QUADERNI

Università degli Studi di Siena
DIPARTIMENTO DI ECONOMIA POLITICA

MARIA ALESSANDRA ROSSI

Decoding the "Free/Open Source (F/OSS) Software Puzzle"a survey of theoretical and empirical contributions

n. 424 - Aprile 2004

Abstract - F/OSS software has been described by many as a puzzle. In the past five years, it has stimulated the curiosity of scholars in a variety of fields, including economics, law, psychology, anthropology and computer science, so that the number of contributions on the subject has increased exponentially. The purpose of this paper is to provide a sufficiently comprehensive account of these contributions in order to draw some general conclusions on the state of our understanding of the phenomenon and identify directions for future research. The exercise suggests that what is puzzling about F/OSS is not so much the fact that people freely contribute to a good they make available to all, but rather the complexity of its institutional structure and its ability to organizationally evolve over time.

JEL Classification: K11, L22, L23, L86, O31, O34.

Keywords: F/OSS software, Innovation, Incentives, Governance, Intellectual Property Rights.

MariaAlessandra Rossi, Dipartimento di Economia Politica, Università di Siena

1. A brief introduction to F/OSS software development: is there an "F/OSS puzzle" to be decoded?

In the early days of the free/open source (F/OSS) phenomenon neither its proponents nor its opponents could imagine its subsequent success. In a 1976 "open letter to hobbyists" Bill Gates wrote: *"Who can afford to do professional work for nothing? what hobbyist can put three man-years into programming, finding all bugs, documenting his product, and distribute for free?"* (Moody, 2001). Later on Linus Torvalds, the father of the popular F/OSS operating system Linux, has frequently acknowledged his surprise at the extraordinary participation Linux has been able to attract over time (Diamonds and Torvalds, 2001). At present, Linux is successfully competing with Windows, the Apache HTTP server runs over 67 per cent of the world's websites as of November 2003 (Netcraft, 2003) and many more F/OSS projects are widely adopted (Perl, Sendmail, FreeSBD, Mozilla, GCC etc.). Moreover, many governments around the world have endorsed to various extents F/OSS software, especially Linux.

The defining characteristics of F/OSS are a) the free availability of its source code, namely of the human-readable instructions that compose a given program, and b) the nature of the license under which it is distributed, which allows both its free modification and its free redistribution. This distinguishes F/OSS not only from proprietary software, distributed in object-code form under a license that sets clear restrictions to its usage and requires payment of a fee, but also from freeware and shareware, that can both be downloaded free of charge but do not allow access to the source code. The combination of the mentioned characteristics allows a scattered group of developers to modify existing programs and redistribute them.

The number of contributions exploring the phenomenon has increased exponentially in the past five years. The purpose of this paper is to provide a sufficiently comprehensive account of these contributions in order to draw some general conclusions on the state of our understanding of the phenomenon and identify directions for future research. Partly as a consequence of the fact that some F/OSS enthusiasts have portrayed the F/OSS world as a world from which *homo oeconomicus* has been banned, the dominant question addressed in the literature has been whether the F/OSSphenomenon represents a puzzle for economic theory or can be accommodated into a standard economic framework. This question will not be crucial to the paper. There are good reasons to believe that what is puzzling about F/OSSis not so much the fact that people freely contribute to a good they make available to all, but rather the complexity of its institutional structure and its ability to organizationally evolve over time. For one thing, in the early days of software programming, free distribution of source code was taken for granted and the cultural frame in which software development took place was akin to the realm of "Open Science" described by Dasgupta and David (1994).

The paper is organized as follows: in section two the various hypotheses advanced to explain voluntary contributions to F/OSSprojects will be reviewed in light of the empirical evidence available, whereas the focus of section three will be on papers dealing with the governance of F/OSSprojects. Section four will broaden the perspective so as to look at the competitive and cooperative relationships of F/OSS communities with commercial actors. In section five the main questions raised by the

F/OSS development model in relation to intellectual property rights will be addressed. Section six will take up the issue of the rationale for government support of F/OSS projects. Section seven concludes.

2. What motivates contributions to F/OSS projects?

Among the many questions raised by OSS, that concerning individual incentives and motivations has received by far the greatest attention. Scholarly contributions on this matter tend to reflect authors' personal conception of human nature and their beliefs about the interpretive power of particular theories of social interaction, with the result that the heterogeneity of the phenomenon has generally been downplayed. Not only F/OSS projects differ significantly from one another in terms of the complexity of the software developed, its degree of modularity, the nature of coordination or the intensity of communication among the various contributors, just to cite few relevant dimensions of the heterogeneity, but within the same project participants differ as regards the intensity of their contributions, the primary motivation for contributing, the level of ideological commitment etc. It is thus unlikely that one-sided interpretations of the motivation leading to contribute to F/OSS software development could capture a sufficient degree of such heterogeneity. Luckily enough, the growing number of empirical explorations of the F/OSS phenomenon can provide an efficient check on the powerfulness of the various explanatory variables proposed.

In what follows a significant number of scholarly contributions on the issue of F/OSS developers' motivation will be reviewed, following an expositional scheme that has come to be widely accepted in the literature (Hars and Ou, 2001; Osteloh, 2002; Lakhani and Wolf, 2003). According to this scheme, individual motivations can be grouped under two broad headings: intrinsic and extrinsic. Extrinsic motivations belong to the realm of "off-the-shelf" economic theory and relate to the immediate or delayed benefits accruing to the individual in a mediated form, in general through monetary compensation. Intrinsic motivation differs from extrinsic motivation primarily because it is valued *per se*. As Ryan and Deci (2000) put it: "*Intrinsic motivation is defined as the doing of an activity for its inherent satisfactions rather than for some separable consequence. When intrinsically motivated, a person is moved to act for the fun or challenge entailed rather than because of external prods, pressures or rewards*"[1]. Lindenberg (2001) has proposed a further distinction between enjoyment-based intrinsic motivation, which is closer to the previous definition, and obligation/community-based intrinsic motivation that refers to the intrinsic benefits that may be associated to adherence to social or community norms.

[1] Essential components of intrinsic motivation are in general assumed to be a) self-determination (Deci, 1980) and autonomy, stemming from the individual's perception of freedom, identity, responsibility, and control and b) competence, which is positively related to intrinsic motivation to the extent that the task at hand is neither far within nor far beyond the individual's level of competence (Deci and Ryan, 1985).

2.1. Extrinsic motivations: reputation, user needs, learning and performance improvement.

One of the most powerful accounts of the role played by extrinsic motivation in explaining contributions to F/OSS projects is surely Lerner and Tirole's (2002) article. The two authors posit that *"a programmer participates in a project, whether commercial or F/OSS, only if she derives a net benefit (broadly defined) from engaging in the activity. The net benefit is equal to the immediate payoff (current benefits minus current costs) plus the delayed payoff (delayed benefits minus delayed costs)"* and derive that the F/OSS phenomenon can be (almost entirely) reconciled with standard economic arguments.

Lerner and Tirole's argument is, indeed, persuasive, and firmly grounded on some empirical facts relevant in the context of OS software. Their focus is on two aspects of F/OSS development: a) the immediate benefits obtained by oss developers in "scratching an itch", namely solving a problem they face and b) the existence of a "signalling incentive" that derives from the gratification associated to peer recognition or from delayed benefits in the form of better job offers relative to non-F/OSS developers.

Lerner and Tirole's main line of argument is thus that once the above mentioned immediate and delayed benefits are factored in an individual developer's cost-benefit calculation, voluntary contributions are not as startling as it might first appear. Indeed, the F/OSS environment represents in many respects a better context than proprietary settings for the reputational mechanism to display its effects. The costs incurred to build up a reputation (including both the effort spent and the opportunity cost of forgone monetary compensation or of not focusing on one's primary mission) might be lowered by a) the "alumni effect", i.e. familiarity with the programming language due to the fact that programs available for free are often used as teaching tools in schools and universities; b) the private benefit of getting a bug fixed or solving another programming problem and c) the enjoyment inherent in a challenging and/or just fun activity. As for the benefits, the authors stress that F/OSS projects fulfill the main requirements spelled out in the literature on signaling incentives (Holmstrom, 1999). Compared to closed source projects, F/OSS software development has a number of advantages in that a) the openness of the source code allows better performance measurement and increases the visibility of one's own contribution to the relevant audience; b) each programmer is fully responsible for her contribution, so that the source code available to anyone for inspection more directly reflects her talent; and c) the labor market is more fluid (for a formal model very much in the spirit of Lerner and Tirole's contribution, see Lee, Mosa and Weiss (2003)).

As subsequent academic commentary has emphasized, proponents of the "signaling incentive" argument tell us only part of the F/OSS story, for as much as the reputation mechanism might be important to the F/OSS community, it does not provide an adequate explanation to some relevant empirical facts. First, the characteristics of F/OSS projects indeed allow developers joining them to gain benefits in terms of reputation and may explain partly or even fully the reason for joining an existing project. What the reputation argument does not make clear is why would a "top-notch programmer" decide to initiate a project in the first place by rendering freely available

on the web code that might have significant commercial value if taken private[2]. Linus Torvalds, the initiator of one of the most successful F/OSS projects (Linux) was not primarily motivated by reputation concerns, but rather from a desire to participate in a community he identified with (Diamond and Torvalds, 2001). Suspects are legitimate that there might be more to F/OSS communities than simply reputation-sustaining devices.

Second, as Steve Weber (2000) points out[3], if reputation were the primary driver of contributions, we would probably observe significantly more direct challenges to project leaders' authority and more "strategic forking", namely more deviations from the main development path motivated by ambition for leadership. After all, reputation and leadership are positional goods (Hirsch, 1976), which entails that in a given community at any point of time aggregate consumption of such goods is equal to zero, so that their enjoyment is affected by whether or not other people consume them in positive amounts (see also Pagano, 1999 and 2002). In other words, not all Linux contributors will be leaders and/or will be able to enjoy good reputations. Indeed, a closer look at day-to-day operation of F/OSS communities shows that forking is a rare event and that *"there is strong social pressure against forking projects. It does not happen except under plea of dire necessity, with much public justification, and with a renaming"*(Raymond, 1999b).

Third, the hierarchical structure of many F/OSS projects casts additional doubts on the dominance of reputation-related motives on programmers' decision to contribute to F/OSS development (McGowan, 2002). This is because the extent to which each programmer will be able to reap reputational benefits depends crucially on the project maintainer's discretional decision, given a hierarchical allocation of decisional power as regards the choice of what pieces of source code should be incorporated into the official release of a program.

Fourth, and most importantly, empirical analyses of the F/OSS community have made available information that counters the reputation-maximizing story. One piece of empirical evidence shows that a wide majority of participants to F/OSS projects does not even actually contribute code, but rather comments and descriptions of the applications (Dempsey et al. 1999), which is hard to consider the best activity to choose if one is willing to engage in reputation-gaining exhibition of skilfulness. Hann et al.'s (2002) study of three F/OSS projects under the control of the Apache Software Foundation obtains somewhat ambiguous result. On one side, they reject the hypothesis of a positive correlation between greater participation to the projects (as measured by the number of contributions made) and wage increases. On the other side, they find a positive correlation between status in the Apache community and wages[4]. Although this second result can be interpreted as consistent with the "signalling incentive" theory, as the authors do, the two results taken together do not lead to a straightforward conclusion. In fact, one might still wonder why would the average contributor put any

[2] Lerner and Tirole (2002, p.215) briefly address this issue, but focus only on the case in which the releaser of the original source code is employed by a corporation and is prevented by the latter from taking the code private.

[3] This point is also made by McGowan (2002, p.39).

[4] This result may be biased because of the history of the Apache Software Foundation, initiated by professionals who still enjoy a high status in the Apache community and who are likely to earn high wages quite independently from their participation in the ASF activities.

effort into F/OSS project if her principal concern was to build up her reputation and her level of contribution did not matter to that end.

Finally, the numbers reported by all the empirical analyses to date (Gosh et al., 2001; Hars and Ou, 2001; Hertel et al. 2003; Lakhani and Wolf, 2001) suggest that reputation *per se* is by no means the most prominent motive for contributing to F/OSS projects. Although these findings may reflect to some extent contributors' self-assessment biases, they should not be dismissed too quickly.

Another driver of voluntary contributions to F/OSS projects pertaining to the category of extrinsic motivations is the already mentioned purpose of "scratching an itch", namely fixing a bug or solving a problem of immediate relevance to the programmer. *User needs* indeed constitute a powerful incentive to create the software code in the first place and, given the low costs associated to the act of making the code available through a means as powerful as the web, the decision to distribute it for free may not appear particularly startling. The existence of *"horizontal user-only innovation networks"* (von Hippel, 2002)[5], innovation by lead users, and free revelation of details of the innovation to manufacturers or even to other users has been documented in a variety of fields (see for example Allen, 1983; von Hippel,1988; Morrison et al, 2000) [6].

In general, free revelation of innovations will occur if the benefits associated to it outweigh its costs. In the specific case of F/OSS development, costs of diffusion are kept down by the availability of an efficient communication technology, i.e. the internet (Kollock, 1999), whereas opportunity costs might be sufficiently low at least for some of the heterogeneous participants to the F/OSS community. Indeed, as reported by empirical investigations of the demographic of F/OSS development (see for instance Gosh et al., 2001), a significant fraction of the contributors to F/OSS projects is made up by students who typically do not regard potential adopters of their innovation as rivals. As for the benefits associated to diffusion, at least two forces are at work: one is a standard network effect according to which users of a given operating system or application benefit from the expansion of the user base; the second is the increase in the number of programmers that can be more directly useful to the original innovator by reporting bugs and making additions to the source code (Lerner and Tirole, 2002; von Krogh, 1998 and 2002).

User needs may explain not only why developers contribute source code, but also why uninteresting and mundane tasks such as providing documentation or support to users are efficiently performed, although they are neither appreciated per se, for the intrinsic pleasure and enjoyment a programmer may derive from them, nor can convey a significant signal of the programmer's ability. Lakhani and Von Hippel (2003) address exactly this issue, empirically analysing the online help system for Apache. Their main finding is that the surprising degree of effectiveness of the voluntary online support for

[5] According to von Hippel (2002), "User innovation networks can function entirely independently of manufacturers when (1) at least some users have sufficient incentive to innovate, (2) at least some users have an incentive to voluntarily reveal their innovations, and (3) diffusion of innovations by users is low cost and can compete with commercial production and distribution."

[6] In a more general context than that of F/OSS context, the following incentives have been pointed out as conducive to free revelation of innovations to other users and/or to manufacturers (Harhoff, Henkel, von Hippel, 2000): a) inducing manufacturers improvement; b) setting a standard advantageous to the user innovator; c) reciprocity and reputation effects; d) low rivalry conditions.

users depends on the opportunity for learning valuable information such activity offers to providers, through the public posting of answers and questions.

Note that an important aspect of the explanation of the motivation to contribute to F/OSS projects in terms of the need to satisfy particular user needs that has been overlooked so far is the heterogeneity of such needs. Were user needs homogeneous, we could expect F/OSS development to resemble a waiting game much more than empirical observation allows us to. Heterogeneity of user needs ensures that the probability of being able to free ride on the contribution of other developers for a specific piece of code is sufficiently low that a programmer may find it worthwhile to develop it by herself (Bessen, 2001)[7].

What is the empirical relevance of user needs as a motivation to participate in F/OSS communities? 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[8]. Gosh et al. (2002) in the so-called FLOSS survey also find a significant percentage of respondents indicating motivations that can be associated to the category of user needs. Similarly, Hertel et al. (2003) find a pragmatic interest in personal advantages associated to improving the functionality of the Linux kernel to be a relevant motive for contributing and to be positively correlated with the intensity of the development effort[9].

However, one should be cautious in concluding that satisfaction of user needs per se represents an adequate explanation for the significant amount of contribution to F/OSS projects that we currently observe. For one thing, the costs of contributing are in fact not as trivial as it might first appear, as publication of code on the internet requires time-consuming publication of the relevant documentation as well. In addition to this, if the mentioned empirical results are put in perspective and the relevance of user needs-related motivations is considered relative to the other motivations indicated, the picture is much less clear-cut. Indeed, Lankhani and Wolf (2003) conclude that *"the F/OSS community has heterogeneous motives to participate and contribute. Individuals may join for a variety of reasons, and no one reason tends to dominate the community or cause people to make distinct choices in beliefs."* Similarly, the FLOSS survey finds a number of motivations distinct from user needs to be relevant to the decision to join and to stay in the F/OSS community, a significant portion of which fares better in absolute terms than user needs[10].

In sum, the empirical evidence available to date confirms that the pursuit of extrinsic rewards does play an appreciable role in motivating contributions to F/OSS

[7] Franke and von Hippel (2003) focus on the heterogeneity of user needs in their study of the Apache Security Software, and propose the extension of the "innovation toolkit" approach adopted in the Apache Security functionality to other markets characterized by heterogeneous demand.

[8] The study, based on the Boston Consulting Group survey, finds work-related user needs to be relevant to 33.8% of participants and non-work-related user needs to be relevant to 29,7% of participant, with only 5% of respondents choosing both types of user needs as important.

[9] It should be mentioned, however, that both Gosh et al.(2002) and Hertel et al.(2003) studies address the question of the relevance of user needs less directly than Lakhani and Wolf's study.

[10] The most prominent reported motive for participation is in fact to "learn and develop new skills"(78,9%), which is followed, interestingly, by more "social" motivations such as the desire to "participate in a new form of cooperation" (34,5%) and "share knowledge and skills"(49,7%), just to cite two of the more often reported motives.

development projects. Learning and performance improvement also tend to be ranked among the top reported motives for contributing in all the surveys reviewed (Gosh et al., 2001; Hars and Ou, 2001; Hertel et al. 2003; Lakhani and Wolf, 2001), although they do not find an appreciable space in the theoretical literature.

2.2. Intrinsic motivations: hedonic motivations, altruism, generalized reciprocity and gift-giving attitude.

Under the heading "intrinsic motivations" a variety of circumstances affecting the decision to contribute can be grouped, ranging from the pure pleasure of programming to more social-oriented motives such as the desire to participate in a collaborative activity or the idealistic convincement that software should be free.

The fact that F/OSS contributors enjoy their programming activities is certainly out of doubt. The characteristics of F/OSS development processes create an environment that tends to be more suitable than proprietary settings to express one's creativity, have fun and experience a sense of satisfaction and accomplishment. This is partly because of the greater freedom of experimentation in writing the code and partly because of greater autonomy in selecting projects that match one's own skill level[11]. Among the few theoretical contributions addressing the issue of enjoyment-based intrinsic motivation, the one that places greater emphasis on human creativity is Moglen's (1999). Professor Moglen attributes to the dramatic decrease in the cost of global interconnection made possible by the Internet the expansion of the possibility of expression of human creativity that is ultimately responsible for the development of the "anarchic" F/OSS mode of production.

The individual enjoyment of programming represents necessarily only part of the intrinsic motivation to contribute to F/OSS projects. Altruism might be an additional component, frequently mentioned by proponents of the F/OSS approach. While it is reasonable to ask why we should expect F/OSS developers to be more altruistic that others, the presence of a concern for increasing the welfare of others should not be ruled out altogether.

Obligation and community-based intrinsic motivations constitute an additional element of the picture. Even casual observation of the working of F/OSS projects suggests the importance of implicit or explicit norms that define the "hacker identity". The sense of community identification indeed pre-exists the popularity of F/OSS software, as the following quote from Linus Torvalds makes clear:

> *"…the act of making Linux available wasn't some agonizing decision that I took from thinking long and hard on it: it was a natural decision within the community that I felt I wanted to be a part of."(Torvalds, 1998)*

Obligation-based intrinsic motivations have been variously described as arising from generalized reciprocity, gift-giving culture, or a clearly defined sense of identification with the F/OSS community that may or may not entail the sharing of an explicit anti-Microsoft attitude.

Kollock (1999) applies to virtual communities the notion of *generalized exchange* developed by Ekeh (1974), suggesting that voluntary contribution of information to

[11] Lakhani and Wolf (2001) indeed report that F/OSS developers enjoy flow states and loose track of time when working on F/OSS projects.

virtual networks is supported by an expectation of *generalized reciprocity*. Whereas anonymous interactions in the digital environment prevent direct reciprocation, it is possible for contributions to occur on the basis of the expectation that a balance might occur within the group as a whole (for a contrasting view, emphasizing the absence of any form of reciprocity, see Moglen, 1999).

The culture of gift-giving has been often mentioned as an important characteristic of F/OSS communities (see Dang Nguyen and Pénard, 1999 for a perspective on gift-giving in the internet more generally). Eric Raymond, one of the most active ethnographers of the F/OSS community, depicts the F/OSS culture as a gift culture in which economic scarcity is not an issue and social status is determined *"not by what you control but by what you give away"* (Raymond, 1999). Bergquist and Ljungberg (2001) and Zeitlyn (2003) subsequently specify the terms of Raymond's analogy between gift-giving and contribution to F/OSS software. While differing in focus, both papers make explicit reference to the classic work of Mauss (1950) and thus acknowledge the fact that gift giving is associated to an obligation to reciprocate and creates social interdependencies. Bergquist and Ljungberg (2001) focus on the idea that gifts in F/OSS communities are at the origin of power relations based on reputation and make a parallel with peer review in academia. Zeitlyn (2003) also makes the point that gift-giving enhances reputation, but stresses the relevance of reputation for generosity as a means to accumulate *"symbolic capital"* in Bourdieu terms. What is perhaps understated in these accounts of the gift-giving culture of F/OSS communities is the role played by F/OSS licenses. The latter ensures that the gift given to the community cannot be privately appropriated and thus allows an indirect form of reciprocation to be sustained. Interpretations that emphasize the power-relations aspect of the gift-giving culture tend to miss this important point.

If the characterization of the F/OSS milieu as a gift giving culture might be controversial[12], much less controversy exists around the fact that F/OSS developers demonstrate an appreciable sense of identification with their community. The extent to which the norms and values of the community are internalised differs across participants, with those who subscribe to the views of the Free Software Foundation[13] being at the extreme of the spectrum of ideological commitment. A coherent set of norms, values and shared beliefs informs interactions in the F/OSS community, to the point that newcomers ("newbies") are thought the basic principles of the F/OSS community culture before they actually start participating (Bergquist and Ljungberg, 2001). Those principles can be (very roughly) summarized by mentioning a) the belief that software should be free; b) the high value attached to a cooperative attitude and the sharing of information; c) appreciation for technical knowledge and high-quality source code; d) the importance of reputation among one's peers; e) the importance of recognizing a developer's valuable contribution of source code by giving it appropriate credit. A shared opposition to Microsoft and the desire to limit the power of large software companies more generally are not necessarily beliefs the whole community

[12] For instance Weber (2000) rightly questions what is exactly to be considered abundant in the digital world besides bandwidth and computing power and suggests that, while the latter are devalued exactly because they are abundant, scarcity still dominates the domain of human creativity and brainpower.

[13] Advocates of the Foundation have a strong ideological commitment to the idea that software should be free, whereas proponents of the F/OSS Definition tend to have a more pragmatic approach and value free software mainly for the concrete benefits it offers (Feller and Fitzgerald, 2002).

ascribes to, although some commentators have stressed the role of perceived competition of OSS projects with commercial software companies as an important motivating factor (Bezroukov, 1999).

Recognition of the existence of a shared set of normative, principled and causal beliefs informs the analyses of Kasper Edwards (2001) and Cohendet, Creplet and Dupouet (2001), that propose a framework based on the combination of the theory of epistemic communities and the theory of situated learning and legitimate peripheral participation (Lave and Wenger, 1991) to understand the motivation of F/OSS developers (on the role of peripheral members in online communities see also Zhang and Storck, 2001). Ilkka Tuomi (2001) combines the theory of "communities of practice" with actor-network theory to understand participation to the Linux kernel. Other scholars have emphasized the similarities between F/OSS communities and a particular kind of epistemic communities, namely scientific communities (Bezroukov, 1999; Raymond, 1999c; Kelty, 2001; David, Arora and Steinmuller, 2001).

The relevance of the sense of identification with the community as a powerful driver of individual participation to F/OSS projects has been assessed by all of the empirical analyses of motivation realized so far (Gosh et al., 2001; Hars and Ou, 2001; Hertel et al. 2003; Lakhani and Wolf, 2001)[14]. In Lakhani and Wolf's (2003) study the components of obligation and community based motivation are highly ranked by participants[15]. Similar results appear in the FLOSS survey (Gosh et al.2002)[16].

Hertel et al.(2003) study contributions to the Linux kernel, and find that identification with the Linux community has the strongest influence on participants engagement, together with the more pragmatic motive of improving own software and with tolerance of time investments. Moreover, 59% of their respondents indicated that they worked in teams. Analysis at the individual team level shows that "instrumentality"(i.e. the perceived importance or indispensability of one's own contribution for the group outcome) and "valence" (i.e. the subjective evaluation of team goals) are the best predictors for time investment in F/OSS programming and willingness to engage in programming in the future, whereas perceived self-efficacy is the best predictor for performance (number of lines of code and "patches" contributed).

As for the other aspects of intrinsic motivation, the mentioned empirical studies confirm their concrete relevance. Hars and Ou (2001) find what they call "self-determination", namely the feeling of accomplishment, competence, enjoyment and effectiveness, to be the second highest-rated motivation, relevant to 79.7% of respondents. Lakhani and Wolf (2003) find that enjoyment-based intrinsic motivation (the sense of creativity and intellectual stimulation experienced when working on the

[14] One study, by Stewart and Gosain (2003), addresses directly the impact of trust and ideology on the effectiveness of team work in F/OSS communities finding a positive influence on team effectiveness of cognitive trust, which is in turn influenced by adherence to community norms. However, they do not find a direct relationship between adherence to community norms and team effectiveness.

[15] One third of respondents indicate for example that the "belief that source code should be open" was an important reason for participation. Self-identification with the hacker community was also confirmed by a 42% of respondents indicating that they "strongly agree with hacker ethic" and another 41% indicating to "somewhat agree" with the statement that the hacker community is the primary source of their identity.

[16] What is interesting about the results of this study is that the decision to join the community and the decision to stay in the community are separately investigated, so that it is possible to get a feeling of the evolution of motivations over time. Obligation and community based motivations ("political" motivations in the language of the study) display a tendency to increase appreciably over time.

project) is the strongest and more pervasive factor[17]. In addition to this, they also report the very interesting finding that, differently from the consistent results of previous studies on the negative impact of extrinsic rewards on intrinsic motivation (Deci, Koestner and Ryan, 1999), being paid for participating to an F/OSS project does not crowd out individual motivation. Altruistic motivations are also found to be of some relevance[18].

On balance, it is difficult to extract from the empirical literature a case for the prevalence of either intrinsic or extrinsic motivations in determining contributions to F/OSS projects. Hars and Ou (2001) conclude for the dominance of extrinsic motivations, although a look at the numbers they report questions such a clear-cut conclusion. The studies by Gosh et al.(2002) and Hertel et al.(2003) do not allow an easy comparison among extrinsic and intrinsic motives, whereas Lakhani and Wolf's study concludes that the main driver of effort is given by enjoyment-based intrinsic motivation. It thus appears that the question whether one or the other category of motivations dominates the other is not as interesting as the question concerning how the different motives interact.

The issue of the interplay between different motives has been addressed by Osteloch et al. (2002) and Frank and Jungwirth (2002). Osterloch et al. (2002) make a case on the complementarity between extrinsic and intrinsic motivations, describing F/OSS projects as instances in which a "tragedy of the commons" in innovation activities can be overcome without strong property rights and central authority because of the heterogeneous nature of participants' motivation. In this interpretation, the contribution of intrinsically motivated developers they describe as "fun seekers" and/or "members of the tribe" is essential to get the project underway and to ensure attention by extrinsically motivated people such as "commercial players", "lead users" or "reputation investors".

The complementarity among different motivations is also stressed by Frank and Jungwirth (2002). What characterizes F/OSS development is – according to the two authors – the crafting of an institutional structure that allows to reconcile the interests of "investors", motivated exclusively by cost-benefit considerations, and "donators" acting out of idealistic motives. The distinction is perhaps a bit harsh, as both kinds of motivations are in general relevant to the same developer, but the focus of the paper is on the working of the institutional structure of F/OSS projects rather than on exploring extensively the nature of programmers' motivations. Central to the disclosure-feedback approach adopted by F/OSS development is the GPL licence that, while sustaining a reputation-based incentive structure and acting as a "non-distribution constraint" akin to those relevant in the theory of non-profit organizations, does not crowd out investors interested in second-use and commercialization of F/OSS software.

[17] In Hertel et al. (2003) survey enjoyment-based and community-based intrinsic motivations are more difficult to disaggregate, whereas Gosh et al. (2002) do not address directly the question of enjoyment-based intrinsic motivations.

[18] They are important for instance to 16.5% of the developers surveyed in Hars and Ou's (2001) study. The percentage increases, perhaps unsurprisingly, when results are disaggregated by programmer "type", with the 24.2% of students and hobby programmers rating high on altruism.

3. The governance of F/OSS projects

F/OSS software development can be considered a success story not only in terms of the number of individuals it has attracted over time, but also in terms of its inherent characteristics, praised not only by its advocates (Raymond, 1998; Jorgensen, 2001; Neumann, 2000) but by rivals as well, as in Microsoft's so-called "Halloween Document"(Valloppillil, 1998). F/OSS projects tend to be complex, and indeed the challenge associated to their complexity is part of the reason why they are able to attract competent and highly motivated developers, as discussed in the previous section. Moreover, F/OSS development seems to defeat the widely accepted assertion in software development that *"adding manpower to a late software product makes it later"*, also known as "Brooks' law" (Brooks, 1975). Indeed, in contrast to Brooks' law, the success of F/OSS development seems driven precisely by the high number of skilled developers that participate in it.
Another issue that has attracted a great deal of scholarly attention concerns therefore the identification of the features of OS software that determine its success and, relatedly, the identification of the characteristics that distinguish it from proprietary development. In what follows this strand of the literature will be introduced by reporting the answers it has provided to three broad questions: what are the principal features of the "F/OSS production mode"? how are F/OSS projects coordinated? and, how is coordination sustained?.

3.1. What are the principal features of the "F/OSS production mode"?

In a ground-breaking article Eric Raymond (1998) has characterized the internal organization of F/OSS projects (more specifically, of Linux) as a bazaar-style programming mode, as opposed to the cathedral-style organizational model adopted by commercial developers. The "bazaar" software development model relies on a number of fundamental principles, partly inherited from the tradition of Unix programming and partly evolved spontaneously along with the Linux software code. Raymond lists a great number of those principles, but for the purposes of the present discussion we deem it sufficient to mention the following (paraphrasing Raymond to some extent):

- *Avoid wasteful duplication of programming efforts by reusing existing software*[19]*;*
- *"Release early, release often. And listen to your customers";*
- *"Given a large enough beta-tester and co-developer base, almost every problem will be characterized quickly and the fix obvious to someone" (Linus' law);*
- *Exploit the distributed intelligence of the system by attaching high value to the users' suggestions and by recognizing their good ideas.*

[19] Weber (2000) notes that this principle is strictly related to the adoption of F/OSS licenses that ensure programmers ongoing access to the source code and therefore softens the pressure to "reinvent the wheel" present under proprietary development.

The metaphor of the bazaar refers to a system of distributed innovation, involving large numbers of developers and characterized by a) the absence of a centralized decision-making unit defining ex-ante the direction of development of the software code; b) concurrent design and debugging; c) the integration of users into the production of software code; d) self-selection of programmers for the tasks that best match their abilities.

As attention towards F/OSS software in general, and its governance structure in particular, has grown considerably, the stylised "cathedral-bazaar" metaphor has been increasingly perceived as inadequate to portrait the F/OSS development process. However, each of the mentioned characteristics identified more or less explicitly by Raymond (1998) has subsequently become the object of academic commentary.

Point a and b are closely related. F/OSS development represents a *process innovation* (Bonaccorsi and Rossi, 2003a; Mateos-Garcia and Steinmuller, 2003) among other things because it adopts on a very large scale a process of parallel development. Parallel development is enabled by the modularization of the source code[20]. What this means is that single programs are assembled through the combination of relatively small and relatively independent components that can perform very complex functionalities in so far as they have effective communication interfaces. Modularity ensures that any change or addition to the source code belonging to a given module will have only limited or no reverberation on the rest of the system, thus allowing different developers to work on different components at the same time without fear of interference (on the implications of a modular architecture for the effectiveness of the F/OSS development model see also Baldwin and Clark, 2003)[21]. Parallel development thus entails both an increase in programming speed and a potential increase in quality in circumstances in which different developers work in parallel on the same component (Feller and Fitzgerald, 2002). These benefits should be contrasted to the possibility that this last circumstance might entail wasteful duplication of effort, as only one of the solutions proposed will be incorporated into the official release of a given F/OSS product. However, one aspect of F/OSS development that might mitigate the mentioned efficiency loss is the *"release early, release often"* principle that implies that information about the availability of a given piece of software code reaches developers of potential alternative solutions early enough to assess the convenience of pursuing their development effort further.

As for point b, Kogut and Methiu (2001) indicate in the concurrent process of design and debugging one of the two sources of efficiency gains associated to F/OSS development (the other being the integration of users into production). According to the two authors, proprietary software development tends to proceed at a pace determined by the *least productive* contributor, following a "weak link" chain production function. By contrast, F/OSS software development fully exploits the intelligence of the community and thus proceeds at a pace determined by the *most productive* member of the community.

[20] Note that, while a modular structure characterizes most F/OSS projects, it is not a necessary condition of F/OSS development. One example of F/OSS project with a non-modular architecture – reported by Kim (2002) – is TouchGraph, a software component for visualizing networks of information.

[21] Note that Linus Torvalds attributes to the modularization of the software code the very success of Linux. (Torvalds, 1999).

The integration of users into the production of software code – point c – is a feature of F/OSS software that has been emphasized in a large number of contributions (Harhoff et al., 2000; Johnson, 2000; Kogut and Methiu, 2001; Bessen, 2002; Kuan, 2002; von Hippel and von Krogh, 2003). Von Hippel and von Krogh (2003) provide a coherent framework for this issue, describing F/OSS as a private-collective model of innovation. F/OSS software development displays significant departures both from the private model of innovation (as typified by the development of proprietary software) and from the collective model of innovation.

The key to understanding the effectiveness of the F/OSS production model – according to the two authors – is the fact that the outcome of the innovative process is not regarded as a pure public good, but possesses significant benefits that can be privately appropriable even if free revelation makes the innovation available to anyone who cares of making use of it. In this perspective, whereas free revelation of an innovation does not necessarily result in a loss of profit for its creator, free-riders are not able to obtain exactly the same benefits from the innovation as those who have contributed to it. This, in turn, explains contributions to F/OSS projects entirely in terms of the private benefits they entail.

The implications of this characterization of F/OSS innovation for the governance of F/OSS projects are at least twofold. First, the relevance of contributions by users limits the need for monitoring and selective incentives to reduce free-riding. Second, differently from standard collective action settings, large groups of users-contributors are not necessarily less effective than small groups. To the contrary, whereas in general increases in group size encourage free riding and raise monitoring costs, in the case of F/OSS projects increasing the user-developer base positively affects developers' incentives both because of network/compatibility effects and because each new participant is in effect a potential debugger/developer (Weber, 2000; von Hippel and von Krogh, 2002). It should be noted, however, that agreement on the latter point is not unanimous[22].

Benkler (2001) expands upon the notion that programmers self-select for the tasks they are best at (point d), adopting it as a building block of a theory of *"commons-based peer production"* that stands in contrast to market-based and hierarchy-based production modes. He compares the three organizational modes along two principal dimensions: a) the effectiveness in elaborating human capital-related information; and b) the effectiveness in combining human capital and other resources. As for a, the relevant metrics for the comparison is the *"information opportunity cost"* entailed by the different production modes, namely the loss of information relative to an ideal state of perfect information. Peer production entails a lower *"information opportunity cost"* with respect to both markets and hierarchies because it does not require either the mediation of the price system or a standard contractual specification of the effort required (both difficult and imperfect because of the tacit nature of human capital) and allows individuals to self-identify for the tasks their human capital is best suited for. Given this process of self-identification, it is apparent that peer production will enjoy an

[22] Foray and Zimmerman (2001), for instance, build a model of coalition in which as the number of agents who derive benefits from using or commercializing the software without contributing to it increases, the incentive system based on user needs, learning and reputation becomes increasingly difficult to sustain.

advantage with respect to the other two production modes also as regards the combination of human capital and other resources (point b). This is because it allows a relative unbounded set of agents to have access to an unbounded set of resources, without incurring the organizational or transaction costs entailed by markets and hierarchies, thus generating increasing returns to scale. The argument holds, however, provided that incentive problems can be solved at least as well in the peer production mode than in the hierarchy-based mode, which Benkler assumes to be generally the case when technology is sufficiently modular to require very small individual contributions. Both of these advantages are crucially important when the physical costs of information production and the costs of communication are low, so that the gains they afford outweigh the costs of coordination of widely distributed agents.

3.2. How are F/OSS projects coordinated?

Empirical analyses aimed at providing a quantitative profile of F/OSS projects tend to be consistent in drawing a picture of the F/OSS community that contrasts with Raymond's image of the bazaar. Departures from the "bazaar" model occur in two directions. On one side, there are studies supporting the notion of F/OSS software as mostly supported by "cave" rather than "community" development (Krishnamurty, 2002). The principal finding of this category of studies is that the median number of developers per project is very low – four in Krishnamurthy's study and as low as one in Healy and Schussman's (2003) study. In addition to this, Krishnamurthy (2002) finds that the level of contact and communication within F/OSS communities tends to be low[23]. On the other side, there are those studies that put into question the idea that community development takes place in a "flat network of interacting peers" each contributing to multiple projects (Healy and Schussman, 2003). Among the main findings of this category are the following:

- Few developers account for most of the contributions (10% of the total number of contributors accounted for 72.3% of the total code base in the Orbiten Free Software 2000 survey)
- The vast majority of contributors is involved in only one (Orbiten Free Software 2000 survey) or very few projects (FLOSS 2002 survey).
- The distribution of activity is highly skewed with respect to all activity measures (number of developers, downloads, site views, number of mailing lists, etc.) and follows power-law-type distributions (Healy and Schussman, 2003; Hunt and Johnson, 2002).

The results mentioned so far have been obtained either on the basis of on-line surveys (Orbiten Free Software 2000 survey; FLOSS 2002 survey) or on the basis of analysis of the Sourgeforce database (Healy and Schussman, 2003; Hunt and Johnson, 2002). Empirical analyses that focus on single, rather large projects, confirm the results

[23] To be fair, however, it should be mentioned that the metaphor of the bazaar was coined with reference to Linux, whereas these studies are based on analysis of the Sourgeforce database, which is the largest repository of F/OSS projects on the internet, but does not include data on the most successful projects such as Linux and Apache – that have their own websites – nor data on GNU projects – hosted on the Savannah website.

on the highly skewed distribution of development effort but emphasize the existence of a core team of developers that controls the development of the majority of the source code. Mockus, Fielding and Herbsleb (2002) study both the Apache web server and the Mozilla web browser, finding evidence of the existence of teams of 10 and 15 people respectively. In addition to this, they also find that the number of people who fix bugs is an order of magnitude larger than the size of the core team, and the number of people who report bugs are an order of magnitude larger than those who fix them. Koch and Schnider (2002) explore the GNOME project, finding results similar to Mockus, Fielding and Herbsleb (2002), although contributions are more dispersed in their case study (52 developers account for 80% of the source code).

To sum up, rather than as a "flat network of interacting peers", the F/OSS community is best characterized as an aggregation of heterogeous contributors performing different roles. In addition to simple users that do not contribute in any direct way to a project's development, at least three roles should be mentioned: 1) project leaders that constitute the "core group" of developers; 2) other developers who submit additions to the source code on a less regular basis; and 3) users of the OSS program that test the program, report bugs or simply submit suggestions.

The existence of such role differentiation raises questions as regards to the extent of the division of labor in the context of F/OSS projects. One of the few contributions directly addressing the issue is von Kogh et al.'s (2003) study of joining behavior and contributions to the Freenet community. The study suggests that joiners tend to observe the development of the project for a while before actually starting to contribute and that newcomers derive benefits from specializing in their contributions. Moreover, only "core developers" who have significantly invested in learning about the complex software architecture are able to perform less-specialized tasks pertaining to the integration of modules. A different approach is taken by Dalle and David (2003), who explore the allocation of programming resources in F/OSS projects through a stochastic simulation model, based on the assumption that developers independently allocate their effort among project tasks and among different projects in order to maximize reputation benefits.

The preceding synthesis of the empirical literature bearing on the characteristics of F/OSS projects was intended to provide two hints as regards to the issue of coordination. The first is that different projects require coordination to a different extent. The second is that, when coordination is a non-trivial problem (e.g. in cases different from the "cave-development" case) its solution should be considered less astonishing than first thoughts would suggest. It is indeed a rather different matter to consider the problem of coordination if the actual developers involved are in the order of magnitude of tens rather than hundreds[24]. Nonetheless, understanding the way coordination is achieved in F/OSS projects is of crucial importance and it is surprising that a relatively little number of theoretical contributions have so far addressed the issue directly.

The most common forms of coordination adopted in the context of F/OSS projects can be referred to as "*benevolent disctatorship*", "*rotating dictatorship*" and "*voting committee*" system (Raymond, 1998b; Ljungberg, 2000). The most prominent example

[24] Consider, however, that in the case of LINUX by July 2000 about 350 contributors were acknowledged in a credits list in the source code of the kernel (Moon and Sproull, 2000).

of the fist category is Linux. Linus Torvalds – the project founder – effectively "owns" the project, meaning that he is entitled to say the last word on the final design of the program. In the specific case of Linux, as the complexity and the number of contributors have grown over time, coordination of the project has become a task too demanding for a single decision-maker, and the model of coordination has evolved to include a number of trusted "lieutenants", each responsible for a single subsystem and its interface with the rest of the project. As the status of project leader, also the status of "co-developer" with direct CVS access[25] is gained on the basis of merit and according to the rule "authority follows responsibility"(Raymond, 1998b).

The adoption of a *rotating dictatorship* is a less widespread practice, but has nonetheless been used, for example in the development of Perl, a programming language used for a variety of programs, including the development of scripts for Apache web servers. Until 1996 a *voting committee* of co-developers was the coordination model adopted by a large project such as the Apache web server, in which decisions were taken on the basis of a system of e-mail voting based on minimal quorum consensus (Fielding, 1999).

Whatever the specific organizational model adopted by different projects, what is important to note is that the anarchistic image of F/OSS development that has sometimes been suggested does not accurately capture the real nature of the phenomenon. Leadership and authority, however defined, are essential to the effective development of F/OSS projects. What remains to be satisfactorily explained is how leadership, authority and therefore coordination are sustained in a context of voluntary collaboration where none of the elements sustaining authority within firms are present. It is to this issue that we now turn.

3.3. How is coordination sustained?

Leadership and authority are key to understanding how successful collaboration among widely distributed developers is achieved and how a major organizational challenge facing F/OSS projects – namely the possibility of "forking" – is overcome. "Forking" entails the release of multiple and concurrent versions of the source code, that would therefore tend to evolve in several and often incompatible directions. Forking has afflicted, for example, very popular programs such as the Berkeley Unix program, Sendmail at the end of the 1980s and more recently the GNU Emacs project. Given the terms of F/OSS licenses, that allow practically anyone to modify the freely available source code and redistribute it, it is rather surprising that the majority of F/OSS projects has not succumbed to forking.

Raymond (1998b) provides an "insider" explanation of why this has been the case. In his essay "Homesteading the Noosphere" he elaborates on the reputation/gift culture argument introduced in "The Cathedral and the Bazaar" explaining the set of taboos, rules and informal institutions that characterize the hacker culture as functional to the "reputation game" that sustains the development of F/OSS software. He analogizes the customs and ownership rules of the hacker community to Lockean-style customs over land tenure. Ownership in the F/OSS domain relates to "*the exclusive right, recognized*

[25] The "Concurrent Versioning System" is a software tool that performs two main functions: 1) store the project's software code together with the documentation and written comments provided by developers; and 2) record changes made to the source code.

by the community at large, to re-distribute modified versions" (Raymond, 1998b). The Lockean-style ownership rules inadvertently but consistently adopted by F/OSS contributors represent means of maximizing reputation incentives. It is in this perspective that the three fundamental taboos of the hacker culture should be understood: (a) forking projects, (b) distributing changes to the projects without the cooperation of the moderators and (c) removing a person's name from a project history, credits or maintainer list. Each of these behaviors reduces the effectiveness of the reputation mechanism by exposing the owner to reputation risks (a and b) and by unfairly denying to a member of the community a deserved reputational reward (c).

In Raymond's story authority is strictly linked to developers' reputation. Note that authority is not to be intended as implying effective direction of the project, or the imposition to the community of a pre-defined "vision" of the development of the program. The notion of authority should be assimilated to some extent to Raymond's conception of ownership and entails mainly a role of coordination and arbitration in case of technical disagreements.

In cases in which the complexity of the project leads to a distribution of design authority and therefore partial property rights (as in the LINUX case), conflicts tend to be solved according to one of two rules: "authority follows responsibility" and "seniority wins". In addition to this, a shared understanding of technical rationality that in general follows the standards set by Unix programming, constitutes an "objective" reference point for solving disputes concerning the opportunity to include alternative patches in the official release of a program (Raymond, 1998b; Weber, 2000).

The enforcement of these rules ultimately depends on the active collaboration of community participants. The two principal sanctioning mechanisms available to the community of F/OSS developers are *"flaming"* and *"shunning"*. The first refers to the violent blaming of individuals for their taboo behavior, whereas the second has deeper consequences for the rule-breaker in that it entails a refusal of cooperation and therefore effective exclusion from the community. Of course the rule-breaker would still have access to the software itself, according to the terms of the F/OSS license, but he would be precluded from benefiting from the collaboration of other developers.

It is important to note that the enforcement of community norms represents a second-order social dilemma, as punishment of rule-breakers is costly to the individuals who inflict the punishment but benefits the community at large, thus displaying the characteristics of a public good (Elster, 1989; Bowles and Gintis, 1998; Osterloch et al., 2002). The literature on collective action (Ostrom, 1990) suggests that the successful solution of this social dilemma is a challenging task in virtual communities, because they lack clearly defined boundaries and a set of well-designed rules. Moreover, whereas monitoring of participants' bahavior is made easier by the digital medium, punishment has to rely on informal sanctions only and therefore risks to be less effective than in territorial communities (Kollock and Smith, 1996). However, in the specific case of F/OSS communities, it is perhaps possible to speculate that Ostrom's conditions are fulfilled to a greater extent than in other virtual communities, partly because the "hacker culture" provides a set of agreed-upon rules and partly because sanctions impose a concrete loss on the rule-breaker when they result in denial of access to the community's pool of knowledge and experience.

Few explanations other than Raymond's "reputation game" argument have been put forward to explain the foundations of authority in the F/OSS community. Mateos-

Garcia and Steinmuller (2003) articulate an argument that is close to Raymond's but places emphasis on the process of challenging the authority rather than stressing cultural norms as Raymond does. As projects grow in complexity – they argue – authority tends to be challenged and distributed in the hands of an "inner circle" of developers, entailing a relative rise in the barriers to entry for outsiders. This sustains the orderly development of the program because prevents anyone who lacks authority from credibly threatening to fork the project, as their chances of attracting a significant number of contributors on the alternate development path are necessarily slight.

Another hypothesis, advanced by McGowan (2002), is that a form of asset specificity may be part of the explanation for those projects that attract a high number of developers or directly challenge a dominant firm. McGowan adopts some of the insights of Hart and Moore's theory of the employment relationship to suggest that *"programmers will be more likely to accept hierarchy in projects that provide unique or unusually valuable returns"*, similarly to employees in a firm, who are more willing to accept fiat power when their productivity depends on access to firm-specific assets.

Interestingly, McGowan only considers the issue of specificity with respect to assets that in Hart and Moore's theory are characterized as physical assets, e.g. the software code, and is therefore led to downsize the relevance of his argument by noting that *"[t]he terms of the open source license effectively preclude project maintainers from denying code to programmers, and programmers therefore could not be induced to accept hierarchy by the threat of being cut off from the relevant code"*. However, McGowan's intuition could be taken a step further by considering the other form of specificity relevant in the GHM[26] theory of the firm, namely the specificity of agents' human capital with respect to the human capital of the other agents. From this perspective, programmers' willingness to accept hierarchy would derive from the threat of being cut off from access to the other programmers' human capital (tacit skills, knowledge and experience) rather than from the threat of being denied access to the source code. The argument would therefore be independent from assumptions concerning the desire to counter Microsoft's commercial dominance and would only require an interest on the part of developers for peers' advice, learning and exchange of ideas, which is consistent with the empirical evidence on motivation reported in section 2.

An alternative hypothesis put forward by McGowan on the basis of insights derived from Herbert Simon's work is that identification with the goals of the community and loyalty towards it lead programmers to work actively to pursue such goals. Given that authority is used *"to coordinate behavior by promulgating standards and rules of the road, thus allowing actors to form more stable expectations about the behavior of the environment (including the behavior of other actors)"*(Simon, 1991), programmers will be willing to accept it as a means to an end. The implications of the sense of belonging and the notion of identity have only recently started attracting the attention of economic scholarship (Akerlof and Kranton, 2000). Indeed, this is an avenue worth pursuing in approaching F/OSS communities.

[26] The expression "GHM theory of the firm" generally refers to the framework set up by the contributions by Grossman and Hart (1986), Hart and Moore (1990) and Hart (1995).

4. F/OSS software in perspective

This section aims at broadening the perspective on F/OSS software by looking at both competitive and cooperative relationships of F/OSS communities with commercial actors. The issue of the competition between open and closed forms of software production has been extensively addressed, especially with reference to the competition between Windows and Linux. The different forms in which interaction between F/OSS communities and commercial actors takes place have also attracted scholarly attention, in light of the increasing involvement of for-profit firms with the F/OSS phenomenon.

4.1. Competition with proprietary software

The competitive relationship between open and closed forms of software provision has been analyzed along two principal dimensions: the relative quality of open vs. closed software products, especially in terms of the ability to meet consumers' needs, and the dynamic of technological competition between open and closed software. Enthusiasm and skepticism seem to be equally abundant among the scholars who have tackled these issues. In what follows we will review the principal theoretical contributions on the two questions in turn.

Kuan (2000) avows the superiority of the quality of F/OSS products, presenting both a theoretical model based on the idea of "consumer integration into production" and a statistical test of the rate of change of software quality based on software bugs data such as the life-expectancy of service request. In the model consumers decide whether to make their own software or to buy it from a commercial vendor. Closed source software provision entails information asymmetries, whereas F/OSS provision is exposed to the possibility of free riding. Thus, the result about the superiority of F/OSS software obtains provided that the free riding problem is overcome as all consumers decide to make their own software.

User needs as drivers of contributions are crucial to the majority of models addressing the issue of F/OSS software quality. Johnson (2000) models F/OSS software development as the private provision of a public good. As in Kuan's model, needs are homogeneous across agents, but valuations and abilities differ. Differently from Kuan, however, Johnson explicitly models the agents' beliefs about the strategies of the other agents. He compares open and closed systems to a constrained social optimum and shows that F/OSS development will in general entail both an inefficient level and an inefficient distribution of development effort, due mainly to the possibility of free riding. At the same time, however, F/OSS development better exploits the talent pool made available by the Internet. A firm providing closed source software, on the other side, although unable to exploit such talent pool and to correctly assess the private valuations of users, will take into account the aggregate enjoyment consumers may derive from the program (see also Xu (2002) for a variant of Johnson's model focusing on the effects of development costs on development effort and Bitzer and Shroder (2002) for a war-of-attrition model of the private provision of a public good).

As Kuan's and Johnson's models, also Bessen's (2002) model takes as a starting point the notion of user-driven innovation. Rather than focusing on the heterogeneity of consumers' evaluations for a given software program, Bessen puts the emphasis on the heterogeneity of user needs and constructs a model in which the choice of the form of

provision of software is endogenous. Given the assumption on the heterogeneity of user needs, free riding is a less pressing concern, provided that the base product is created in the first place. This is mainly because the low probability that a specific feature will be developed by another consumer reduces the incentives to wait. In Bessen's model the F/OSS form of provision is more efficient because it allows the complex and sophisticated needs of some consumers to be met in markets where incomplete contracts and asymmetric information prevent proprietary software from serving all applications.

It is noteworthy that in both Johnson's (2000) and Bessen's (2002) models, the existence of a core group of developers that aggregate a sufficient amount of effort to provide a base code deeply affects the outcome of the model. This is consistent with both casual empiricism and self-reflection by advocates of the F/OSS community. Raymond (1998a), for instance, explicitly includes among the requirements for the "bazaar" mode of software production the availability of an initial base code, specifying that the bazaar mode is not suited to the provision of this initial amount of software code. The same observation is critical to models that capture the competitive dynamics of open vs. closed systems of software development through computer simulations, although in this case the existence of a core group of agents is relevant to the *diffusion* rather than to the *production* process.

In Dalle and Jullien's (1999) model of the diffusion of the Linux operating system, this aspect is captured though the introduction of individual *adoption thresholds*, dependent on idiosyncratic preferences for the dominant standard (in their analysis, Windows NT). As long as the parameter denoting such preferences is sufficiently low, a threshold effect may be set forth and F/OSS programs may overcome the existing proprietary standard. The competitiveness of F/OSS programs depends, however, also on the efficiency of its organizational structure (that benefits from adoption externalities to a greater extent than proprietary software) and on the degree of compatibility with existing proprietary solutions, so that the success of F/OSS should not be taken for granted. In particular, whereas simple reactions by commercial software producers (price reductions, increased R&D investments, changes in software distribution) may not be sufficient to contrast the path-dependent process of technological competition, hybridization strategies implying the adoption of F/OSS features may succeed as they modify the underlying preferences.

Network effects and externalities are also at the heart of Bonaccorsi and Rossi's model of the rivalry between F/OSS and proprietary software. Building on Marwell and Oliver's theory of "critical mass", the two authors build an agent-based simulation model in which the adoption of F/OSS software by a population of heterogeous agents is affected by its perceived intrinsic value, the negative network externality associated to the existence of an incumbent standard, the positive network externality associated to other agents' use and development of F/OSS programs, and the intensity of the competitive reaction by the incumbent. Bonaccorsi and Rossi's argument is similar in vein to Dalle and Jullien's but leads to some extent to different results. Most notably, their simulations suggest that F/OSS and commercial software are likely to coexist, even in the limit. Even in a setting characterized by a biased distribution of beliefs in favour of F/OSS and no incumbent advantage, a strong competitive reaction by incumbent commercial players (for instance in the form of aggressive R&D spending) can put a brake or even block the diffusion of F/OSS software.

A different kind of strategic reaction by the incumbent software vendor (in terms of pricing, rather than R&D spending) and demand-side learning are at the heart of Casadeus-Masanell and Ghemawhat (2003) dynamic model of mixed-duopoly competition[27] between Windows and Linux. The main results of the model are that, depending on the speed of demand-side learning, either Windows and Linux coexist in the long run, or Windows pushes Linux out of the market. This occurs as long as Windows' pricing decision is not myopic. By adopting a strategic pricing decisions, Microsoft can influence consumers' valuations for its product at period t+1. However, the presence of cost asymmetries may overturn this result in favor of Linux.

A starkly different approach in modeling the impact of F/OSS on commercial software markets is taken by Mustonen (2003) that attributes to a monopolist's wage policy the possibility to influence the occupational choices of programmers and the quality of both the proprietary and the F/OSS software program.

4.2. Cooperation with commercial actors

The participation of commercial firms to F/OSS projects brings a number of benefits to the F/OSS community. Most obviously, it implies an enlargement of the user base and consequently an increase in popularity of the project , with the usual benefits in terms of network effects. The contribution of commercial actors is particularly relevant in that it can help to bridge the gap between technical and commercial software users. As mentioned in section 2, F/OSS developers tend to direct their contributions towards the solution of challenging programming tasks and to neglect those aspects of software development that, though less stimulating, are useful to less technically skilled users, such as for instance user-friendly interfaces. Collaboration with firms can thus provide financial incentives to target some effort in this direction (O'Mahoney, 2002).

What are the motivations for firms to be active in the F/OSS world? The FLOSS survey (Wichmann, 2002a and 2002b) reports four motivations relevant to large firms: 1) standardization; 2) use of F/OSS software as a low-cost component; 3) strategic considerations; and 4) enabling compatibility. The first motivation relates to the fact that F/OSS platform offers a specific advantage as a common standard because the license under which it is distributed effectively prevents any of the commercial developers adopting the standard from appropriating the software at a later point of time. The second motivation is easy to understand: the fact that F/OSS software is free makes it an ideal candidate for being offered to customers as part of a hardware-software bundle to meet specific needs. Contributions to F/OSS software development may also be motivated by competitive goals, such as the desire to pose a threat to dominant firms in the market (3). Finally, firms might simply be interested in making their software or hardware compatible with F/OSS software and develop plug-ins necessary to do so (4).

When attention is shifted to SMEs, the nature of motivations obviously changes considerably and becomes to many respect closer to the nature of individual extrinsic and social motivations (Bonaccorsi and Rossi, 2003b). User needs are among the main

[27] The expression "mixed-duopoly competition" refers in the paper to competition between a for-profit and a not-for-profit firm.

drivers of contribution. The decision to contribute to the OS community software developed for the firm's own need should not be too surprising, in light of the fact that a firm might face high transaction costs in trying to sell software developed for internal use if it is not related to his main area of business. Moreover, if the software is not key to the firm's competitive advantage and making it freely available does not risk disclosing firm-specific information releasing the source code does not pose significant threats to the firm (Henkel, 2002).

It should be emphasized that not all the firms, large and small, contribute back to the community. In the case of the Italian firms surveyed by Bonaccorsi and Rossi (2003b), a comparison between individual and firms' contribution shows evidence of a significant free-riding behavior on the part of firms[28]. Indeed, firms join less projects than individual programmers do and they do not devote as much programming effort as individuals do to the projects they join. However, they also find that firms that participate in F/OSS development out of social motivations not only tend to display higher project membership and higher intensity of contribution in terms of lines of code, but also tend to have their contributions accepted into the official release of the project more frequently than firms motivated only be extrinsic motivations.

The extent to which firms actively contribute to F/OSS development depends on the particular business model adopted. A comprehensive overview and evaluation of the business models that firms have elaborated so far is outside the scope of this paper[29]. However, following Lerner and Tirole (2002) a rough distinction could be drawn between "reactive" and "proactive" strategies. The first kind of strategies entails a symbiotic relationship with an F/OSS project, with firms commercially providing complementary products and services that the F/OSS community fails to provide efficiently. This is the case of Red Hat, VA Linux or Caldera, who package the software in a user-friendly way and provide additional material such as manuals and support services. A more "proactive" strategy consists in a decision to release proprietary software code, generally with the aim of profiting from a complementary segment. The groundbreaking initiative in this regard can be safely identified in the decision by Netscape to release the source code for his web navigator (Mozilla) in 1998. Somewhere in between are hybrid business models – such as the one adopted by Sendmail Inc. – that lead firms to develop proprietary products complementary to F/OSS software products and sell them without contributing them back to the community (Ljungberg, 2002).

One issue that has been largely overlooked in the literature concerns the long-term impact of the collaboration with commercial actors on developers' incentives. More research could help to shed light on the risks that an increase in the number of agents that directly profit from F/OSS software might entail for the long-term survival of F/OSS communities norms. In addition to this, while scholars have so far focused either on competition or on cooperation between commercial actors and F/OSS communities, interesting insights could be gained by integrating the two aspects. Economic modeling could try to capture the strategic interests of commercial companies in supporting F/OSS software and how the symbiotic relationship between firms and F/OSS

[29] For interesting insights on the variety of business models related to F/OSS software, see for example Behlendorf (1999), Young (1999), Hecker (2000), Babcock (2001), Hawkins (2002), Nilendu and Madanmohan (2002).

communities affects the competitive relationship between F/OSS and closed source products.

5. F/OSS and Intellectual Property-related Issues

Throughout the previous sections, a crucial aspect of the F/OSS phenomenon has been overlooked and hinted at from time to time with the purpose of undertaking a thorough discussion of it at a later stage. It is now time to explore in greater detail the basic institutional innovation at the heart of F/OSS projects (Frank and Jungwirth, 2002), namely the adoption of a novel form of licensing.

5.1. Copyright, Copyleft and Open Source licenses

The importance of F/OSS licenses cannot be overstated. F/OSS licenses play a crucial role with respect to all of the issues that have been addressed so far: the motivation of developers, the coordination of projects, the effectiveness of the F/OSS development model and the relationship with commercial firms. A quick look at the four most salient criteria contained in the OSD[30] immediately suggests why this is the case[31]. The principle of Free Redistribution *("[t]he license may not restrict any party from selling or giving away the software […]")* ensures the viability of F/OSS business models. A second principle requires the availability of the source code, crucial to the functioning of the distributed development model together with the requirement that the license *"allows modifications and derived works, and […] allows them to be distributed under the same terms as the license of the original software."* Finally, among the most relevant criteria for compliance with the OSD is the requirement concerning the integrity of the source code, essential to the process of reputation building emphasized in the previous sections.

There is often some confusion concerning the question whether F/OSS software belongs to the public domain. Academic commentary has extensively addressed the issue (Gomulkiewitz, 1998; Lee, 1999; Perens, 1999; Lee, 2003; McGowan, 2001; Kennedy, 2001). F/OSS software is to be distinguished from software that is simply put into the public domain because, although it is freely available to all, developers do not surrender their rights to their software creations. Rather, they retain copyright over their work and adopt licenses to ensure free access and modification of the source code. As McGowan puts it, *"[t]he licenses, and the GPL in particular, represent an elegant use of contractual terms and property rights to create social conditions in which software is produced on a model of openness rather than exclusion".* In other words, *"rather than diminishing the commons, the copyright in open source software protects the commons."* (McJohn 2000). Were F/OSS software to be in the public domain, everyone

[30] The Open Source Definition (OSD) defines the *"rights that a software license must grant you to be certified as Open Source"* (Perens, 1999). We refer here to the OSD version 1.0. The latest version of the license is available at http://www.opensource.org/osd.html.

[31] The other six principles embodied in the OSD are as follows: 5) No Discrimination Against Persons or Groups; 6) No Discrimination Against Fields of Endeavor; 7) Distribution of License: "[t]he rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties."; 8) License Must Not Be Specific to a Product; 9) License Must Not Restrict Other Software; 10. License Must Be Technology-Neutral.

could appropriate it, modify, even obtain copyright over it and remove the author's name (Kennedy, 2001) and the structure of incentives that currently sustains F/OSS development would be jeopardized.

The Open Source Definition is not itself a license in legal terms. It is *"a specification of what is permissible in a software license for that software to be referred to as Open Source"* (Perens, 1999). The first "open source license" – using the term a bit loosely – predates the OSD. Among the first F/OSS licenses is the GPL (General Public License), created by Richard Stallman, founder of the Free Software Foundation, and also called copyleft, to hint at the kind of rights creators grant to the public through the license. In the words of its creator: *"Copyleft uses copyright law, but flips it over to serve the opposite of its usual purpose: instead of a means of privatising software, it becomes a means of keeping software free"* (Stallman, 1999). The OSD was released only in 1998, partly with the purpose of clarifying the ambiguity associated to the term "free software" – hard to digest for potential commercial developers – and thus enhance the diffusion of the software beyond the borders of the original community of developers.

The distinctive characteristic of copyleft licenses is that they are "viral", as they impose that any derivative work that is distributed or published must be licensed as a whole under the terms of the same license (section 2(b) of the GPL license). The term "viral" is probably unfortunate and not entirely appropriate, as it has generated an *"unreasonable fear of infection"* (Rosen, 2001a) among commercial firms. It is not the mere use of GPL-ed software that generates an obligation to release one's own software code under the GPL. The requirement of reciprocity only applies to works based on the program[32].

Not all OSD-compliant licenses have a persistent nature. In fact, a taxonomy of OSD-compliant licenses can be build with reference to the way different licenses address the issue of derivative works (Kennedy, 2001), distinguishing among a) the GPL and LGPL (Library General Public License); b) the Berkeley Software Distribution (BSD), the MIT and Apache licenses; c) commercial licenses such as Mozilla Public License; and d) other open source licenses (see for instance Rosenberg, 1998). The more permissive licenses in the taxonomy are those belonging to the b category: they do not impose any burden of reciprocity and allow the greatest freedom of distribution, modification and license change. The LGPL lies somewhere in between the GPL and category b, as it allows for license change if the software is bundled into new work. Commercial F/OSS licenses can be more or less restrictive as regards the issue of derivative works, but tend to impose some specific requirement different from the other licenses, such as for instance the imposition that ownership of any contribution should be attributed to the holder of the original license. As reported by Lerner and Tirole (2002b), the GPL license is, at present, the most widely used among the licenses complying with the OSD.

Many commentators have analysed the choice of the appropriate kind of license for a business firm (Perens, 1999; Horne, 2001; Rosen, 2001b; Nadan, 2002; Oksanen and Valimaki, 2002; Stinebruner, Humphrey and Davis, 2002). Hawkins (2002) explores the

[32] Section 2(b) of the GPL defines a *"work based on the program"* as *"either the program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language."*

choice of a commercial firm between copyleft and non-copyleft licenses in a very simple model showing, among other things, that the a GPL license is never profitable if the possibility of competition with other firms is ignored, because GPL licenses can only be used by a subset of the public that can use the non-viralGPL ones. Persistent licenses such as the GPL can be an optimal choice, however, when the presence of a market competitor that can adopt the product is considered. In the latter case, the persistent license forces the competitor to release changes.

Lerner and Tirole (2002b) and Bonaccorsi and Rossi (2003c) address the issue from an empirical perspective. The first two authors find that restrictive licenses are more likely to be adopted when the software is directed at end-users, whereas less restrictive licenses are more frequently adopted for projects geared to developers, the Internet, or proprietary operating systems. Bonaccorsi and Rossi (2003c) show that adoption of copyleft licenses is positively related to the strength of social and ideological motivations on the part of firms and to their degree of involvement into the F/OSS community. They also stress the fact that firms commonly recur to mixed forms of licensing. The possibility of dual licensing, i.e. releasing the software both under an F/OSS and a proprietary software license, has been also theoretically addressed (Rosen, 2001b; Valimaki, 2003).

F/OSS licenses raise a number of interesting legal issues. First, there is the question of *enforceability*. Of the very few legal suits involving F/OSS projects of which we are aware[33], none has been decided yet. However, legal scholars have usually argued in favour of the enforceability of F/OSS licenses under copyright law (Lee, 1999; Bobko, 2000; Ravicher, 2000; Kennedy, 2001; McGowan, 2001; Jarvinen, 2002).

A related question concerns the attribution of ownership rights and thus of the right to enforce the copyright and licensing terms (Kennedy, 2001). F/OSS software is the result of a process of distributed innovation, which opens up the possibility of distributed ownership. It is possible to envisage three scenarios (Oksanen and Valimaki, 2002): a) *bundled work and authorship*: when the development process is uncoordinated and each developer retains rights to his own contribution; b) *joint authorship*: when development is centrally coordinated and each author has a copyright over the entire program; c) *new authorship*: when software is completely rewritten and the copyright-holder is the author of the re-written software. To these possibilities a further complication should be added, namely the possibility that the developer's employer obtains right to the software.

One way out of this potentially confusing fragmentation of rights has been pursued by the Free Software Foundation and by other non-profit foundations operating in the F/OSS domain: encouraging developers to transfer copyright to the foundation so as to facilitate enforcement. This practice, together with the acquisition of trademarks and the active protection of projects' brands, represents one of the strategies Siobhan

[33] One case, initiated in 2002, involves MySQL AB and Progress Software Corporation. The main issue at stake concerns the question whether the latter is allowed to combine his own proprietary software with MySQL AB's GPL-ed software and sell it under his own commercial license (Oksanen and Valimaki, 2002). The second case has attracted attention of a wider audience, as it involves suits and countersuits between SCO (formerly Caldera), on one side, and the biggest commercial Linux contributors and distributors, i.e. IBM SuSe and Red Hat. After acquiring the UNIX original AT&T's source code from Santa Cruz Operations, and turning its name into SCO, Caldera has sued IBM for damages of $1 billion (later increased to $3 billions) for infringement of several lines of his UNIX code. IBM SuSe and Red Hat reacted by suing SCO for infringement of the Linux open source license.

O'Mahoney (2003) has found to be adopted by F/OSS developers in order to *"guard the commons"*.

Another legally controversial issue is whether the provision disclaiming warranties and limiting liability often contained in F/OSS license – entirely reasonable from the perspective of developers contributing to a public good – contrasts with national consumer protection laws (Jarvinen, 2002). The issue is particularly relevant in light of the expanding scope of participation of commercial firms to F/OSS projects.

5.2. F/OSS and the rationale for intellectual property protection of software

The F/OSS phenomenon has been regarded by many authors as suggestive of the need for a radical rethinking of the current state of intellectual property protection for software programs (Moglen, 1999; Kogut and Methiu, 2001; Benkler, 2002; Bessen, 2002; Osterloch et al., 2002; Karin, 2002). The argument advanced in this regard is generally twofold. On one side, it is emphasized that the success of the voluntary and distributed model of innovation that goes under the F/OSS label puts into question the traditional "market failure approach" to innovation, according to which exclusive rights to newly created knowledge are necessary to ensure appropriability of the benefits from innovative efforts. On the other side, attention has been cast on the possibility that the current system of intellectual property protection may pose a threat to the survival and further development of F/OSS software. This is the case in those countries, most notably the United States, where intellectual property protection has been stretched to encompass the use of software patents in addition to copyright and trade secrecy.

The issue of the appropriate form of legal protection of software programs is, to some extent, an unresolved issue. As argued by Zimmerman (1999) and Jullien and Zimmermann (…) when copyright protection is applied to software, it is unable to provide an adequate balance for the two traditional objectives of intellectual property rights, namely providing incentives to innovation by restricting access to intellectual creations while at the same time encouraging the innovator to disclose his innovation so as to ensure diffusion. The inadequacy of copyright – that protects the expression of an idea, rather than the idea itself – is mainly due to the dual nature of software, that is information and technology at once, and to the fact that when the copyrighted software code is diffused in the form of object-code it is in fact not accessible. Indeed, the possibility to distribute software programs in binary form only implies that the most widely adopted form of protection for software programs is secrecy[34].

However, recent court rulings in the United States have opened up the possibility of obtaining patent protection for software. As a result, hardware and software companies have engaged in a *"dramatic patent 'land grab'"* (Bessen, 2002) that has made the number of software patents granted soar in recent years. The notion that patents are an

[34] See for instance the essay by Lee T.Gesmer on "Trade secret protection of computer programs", available at http://www.lgu.com/publications/tradesecrets/5.shtml, or Dennis S.Deutch "Trade secret protection for Software" on Computer Forensic Online, available at http://www.shk-dplc.com/cfo/issue%201/secret.html.

imperfect mechanism to encourage innovation is well-acknowledged in the literature[35]. What makes their use even more problematic in the case of software is the nature of the innovation process that takes place in the software industry. Software products are *complex* products, and are the outcome of an innovation process that is both *sequential* and *complementary* (Bessen and Maskin, 2000). The sequential (or cumulative) nature of innovation bears upon the fact that each invention constitutes an input for subsequent innovative efforts. The complementarity of software innovations refers to at least two aspects: on one side, complementary pieces of software code are often combined to produce a complex and coherent product; on the other side, research efforts may be complementary in the sense that the activity of each developer, be it truly innovative or even imitative to some extent, exerts a positive externality on the other developers, increasing the overall probability of innovation.

To be sure, software is not the only innovative domain with such characteristics. Kogut and Methiu (2001) mention for instance Heller and Eisenberg's (1998) "tragedy of the anticommons" – the problem of excessive fragmentation of exclusive rights leading to under-utilization of knowledge – that has been first identified in the biomedical domain. What is peculiar about software – it is argued – is that it is a domain in which a spontaneous solution to the problem of inciting innovation has emerged in the form of the F/OSS mode of development.

As mentioned, other authors have emphasized the threat that strengthened patent rights for software innovations may pose to the continued existence of the F/OSS production mode. Benkler (2002) urges to consider the loss of information generated through peer production as an additional cost that excessively strong intellectual property rights would impose on society. Patents increase the cost of using existing information, and thus negatively affect the working of production systems – such as F/OSS development – based on the free matching between information inputs and human capital.

Threats to F/OSS projects may take a very concrete form. In particular, they may take the form of patent litigation, an increasingly important strategic tool in the hands of commercial companies[36]. Bessen (2002) highlights some features of software patents and of the US patent system in general that may make matters worse for F/OSS developers. First, US courts have shown a tendency to lower the standard for non-obviousness and to reinforce the presumption of validity. Second, software patents tend to be of a very poor quality, given the difficulty of assessing the existence of relevant prior art (see also Kahin, 2002). Third, software products are complex artifacts often made out of a large number of patentable components. These characteristics generate a situation that is prone to the emergence of "patent thickets": large companies find it profitable to build large portfolios of patents that can be used both as an offensive and a defensive weapon. It is clear that unpaid F/OSS developers cannot afford to resist to patent infringement suits, let alone to directly engage in such strategic games. Although Bessen notes that the patent threat to F/OSS communities has not materialized so far,

[35] Mazzoleni and Nelson (1998) provide a concise argument in favour of a more cautious attitude towards strenghtening the intellectual property system. See also Gallini (2002) for an overview of theoretical and empirical contributions on the effects of patent law.

[36] See Lanjouw and Shankerman (2001) for an empirical analysis of the growing importance of IP litigation, and Meurer (2003) for a legal perspective on opportunistic and anticompetitive IP litigation.

the threat patents pose to software innovation more generally seem to be tangible enough to question the wisdom of keeping software patents in place[37].

The F/OSS phenomenon thus suggests the need for additional research on the effects of the patent regime on innovative domains characterized by a strong degree of complementarity and cumulativeness. Such research is of special relevance in light of the debate surrounding the European Software Patents Directive.

5.3. F/OSS and interface standards

F/OSS software also opens up issues at the intersection between intellectual property and competition law. In particular, attention has been devoted to the opportunity that F/OSS software offers to overcome the problems associated to software interoperability in a novel way (Zimmerman, 1999; Jullien and Zimmerman, …). Two notable characteristics of software products are the (positive) network externalities that they engender and, relatedly, the role of interface standards. Network externalities may be both direct – when consumers' utility is positively related to the overall number of consumers using a certain good – and indirect – when a large consumer base favors an increase in supply of complementary products. In the second case, the ability of consumers to benefit from greater availability of complementary products crucially depends on the compatibility among different applications and therefore on the specification of interfaces.

Interoperability is ensured by the existence of standards. Zimmerman (1999) distinguishes among three kinds of solutions to the problem of interoperability: the existence of a *dominant position* that effectively imposes a standard based on proprietary interfaces; a *standard de facto*, emerging either in a decentralized fashion or as a result of the concerted effort of firms through a consortium or committee; a *standard de jure*, introduced through a concerted effort coordinated by a national or supra-national authority. According to Zimmerman (1999) and Jullien and Zimmerman (…) F/OSS development gives rise to a completely new approach to the problem of interoperability, as it generates a dynamic process of standards creation and evolution that has an entirely public character given the free availability of the source code. In particular, F/OSS software represents a valid alternative to the imposition of standards *de jure* – the so-called "*normalisation*".

6. Government policies toward F/OSS software development

Rather surprisingly, F/OSS software has swiftly found a place in the agenda of policymakers. Governments around the globe have considered the issue of providing direct or indirect support to F/OSS activities, turning F/OSS in a rather political issue. Countries as far apart as Germany, Brazil, Italy and Singapore, among others, have all endorsed F/OSS software to some extent, according a preference for adoption of F/OSS in governmental offices, offering temporary tax reductions and financial grants to fund Linux-related projects or though some other means (Hahn, 2002). In the face of such widespread positive acceptance of the F/OSS phenomenon by governments, a number

[37] Bessen and Maskin (2000) found not only that the effect of software patents has been neutral at best, significantly negative at worst, but also that large patent portfolio holders have actually decreased their R&D spending relative to sales.

of authors has considered the economic rationale for direct or indirect public support to F/OSS projects.

One line of argument proceeds by articulating the question "is the software market broken? If so, should we fix it by promoting F/OSS software?". In other words, the rationale for public support of F/OSS software is pinned down to the existence of significant market failures in the software market (Evans and Reddy, 2002; Schmidt and Schnitzer, 2002; Comino and Manenti, 2003). A second line of argument bases the evaluation of the merits of public intervention on the comparison between F/OSS and proprietary software (Bessen, 2002; Evans and Reddy, 2002; Lessig, 2002; Schmidt and Schnitzer, 2002; Smith, 2002). The two perspectives are complementary, and often combined in a unitary approach, as for instance in Schmidt and Schnitzer (2002).

The two authors list a number of potential sources of market failure and compare the relative advantages of F/OSS and proprietary software in mitigating these problems. Software markets possess three distinctive features: a) large economies of scale; b) crucial role of innovation; and c) network effects.

F/OSS software enjoys an advantage relative to proprietary software only with respect to the first aspect. F/OSS software is priced at marginal cost, so that it maximizes static efficiency by ensuring the widest possible access to the software code once it is produced. On the contrary, the need for recouping fixed production costs induces commercial firms to charge prices above marginal costs and thus reduces access. The authors are less sanguine about the innovative potential of F/OSS projects. They emphasize, on one side, that the profit motive is likely to constitute an incentive to innovate stronger than whatever mix of incentives sustains F/OSS projects and, on the other side, that commercial companies have a greater incentive to tailor their products to the needs of the average costumer (this is a rather uncontroversial observation, on which all the other authors agree upon). Even less sanguine on this point are Evans and Reddy (2002), who insist on the fact that F/OSS projects have a predominantly imitative nature. As for the third feature, Schmidt and Schnitzer caution against the possibility that, in presence of strong network effects, government direct or indirect support for F/OSS software may tip the market in the direction of F/OSS software by influencing consumers' expectations. Their Hotelling-like horizontally differentiated model of competition between an open-source product and a for-profit closed product shows that even when network effects are less pronounced, support to F/OSS software may have negative consequences: reduce competition, increase prices and lower both innovation and social welfare.

Both Schmidt and Schnitzer (2002) and Evans and Reddy (2002) are skeptical about the opportunity of motivating public support of F/OSS on grounds of the existence of some sort of market failure in the software market. Comino and Manenti (2003), on the contrary, highlight the presence of a potential instance of market failure linked to the presence of a large fraction of users uninformed about the availability of F/OSS products. They conclude that government action devoted to removing this informational asymmetry enhances welfare, whereas direct subsidies to consumers of F/OSS products are always welfare-decreasing. A different instance of market failure is identified by Bessen (2002), who attributes to F/OSS software the merit of extending the market for software programs, limited by the combination of asymmetric information and contractual incompleteness (see section 4.2). However, the only form of government

intervention that Bessen recommends is the removal of a market failure the government itself has created by strengthening patent protection in the software domain.

When the rationale for public support to F/OSS software is assessed with reference to the comparison between F/OSS and proprietary software, a commonly made assumption is that the government should act as any commercial actor and choose on the basis of objective quality, security and efficiency criteria (Bessen, 2002; Evans and Reddy, 2002; Hahn, 2002; Lessig, 2002). As in other contexts – it is argued – it is the market that should tilt the competition in favor of one product or the other. The differences among the various arguments largely rest upon the question whether governments should also take into account additional considerations when making their decision. Schmidt and Schnitzer (2002) and Evans and Reddy (2002) strongly favor the firm-like approach. Lessig (2002), on the contrary, while supporting the idea that governments should take a neutral stance towards open and proprietary software, argues that *"between two systems for producing a public good, one that releases the information produced by that good freely and one that does not, all things being equal, public policy should favor free access"*. Moreover, he claims that the government has an interest in maintaining an open platform that should be factored in when considering the adoption of a platform for governmental agencies.

Public support of F/OSS software may take many forms. The most commonly chosen option so far has been the mandatory adoption of F/OSS software in government offices. Direct subsidization has not fared as well as the first option in the real world, but is a theoretical possibility explored by Schmidt and Schnitzer (2002) with reference to the possibility of direct subsidization of developers and of institutions that coordinate F/OSS development, and by Comino and Manenti (2003) in relation to the subsidization of users. Finally, it has also been suggested that an effective form of government support of F/OSS software would be either a radical change in patent policy, namely the outright elimination of software patents (Bessen, 2002; Lessig, 2002), or a more moderate reform in the direction of stronger standards and shorter lives for patents (Evans, 2002). In sum, is seems reasonable to extract from the literature a case against direct government support of F/OSS software.

A distinct issue is whether the government should allow and/or require publicly-funded software to be licensed under the terms of the GPL license. Schmidt and Schnitzer (2002), Smith (2002) and Evans and Reddy (2002) make a case against public funding of GPL-ed software on the basis of the fact that the "viral" nature of the GPL license would prevent some commercial companies from incorporating the GPL-ed software in their own innovations. Lessig (2002) concedes that this argument may have some merit on fairness grounds, but questions its relevance in terms of efficiency by noting that *"it is not clear why the fact that [GPL licenses] would exclude some private companies from developing the software should, on its own, matter"*. A stronger case in favor of adopting GPL licenses for publicly-funded software innovations is made by Aigrain (2002), who argues that GPL licenses best serve the aim of protecting the digital commons, especially by safeguarding from private appropriation those software tools that *"play a critical role as part of the common infrastructure of the information society"*. Further investigation on the implications of the adoption of GPL vs. non-GPL licenses in terms of social welfare is surely needed.

7. Conclusion

F/OSS software is a complex and heterogeneous phenomenon. Several theoretical and empirical studies have uncovered a number of aspects of this heterogeneity, showing that the nature of individual motivations as well as the extent of ideological commitment to the idea that software should be free differ across participants. Also, different projects are characterized by differing degrees of complexity and modularity, which in turn affects both the overall costs of coordination and the extent of hierarchical organization.

However, the rapid proliferation of research activity on the subject has by no means exhausted the range of interesting issues opened up by this multifaceted phenomenon. There are good reasons to think that the best lesson to be learned from approaching the F/OSS world does not concern the question whether it constitutes a puzzle or not in light of economic theory, but resides in the opportunity it offers to question parochial methodological approaches. The real puzzle to be solved resides in explaining the multiple dimensions of the complementarity between not only individual motivations, but also between individual motivations and institutional factors such as the structure of social interaction and the nature of social norms. Indeed, it is to these aspects that the complexity and the ability of F/OSS projects to evolve over time should be attributed.

The paper suggests some directions for further research. First, whereas the nature of the different types of individual incentives has been explored in sufficient detail, more could be done to provide an explanation of how the heterogeneous motivations guiding behavior of the different actors involved in F/OSS development integrate coherently. The issue is relevant in light of the fact that different motivations are likely to affect the choice of projects and the outcome of the development effort and thus the evolution of the F/OSS phenomenon.

Second, few attempts have been made at explaining how authority and hierarchy are sustained in a context of voluntary collaboration where none of the elements sustaining authority within firms are present. An exploration of this issue could also lead us to conclude, as some have argued (Naidu, 2004), that what we observe in F/OSS project is actually not akin to hierarchy at all. Our understanding of the governance of F/OSS projects would also benefit from further exploration of the role of identity and the sense of belonging to the community of developers that, although empirically relevant, tend to be neglected in theoretical analyses of the F/OSS phenomenon. Moreover, the dynamic of enforcement of community norms and of punishment of rule-breakers also deserves further consideration.

Third, it would be worth speculating on the long-term impact of the collaboration with commercial actors on developers' incentives and the possibility that monetary compensation will crowd out individual incentives. More research could help to shed light on the risks that an increase in the number of agents that directly profit from F/OSS software might entail for the long-term survival of F/OSS communities norms. Also, economic modeling could try to capture the strategic interests of commercial companies in supporting F/OSS software and how the symbiotic relationship between firms and F/OSS communities affects the competitive relationship between F/OSS and closed source products.

Fourth, the issue of the choice between copyleft vs. non-copyleft licenses has been mainly analysed from the firm's point of view. A question with strong implications for the public interest concerns the effects of widespread adoption of copyleft licenses in terms of social welfare. In particular, the issue of the kind of licence that should be adopted for publicly-funded software deserves further consideration. One could speculate for instance that encouraging the diffusion of copyleft licenses could be an alternative open to EU regulators faced with the question whether to extend patentability to software programs. The US decision in this direction may put European firms at a disadvantage, as they are exposed to the possibility of unintentional infringement of US patents. Instead of reacting by homogenising patent policies, an alternative option could be to encourage the adoption of copyleft licenses in order to create an area of free exchange of source code and increase the stock of "prior art" so as to hold back the relentless march towards knowledge privatisation.

Finally, a better overall understanding of the F/OSS development model could shed light on the possibility to extend the model to other domains, so as to mitigate the effects of reliance on second-best incentive mechanisms such as the intellectual property system. Some proposals in this direction have already been made with regard to the application of F/OSS principles to the biotech domain (Burk, 2002; Maurer, 2003).

References

Agrain, Philippe, 2002. "A framework for understanding the impact of GPL copylefting vs. noncopylefting licenses", working paper available at: http://opensource.mit.edu/papers /aigrain2.pdf

Akerlof, George A. and Rachel E. Kranton, 2000. "Economics and Identity", *Quarterly Journal of Economics* 653, 715-753.

Allen, Robert C. 1983. "Collective Invention." *Journal of Economic Behavior and Organization* 41, 1-24.

Babcock C., 2001. "F/OSS code: a corporate building block", *Interactive Week*, http://zdnet.com/intweek/stories/news/0,4164,2717905,00.html.

Baldwin Carliss Y. and Kim B. Clark, 2003. "Does Code Architecture Mitigate Free-Riding in the F/OSS Development Model?", working paper, Harvard Business School.

Behlendorf, Brian, 1999. "F/OSS as a Business Strategy", in In DiBona, C., Ockman, S., Stone, M. Eds. *F/OSSs: Voices from the F/OSS Revolution*. O'Reilly & Associates, Sebastopol, CA.

Benkler, Yochai, 2003, "Coase's Penguin, or Linux and the Nature of the Firm", *Yale Law Journal*, 112, Winter 2002–2003.

Bergquist, M. and J. Ljungberg, 2001 "The power of gifts: Organizing social relationships in F/OSS communities", *Information Systems Journal* 11**,** 305-320.

Bessen, J. and E. Maskin, 2000. "Sequential innovation, patents and imitation", MIT Economic Department Working Paper, Cambridge, MA.

Bessen, James, 2001 “F/OSS Software: Free Provision of Complex Public Goods.” Research on Innovation paper, available at: www.researchoninnovation.org/opensrc.pdf.

Bessen, James, 2002. “What Good is Free Software?”. In In Hahn ed., *Government Policy toward F/OSS Software.* AEIBrooking Joint Center for Regulatory Studies, Washington D.C.

Bezroukov, Nicolaj, 1999. "F/OSS software development as a special type of academic research critique of vulgar raymondism," *First Monday*, 4.

Bitter, Jurgen and Philipp J.H. Schroder, 2002. “Bug-Fixing and Code-Writing: The Private provision of F/OSS Software”, DIW Berlin Discussion Paper 296.

Bobko, P. K., 2001. “F/OSS Software and the demise of copyright”, *Rutgers Computer and Technology Law Journal,* 27, 51-92.

Bonaccorsi, Andrea, and Cristina Rossi, 2003a. “Why F/OSS can succeed”, *Research Policy*, 327, 1243-1258.

Bonaccorsi Andrea and Cristina Rossi, 2003b. “Contributing to common pool resources. A Comparison Between Individuals and Firms”, working paper, ‚Sant’Anna School of Advanced Studies, available at: http://opensource.mit.edu/papers/bnaccorsirossidevelopers.pdf.

Bonaccorsi, Andrea and Cristina Rossi, 2003c, “Licensing schemes in the production and distribution of F/OSS software. An empirical investigation”, working paper, Sant’Anna School of Advanced Studies, available at: http://opensource.mit.edu/papers/ bnaccorsirossilicense.pdf.

Bowles, Samuel and Herbert Gintis, 1998: ”The Evolution of Strong Reciprocity”, *mimeo*, University of Massachusetts at Amherst.

Burk, D.L., 2002. “Open Source Genomics,” *Boston Univ. Journal of Science and Technology Law* 8, 254.

Cohendet, P., Creplet, F. e O. Dupouët, 2001 “Organisational innovation, communities of practice and epistemic communities: the case of Linux”, in A. Kirman e J. B. Zimmermann, eds, *Economics with heterogeneous interacting agents*. Springer, Berlin.

Dalle, Jean-Michel and Nicolas Jullien, 2000. “NT vs. Linux, or some explorations into the economics of free software,” In G. Ballot and G. Weisbuch, eds, *Application of simulation to social sciences*, Paris, France: Hermès, 399-416.

Dalle, Jean-Michel and Nicolas Jullien, 2003. “ ‘Libre’ software : turning fads into institutions?”, *Research Policy*, vol. 321, 1-11.

Dang Nguyen G. and T. Pénard, 1999 "Don et coopération dans Internet : une nouvelle organization économique ?", Terminal 80-81.

David, Paul A., Seema Arora and W. Edward Steinmueller, 2001. “Economic Organization and Viability of F/OSS Software: A Proposal to The National Science Foundation,” SIEPR, Stanford University, 22 January.

Deci, E. L. 1980. *The Psychology of Self-Determination*. Lexington, MA: Lexington Books.

Deci, E. L. and R.M. Ryan, 1985. *Intrinsic Motivation and Self-Determination in Human Behavior.* New York: Plenum Press.

Deci, E. L., R. Koestner and R. M. Ryan, 1999. “Meta-Analytic Review of Experiments: “Examining the Effects of Extrinsic Rewards on Intrinsic Motivation”, *Psychology Bullettin* 125(3), 627-668.

Dempsey B. J., Greenberg J., Jones P. and D. Weiss 1999. “A quantitative profile of a community of F/OSS Linux developers”, SILS Technical Report TR- 1999-05.
Elster, J. 1989. *The cement of society: A study of social order.* Cambridge University Press, New York, NY.
Evans, D. S., 2002. “Politics and Programming: Government Preferences for Promoting F/OSS Software”, in Hahn ed., *Government Policy toward F/OSS Software*. AEIBrooking Joint Center for Regulatory Studies, Washington D.C.
Evans, D. S. and B. Reddy, 2002 “Government Preferences for Promoting F/OSS Software: a Solution in Search of a Problem”, N.E.R.A. working paper.
Feller J. and B.Fitzgerald, 2002. *Understanding F/OSS Software Development.* Addison Wesley, Boston, MA, USA.
Fielding RT, 1999. “Shared leadership in the Apache project”, *Communications of the ACM* 424, 38–39.
Foray D. et Zimmermann J.B. 2001, "L'économie du logiciel libre: organisation coopérative et incitation à l'innovation", *Revue Economique*, 51, 77-93.
Franck, E., Jungwirth, C., 2001, “Reconciling investors and donators—the governance structure of F/OSS”, University of Zurich Working Paper.
Franke, Nikolaus and Eric von Hippel, 2003 “Satisfying heterogeneous user needs via innovation toolkits: the case of Apache security software,” *Research Policy* 327, 1199-1215.
Hirsch, Fred 1976. *The Social Limits to Growth.* Routledge & Kegan Paul, London.
Gallini, Nancy T., 2002. “The Economics of Patents: Lessons from Recent U.S. Patent Reform”, *Journal of Economic Perspectives*, 162, 131-154.
Ghosh, Rishab Aiyer, 1998. “Cooking pot markets: an economic model for the trade in free goods and services on the Internet,” *First Monday* 3, available at : www.firstmonday.org/issues/ issue3_3/ghosh/index.html.
Ghosh, Rishab Aiyer, Rudiger Glott, Bernhard Kreiger and Gregario Robles, 2002 “The Free/Libre and F/OSS Software Developers Survey and Study—FLOSS Final Report”, available at: http://www.infonomics.nl/FLOSS/report.
Gomulkiewicz R. W. 1998, “The license is the product: comments on the promise of article 2B for software and information licensing”, *Berkeley Technology Law Journal On Line* 133, available at: http://www.law.berkeley.edu/journals/btlj/articles/vol13/Gomulkiewicz/html/reader.html.
Gomulkiewicz, R.W., 1999, “How copyleft uses license rights to succeed in the F/OSS software revolution and the implications for Article 2B”, *Houston Law Review* 36, 179–194.
Grossman, Sanford J. and Oliver D. Hart. 1986. “The Costs ad Benefits of Ownership: a Theory of Vertical and Lateral Integration,” *Journal of Political Economy* 944, 691-719.
Hahn, R. W. 2002. *Government Policy toward F/OSS Software*. AEIBrooking Joint Center for Regulatory Studies, Washington D.C.
Hann, Il-Horn et al. 2002, “Delayed Returns to F/OSS Participation: an Empirical Analysis of the Apache http Server Project”, working paper available at: http://www.idei.asso.fr/commun/Conferences/Internet/OSS2002/Papiers/Hann.PDF.
Hansmann, Henry B. 1980. "The Role of Nonprofit Enterprise." *Yale Law Journal*, 89, 835-901.

Harhoff, D., Henkel, J. and E. von Hippel. 2000. "Profiting from voluntary information spillovers: How users benefit by freely revealing their innovations", *MIT Sloan School of Management Working Paper*, available at: www.opensource.mit.edu/papers/evhippel-voluntaryinfospillover.pdf.

Hars, A., Ou, S., 2000. "Working for free? Motivations of participating in F/OSS projects", in: *Proceedings of the 34th Hawaii International Conference on System Sciences, IEEE.*

Hart, Oliver and John Moore. 1990. "Property Rights and the Nature of the Firm," *Journal of Political Economy* 986, 1119-1158.

Hart, Oliver D. 1995. *Firms, Contracts and Financial Structure*. Oxford: Clarendon Press.

Hawkins R. E. 2002. "The economics of the F/OSS Software for a competitive firm", working paper available at: http://opensource.mit.edu/papers/hawkins.pdf.

Healy K., Schussman A. 2003 "The ecology of F/OSS software development", working paper available at: http://opensource.mit.edu/papers/healyschussman.pdf.

Hecker, F., 1999 "Setting up shop: the business of Open-Source software". *IEEE Software* 16 1, 45–51.

Heffan I. V. 1997 "Copyleft: licensing collaborative works in the digital age", *Stanford Law Review* 49, 1487-1509.

Heller, Michael A. and Rebecca S. Eisenberg. 1998. "Can Patents Deter Innovation? The Anticommons in Biomedical Research," *Science* 280, 698-701.

Henkel, Joachim 2002, "F/OSS Software from Commercial Firms: Tools, Complements, and Collective Invention", GEABA Discussion Paper 02-27.

Hertel G., Niedner S., Hermann S. 2003."Motivation of software developers in the F/OSS projects: an Internet-based survey of contributors to the Linux kernel", *Research Policy*, 327, 1159-1177.

Holmstrom, B., 1999, "Managerial Incentive Problems: A Dynamic Perspective", *Review of Economic Studies*, 66, 169-182.

Horne N. T. 2001. "F/OSS software licensing: using copyright law to encourage free use", *Georgia State University Law Review*, 17, 863-xx.

Hunt F., Johnson P. 2002. "On the Pareto distribution of Sourceforge projects". In *Proceedings of the F/OSS Software Development Workshop*, 122-129, Newcastle, UK.

Järvinen H. 2002 "Legal aspects of F/OSS licensing", available at: http://www.cs.helsinki.fi/ u /campa/teaching/oss/papers/jarvinen.pdf.

Johnson, Justin Pappas 2001 "Economics of F/OSS software", working paper available at: http://opensource.mit.edu/papers/johnsonopensource.pdf.

Jorgensen, N. 2001 "Putting it all in the trunk: incremental software development in the FreeBSD F/OSS Project", *Information Systems Journal*, 114.

Kahin, Brian 2002, "The Patent Theat to F/OSS Software", in *Report of the Georgetown University 2002 F/OSS Summit: Public Interest and Policy Issues*.

Kasper E. 2001. "Epistemic communities, situated learning and F/OSS software development", working paper available at: http://opensource.mit.edu/papers/kasperedwards-ec.pdf.

Kelty, Christopher M. 2001 "Free Software/Free Science", *First Monday* 612, available at: www.firstmonday.org/issues/issue6_12/kelty/index.html.

Kennedy D. M. 2001 "A primer on F/OSS licensing legal issues: copyright, copyleft and copyfuture", *Saint Louis University Public Law Review*, XX

Kim E. E. 2003. "An introduction to F/OSS communities", working paper available at: http://www.blueoxen.org/research/00007/index.html.

Koch, S., Schneider, G., 2002. "Effort, co-operation and co-ordination in an F/OSS software project: GNOME". *Information Systems Journal* 12 1, 27–42.

Kogut, Bruce and Anca Metiu, 2001. "Open-Source software development and distributed innovation," *Oxford Review of Economic Policy* 172, 248-264.

Kollock, Peter 1999, "The Economies of Online Cooperation: Gifts and Public Goods in Cyberspace", in Marc Smith and Peter Kollock eds., *Communities in Cyberspace*. Routledge, London.

Kollock, Peter and Marc Smith, 1996 "Managing the Virtual Commons: Cooperation and Conflict in Computer Communities", in Susan Herring ed. *Computer-Mediated Communication: Linguistic, Social and Cross-Cultural Perspectives*. John Benjamins, Amsterdam.

Krishnamurthy, Sandeep. 2002 "Cave or Community? An Empirical Examination of 100 Mature F/OSS Projects." , First Monday 76, available at: http://firstmonday.org/issues/ issue7_6/krishnamurthy/index.html.

Kuan, Jennifer, 2001. "F/OSS Software as Consumer Integration into Production", working paper available at: www.papers.ssrn.com/ paper.taf?abstract_id=259648.

Kuwabara, Ko, 2000. "Linux: A Bazaar at the Edge of Chaos," *First Monday*, 53, available at: www.firstmonday.org/issues/issue5_3/kuwabara/index.html.

Lakhani, K., Wolf, B., Bates, J., DiBona, C., 2002. The Boston Consulting Group Hacker Survey, Release 0.73. Located at: http://www.bcg.com/opensource/ BCGHackerSurveyOSCON24July02v073.pdf.

Lakhani, Karim and Eric von Hippel, 2003 "How F/OSS software works: 'free' user-to-user assistance," *Research Policy* 326, 923-943.

Lakhani, Karim and Robert G. Wolf, 2003, "Why Hackers Do What They Do: Understanding Motivation Efforts in Free/F/OSS Projects", working paper 4425-03, MIT Sloan School of Management.

Lanjouw, Jean O. and Mark Shankerman 2001, "Characteristics of Patent Litigation: a Window on Competition", *RAND Journal of Economics* 321, 129-151.

Lave, J. and E. Wenger, 1991. *Situated Learning. Legitimate peripheral participation*, University of Cambridge Press, Cambridge.

Lee J. 2003 "*F/OSS software licenses and innovation"*, working paper available at: http://VirtualGoods.tuilmenau.de/2003/licenses_innovation_leejz.pdf.

Lee S. H. 1999 "*F/OSS software licensing"*, working paper available at: http://eon.law.harvard.edu/openlaw/gpl.pdf.

Lee, Samuel, Nina Moisa and Marco Weiss, 2003, "F/OSS as a Signaling Device: an Economic Analysis", working paper available at: http://opensource.mit.edu/papers/ leemoisaweiss.pdf.

Lerner, J., Tirole, J., 2002a. "Some simple economics of F/OSS", *Journal of Industrial Economics* 52, 197–234.

Lerner, J., Tirole, J., 2002b, "The Scope of F/OSS Licensing", Harvard Business School working paper, available at: http://www.people.hbs.edu/jlerner /simple.pdf.

Lessig, Lawrence 2002, "F/OSS Baselines: Compared to What?", in In Hahn ed., *Government Policy toward F/OSS Software*. AEIBrooking Joint Center for Regulatory Studies, Washington D.C.
Lindenberg, S., 2001."Intrinsic Motivation in a New Light", *Kyklos* 542/3, 317-343.
Ljungberg, J., 2000. "F/OSS movements as a model for organizing", *European Journal of Information Systems,* 94, pp. 208-216.
Manenti, Fabio and Stefano Comino, 2003, "F/OSS vs. Closed Source Software: Public Policies for the Software Market", working paper, University of Padova.
Mateos-Garcia, Juan and W. Edward Steinmueller, 2003."The F/OSS Way of Working: A New Paradigm for the Division of Labour in Software Development?', INK F/OSS Research Working Paper No. 1, SPRU-University of Sussex, Brighton, England.
Maurer, Stephen M., 2003. "New Institutions for Doing Science: from Databases to Open Source Biology," paper presented at the European Policy for Intellectual Property Conference, Maastricht.
Mauss, M., 1954. *The Gift. Forms and Functions of Exchange in Archaic Societies.* Cohen and West, London.
Mazzoleni, R. and Richard R. Nelson, 1998. "The benefits and costs of strong patent protection: a contribution to the current debate", *Research Policy* 27, 273-284
McGowan, D., 2001. "Legal implications of open-source software", *University of Illinois Law Review* 1, 241–304.
McJohn, Stephen M. 2000. "The Paradoxes of Free Software", *George Mason Law Review*, fall, 25-68.
Meurer, Michael J. 2003, "Controlling Opportunistic and Anticompetitive Intellectual Property Litigation", *Boston College Law Review* forthcoming.
Mocus A., Fielding R. et Herbsleb J. 2000, "A case study of F/OSS software development: the Apache server" *Proc. 2000 Int'l Conference on Software Engineering* ICSE2000, Limerick, Ireland, June 4-11, 263-272.
Moglen E. 1999 *Anarchism triumphant: free software and the death of copyright*. First Monday 48, available at: http://www.firstmonday.dk/issues/issue48/moglen/index.html.
Moody, Glynn. 2001. *Rebel Code: The Inside Story of Linux and the F/OSS Revolution*. Cambridge, Massachusetts: Perseus Publishing.
Moon, J. Y. and L. Sproull 2000 "Essence of distributed work: The case of the Linux kernel", *First Monday* 511.
Morrison, Pam, John Roberts and Eric von Hippel 2000, "Determinants of User Innovation and Innovation Sharing in a Local Market ," *Management Science* 4612, 1513-1527.
Mustonen, Mikko 2003 "Copyleft – the economics of Linux and other F/OSS software", *Information Economics and Policy*, 15, 99-121.
Nadan, Christian H. 2002. "F/OSS Licensing: Virus or Virtue?", *Texas Intellectual Property Journal 10, 349.*
Neumann, P.G. 2001 "Robust non-proprietary software", *IEEE Symposium on security and privacy,* May.
Nilendu Pal and Madanmohan T. R., 2001. "Competing on F/OSS: Strategies and Practise", working paper available at: http://opensource.mit.edu/papers/madanmohan.pdf.

O'Mahony S. 2003. "Guarding the commons: how do community managed software projects protect their work", *Research Policy*, 327, 1179-1198.
O'Mahony, S., 2002. *Community Managed Software Projects: The Emergence of a New Commercial Actor.* Unpublished Ph.D.Dissertation. Stanford University, 2002.
Oksanen V., Valimaki M. 2002 "Evaluation of F/OSS licensing models for a company developing mass market software", available at: http://www.hiit.fi/de/valimakioksanen_lawtech_2002.pdf.
Osterhout J 1999. "Free software needs profit", *Communications of the ACM* 424, 38–39.
Osterloh M., Rota s., Von Wartburg M. 2002 "*F/OSS – new rules in software development*", working paper available at: http://www.unizh.ch/ifbf/orga/downloads/opensourceaom.pdf.
Ostrom, E. 1990. *Governing the commons: The evolution of institutions for collective action.* Cambridge University Press, New York, NY.
Pagano U. 1999 "Is Power an Economic Good? Notes on Social Scarcity and the Economics of Positional Goods". In Bowles S., Franzini M., Pagano U. 1999 *The Politics and the Economics of Power.* Routledge, London , 63-85.
Pagano, U. 2002, "Legal Positions and Institutional Complementarities", *Quaderni del Dipartimento di Economia Politica n.360*, Università di Siena.
Perens, Bruce, 1999. "The F/OSS Definition," In DiBona, C., Ockman, S., Stone, M. Eds. *F/OSSs: Voices from the F/OSS Revolution*. O'Reilly & Associates, Sebastopol, CA.
Ravicher, Daniel B., 2000. "Facilitating Collaborative Software Development: the Enforceability of Mass-Market Public Software Licenses", *Virginia Journal of Law and Technology* 115, 1522-1687.
Raymond, Eric S., 1999. "A Brief History of Hackerdom". In DiBona, C., Ockman, S., Stone, M. Eds. *F/OSSs: Voices from the F/OSS Revolution*. O'Reilly & Associates, Sebastopol, CA.
Raymond, Eric S., 1998a. "The Cathedral and the Bazaar," *First Monday*, 33, available at: www.firstmonday.org/issues/issue3_3/raymond/index.html.
Raymond, Eric S., 1998b. "Homesteading the Noosphere," *First Monday*, 310, available at www.firstmonday.org/issues/issue3_10/raymond/index.html.
Rosen L. 2001a "The unreasonable fear of infection", available at: http://www.rosenlaw.com /html/GPL.PDF.
Rosen L. 2001b "Which F/OSS license should I use for my software?", available at: http://www.rosenlaw.com/html/GL5.pdf.
Rosenberg D. K. 1998 "Evaluation of public software licenses", available at: http://www.stormian.com/Public Licenses.html.
Schmidt, K. and Schnitzer, M. 2003. "Public Subsidies for F/OSS? Some Economic Policy Issues of the Software Market", CEPR Discussion Paper, No. 3793.
Schweik C. M., Semenov A. 2003 "*The institutional design of F/OSS programming: implications for addressing complex public policy and management problems*" First Monday, http://firstmonday.org/issues/ /issue8_1/schweik/index.html.
Simon, Herbert A. 1991,"Organizations and Markets", *Journal of Economic Perspectives* 5, 25-XX.

Smith, B. L. 2002. “The Future of Software: Enabling the Marketplace to Decide”. In Hahn ed., *Government Policy toward F/OSS Software.* AEIBrooking Joint Center for Regulatory Studies, Washington D.C.

Stallman 1984. “What is copyleft?”, available at: http://www.gnu.org/copyleft/copyleft.html.

Stallman, R., 1999. “The GNU operating system and the free software movement”. In DiBona, C., Ockman, S., Stone, M. Eds. *F/OSSs: Voices from the F/OSS Revolution*. O'Reilly & Associates, Sebastopol, CA.

Stewart, Katerine J. and Sanjay Gosain, 2003, “Impact of Ideology, Trust and Communication on Effectiveness in F/OSS Software Development Teams”, working paper, available at: http://opensource.mit.edu/papers/stewartgosain.pdf.

Torvalds, L. 1998, “What motivates free developers?”, Interview in *First Monday* 33, available at: http://firstmonday.org/issues/issue3_3/torvalds/index.html.

Torvalds, L. 1999, “The Linux Edge”, in DiBona, C., Ockman, S., Stone, M. Eds. *F/OSSs: Voices from the F/OSS Revolution*. O'Reilly & Associates, Sebastopol, CA.

Tuomi, Ilkka, 2001. “Internet, Innovation, and F/OSS: Actors in the Network”, *First Monday* 61.

Valloppillil, V., 1998, “F/OSS Software: A New? Development Methodology?” [known as the “Halloween Document”], working paper available at http://www.opensource.org/ halloween/halloween1.html.

von Hippel, E., von Krogh, 2003. “The private-collective innovation model in F/OSS Software development: issues for organization science”. *Organization Science*, in press.

von Hippel, Eric 1988 *The Sources of Innovation.* New York: Oxford University Press.

von Krogh G., Spaeth S., Lakhani K. R. 2003. “Community, join, and specialization in F/OSS software innovation: a case study”, *Research Policy*, 327, 1217-1241.

von Krogh, G. 1998 "Care in Knowledge Creation", *California Management Review,* 403, 133-153.

von Krogh, G. 2002 "The communal resource and information systems", *Journal of Strategic Information Systems* 112, 85-107.

von Krogh, Georg, Sebastian Spaeth and Karim R. Lakhani, 2003, “Community, joining, and specialization in F/OSS software innovation: a case study”, *Research Policy* 327, 1217-1247.

Weber, Steven, 2000. “The Political Economy of F/OSS Software”. BRIE Working Paper 140, Economy Project Working Paper 15.

Wichmann T. 2002a “Basics of F/OSS software markets and business models”. *Free/Libre and F/OSS Software: Survey and Study, FLOSS Final Report*, International Institute of Infonomics, Berlecom Research GmbH.

Wichmann T. 2002b.“Firms’ F/OSS activities: motivations and policy implications”, *Free/Libre and F/OSS Software: Survey and Study, FLOSS Final Report*, International Institute of Infonomics, Berlecom Research GmbH.

Xu, Xiaopeng, 2002, “Development Costs and F/OSS Software”, working Paper, University of California at Berkeley.

Young R 1999. “Giving it away: how Red Hat software stumbled across a new economic model and helped improve an industry”. In DiBona, C., Ockman, S.,

Stone, M. Eds. *F/OSSs: Voices from the F/OSS Revolution*. O'Reilly & Associates, Sebastopol, CA.
Zang, Wei and John Storck, 2001, "Peripheral members in On-line Communities", working paper available at: http://opensource.mit.edu/papers/zhang.pdf.
Zeitlyn, David, 2003, "Gift economies in the development of F/OSS software: anthropological reflections", *Research Policy* 327, 1287-1291 .
Zimmerman, Jean-Benoit, 1999, "Logiciels et propriété intellectuelle: du copyright au copyleft", working paper CNRS-Grequam, Marseille.