents voting R^O and R^C are respectively

$$\pi_O(\omega) = \omega \left[\frac{Q - (m - x)}{n} \right] + (1 - \omega) \frac{(Q - l)}{n} \tag{18}$$

$$\pi_C(\omega) = \omega \frac{(Q - m)}{n} + (1 - \omega) \left\{ \left[\frac{Q - (l - y)}{n} \right] + \frac{z}{n_r} \right\} \tag{19}$$

These expected payoff functions are reported in Figure 2, assuming $x > y$. Once such functions have been computed and before choosing on a new vote, each agent meets with another agent randomly selected from her sub-population and compares the respective decisions. For instance, an agent a in sub-population t has the opportunity to observe the vote expressed by another agent, named b, and to know her expected payoff. If b expressed the same vote as a, a does not update. But if b expressed a different vote, a compares the two expected payoffs and, if b has a greater expected payoff, switches to b's vote with a probability

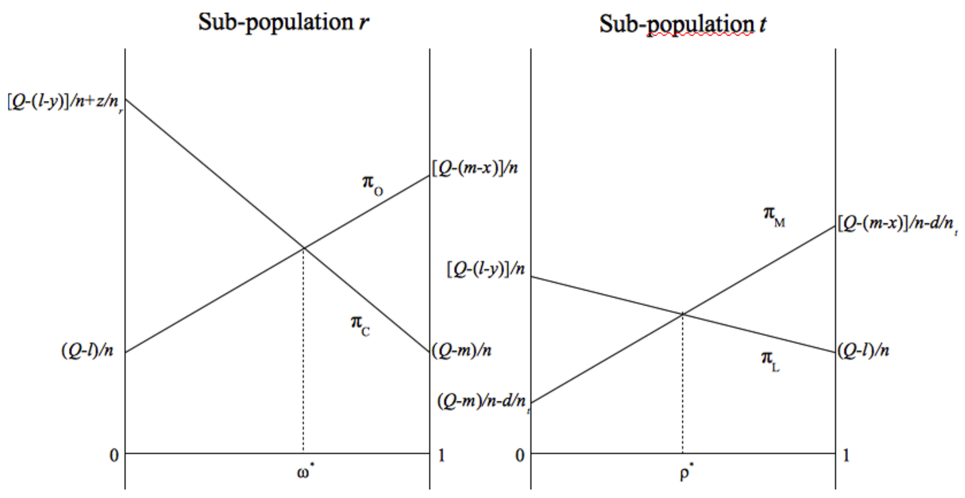

Figure 2: Expected payoffs to agents from sub-populations r and t. Note that ω and ρ are respectively the fractions of agent voting T^M and R^O in the previous period.

equal to $\beta > 0$ times the payoff difference, retaining her own vote otherwise. While this updating process is not very sophisticated, it may realistically reflect individual cognitive capacities and it ensures that the standard economic assumption of utility maximization is preserved - i.e. each agent votes for the outcome that, given the distribution of vote in the other sub-population, maximizes her expected payoff.

Once this updating process is defined it is easily shown (see Bowles, 2006) that the state of the population, which in any time period τ is given by $\{\rho_\tau, \omega_\tau\}$, evolves according to the following system of replicator equations:

$$\frac{d\rho}{d\tau} = \rho(1-\rho)\beta(\pi_O(\omega) - \pi_C(\omega)) \tag{20}$$

$$\frac{d\omega}{d\tau} = \omega(1-\omega)\beta(\pi_M(\rho) - \pi_L(\rho)) \tag{21}$$

Given equations (20) and (21) we are mainly interested in the stationary states of the population, namely the states for which $d\rho/d\tau = 0$ and $d\omega/d\tau = 0$. Such states qualify as fixed-points of the dynamical system, and as voting equilibria of the population. It is easy to see that $d\rho/d\tau = 0$ for $\rho = 0$, $\rho = 1$ and $\omega = \omega^*$, while $d\omega/d\tau = 0$ for $\omega = 0$, $\omega = 1$ and $\rho = \rho^*$, where

$$\omega^* = \frac{y + \delta z}{x + y + \delta z} \tag{22}$$

$$\rho^* = \frac{y + \xi d}{x + y} \tag{23}$$

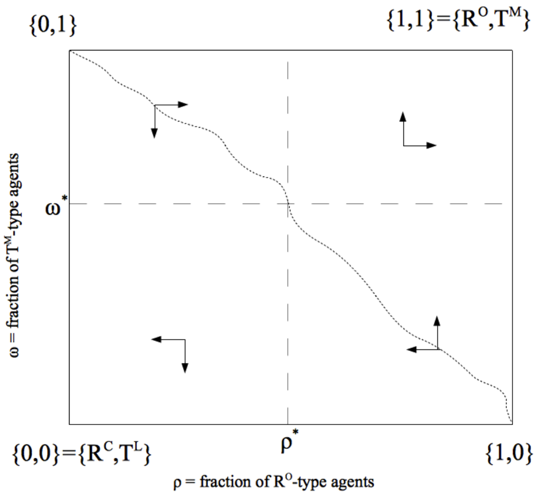
{0,1}
{1,1}={R^O,T^M}
ω*
ω = fraction of T^M-type agents
{0,0}={R^C,T^L}
ρ*
{1,0}
ρ = fraction of R^O-type agents

Figure 3: Asymptotically stable states and out-of-equilibrium dynamics. Note: the arrows represent the disequilibrium adjustment in the number of agents voting in favor of open-source production (horizontal movements for the domain of copyrights and vertical for the domain of technology).

and $\delta = n/n_r$. Because $d \leq x/\xi$, both ω^* and ρ^* are included in the closed interval $[0,1]$. The vector field in Figure 3 offers a graphical representation of such solutions, where the arrows indicate the out-of-equilibrium adjustment. The state $\{\rho^*, \omega^*\}$ is stationary, but is a saddle: small movements away from ρ^* and ω^* are not self-correcting. (Two additional unstable stationary states, namely $\{1,0\}$ and $\{0,1\}$ are of no interest) The asymptotically stable states are $\{1,1\}$ (corresponding to an *open-source organizational equilibrium*, i.e. $\{R^O, T^M\}$ in Table 1) and $\{0,0\}$ (corresponding to a *closed-source organizational equilibrium*, i.e. $\{R^C, T^L\}$ in Table 1). On this basis, the following holds:

Remark 1. *In the evolutionary game correspondent to the stage game G there exist only two asymptotically stable voting equilibria: open-source production, i.e.* $\{1,1\}$*; and closed-source production, i.e.* $\{0,0\}$.

Under this formulation, there being more than one absorbing state, the dynamic process is non-ergodic, i.e. its long-run average behavior is dependent on initial conditions. If the population starts with a distribution of votes in the area above the dashed downward-sloping line in Figure 3, i.e. in the basin of attraction of $\{1,1\}$, the population will adopt an open-source production. On the contrary, if the initial state is a point in the area below the dashed line, i.e. in the basin of attraction of $\{0,0\}$, the population will adopt a closed-source production. While such a dynamics may offer some insights on the reasons why organizations exhibit path dependency in the way in which new software is developed (Fact 3), e.g. their members have formed preferences in favor of one organization of production rather than the other, it does not explain why a specific organization of production is adopted in the first place. In order to address this point we need to transform the dynamical system into an ergodic process and, following Bowles (2006) and Naidu et al. (2010), I do so by introducing the possibility that agents engage in intentional idiosyncratic plays.

Suppose that every period there is a probability $\lambda \in (0,1)$ that each agent is selected to undertake an intentional non-best response meant at explicitly influencing the voting process in favor of the organization of production that she prefers, i.e. the one that ensures the greatest individual payoff. This transforms the dynamics into an ergodic process.

If one organization of production ensures to all agents in both sub-populations a payoff greater than the other, once such a organization is adopted no idiosyncratic plays occur and the population will remain in that state forever. In line with the Neo-institutional argument discussed in Section 2, therefore, highly efficient and conflictless organizations of production tend to be favored in the evolutionary dynamics.

The result is different, however, if we consider situations in which there exist conflicts of interest between the agents belonging to the two sub-populations. Let's consider, for instance, the following parameter ranges:

$$x > y \qquad , \qquad d < \frac{x-y}{\xi} \qquad , \qquad \frac{(x-y)}{\delta} < z < x - y - d \tag{24}$$

Looking at the payoff in Table 1 the first three inequalities imply that t-agents prefer $\{R^O, T^M\}$ while r-agents prefer $\{R^C, T^L\}$; the third inequality means instead that open-source production is more efficient than closed-source production, where efficiency is measured in terms of joint surplus available to the population as a whole. Under these conditions, when the population's state is in the basin of attraction of $\{1,1\}$, the r-agents that are selected for idiosyncratic plays have an incentive to vote for R^C because in so doing they may induce their best responding partners to vote T^L next period. For the same reason t-agents have an incentive to idiosyncratically vote for T^M when the population's state is in the basin of attraction of $\{0,0\}$. The combination of these effects may lead to the "tipping" of the population from one absorbing state to the other. The state at which the population will spend most of the time depends on the amount of non-best responses necessary to induce the best-responding partners to change their strategies, i.e. ω^* for sub-population t and $1-\rho^*$ for sub-population r. In particular, $\{1,1\}$ (and thus open-source production) is relatively persistent if and only if $\omega^* \leq 1-\rho^*$, which is the case when:

$$z \leq \frac{x^2 - y^2 - \xi(x+y)d}{\delta(y+\xi d)} = z^* \tag{25}$$

where, for the ranges of parameter described by the conditions included in the (24), $z^* > 0$. From this result, it follows that:

Remark 2. *In the evolutionary game correspondent to the stage game G, assuming that conflict of interests exists, there exist a* $z^* = [(x^2-y^2) - \xi(x+y)d]/\delta(y+\xi d)$ *such that if* $z \leq z^*$ *open-source production is persistent.*

The intuition behind Remark 2 is straightforward. So long as the rent obtainable from the sale of proprietary software is sufficiently small, r-agents do not have much to loose in adopting an open-source production and therefore are more vulnerable to the idiosyncratic plays of their counterparts. At the same time, since open-source production is less costly than closed-source production, t-agents strictly prefer the former and will try their best to get it. The combination of these two effects, make open-source production at the same time more likely to emerge and less likely to be abandoned.

The range of values for which open-source production is relatively persistent strictly depends on two main factors: the design cost of modularity d and the inverse of the relative sizes of the two sub-populations δ and ξ. For what concerns d it is easily shown that $\partial z^*/\partial d < 0$, so that open-source production becomes relatively more persistent the lower the value of d. This result could be used to interpret the impact of the so-called "digital revolution": by enabling a greater modularization of software component, the advent of digital computing has dramatically reduced d and has therefore facilitated the viability and persistence of open-source productions. On this issue a similar point has been raised also by Benkler (2002, 2006) and Baldwin and von Hippel (2009) among the others.

With respect instead to δ and ξ, their impact on the dynamics is less clearcut. In order to simplify the analysis, I reported in Figure 4 the value of z^* (vertical axis) for different combinations of δ and ξ, assuming $x = 10$, $y = 2$ and normalizing the size of the population to one (i.e. $n = 1$).[14] On the horizontal axis is reported the ratio $\delta/\xi = n_t/n_r$, and the curves are drawn for $d = 0.7, 0.5, 0.3$. The graph shows that, if we start from a situation in which the population is

[14]These values are chosen for convenience in such a way that the effects are expressed in the right order of magnitude. The general results, however, do not depend on them.

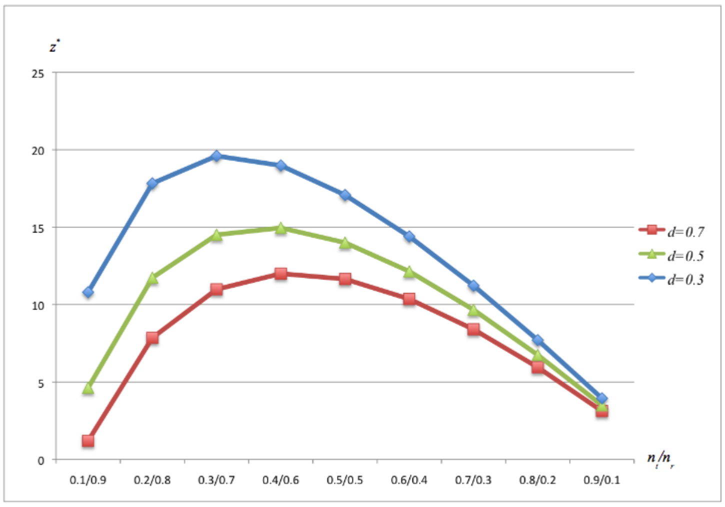

Figure 4: Variation of z^* as a function of the ratio n_t/n_r. Notice that for $z \leq z^*$ open-source is persistent. The curves are drawn assuming $x = 10$, $y = 2$, and $n = 1$. Three different values of d are considered: 0.7, 0.5 and 0.3. As the ratio n_t/n_r increases, z^* presents and inverted-U-shaped behavior, therefore making open-source initially more and then less persistent for any given value of z.

composed mainly by r-agent and we progressively increase the ratio n_t/n_r, z^* first increases - therefore making open-source production relatively more persistent for a given z, and then (after a certain threshold) decreases. The reasons behind such a behavior are related to the combination of two main forces that operate as the relative size of the two sub-populations changes. The first force is associated with the effects on the value of the payoffs at the two stable equilibria. As n_t increases and n_r decreases, in fact, the share of the design cost that goes to each t-agent in the case of open-source production diminishes, while the share of the benefits remains constant (n is fixed). This makes open-source production relatively more attractive. The same, obviously, holds for r-agents, since a reduction in n_r increases the share of the rent that would go to each of them in the case closed-source production were to take place. For low value of n_t/n_r, however, the effect of an increase in n_t for t-agents is predominant over a reduction in n_r for r-agents, and the range of parameter for which open-source production is persistent enlarges (i.e. z^* increases).

After a certain threshold, however, this trend is inverted. As n_t and n_r begin to have similar dimensions a further reduction in the share of design costs due to an increase in n_t is more than offset by the increase in the share of rent that follows a reduction in n_r, and z^* diminishes. In addition, when n_r becomes particularly small there is a second force that starts to play a role against the persistence of open-source production, which is associated with an increase in the voting power of r-agents. For a given level of λ (i.e. the probability that an agent is selected for idiosyncratic play), in fact, the smaller n_r, the more r-agents can influence the the voting process in favor of the organizational equilibrium that they prefer, namely closed-source production. Notice that in this model such a power does not depend on their greater ability to coordinate on a common and advantageous position. Rather it is a consequence of the fact that, being small, they experience more "tipping" opportunities.

This inverted-U-shaped behavior of z^* with respect to the ratio n_t/n_r is partially related to the fact that, in this setting, the design of a modular code architecture assumes the characteristics of a quasi-public good. In order to have the degree of modularity which is necessary for open-source production to emerge, in fact, a big initial investment has to be put forward by one part of the population (i.e. the sub-population of t-agents) with the resulting benefits being shared by the population as a whole.[15] Referring to the literature on the theory of economic organizations (e.g. Alchian and Demsetz, 1972) this result could be interpreted as a version of the "$1/n$ problem" where, as long as n_r is too big compared to n_t, the marginal cost of modularity is higher than its marginal benefit, therefore making open source production unsuitable. An interesting implication of this result is that the persistence of open-source productions is not affected by the size of n, but rather by the *relative* size of the two sub-populations. As a consequence open-source production could still be viable even when a large population of agents is involved, as long as, a sufficiently big portion of agents shares in the cost of software design.[16]

Equation (25) together with the conditions imposed by the (24) can also be used in order to investigate the relationship between production efficiency and the stability of the different organizational solutions. The following proposition, in particular, summarizes the main finding:

Proposition 5. *Suppose that open-source production is more efficient than closed-source production and conflict of interests exists. Assume also $n_r < n_t$. Then, there exist a range of values for the triple (y, z, d) such that closed-source production (despite being inefficient) is relatively persistent. Moreover, such a range is greater the bigger is δ.*

Proof. From equation (25) it follows that state $\{0, 0\}$ (i.e. closed-source pro-

[15] I am grateful to Ugo Pagano who suggested me the idea of modularity as a quasi-public good.

[16] This could explain why open-source projects generally succeed only when they manage to attract a sufficiently high number of software developers. Kollock (1999) refers to the latter as the minimal contribution condition.

duction) is relatively persistent if and only if:

$$z > \frac{x^2 - y^2 - \xi(x+y)d}{\delta(y+\xi d)} = z^* \tag{26}$$

Considering equation (26) jointly with the conditions imposed by the (24), closed-source production can be relatively persistent *and* inefficient as long as the following holds:

$$\frac{x^2 - y^2 - \xi(x+y)d}{\delta(y+\xi d)} < z < x - y - d \tag{27}$$

For $d = 0$, the (27) reduces to

$$\frac{x^2 - y^2}{\delta y} < z < x - y \tag{28}$$

where the closed interval defined by the (28) exists if and only if:

$$n_r < n_t \qquad and \qquad \frac{x}{\delta - 1} \leq y < x \tag{29}$$

When the (29) holds, the (28) together with the fact that $\partial z^*/\partial d < 0$ suffices to proof the first part of the proposition. The second part follows from the fact that $\partial z^*/\partial \delta < 0$. $\square$

Figure 5 summarizes the content of Proposition 5. The constraints imposed by the (24) are reported in the (d, z) plane as dotted lines, under the assumption that $n_r < n_t$ and $x > y$. The downward sloping 45-degree line defines the portions of the plane for which open-source production as opposed to close-source production is efficient: any point above the line identifies a combination of parameters (x, y, z, d) for which closed-source production is efficient, while any point below the line is a combination for which open-source production is efficient. The entire curve represents instead z^*, so that for any point above such curve closed-source production is relatively persistent. As it is easy to see, so long as z^* has a vertical intercept which is lower than $x - y$ (condition (28) above), there exist a whole set of parameter (the shaded area in the graph) for which closed-source production is at the same time relatively inefficient *and* persistent.

The fact that some organizations of production in the software industry effectively locate in this area of the graph is difficult to say theoretically, and it turns out to be mainly an empirical question. What Figure 4 shows is the possibility that, under some plausible assumptions, relatively inefficient organizations of production *can* persist over time, and are in principle difficult to displace by

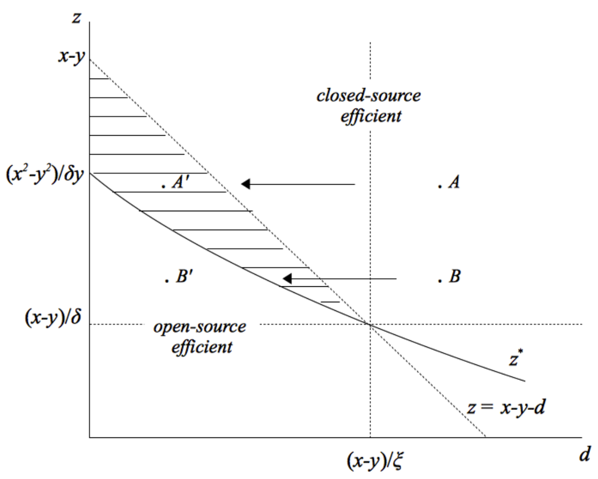

Figure 5: Production efficiency and persistence. Note: the shaded area reflects the set of points for which closed-source production is at the same time inefficient and persistent. The arrows indicate the shift caused by the diffusion of digital technologies. Starting from an economy mainly populated of closed-source productions, the reduction in the design cost of modularity brought about the emergence of an ecology of organizational forms in which open and closed-source production co-exist.

the sole force of competition. Moreover, Figure 4 can be useful also in trying to make sense of the deep change brought about by the advent of digital technologies. Let's consider for instance points A and B in the graph, which represent a combination of parameters characterized by given rents (higher in A than in B) and high design cost of modularity. These points could be interpreted as two different organizations of production that were engaged in software development prior to the digital revolution. At that time technology made it very expensive to design modular platforms, so that a close-source type of production tended to be at the same time efficient and persistent. The economy, as a consequence, exhibited a rather homogeneous nature in terms of organizational demography, with close-source production being the prevailing form (both A and B fall, in fact, in this category).

After the widespread diffusion of the Internet and the huge fragmentation of computational capabilities, however, the design cost of modularity dropped sensibly. At the same time no reform has been introduced on the property rights domain, so that rents were kept close to their initial level. As a consequence the organizations of production experienced a horizontal shift in the (d, z) space as the one depicted in the figure ($A \rightarrow A'$ and $B \rightarrow B'$). In the cases in which the rent prior to the drop in d was low, point B in the graph, the organizations most likely experienced a change form closed-source to open-source production (point B') which had become, by that time, relatively efficient (e.g. Netscape Communicator 4.0 turned into Mozilla Firefox). On the contrary, when the rent before the reduction in d was high, point A, the organization still had an incentive in maintaining a closed-source type of production (point A') despite the fact that, under the new technological environment, it had become relatively inefficient (e.g. Internet Explorer). As a result, in the economy as a whole, there has been the emergence of variegated ecology of organizational forms in which open-source and closed-source productions co-exist, *independently of* their relative (in)efficiency.

From these results together with the findings obtained in the previous sections, two main implications for the study of open and closed-source productions follow. First, the emergence of different organizations of production in the software industry can be explained not only by the presence of technological rigidities (as suggested in Section 3) but also by the fact that different softwares present different costs advantages (e.g. the presence of a motivated communities of contributors) and, especially, rent opportunities (due for instance to its diffusion). Second, the result contained in Proposition 5 raises the question on whether government intervention should be resorted to in order to increase efficiency in the economy. In this sense policy interventions could go along two main lines: first, a reform of the intellectual property rights law aimed at reducing rents from software sales; and second, a deeper involvement of public institutions (e.g. Universities) in the design of modular technologies with the objective of reducing the under-supply of modularity which is due to its own quasi-public good nature.

5 Conclusion

This paper tries to explain the sustained co-existence of open and closed-source productions in the software industry. Starting from some empirical facts (Facts 1, 2 and 3), it models open and closed-source productions as two distinct organizational equilibria that co-exist thanks to a two-ways causality between property rights and technology. On this basis the paper makes four main points: first, the organization of software development is strongly affected by the degree of technical malleability, i.e. the degree of substitutability between modularity and labor; second, there exist an institutional complementarity between the property rights and the technology used in software development; third, in the presence of high technical malleability the emergence of different organizations of production in the software industry can be motivated by differences in rent opportunities and design costs; and finally, there exist a wide range of parameters for which production inefficiency is persistent. The combination of these four points may offer an answer to questions Q1, Q2 and Q3 outlined in the Introduction.

The paper adds to the previous literature on FOSS in three ways. First of all it considers a causality in the emergence of open-source productions that has been often neglected by the previous contributions, i.e. the causality that runs from property rights to technology. Although such an argument is a well known one in social sciences, no author has previously investigated its applicability to software development. This paper fills such gap by the mean of a simple model and obtains as a result a much better sense of the empirical evidence. Secondly, the paper suggests an analogy between the evolution of organizational forms in the software industry and speciation. Similarly to the case of natural species, in fact, open and closed-source productions are presented as entities in which each part of the whole tends to become optimal *given* the nature of the other parts. For this reason, the history of their evolution is likely to be "punctuated" by sudden complementary changes followed by periods of one-by-one adjustments, with each organizational form taking distinct and mutually exclusive evolutionary trajectories. Such an analytical framework is pretty much consistent with the simultaneous shocks occurred in the late 80's as a consequence of the diffusion of the Internet and the development of free-software licenses (e.g. GPL), and the subsequent emergence and persistence of open and closed-source productions. Finally, this paper presents a clear argument in favor of the possibility that, despite of their relative (in)efficiency, both open and closed-source productions will continue to co-exist. Such an argument, obviously, does not in any sense undermine the relevance of open-source production as an effective ways of organizing software development. Rather, it limits the possibility of defending the present viability of closed-source productions by establishing a link between their protracted existence and a competitive selection process based on efficiency.

A Appendix - Figure 1

Software package	**Software Type**	**Version 1.0**	**Present version**
Google Chrome	Web browser	Open (BSD)	Open (BSD)
Internet Explorer	Web browser	Closed	Closed
Mozilla Firefox	Web browser	Closed	Open (MPL, GPL, LGPL)
Opera	Web browser	Closed	Closed
Safari	Web browser	Closed	Closed
Microsoft Access	Database	Closed	Closed
ADS	Database	Closed	Closed
Alpha Five	Database	Closed	Closed
Apache Derby	Database	Closed	Open (Apache License
Black Ray	Database	Open (GNU)	Open (GNU)
CA Datacom	Database	Closed	Closed
CSQL	Database	Open (GNU)	Open (GNU)
Cubrid	Database	Open (GNU/BSD)	Open (GNU/BSD)
Dataease	Database	Closed	Closed
Dataphor	Database	Closed	Open (BSD)
Db2	Database	Closed	Closed
Elevate db	Database	Closed	Closed
Empress	Database	Closed	Closed
Extremedb	Database	Closed	Closed
Filemaker	Database	Closed	Closed
Firebird	Database	Open (IPL, IDPL)	Open (IPL, IDPL)
Gladious Db	Database	Open (GPL)	Open (GPL)
H2(dbms)	Database	Open (MPL)	Open (MPL)
Helix	Database	Closed	Closed
Hsqldb	Database	Open (BSD)	Open (BSD)
Ingres	Database	Open (GPL)	Open (GPL)
Inter System Cache'	Database	Closed	Closed
Interbase	Database	Closed	Closed
LinterSQL	Database	Closed	Closed
Maxdb	Database	Closed	Open (GPL)
Mckoi	Database	Open (GPL)	Open (GPL)
Mimer SQL	Database	Closed	Closed
MonetDB (MDB)	Database	Open (MDB License)	Open (MDB License)
MySQL	Database	Open (GPL)	Closed
Nonstop SQL	Database	Closed	Closed
Oracle DB	Database	Closed	Closed
Postgree SQL (PSQL)	Database	Open (PSQL License)	Open (PSQL License)
RDM	Database	Closed	Closed
RDM Server	Database	Closed	Closed
Rocket U2	Database	Closed	Closed
Sas	Database	Closed	Closed
Scimore DB	Database	Closed	Closed
Solid DB	Database	Closed	Closed
SQL Server	Database	Closed	Closed
SQlite	Database	Open (Public Domain)	Open (Public Domain)
Tdbengine	Database	Closed	Closed
Timeten	Database	Closed	Closed

Software package	**Software Type**	**Version 1.0**	**Present version**
Txt SQL	Database	Open (GPL)	Open (GPL)
Vertica DB	Database	Closed	Closed
Visual Fox Pro	Database	Closed	Closed
24Seven Office	Finance	Closed	Closed
Access	Finance	Closed	Closed
Accpac	Finance	Closed	Closed
Accunts Portal	Finance	Closed	Closed
Acumatica	Finance	Closed	Closed
Adempiere	Finance	Open (GPL)	Open (GPL)
AME	Finance	Closed	Closed
Arixcel	Finance	Closed	Closed
Baan	Finance	Closed	Closed
Banana	Finance	Closed	Closed
Cgram	Finance	Closed	Closed
Coa	Finance	Closed	Closed
Compiere	Finance	Open (GPL)	Open (GPL)
CMS	Finance	Closed	Closed
Crucnch Acc.	Finance	Closed	Closed
Cubit Acc.	Finance	Closed	Closed
CYMA	Finance	Closed	Closed
EasyAS	Finance	Closed	Closed
FinanceToGo	Finance	Closed	Closed
FisrtOffice	Finance	Closed	Closed
FlexAcc.	Finance	Closed	Closed
Fortora Fresh	Finance	Closed	Closed
GNU Cash	Finance	Open (GPL)	Open (GPL)
Grisbi	Finance	Open (GPL)	Open (GPL)
HansaWorld	Finance	Closed	Closed
Home Bank	Finance	Open (GPL)	Open (GPL)
iBank	Finance	Closed	Closed
inniAccounts	Finance	Closed	Closed
Intacct	Finance	Closed	Closed
Integrater Office Acc.	Finance	Closed	Closed
IRIS	Finance	Closed	Closed
JFin	Finance	Open (GPL)	Open (GPL)
jGnash	Finance	Open (GPL)	Open (GPL)
KMyMoney	Finance	Open (GPL)	Open (GPL)
Lawson Software	Finance	Closed	Closed
LedgerSMB	Finance	Open (GPL)	Open (GPL)
Mamut Software	Finance	Closed	Closed
Microsoft AX	Finance	Closed	Closed
Microsoft GP	Finance	Closed	Closed
Microsoft NAV	Finance	Closed	Closed
Microsoft SL	Finance	Closed	Closed
Microsoft Money	Finance	Closed	Closed
Microsoft Off. Acc.	Finance	Closed	Closed
Mifos	Finance	Open (Apache License)	Open (Apache License)
Mint.com	Finance	Closed	Closed
Moneydance	Finance	Closed	Closed

Software package	**Software Type**	**Version 1.0**	**Present version**
MyOB	Finance	Closed	Closed
NetSuite	Finance	Closed	Closed
NolaPro	Finance	Closed	Closed
NOSA XP	Finance	Closed	Closed
Open Bravo	Finance	Open (MPL)	Open (MPL)
OpenERP	Finance	Open (GPL)	Open (GPL)
Open System Acc.	Finance	Closed	Closed
Openda QX	Finance	Closed	Closed
Oracle E-Business Suite	Finance	Closed	Closed
Peachtree	Finance	Closed	Closed
Pegasus	Finance	Closed	Closed
POS Solutions	Finance	Closed	Closed
PostBooks	Finance	Closed	Open (CPAL)
QuantLib	Finance	Open (BSD)	Open (BSD)
Quasar Acc.	Finance	Closed	Closed
QuickBooks	Finance	Closed	Closed
Quicken	Finance	Closed	Closed
Red Wing Software	Finance	Closed	Closed
Sage	Finance	Closed	Closed
Sage Line 50	Finance	Closed	Closed
Sage Pastel Evolution	Finance	Closed	Closed
Sage PFW ERP	Finance	Closed	Closed
SAP Business One	Finance	Closed	Closed
SAP ERP	Finance	Closed	Closed
Simply Acc.	Finance	Closed	Closed
SQL-ledger	Finance	Open (GPL)	Open (GPL)
Tally	Finance	Closed	Closed
Traverse	Finance	Closed	Closed
Trton	Finance	Open (GPL)	Open (GPL)
Turbo Cash	Finance	Open (GPL)	Open (GPL)
Ubikwiti	Finance	Closed	Closed
Xero	Finance	Closed	Closed
YNAB	Finance	Closed	Closed
Ability Off.	Office Suite	Closed	Closed
Cellframe Off.	Office Suite	Closed	Closed
Easy Off.	Office Suite	Closed	Closed
EI Off.	Office Suite	Closed	Closed
Ichitaro	Office Suite	Closed	Closed
Wordperfect Off.	Office Suite	Closed	Closed
Neo Off.	Office Suite	Open (GPL)	Open (GPL)
iWork	Office Suite	Closed	Closed
Mariner Writer	Office Suite	Closed	Closed
Auis	Office Suite	Open (GPL)	Open (GPL)
feng Off.	Office Suite	Closed	Closed
Contact Off.	Office Suite	Closed	Closed
Share Off.	Office Suite	Closed	Closed
Think Free Off.	Office Suite	Closed	Closed
Zoho Off.	Office Suite	Closed	Closed
Acrobat	Office Suite	Closed	Closed

Software package	**Software Type**	**Version 1.0**	**Present version**
Microsoft Off.	Office Suite	Closed	Closed
Open Off.	Office Suite	Closed	Open (LGPL)
Lotus Symphony	Office Suite	Closed	Closed
Go-oo	Office Suite	Open (LGPL)	Open (LGPL)
Koffice	Office Suite	Open (GPL, BSD)	Open (GPL, BSD)
Gnome Off.	Office Suite	Open (LGPL)	Open (LGPL)
Siag Off.	Office Suite	Open (LGPL)	Open (LGPL)
Kingsoft Off.	Office Suite	Closed	Closed
Softmaker Off.	Office Suite	Closed	Closed
Star Off.	Office Suite	Closed	Closed
Free BSD	Operative System	Open (BSD)	Open (BSD)
iPhone OS	Operative System	Closed	Closed
Linux OS	Open (GPL)	Open (GPL)	
Mac OS X	Operative System	Closed	Closed
Open VMS	Operative System	Closed	Closed
Symbian	Operative System	Closed	Open (EPL)
Microsoft Windows	Operative System	Closed	Closed
Apache	Web server	Open (Apache License)	Open (Apache License)
Microsoft Windows Serv.	Web server	Closed	Closed

References

Armen A. Alchian and Harold Demsetz. Production, information costs, and economic organization. *The American Economic Review*, 62(5):777–795, 1972.

Masahiko Aoki. *Toward a Comparative Institutional Analysis.* MIT Press, Cambridge, 2001.

Carliss Y. Baldwin and Kim B. Clark. The architecture of participation: Does code architecture mitigate free riding in the open source development model? *Management Science*, 52:1116–1127, 2006.

Carliss Y. Baldwin and Eric von Hippel. Modeling a paradigm shift: From producer innovation to user and open collaborative innovation. MIT Sloan Research Paper No. 4764-09, 2009.

Yochai Benkler. Coase's penguin, or linux and the nature of the firm. *The Yale Law Journal*, 112(3):369–446, 2002.

Yochai Benkler. "sharing nicely": On shareable goods and the emergence of sharing as a modality of economic production. *Yale Law Journal*, 114:273–358, 2004.

Yochai Benkler. *The Wealth of Networks: How Social Production Transforms Markets and Freedom.* 2006.

James Bessen. Open source software: Free provision of complex public goods. In Jürgen Bitzer and Philipp J.H. Schröder, editors, *The Economics of Open Source Software Development.* Elsevier B.V., Amsterdam, 2006.

Samuel Bowles. The production process in a competitive economy: Walrasian, neo-hobbesian, and marxian. *The American Economic Review*, 75(1):16–36, 1985.

Samuel Bowles. *Microeconomics: Behavior, Institutions, and Evolutions.* Princeton University Press, Princeton, 2006.

Samuel Bowles and Herbert Gintis. The power of capitalism: On the inadequacy of the conception of the capitalistic firm as private. *Philosophical Forum*, 14: 225–245, 1983.

Samuel Bowles and Herbert Gintis. A political and economic case for the democratic enterprise. *Economics and Philosophy*, 9:75–100, 1993.

Ronald H. Coase. The nature of the firm. *Economica*, 4:386–405, 1937.

Ben Craig and John Pencavel. The behavior of worker cooperatives: The plywood companies of the pacific northwest. *The American Economic Review*, 82(5):1083–1105, 1992.

Harold Demsetz. Toward a theory of property rights. *The American Economic Review*, 57(2):347–59, 1966.

Matthijs den Besten, Jean-Michel Dalle, and Fabrice Galia. The allocation of collaborative efforts in opnes-source software. *Infromation Economics and Policy*, 20:316–322, 2008.

Gregory K. Dow. Why capital hires labor: A barganining perspective. *The American Economic Review*, 83(1):118–134, 1993.

Gregory K. Dow and Louis Putterman. Why capital suppliers (usually) hire workers: What we know and what we need to know. *Journal of Economic Behavior and Organization*, 43:319–336, 2000.

Niles Eldredge and Stephen Jay Gould. Punctuated equilibria: An alternative to phyletic gradualism. In Thomas J. M. Schopf, editor, *Models in Paleobiology*, pages 82–115. Freeman, Cooper and Company, San Francisco, 1972.

Herbert Gintis. Financial markets and the political structure of the enterprise. *Journal of Economic Behavior and Organization*, 11(3):311–322, 1989.

Herbert Gintis. *Game Theory Evolving: A Problem Centered Introduction to Modeling Strategic Interaction.* Princeton University Press, Princeton, 2nd edition, 2009.

Paola Giuri, Francesco Rullani, and Salvatore Torrisi. Explaining leadership in virtual teams: The case of open source software. *Information Economics and Policy*, 20:305–315, 2008.

Sanford J. Grossman and Oliver D. Hart. The costs and benefits of ownership: A theory of vertical and lateral integration. *The Journal of Political Economy*, 94(4):691–719, 1986.

Lu Hong and Scott E. Page. Groups of diverse problem solvers can outperform groups of high-ability solvers. *Proceedings of the National Academy of Sciences*, 101(48):16385–89, 2004.

Michael C. Jensen and William H. Meckling. Theory of the firm: Managerial behavior, agency costs and ownership structure. *Journal of Financial Economics*, 3(4):305–360, 1976.

Justin Pappas Johnson. Collaboration, peer review and open source software. *Information Economics and Policy*, 18(4):477–497, 2006.

Peter Kollock. The economies of online cooperation: Gifts and public goods in cyberspace. In Marc A. Smith and Peter Kollock, editors, *Communities in Cyberspace*. Routledge, New York, 1999.

Richard E. Langlois and Giampaolo Garzarelli. Of hackers and hairdressers: Modularity and the organizational economics of open-source collaboration. *Industry and Innovation*, 15(2):125–143, 2008.

Josh Lerner and Jean Tirole. Some simple economics of open source. *The Journal of Industrial Economics*, 50(2):197–234, 2002.

Josh Lerner and Jean Tirole. The economics of technology sharing: Open source and beyond. *The Journal of Economic Perspectives*, 19(2):99–120, 2005.

Stephen A. Marglin. What do bosses do? the origins and functions of hierarchy in capitalist production. *Review of Radical Political Economy*, 6:60–112, 1974.

David McGowan. Legal implications of open-source software. *University of Illinois Legal Review*, 2001(1):241–304, 2001.

Paul Milgrom and John Roberts. Rationalizability, lerning and equilibrium in games with strategic complementarities. *Econometrica*, 58(6):1255–1277, 1990.

Suresh Naidu, Sung-Ha Hwang, and Samuel Bowles. Evolutionary bargaining with intentional idyosincratic play. *Economic Letters*, forthcoming, 2010.

Ugo Pagano. Organizational equilibria and institutional stability. In Samuel Bowles, Herbert Gintis, and B. Gustafsson, editors, *Markets and Democracy: Participation, Accountability and Efficiiency.* Cambridge University Press, Cambridge, 1993.

Ugo Pagano and Robert Rowthorn. The competitive selection of democratic firms in a world of self-sustaining institutions. In Ugo Pagano and Robert Rowthorn, editors, *Democracy and Efficiency in the Economic Enterprise.* Routledge, London, 1994a.

Ugo Pagano and Robert Rowthorn. Ownership, technology and institutional stability. *Structural Change and Economic Dynamics*, 5(2):221–242, 1994b.

Louis Putterman. Some behavioral perspective on the dominance of hierarchical over democratic forms of enterprise. *Journal of Economic Behavior and Organization*, 3(2-3):139–160, 1982.

Eric Steven Raymond. *The Cathedral and the Bazaar.* O'Reilly and Associates, Inc., Sebastopol, 1999.

Maria Alessandra Rossi. Decoding the free/open source software puzzle: A survey of theoretical and empirical contributions. In Jürgen Bitzer and Philipp J.H. Schröder, editors, *The Economics of Open Source Software Development.* Elsevier B.V., Amsterdam, 2006.

Robert Rowthorn. Neo-classicism, neo-ricardianism and marxism. *New Left Review*, 86:63–82, 1974.

Paul Samuelson. Wage and interest: A modern dissection of marxian economic models. *The American Economic Review*, 47(6):884–912, 1957.

Herbert A. Simon. The architecture of complexity. *Proceedings of the American Philosphical Society*, 106:467–482, 1962.

Oliver E. Williamson. *The Economic Institutions of Capitalism.* The Free Press, New York, 1985.