Rullani, Francesco

Working Paper

Dragging developers towards the core: How the free/libre/ open source software community enhances developers' contribution

LEM Working Paper Series, No. 2006/22

Provided in Cooperation with:
Laboratory of Economics and Management (LEM), Sant'Anna School of Advanced Studies

Suggested Citation: Rullani, Francesco (2006) : Dragging developers towards the core: How the free/ libre/open source software community enhances developers' contribution, LEM Working Paper Series, No. 2006/22, Scuola Superiore Sant'Anna, Laboratory of Economics and Management (LEM), Pisa

This Version is available at:
https://hdl.handle.net/10419/89377

Laboratory of Economics and Management
Sant'Anna School of Advanced Studies

Piazza Martiri della Libertà, 33 - 56127 PISA (Italy)
Tel. +39-050-883-343 Fax +39-050-883-344
Email: lem@sssup.it Web Page: http://www.lem.sssup.it/

LEM
Working Paper Series

Dragging developers towards the core:

How the Free/Libre/Open Source Software community enhances developers' contribution

Francesco Rullani*

*Sant'Anna School of Advanced Studies, Pisa, Italy
and
IVS – Copenhagen Business School

2006/22 **September 2006**

Dragging developers towards the core
How the Free/Libre/Open Source Software community enhances developers' contribution

Francesco Rullani

LEM - Sant'Anna School of Advanced Studies
IVS – Copenhagen Business School

This version*: January, 2006*

"Hackers produce more than software,
they produce hackers"
Lars Risan[1]

Abstract

The paper presents a *dynamic perspective* on the landscape of Free/Libre/Open Source Software (FLOSS) developers' motivations and tries to isolate mechanisms sustaining developers' contribution over time. The first part of the paper uses data gathered by the empirical studies relative to the FLOSS case to judge the relative importance of each group of incentives detected by the literature. In the second part of the paper, the same data are used to further characterize developers' motivations in dynamics terms, showing how the relative importance of different incentives changes over time. Drawing inspiration from these results, the third part of the paper identifies a specific mechanism fostering developers' contribution to the community activities, namely that: "*Independently of developers' exogenous preferences, the more their exposure to the FLOSS community social environment, the more their contribution to the community activities*". The key point of this hypothesis is that, if the exposure to the FLOSS community social environment is able to foster developers' contribution *beyond* the level granted by their predetermined preferences, this leads directly to the evidence that the FLOSS community is provided with a mechanism sustaining and enhancing developers' incentives to produce and diffuse code. In the last part of the paper, data relative to 14,497 developers working on SourceForge.net during two years (2001-2002) are employed to estimate a model testing the aforementioned hypothesis. Endogeneity problems are explicitly accounted for, and robustness checks are performed in order to make sure that the observed confirmation of the hypothesis is actually an empirically grounded result.

Acknowledgements*: I am grateful to Alfonso Gambardella, Paul David, Bronwyn Hall, Paola Giuri, Salvatore Torrisi, Alessandro Nuvolari, Matteo Ploner, Gaia Rocchetti, Francesco Rentocchini and all the participants to the LEM, and CESPRI seminars, to the DIME workshop on "Motivations and Incentives" (Lisbon, WP 1.1), to the CCC Conference (Lausanne) and to the workshop "Mediterranean Research on Free/Libre and Open Source Software" (Venice). I thank the SourceForge.net staff for having provided our group with the data and the Italian developers of SF.net who helped us in interpreting the data. I also acknowledge the financial support of the project "Economic Change: the micro foundations of institutional and organisational change: EconChange", CE, DGXII, FP V. The usual disclaimers apply.*

Reference address*: Francesco Rullani, Laboratory of Economics and Management, Sant'Anna School of Advanced Studies, Piazza dei Martiri della liberta, 33, 56127 Pisa, Italy (rullani@sssup.it) and IVS, Copenhagen Business School, Kilevej 14A, 2000, Frederiksberg, Denmark, (fr.ivs@cbs.dk).*

Keywords*: Free/Libre/Open Source Software, incentives to innovate, dynamics of motivations, cooperation, community.*

JEL classification*: O31, L86*

[1] *See http://folk.uio.no/lrisan/Linux/Identity_games/*

1. Introduction

In recent years, new institutional and organizational forms have emerged. Knowledge-based communities, where knowledge is produced and shared among the community members have moved to the center of the social and economic scene, and open organizations have become increasingly object of a wide range of studies (David and Foray, 2003). These organizations have the peculiarity of widening the division of the innovative labor providing their members with significant incentives to innovate *and* to diffuse the produced knowledge. As Arrow showed (1962), the se two conditions are not naturally coupled. Their contemporaneous realization is instead reached by means of "artificial" and peculiar structures such as Science and Technology (David and Dasgupta, 1987, 1994) or Hierarchy (Williamson, 1975), or determined by specific contingencies directly changing the terms of actors' payoff functions (temporarily), as it is the case for Collective Inventions (Allen, 1983; Nuvolari, 2005, 2004). In order to understand how these organizations really work and the extent at which their model is applicable to other domains of knowledge production the main question economists need to answer is then: how do these organizations overcome Arrow's contradiction? What sustains individuals' incentives to innovate even in an "open environment".

The study undertaken here contributes in building an answer to this question analyzing the Free/Libre/Open Source (FLOSS) model of production. FLOSS is produced by a community of developers spread worldwide, who do not directly gain any monetary reward from their activity and nonetheless keep producing software and diffusing it on the internet for free. Thus, FLOSS represents one of the most interesting examples of knowledge-based communities.
In particular, the present paper tries to scan the landscape of FLOSS developers' incentives to isolate the principles able to sustain developers' contribution *over time*.
The first part of the paper argues that data gathered by other empirical studies show that developers' motivations do change over time. On this basis, part two identifies the "exposure to the community social environment" as the mechanism behind this change. If this mechanism is actually able to foster developers' contribution *independently* of developers' pre-determined preferences, then it is possible to state that the FLOSS community is provided with a mechanism solving the contradiction between knowledge production and diffusion pointed out by Arrow. The econometric analysis developed in part three supports precisely this argument.

2. The context and the purpose of the paper

2.1. Institutions and production of knowledge

Knowledge contains a contradiction. It is intrinsically cumulative, so that its dissemination should enable a wider and faster production of new knowledge. At the same time, the incentive to produce it is inhibited by its non-rivalry and (partial) non-excludability, and because, once codified, nowadays it can be replicated and transferred at virtually no costs. These properties limit to a set of special cases the production of rents springing directly from the innovation and aimed at compensating the innovators for the (sunk) costs and risks they bore in the first place, during the research activity. Thus, while knowledge features call for a collective mode of production, the same features cause a "pure" market solution to usually fail as a coordination mechanism (Arrow, 1962).
Dasgupta and David (1987, 1994) showed that Science and Technology emerged as institutions able to solve the contradiction. Similarly, Williamson's (1975) conceptual description of the firm pictures an institutional structure able to provide researchers with the incentive to innovate and share the results within the company (even if with the *caveats* highlighted by the Agency Theory): hierarchy. Even in the absence of these structures, Collective Inventions, where economic agents share knowledge freely and openly, occasionally (and temporary) occur (Allen, 1983; Nuvolari, 2005, 2004). Arrow's intuition (1962) creates then a dichotomous conceptualization of knowledge

production. On one side, the institutions and circumstances sketched above, enabling the collective production of knowledge. On the other side, environments where such conditions are not realized and the consequent "pure-market-like situation" results in the innovators' lack of incentives in investing in the research activity.

2.2. What is FLOSS community and why should we focus on it?

In recent years, the advent of the information society and the diffusion of the Information and Communication Technologies (ICT) created the conditions for a wider division of the innovative labor. A high number of dispersed individuals can be mobilized through the internet, enabling the system to reach the necessary critical mass to effectively produce knowledge collectively. At the same time, codified knowledge can be highly modularized and new knowledge can be easily matched with the existing one thanks to the improvements in the interfaces (e.g. the *concurrent versioning system*, CVS, for software production). These changes created room for the emergence of knowledge-based communities (David and Foray, 2003), where agents collectively produce and "freely" exchange knowledge without a formal and central authority, without IPR or state intervention.

These changes were particularly evident in the software industry. From the fifties to the seventies, software was produced in scientific-research like way. During the eighties, however, many companies and developers started to apply for patents and copyright for their code, or "closed" the source code of their programs in order to be able to extract monetary rents from it (Benussi, 2005; Nuvolari, 2005). The source code is the interface between the Machine, which reads binary information, and the Man, who understands natural-language-like instructions. If the source code of a program is attached to the program itself, it can be red by other developers, who can also act on it and improve the program. If, on the contrary, the code is kept secret, the structure of the program can be revealed only through costly (and often illegal) practices as reverse engineering. Closing the code, thus, means blocking the diffusion of the knowledge embodied in the program, and assigning *de facto* to the original developer the complete control on the innovation. In opposition to this entry in the "Technology realm", Richard Stallman, a researcher at MIT, designed the General Public License (GPL), and released the software he produced under its terms. The GPL enables other developers to copy and distribute the software, to access the source code of the program, to modify it and to redistribute it. At the same time, it requires that the resulting software is distributed under the same terms of the original one. Many other developers improved the original Stallman's software and applied to their own original code the GPL or a similar -sometimes less restrictive- licenses[2].

In few years the FLOSS community, i.e. the community of users and developers of software licensed under an Open Source license (see http://www.opensource.org/), grew and spread worldwide thanks to the diffusion of the ICT, to the modularity of the production process (Giuri *et al*., 2005) and to the flexible modularity of the code architecture (Narduzzo and Rossi, 2004). Nowadays, about 70% of web servers on the internet run Apache, a software distributed under a specific Open Source license (the "Apache web license"), and Linux is seriously threatening the incumbents on market for desktops. This candidates the FLOSS community as one of the most successful cases of the knowledge-based communities described by David and Foray (2003).

2.3. The aim of the research

The situation depicted above is at odds with the specific license regime under which FLOSS products are released. FLOSS can be freely copied and redistributed, and the majority of the developers do not extract a direct monetary rent from the innovation they produce. This should

[2] See Giuri *et al*., 2002; Gambardella and Hall, 2006; Lerner and Tirole, 2005; for a discussion on the effects of GPL and of the different degrees of openness of the other licenses used by the FLOSS community.

result in a low rate of innovation, and eventually in the disappearance of the cooperative regime.
But -by now- it does not. How is this possible? What is the mechanism the FLOSS community puts forward to enhance its developers' contribution and thus solve Arrow's contradiction?
The aim of the present paper is trying to answer these questions. In doing so, it also helps to shed light on the mechanisms behind the knowledge-based communities David and Foray (2003) place at the basis of a knowledge-based society.

The managerial and economic literature has assessed several dimensions of the FLOSS model. In order to understand how the FLOSS community solves the contradiction between incentives to innovate and knowledge diffusion, the focus has to be placed on the agents' motivations to create and diffuse software. Besides few exceptions (e.g. Shah, 2006; Bagozzi and Dholakia, 2006; von Krogh *et al*., 2003a; Ghosh *et al*., 2002), the literature on this topic usually treats developers' incentives as static. However, "future research should explain [...] the motives of joiners and how they change over time as they work their way into the project. This will help get a more complete picture of those factors enabling growth and continuation of projects." (von Krogh *et al*., 2003b, p. 1235).
Thus, the first section of the paper elaborates on empirical literature about FLOSS to explore the *dynamic* structure of developers' incentives and to stress that motivations do change over time.
As a further step, the paper tries to identify a possible "*engine*" behind these changes. In the second section the observed evolution of the incentive scheme is analyzed to identify the mechanism driving the transformation of developers' preferences. In particular, what emerges is that -independently of developers' exogenously determined preferences- the more developers' exposure to the FLOSS community social environment, the more their contribution to the community activities.
The last contribution of the paper, developed in the third sections, is a test of this hypothesis by means of regression analysis. The confirmation of the hypothesis puts forward the need to reframe in a dynamic fashion the motivational sets used by the literature to explain FLOSS existence. It also sheds light on the FLOSS model and thus helps to understand a series of related issues, such as the sustainability of FLOSS as a model for innovation.

3. What drives developers' effort and cooperation

3.1. How a user becomes a developer

Mateos Garcia and Steinmueller (2003) and von Krogh *et al*. (2003b) notice that the process by which members of the community are recruited and accepted plays a central role in shaping the evolution of the community itself, and in determining its capabilities to survive and grow.
This process seems to begin with a period of "gestation" through which FLOSS users can move from the outer circles toward the core of the community becoming developers. This is the period when the user *learns* about the community rules, habits, language and "vision", and when the community *examines* the user, evaluating her capabilities, reliability and trustworthiness. In particular, analyzing the case of Freenet, von Krogh *et al*. (2003a; 2003b) show that would-be developers usually approach the community social environment initially "lurking" at the mailing lists of the project. This way, they gather information on the community social environment, on the topics developed in the community debate, as well on the software architecture. At a certain point, if they find something interesting for them and -in their opinion- useful to the community, they provide a feedback posting comments and/or code to the mailing list. At this point users begin a dialog with the community members by which the latter can evaluate the former's ideas and software contributions. The provision of software is a fundamental moment of the joining process, because it gives the community the possibility to evaluate the actual skills of the user. And in fact "[The interview with developer #405] shows the developer favors hand-on solutions to technical problem, and that demonstration of technical knowledge in the form of software code submission

matters more than signaling of interest and experience" (von Krogh *et al*., 2003b; p. 1229). If the would-be developer behaves as described, her admission to the "inner" circles of development (e.g. getting access to the CVS) is more likely (von Krogh *et al*., 2003b). A similar result is obtained by David and Rullani (2006), who find that developers active in the long-run are also those who become active after a first period of lurking.

Thus, on the one hand, the participation of the user into a process of "regulated" interaction is a fundamental tool to let community members judge her real skills, commitment, vision of the project and willingness to absorb and respect the rules of the community. On the other hand, the exposition to the community enables the newcomer to understand what the social structure and the technical needs of the community are, and to relate them to her interests and potentials.

3.2. Static and dynamic assessments of developers' incentives sets

Having described the process by which a user becomes a developer, a study of the motivations driving that process is needed. The empirical and the theoretical literature have highlighted four main "fields" of incentives at work in FLOSS case, summarized in table 1.

Table 1. Developers' motivations as described by the literature.

#	Motivation	Literature	Category
1	*monetary* rewards, *carrier* concerns and *signaling*[3], when individuals decide to produce code and freely distribute it in order to signal their capabilities to the job market or because they have some form of monetary gains (e.g. they run or are employed in a FLOSS-based firm).	Lerner and Tirole, 2002; Ghosh, 1998; Roberts *et al*., 2006.	*Economic incentives*
2	*own-use*, when a specific software is realized to answer the particular needs of its creator and distributed in order to exploit the economies of scale derived by the collective work (the "cooking pot" defined by Ghosh, 1998) or to influence the trajectory of the product development.	von Hippel, 2002; von Hippel and von Krogh, 2003; Shah, 2006; Ghosh, 1998; Weber, 2000; Jeppesen and Frederiksen, 2006; Lakhani and von Hippel, 2003; von Krogh *et al*., 2003a.	*Own-use*
3	*learning*, when developers interact to participate in a social environment where they can acquire new skills and where new problems and new solutions are discussed and implemented.	Ghosh 2003a; von Hippel and von Krogh, 2003; Bonaccorsi and Rossi, 2006; Lakhani and von Hippel, 2003; von Krogh *et al*., 2003a.	*Learning*
4	a group of *physiological motivations* -such as fun or creativity- and *social motivations* -such as group kinship or identity, hacker culture, ideology or the need for reciprocation, reputation and peers' regard, enhancing one's "ego", seeking a higher status in the community-.	Torvalds and Diamond, 2001; Lakhani and Wolf, 2005; Weber, 2000, 2004; Osterloh and Rota, 2004; Bitzer *et al*., 2004; Lerner and Tirole, 2002; Dalle and David, 2005; Dalle *et al*., 2004; Bagozzi and Dholakia, 2006; Frederiksen, 2006; Bonaccorsi and Rossi, 2006; Shah, 2006; Himanen *et al*., 2001. Hertel *et al*., 2003; Zeitlyn, 2003; Raymond, 1998.	*Social and psychological incentives*

Following the classification represented in the table, FLOSS developers can be moved by the need to signal their skills to a wide community in order to reach a better position in the labor market, or simply by the fact that FLOSS is a means to earn money (Lerner and Tirole, 2002; Roberts *et al*.,

[3] Lerner and Tirole (2002) label "signaling" any activity undertaken for the purpose of acquiring reputation, both among peers (peers' regard) and on the job market (career concerns). For the purposes of the present study, these two factors have to be conceived as belonging to different realms of developers' motivations (social and psychological incentives and economic incentives, respectively). The label "signaling" is here attached only to the economic side of the reputation concerns.

2006). At the same time, firms' difficulties in identifying and matching each developer's needs and the disproportion between the small cost of free revealing and the high revenue of gathering other developers adopting and possibly working on the software one needs creates the incentive for lead users to engage in the FLOSS production (Franke and von Hippel, 2003). Besides these "economic" explanations, also the social (Bagozzi and Dholakia, 2006) and psychological (Lakhani and Wolf, 2005) sides of developers' incentives have been proved to be crucial in determining individuals' active participation in FLOSS projects. Eventually, both theoretical (e.g. von Hippel and von Krogh, 2003; von Krogh *et al*., 2003a; 2003b) and empirical literature (Ghosh *et al*., 2002; David *et al*., 2003, Lakhani *et al*., 2002) highlighted the importance of learning opportunities the FLOSS offers to those who actively contribute to its construction.

As shown above, in their initial experience of FLOSS developers were basically FLOSS users. The time dimension of this period gives a clear proof of its importance. Among the responders to the FLOSS-US survey (David *et al*., 2003) only 19.8% of the developers used FLOSS for about six months before starting to develop the software. The rest of the responders (80.2%) declared a longer period of FLOSS usage, oscillating between "about a year" and "more than 3 years". Notice that the most numerous group (29.2%) answered "about 2-3 years". The survey also assesses developers' reasons to approach FLOSS. Even if the question provides answers only related to two specific environments (leisure or work time), it is interesting to notice that "for recreation" was marked by a substantial number of users. In this period, when developers are not part of the community yet, the users approached FLOSS not only because they needed to use it in their workplace (own-use) but also for recreation.

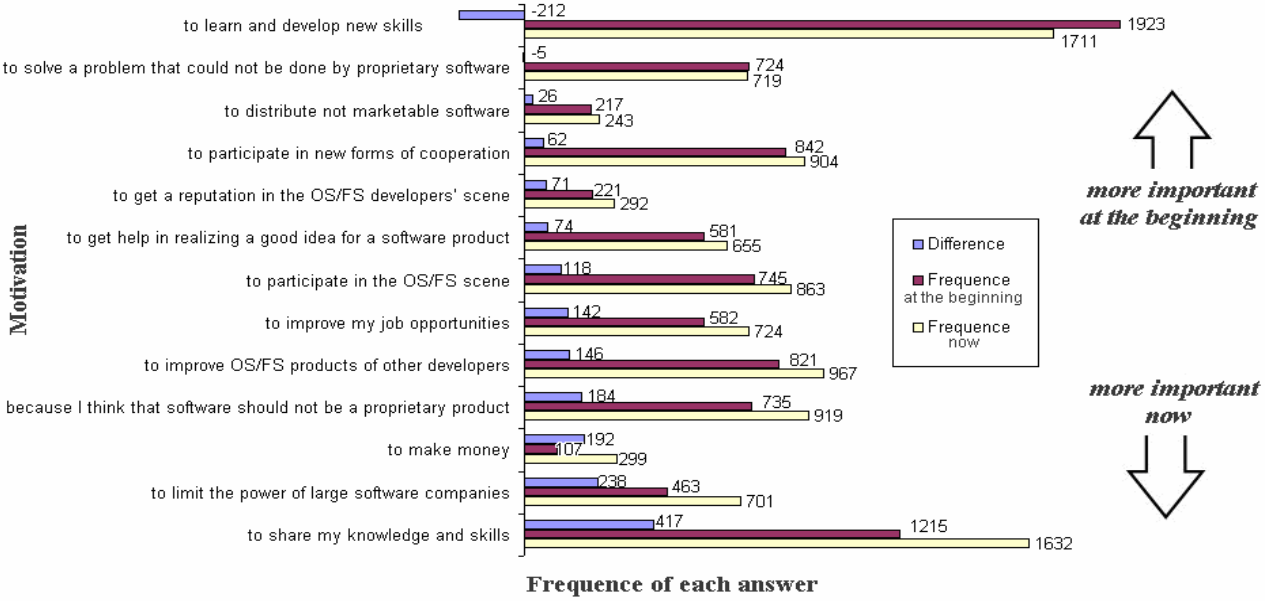
The previous phase ends when the user becomes a developer. In this transformation, the role of "fun" or "own-use" is less important. In the FLOSS-US survey out of 12 possible motivations to *start developing* FLOSS the answers clearly related to fun and creativity ("I liked the challenge of fixing bugs and problems in existing software") or own-use of software ("I needed to fix bugs in existing software" and "I needed to perform tasks that could only be done with modified versions of existing software") are "very important" and "important" only for a small subset of individuals, so that they rank only eleventh, tenth and eighth, respectively. Notice also that the work-related option, "My employer wanted me to collaborate in open source development", ranks last.
This means motivations have changed. Once the developer has approached the community, the interaction with this new environment enriches her set of motivations, and assigns more weight to new "items". This can be clearly seen when considering the social side of user/developers' motivations. In fact, "As a developer of free software, I wanted to give something back to the community", which is clearly a product of the user's interaction with the community, and "I wanted to interact with like-minded programmers" rank second and seventh. Notice that the answers "I thought it was the best way for software to be developed" and "I thought we should all be free to modify the software we use" rank fourth and first, respectively. Thus ideological as well as technological reasons enter the picture. Eventually, consider that also learning opportunities become crucial: the option "I saw it as a way to become a better programmer" ranks third[4].
In nuce, there seems to be a dynamic process developing along the relationship between the individual and the community, which eventually result in a change of the motivations driving the former's entry into the "inner" circle of the developers.

Consider now figure 1 obtained from the FLOSS-EU survey (Ghosh *et al*., 2002). The motivations driving developers' contribution to the activity of the community in the first place (red bars) are mainly centered on learning and social leverages. In this case, also own-use has a significant effect, while signaling and economic incentives have only a marginal role.

[4] Notice that in the specific question of the FLOSS-US we are considering now is not provided an answer clearly and univocally related to monetary opportunities or signaling.

Figure 1. Developers' motivations at the beginning and now ordered by their difference.

Remembering the time you started developing and/or distributing OS/FS, what was the reason for this? And for what reason(s) do you go on with developing and/or distributing OS/FS? (max 4 answers)



Source: FLOSS survey (Ghosh et al., 2002).

Having described what drives users' transformation into developers, we need to focus now on the evolution of their involvement into the community. The relative importance of each motivational field can be assessed in an evolving context thanks precisely to figure 1. In figure 1 motivations have been ranked according to the difference between the number of responders who marked them in the two moments in time "today" and "at the time you started developing and/or distributing OS/FS"[5]. Consequently, the motivations placed at the top of the figure have to be considered more important at the beginning, while the incentives listed in the bottom part of the figure are those which acquire more importance over time.

First of all, notice that *there is* a change in the importance of each incentive over time. Motivations, as said, are not given at the beginning, "once and forever". On the contrary from figure 1 built on the FLOSS-EU survey data collected by Ghosh *et al*. (2002) and from the discussion in their report and in Glott *et al*. (2004) we derive the strong feeling that the structure of motivations *evolves*, changing the weight of the factors in agents' payoff functions.

Secondly, own-use related answers are generally important, but they do not seem to become more relevant over time. The frequencies related to this incentive field such as "to solve a problem that could not be done by proprietary software" or "to get help in realizing a good idea for a software product" do not change much from the first to the current developers' involvement. Learning has a similar behavior, inasmuch as it is generally important but does not acquire significance over time. On the contrary, it gradually becomes less central. Economic incentives, represented by answers such as "to improve my job opportunities", "To make money" and "To get a reputation in the OS/FS developers' scene", remain largely peripheral along the whole time scale. There is an increase in the number of developers marking some of these answers, but in absolute terms they remain marginal. On the contrary, social and psychological incentives are always crucial, and increase their relevance over time. This result is consistent with the empirical observations gathered by Shah (2006), where

[5] The answer "I do not know", which was given by 46 responders with respect to their first contributions and by 37 with respect to their current involvement, is not included in the figure.

developers initially contributing just to satisfy a need gradually change the set of their motivations so that, once the needs are fulfilled, they keep contributing basically as hobbyists.
Moving now to a finer-grain analysis, it is possible to search for the mechanisms behind this evolution.

3.3. Gradually identifying what "learning" really means

Learning is and remains the most important driver of participation. However, it figure 1 states it loses relevance over time. However, this result is at odds with other empirical evidence that showed the increasing importance of skill-improvements opportunities offered by FLOSS collaboration (Glott *et al*., 2004; Ghosh and Glott, 2005). An explanation for this can be given considering that learning is a complex object. It can be considered *per se* as well as considered an instrument to satisfy any of the other incentives discussed above. For example, learning enhances career opportunities (Ghosh and Glott, 2005) as well as the possibility for developers to create precisely the features they need, and it is at the same time a social and psychological process acting on the intrinsic motivations side. Thus, when developers enter the community and learning becomes an everyday activity, their experience of the community social environment increases their capability to better define the basic motivations behind their decision to mark "learning" in the first place.
The message this delivers is that entering the community, developers enter a social environment where there are a lot of opportunities related to social interaction, fun, own-use, reputation and signaling. The "blossoming" of a wide range of different (unexpected) opportunities pushes developers to consider the possibility to easily acquire a minimum quantity of each one of these different "motivational goods". A proof of this can again be found in figure 1 taking into account the number of answers given by the developers in the two considered time periods. Developers where allowed to mark up to four answers describing initial and current motivations behind their contribution to FLOSS. As figure 1 shows, the same number of developers[6] marked 9,176 answers with respect to the beginning of their experience in FLOSS, and 10,629 answers (15% more) when they are asked to evaluate their current motivations[7]. As said, such an evolution makes clear that the number of opportunities each developer began to face during her work in the FLOSS increased. Notice that these opportunities have to be considered as unexpected, otherwise they could have been treated as expectations and embodied in the motivations "at the beginning", generating the same series of answers in the two time periods.

3.4. Expectations of monetary rewards and the role of job opportunities

In the previous picture, economic incentives seem to have only a marginal role. Only few developers referring both to their first and to their current involvement state that their behavior was led by expected job opportunities or career concerns. When considering the each motivation variation over time as a percentage of the initial frequency, the results are however different, as table 2 shows.

[6] The survey reports for the initial period 2438 answers from 2774 insertions, and for the current period 2428 answers from 2774 insertions.
[7] There is a bias connected to the fact that some of the answers could have been given just for one of the two cases (initial/current), e.g. because for some developers the current project is actually the first one. Nonetheless, since the numbers of the responders is fairly equal for the two questions, it is plausible to assume that the possible error is very small, and that the qualitative results are not influenced by it.

Table 2. Developers' motivations (variation between the frequencies at the beginning and now)

Remembering the time you started developing and/or distributing OS/FS, what was the reason for this? And for what reason(s) do you go on with developing and/or distributing OS/FS? (max 4 answers)	Variation: $\left(\frac{F_n - F_b}{F_b}\right)$
to make money	*179%*
to limit the power of large software companies	*51%*
to share my knowledge and skills	*34%*
to get a reputation in the OS/FS developers scene	*32%*
because I think that software should not be a proprietary product	*25%*
to improve my job opportunities	*24%*
to improve OS/FS products of other developers	*18%*
to participate in the OS/FS scene	*16%*
to get help in realizing a good idea for a software product	*13%*
to distribute not marketable software	*12%*
to participate in new forms of cooperation	*7%*
to solve a problem that could not be done by proprietary software	*-1%*
to learn and develop new skills	*-11%*

Source: FLOSS survey (Ghosh et al., 2002). Notation: $\boldsymbol{F_b}$*= frequency at the beginning;* $\boldsymbol{F_n}$*= frequency now.*

In the table above, opportunities to "make money" or signal one's own skills through FLOSS development acquire the greatest importance over time. In other words, monetary and signaling-related motivations are the most elastic with respect to time. They are able to "attract" a number of developers which is smaller than other motivations in absolute terms, but greater in relative terms. Why?

In the FLOSS-US (David *et al*., 2003) survey to the question "How important were the following factors when you first started developing open source/free software?" the answer "My employer wanted me to collaborate in open source development" is the last in terms of importance, consistently with the FLOSS-EU survey (see figure 1 and Ghosh *et al*., 2002). Thus, at the beginning, career concerns and expectations of monetary rewards seem to be not relevant.

FLOSS-US data, however, state that firms' financial support for FLOSS projects increased over time (question 30 and 31), which means that opportunities to receive monetary rewards for FLOSS development increased. This can be related to the developers' expectations presented again in the FLOSS-US survey. When the survey was launched (January 28, 2003), most of the developers were already part of the community (85% of them first started participating in open source/free software development in 2001 or before). After at least one year spent working on FLOSS, they expected not only to work for business enterprises based on FLOSS, but also to leave the position of employees to acquire managerial roles or become autonomous consultants.

Thus, the importance of monetary rewards increases with agents' involvement in the community because *developers become aware of the actual earning possibilities they begin to face*.

A similar interpretation can be given to the results reported by Ghosh (2003b), where earnings opportunities are compared with the actual earnings obtained by each developer. While politically and socially motivated developers do not expect to get paid from their development (and in fact most of them do not earn a lot of money either directly nor indirectly, Ghosh, 2003b), and while "those who report career and monetary motives get what they want", it is interesting noticing that "although those with purely product-related motives [...] are less likely to earn money from FLOSS than other groups, they earn the most, on average" (Ghosh, 2003b, p. 19-20)[8]. So, as long as we talk

[8] Notice that the categories used by Ghosh (2003b) to construct the clusters of developers are very peculiar. As the author reports: "while many developers express social or community-related motives, only those who also express career concerns were included in the second category, while only those who also express political views were placed in

about own-use motivated developers, they face in fact *unexpected* earning opportunities which at the end result in a higher income.
The "economies of scope" embodied into the communitarian environment put under another light the growing importance of monetary rewards and career concerns. As for learning, their dynamic evolution should be conceptualized as the results of an unexpected "income effect", enabling developers to acquire a greater amount of "motivational goods".

3.5. Own-use incentives: conditio sine qua non or dynamic drivers?

The previous analysis clearly shows that own-use based motivations and learning have a relevant role at the beginning. On the contrary, when elasticity with respect to time is taken into account (table 2), it is easy to see that the change of their importance over time is limited. Thus they seem to be mainly *environmental conditions*, in the sense that they are a set of pre-determined preferences enabling the FLOSS community to exist. In other words, they provide developers with a substantial and constant "endowment" of motivation, but they are unable to increase their level of participation to the community over time. However, this is just a superficial observation. If developers are stimulated to participate in the community production of software to fulfill their exogenously determined necessities, they should experience a decrease in their interest when the software they needed is actually created and diffused. This is precisely the result found by Shah (2006). However, figure 1 shows that own-use motivations remain crucial over time. Thus, the "income effect" described before has to be at work also on this side of developers' motivations. In particular, it is plausible to think that interacting with other members developers discover new opportunities of consumption. They can recognize new possible adaptations of the software they are interested in, or new needs they were not aware of before. The result is that the community continually feeds the set of developers' needs. Again, the exposure to the community social environment has the property to make developers aware of (unexpected) opportunities.

3.6. The social and psychological dimensions of the community

Several theories have been applied to the FLOSS case in order to unfold the social and psychological side of developers' incentives (Bagozzi and Dholakia, 2006; Hertel *et al*., 2003). The comparison with societies based on gifts exchange (Raymond, 1998) or on peer regard (Dalle *et al*. 2004) has been used to describe the FLOSS community. It has been also recognized often as a community of practice or an epistemic community (Cohendet *et al*. 2000; Lin, 2004; Edwards, 2001). In a related work I tried to show that the social and psychological dimension of the relationship between the developer and the community can be related to the 'reflexive mechanisms' described by Giddens (1991). In particular the concept of 'reflexive identity' (Lindgren and Wåhlin, 2001; Rullani, 2006) is particularly useful in this context. The reflexive identity process is based on the continuous interaction between the debate underpinning the community social environment and members' identities. In particular, this process is very close to the one described by the community-of-practice theory (Wenger, 1998). The contradiction between the vision of the world the community embodies and its members' identities results in a certain amount of individual dissonance. The need to reduce the dissonance forces each developer to choose between internalizing the rules of the community and accumulating a certain level of conflict. In the latter case, depending on the extent of the conflict, the consequence can be a revolution, changing the vision of the world embodied by the community, or the splitting and/or dissolution of the community itself. In the first case, the social environment of the community gradually enters the psychological dimension of the individual, and makes her internalize the community rules, habits, and vision of the world (Kuran, 1989, 1995, 1998; Kirman and Teschl, 2006). It is easy to see how

---

the third category. The last category is necessarily small since it comprises those who expressed only product-related motives (participating in the developer community to improve a software product, say). Of course, there are many ways of forming these categories and how one forms them would change how they correlate to other variables".

this principle can be placed as the basis of the dynamics of developers' incentives, linking together the interaction typical of the FLOSS community and the social motivations of its members.

The same principle can also be related to "creativity", "fun" and the other categories of intrinsic motivations the literature has recognized as fundamental in software development (Luthiger, 2005; Osterloh and Rota, 2004; Bitzer *et al*., 2004; Lakhani and Wolf, 2005). The link is based on the idea that intrinsic motivation is enhanced by the exposure to a lively social environment, where new challenges are continuously elaborated and spread between members. In other words, a communicative community is able to produce the 'raw material' to constantly feed its members' intrinsic motivations.

4. Exposure to the community social environment and contribution: the hypothesis to test

4.1. The nature of the determinants of developers' contribution

The previous analysis has shown that the individual and the community engage in a relationship which becomes "thicker and thicker". Being exposed to the community social environment, developers acquire information on new job as well as new consumption opportunities the participation in the FLOSS construction offers them. At the same time they identify more precisely where the acquisition of a wider and deeper set of skills can lead them, and can better specify the direction of their involvement in the community activities. Moreover, the exposure to the community environment continually renews the set of challenges they can face to "have fun" or "feel creative" and triggers a series of social processes able to tie each developer to the community, to make her feel as a part of a social entity based on duties and rights, rules and psychological and social gains and punishments.

This evolution of developers' motivations is coupled to a precise dynamics of their involvement into the community activity. The FLOSS-US survey shows that "developers tend to play more roles in their OS/FS projects, and perhaps more leadership oriented roles, as their careers progress. […] Respondents play every role more commonly in their most recent projects than in their first projects" (David *et al*., 2003, p 33). The community reception of developer's contributions also changes. The longer the developer's experience in the community, the more the amount of her submitted code included in a project version, while the less the time for that code to be included. Moreover, consider that the average hours per week developers spend in contributing to FLOSS increases with the duration of their participation, and that a similar results have been found for the distributions of the maximum amount of hours per day and for the pattern of the hours spent coding during the most intense period of work. In the Boston Consulting Group survey (Lakhani *et al.*, 2002), 28.6% of the responders said "On average, I spend more time than when I first started", while "The time I spend has stayed about the same" was the answer of 14.3%, "On average, I spend less time than when I first started" of the 19.3%, and "My involvement is completely variable" of the 37.4%. Eventually, and likely as a consequence, the projects in which developers are involved reach maturity faster (again FLOSS-US).

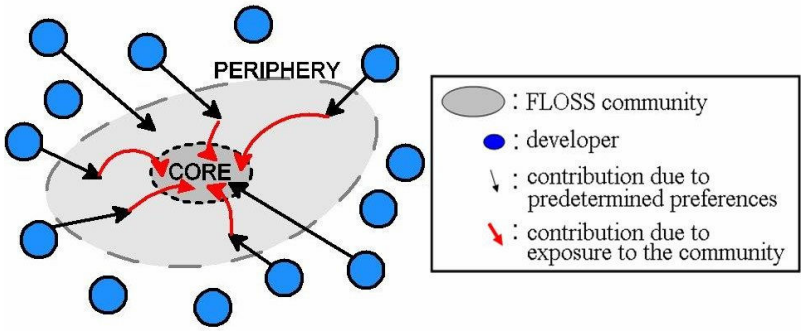
Thus, as the exposure to the community social environment increases and changes the set of members' motivations, developers enter the inner circles of the community and increase their participation, contribution and commitment.

4.2. The hypothesis

Notice that what has been described since now is a *dynamic* structure. Developers' initial distribution among the different typologies of incentives is taken as the baseline, so that the described relationship between developers' motivations and contribution has been derived *independently of developers' pre-determined preferences*. In other words, the messages we get from the previous analysis is that, whatever is the set of motivations leading the developer to join the

community and contribute to it in the first place (black arrows in figure 2), the exposure to the community social environment results in the amplification of certain motivational sets and eventually in an increase of the level of her participation and contribution (red arrows in figure 2).

Figure 2. Determinants of developers' contribution to the community.



In order to uncover the causal relationships implicit in the previous discussion, the preceding observation is expressed as a hypothesis to be tested:

Hypothesis: *independently of their pre-determined preferences, the more the developers' exposure to the FLOSS community social environment, the more their contribution to the community activities.*

Next section is aimed precisely at testing this statement.

5. Data and variables

5.1. The dataset

To test this hypothesis, I can rely on data provided by Sourceforge.net (*SF.net* henceforth), the largest repository for FLOSS projects, and relative to the activity of 544,669 developers along 840 days from September 2000 to December 2002.
Developed and maintained by VA Software (former VA Linux), one of the most successful companies acting in the FLOSS scenario, *SF.net* (http://soureceforge.net) is an on-line platform for software development enabling users and developers to coordinate their activities, and to host and distribute the code they produce. On *SF.net*, registered developers can contribute in a wide range of different forms. They can join the teams of existing projects or found their own projects and then recruit other developers, they can report bugs, send patches, signal features which could be developed, or, when they gain the access to the software basis of a project, directly change the code produced by other developers through the Concurrent Versioning System[9] (CVS). The platform provides also instruments to organize the team work, such as spaces where project members can list the jobs they ask help for, or for the distribution of on-line surveys or file releases, and tools to facilitate the communication among participants (forums, mailing lists, news reports, forms to manage support requests, etc.). Each one of these activity is recorded and stored in the dataset[10].

5.2. The sample

Several studies have shown that FLOSS production is characterized by a large number of "passive" lurkers (i.e. observers), registering to *SF.net* but not participating in any form. Most of these

[9] When a developer connects to the software basis of a project, the Concurrent Versions System synchronizes the code on the developer's computer and the code stored in *SF.net*, in order to avoid the emergence of different versions of the same program.
[10] Howison and Crowston (2004) and Comino *et al*. (2005) have identified some problems connected to the use of this dataset. The version I am using here is the same used in Giuri *et al*. (2005) and David and Rullani (2006), to which the reader can refer to understand how the aforementioned problems have been solved.

developers are likely not even lurkers, and visit the web site of the platform only rarely. They cannot be considered as part of the community. In order to construct a meaningful sample, only the those developers who were at least once part of a project team (even if only for few moments or submitting a project that was rejected by the *SF.net* staff) were selected. Moreover, I could follow developers' activity in detail only from September 2000, so I decided to exclude those individuals who registered earlier to the platform in order to be able to account for the whole history of each one of the included developer. The sample reduced then to 71,728 individuals.
A further reduction of the sample had to be performed in order to account for the properties of the employed estimation models, namely negative binomial and logistic regression models, both specified within a fixed effect panel structure. In order to use these models I have to rely only on developers granting a certain variability of the dependent variable over the time periods. If a developer does not satisfy this condition, her contribution to the estimate is irrelevant. Thus, these observations are simply dropped, and the sample size is then reduced to 14,497. Tables 6a-6b enable the comparison of means, standard deviations, minimum and maximum of both groups across the employed variables.
Eventually, consider that each developer's activity on *SF.net* was observed for 28 thirty-days long periods from September 2000 to December 2002. But the focus here is on the evolution of developers' participation from their entry in *SF.net* to the last observable period. Thus, in the analysis, the time dimension of each variable obtained this way was changed from "real months" to developers' periods of experience in *SF.net*. What is referred to as 'the first period' is not September 2000, but each developer's first 30 days after her registration to *SF.net*. Notice that, since the last period stored in the dataset is December 2002, the emerging panel structure is unbalanced.

5.3. The dependant variable

The level of developers' contribution can be captured by several variables. In the database, the main dimension of developers' activity, i.e. the number and the magnitude of the changes to the code basis of each project they have acted on, is missing. However, information on whether developers have founded a new project, and when, can be retrieved after having cleaned the dataset as explained in David and Rullani (2006).
Founding a new project is a costly investment for a developer. The time and the effort needed to prepare the proposal to be submitted to the *SF.net* maintainers are just the first step. If the project is accepted, the founder has to manage all the main aspects relative to the projects, and if it is able to attract other developers, she has to manage and direct the whole development activity. At the same time, it is a risky activity, because the borne costs are sunk costs, and if the project fails, or becomes too successful and can be managed only increasing the effort, or if it eventually becomes something different from what the founder wanted, these costs will not produce any gain. Thus, when a developer decides to found new projects, this can be seen as a signal of a high propensity to contribute.
And precisely because what is relevant here is developers' propensity to contribute to the community activity, I am not interested in the number of accepted or successful projects among those proposed by each developer. Instead, I will focus simply on the number of projects the developer has founded per period, whatever is their fate.

In addition to this measure, other proxies for developers' contribution can be built.
Each developer's role into existing projects can be proxied by the number of news and job requests she has posted. Posting the news relative to a specific project or uploading a list of tasks the development team asks help for are activities signaling the central role of the developer who undertakes them. Moreover, *SF.net* provides also a specific tool, the tracker system, to manage bug reports, patches, support and new feature requests. Developers belonging to a project are not likely to use these instruments to signal to the other members the problems they have found. They can

directly act on the software basis of the project, or simply embody the suggestions and the lines of code in messages sent directly to the other team members. What this variable captures, then, is mainly the developer's contribution to projects she does not belong to. These three variables can be coupled with the first one to represent the whole spectrum of activities a developer can undertake in *SF.net* as founder of a new projects, member of existing projects, or contributor of projects she does not belong to. Thus, in order to study developers' contribution to the community, two main variables are employed:

Table 3. Dependant variables.

Variable	Description
$FOUNDED\_PRJ_{it}$	Number of founded projects by developer *i* at time *t*.
$CONTRIBUT_{it}$	A dummy variable taking value 1 if $FOUNDED\_PRJ_{it}>0$ or if at least one of the following variables is positive at time *t* (i.e. it equals 1 if developer *i* has founded a new group, or posted job requests or news on a project website, or sent a contribution through the tracker system):
- $DEVEL\_JOBS_{it}$	Number of jobs required by developer *i* at time *t*.
- $DEVEL\_NEWS_{it}$	Number of messages posted by the developer *i* to the section "news" on a project's website at time *t*.
- $DEVEL\_TRACK_{it}$	Number of trackers (defined as "bugs", "patches" and "feature requests"[11]) posted by developer *i* to the tracker system of a project at time *t*.

5.4. The main regressors

The hypothesis to be tested highlights the positive effect of the community social environment developer *i* is exposed to on *i*'s contribution. What has to be captured by the main regressors, then, is the level of the community social environment *around* each developer and *independently* of its participation and communicative attitude.

The database provides data relative to the messages sent to each project's forums, and to the surveys promoted by a single project or by *SF.net* maintainers and answered by each developer. On this basis, the following variables have been built:

Table 4. Main regressors.

Variable	Description
$COLLEAGUE\_MSG_{it}$	Number of the projects developer *i* does NOT belong to to which her colleagues (member of *i*'s projects) have sent forum messages at time *t*.
$COLLEAGUE\_SURV_{it}$	Number of surveys answered by developer *i*'s colleagues (member of *i*'s projects) at time *t*.

These two variables have been chosen because they are likely correlated to the level of communication surrounding each developer. This assumption is based on the idea that they represent the propensity to communicate of those developers who work at the same projects of *i*, here called "colleagues"[12]. The more the colleagues' propensity to communicate, the more their use of forums and the higher the number of the surveys they answer to. Notice that, as I will discuss later, the variables have been built in such a way that they minimize the influence developer *i* can have on the environmental communication around her. This will help in solving the problem of endogeneity.

As table 7b shows, the correlation between the two variables is not so high as one could expect (0.3596 and significant at a 5% level). This is due the fact that they both capture the same phenomenon, i.e. developer *i*'s colleagues' communicative attitude, but in different forms. Forums allow for continuous communication, and can contain whatever type of messages. Surveys are

[11] 'Support Requests' was excluded, so that the new variable is much more effective in capturing only valuable contributions.

[12] For a wide description of the communication in a virtual community of innovative users see Frederiksen (2006) and, for the specific case of FLOSS, Krishnamurthy (2002) and Kuk (2006).

instead specific instruments used only in particular context to acquire precise information on a relevant topic (ranging from judgments on an on-going project to ideas about the FLOSS movement as a whole). Moreover, they allow only for a limited expression of developers' opinions. The difference between the two instruments can then be red as a difference in the nature of the communication they represent. On the one hand, forum messages can be assimilated to informal communication, where the timing, the level and the content of the interaction can vary according to each participant interests and wishes. On the other hand, surveys responses are much closer to formal communication, being structured in terms of the treated topics and of the timing of communication.

Eventually, notice that $COLLEAGUE\_MSG_{it}$ has been chosen after having evaluated the reliability of the data relative to forum messages. Forums are not always controlled by team members or by the *SF.net* staff, and sometimes they are subjected to disturbances or spam, or to double posting. The scan of a random sample of messages showed that these phenomena are limited and seem randomly distributed. However, since in the following analysis forum messages will be used to build one of the main regressors, the data should be as reliable as possible. Being impossible to scan each single message to check its validity, the strategy that has been used was to reduce the variability of each project's number of messages collapsing all the messages sent by each *i*'s colleague to a group *i* does not belong to into only one unit and then adding them up to obtain $COLLEAGUE\_MSG_{it}$. A variable relative to the number of messages has also been built and used to check for the robustness of the results obtained with $COLLEAGUE\_MSG_{it}$. It substantially confirms them, even if at a lower degree of stability, as expected.

5.5. The controls

The dataset allows capturing the main characteristics of developers' participation in *SF.net* and their evolution over time. This, coupled with the capability of the panel estimation to account for developers' time-invariant traits, is useful to "cut out" developer *i*'s characteristics from the effect of the exposure to the community social environment on her contribution. In particular, as I will explain in the next section, introducing these controls helps in avoiding endogeneity, so that the results of the regression become effective in testing the hypothesis.
The dimensions I can control for are reported in table 5, together with the description of the variables used in the regression.

Table 5. Controls.

Variable	Description	Dimension
$DEVEL\_MSGS_{it}$	Number of messages sent by developer *i* to the forums at time *t*.	*Developer's communicative attitude*
$DEVEL\_NEWS_{it}$*	Number of messages posted by the developer *i* to the section "news" on a project's website at time *t*.	*Developer's participation*
$DEVEL\_PRJ\_IN_{it}$	Number of projects developer *i* belongs to at time *t*.	*Developer's membership*
$DEVEL\_PRJ\_1_{it}$	A dummy variable taking value 1 if developer *i* belongs to project 1 (i.e. the *SF.net* maintenance project) at time *t*.	
$DEVEL\_EXPER_{it}$	t^{th} 30-day period from developer *i*'s entry in *SF.net*.	*Developer's experience*
$PRJ\_COLLEAGUES_{it}$	Number of colleagues, i.e. other members of the projects developer *i* belongs to at time *t* without repetition and without counting developer *i*.	*Characteristics of the projects the developer belongs to*
PRJ_AGE_{it}	Average registration period of the projects the developer *i* belongs to at time *t*.	
PRJ_GPL_{it}	Number of projects the developer *i* belongs to at time *t* that have chosen the General Public License as their first license.	
PRJ_LINKS_{it}	Each project's average number of links is computed as the mean number of projects participated by its members. This value is here averaged over the projects developer *i* belongs to at time *t*	

$PRJ\_FILE\_REL_{it}$	Average number of file releases of the projects developer *i* belongs to at time *t*.	*Performance of the projects the developer belongs to*

** The variable is dropped when considering $CONTRIBUT_{it}$ as the dependent variable.*

Notice that variables such as $PRJ\_COLLEAGUES_{it}$ and $DEVEL\_PRJ\_IN_{it}$ are crucial in normalizing the measures of the main regressors. The more the number of projects a developer is member of and/or the more the number of her colleagues, the more the surveys and the forum messages surrounding the developer. Even if this is precisely the phenomenon I want to address the analysis to, it must be "cleaned" from every possible scale effect not directly connected to the communication side of the community social environment. $PRJ\_COLLEAGUES_{it}$ and $DEVEL\_PRJ\_IN_{it}$ control precisely for this dimensions and point $COLLEAGUE\_MSG_{it}$ and $COLLEAGUE\_SURV_{it}$ directly to community communication.
Eventually, consider that the position of the developer in the network structure of the community is taken into account not only by $PRJ\_COLLEAGUES_{it}$ and $DEVEL\_PRJ\_IN_{it}$, but also through a specific control: PRJ_LINKS_{it}. To obtain this variable I first calculated for each project the average number of projects participated by its members (including *i*). Then I averaged this value over the projects *i* belongs to at time *t*. The result is the number of projects participated by the average user populating the average projects *i* belongs to. Thus, this variable, coupled with $PRJ\_COLLEAGUES_{it}$ and $DEVEL\_PRJ\_IN_{it}$, helps in controlling for developer *i*'s position in the network of the *SF.net* community.

Tables 6a-6b and 7a-7b presented in the appendix report the main statistics concerning the variables describe since now and their correlations matrixes.

6. Estimation strategy

6.1. The model

The first regressions are relative to the influence of the level of communication each developer *i* is exposed to (and proxied by $COLLEAGUE\_MSG_{it}$, $COLLEAGUE\_SURV_{it}$) on the number of new founded projects ($FOUNDED\_PRJ_{it}$). In this case, the final model presented in table 8 is a negative binomial regression model that exploits the panel structure of the data through fixed effects. This choice has been driven by the fact that the dependent variable $FOUNDED\_PRJ_{it}$ is a count variable and by a series of observations and tests selecting the negative binomial model with fixed effect as the best option.[13]
A second measure of developers' participation in the community is also applied. $CONTRIBUT_{it}$ is a dummy variable taking value 1 if the developer has at least founded a new project, posted a job request or a news message, or sent a contribution to the tracker system. A logistic model is then estimated, and its results compared to the previous analysis.[14] In particular, the individual effects

[13] To estimate the model I used the command `xtnbreg,fe` in STATA. This command includes the fixed effect in the overdispersion parameter. To account for a more intuitive definition of the fixed effect, I run the regressions also using a Poisson specification (command: `xtpoisson, fe`) where overdispersion is excluded by definition and the fixed effect is defined as usual. The results are identical in every respect. Going back to the negative binomial specification, to control at the same time for overdispersion and excess of zeros -but expecting much less precision because of the necessity to pool the data- I used a zero-inflate negative binomial model both with and without the clustering of the standard errors' robust estimation around each developer's group of observations (i.e. I used the command: `zinb` with and without the option `robust cluster(developer_id)`). The analysis confirms the results, even if at a lower level of precision and strength, as expected.

[14] $DEVEL\_NEWS_{it}$ is used here to "clean" the effect of the community social environment on the foundation activity of the developer from the influence her effort in other typologies of her activities. Notice that this variable will not be used in the estimation of the logit model. In this case, I am interested in capturing as much as possible dimensions of developers' activities, and thus $DEVEL\_NEWS_{it}$ enters directly the definition of the dependant variable $CONTRIBUT_{it}$. Thus it cannot be used as a regressor. In the negative binomial model also the other variables used to construct $CONTRIBUT_{it}$

are considered as fixed, given the fact that the Hausman test rejects the hypothesis that the regressors are not correlated to the errors.
Notice that all the excluded models (pooled, Poisson, random effects) have also been estimated to check for the robustness of the results.

6.2. Accounting for independence of developers' predetermined preferences

The hypothesis requires *independency of developers' predetermined preferences*. Thus, the equations must contain a control able to account for the basic traits of the developers. This control has been introduced through the use of fixed effects, where time-invariant characteristics are accounted for and absorbed by individual specific variables. The panel structure of the data, in other words, allows estimations focused on the dynamics of the process, so that predetermined preferences, i.e. the basic traits of the developers, are controlled for and do not affect the coefficients of the main regressors. This can be seen considering the following example. Imagine two developers with a different set of predetermined preferences. Developer *i* is very sensitive to the ideological side of the "free regime" of FLOSS, while developer *j* instead is more concerned with the satisfaction of her needs through the usage, and improvements, of FLOSS. This difference in the two developers' predetermined preferences is likely to generate a different mean level of contribution. This is what the fixed effect captures. As a consequence, the introduction of fixed effects allows an estimation "free" of the developers' predetermined preferences and instead focused on the *dynamics* of their contributions, i.e. on when, how -and especially as consequence of what- these contributions depart from the average levels due to their predetermined preferences.

6.3. Accounting for endogeneity

The introduction of the fixed effects moves the attention onto the dynamics of the process. In this realm, however, the problem of endogeneity emerges clearly. The main sources of endogeneity are basically three, i.e. increasingly active developers could increasingly 1) **attract** more communicative developers or 2) **stimulate** the communication around them or 3) **move** to projects composed by more communicative developers.

The "links" that have to be broken are then three:
1) *attract*: more *communicative* developers could be increasingly attracted by more *productive* developers. This could be due, for example, to their intention to establish themselves as crucial knots into the network of the community even if they are not very skilled or very committed to programming. In this case communicating more *and* moving closer to productive developers could be a suitable strategy to obtain a similar result. It is obvious that this is a source of endogeneity. The intrinsic rigidity of the dynamic structure of this mechanism can be used to rule it out. In order for communicative developers to detect the productive ones, the former have to observe first the activity of the latter. Moreover, if the increase in the level of communication surrounding a productive developer is just a consequence of her level of contribution, when other communicative developers join her, this should not affect her level of contribution. Given this, lagging the independent variables assures that this process cannot be at work, because in this case a positive coefficient of $COLLEAGUE\_MSG_{it}$ or $COLLEAGUE\_SURV_{it}$ represents a positive effect of today communicative environment on the *future* developers' level of activity. The endogeneity mechanism described above works the other way round, and can then be ruled out.

2) *stimulate*: it could be that increasingly productive developers become also more communicative, and thus increasingly stimulate the communication surrounding them. In order to take this into account, the main regressors have been built trying to reduce as much as possible the influence of

---

have been used as controls, namely $DEVEL\_JOBS_{it}$ and $DEVEL\_TRACK_{it}$, but they were less significant than $DEVEL\_NEWS_{it}$, i.e. less effective in detecting the action of developer *i*'s degree of activity on the number of new projects she founds.

developer *i* on the level of the surrounding communication. Moreover, the effect of the communication surrounding developers on the level of their activity has been "freed" by the effect of their own communicative attitude.
These effects have been obtained through the following techniques.
First of all, developer *i*'s colleagues' communicative attitude represented by $COLLEAGUE\_MSG_{it}$ has been obtained considering only the messages they have sent to projects developer *i* does *not* belong to. In a previous version of the paper this distinction was not made, and the variable contained all the messages sent by the developer's colleagues to every project they belonged to. In this case the endogeneity was given by the fact that in those projects where the developer and her colleagues worked together, the former could have been the stimulus, the cause, of the communication. Thus, the use of $COLLEAGUE\_MSG_{it}$ removes this source of endogeneity.
A similar argument can be stated for the second regressor: $COLLEAGUE\_SURV_{it}$. Surveys are produced and diffused by developers who have organizational roles in the community. They can be *SF.net* maintainers or the managers of specific projects. The public the surveys are directed to changes consequently. It can be the whole *SF.net* community or the developers of a specific project, respectively. The dataset does not allow for the detection of the source of each survey. However, the observation of other data relative to the activity on the *SF.net* website suggests that most of the surveys are carried out by *SF.net* maintains. Only few developers can directly affect *SF.net* maintainers' decisions. Given the high number of observations, the influence of this effect should be irrelevant. Moreover, $DEVEL\_PRJ\_1_{it}$ has been introduced to control precisely for this. The variable is a dummy variable taking value 1 if developer *i* belongs to project 1, the *SF.net* maintenance project. If developer *i* is part of it, she has direct access to the group of *SF.net* maintainers, can talk to them and may influence their decisions on the launch of new surveys. Introducing $DEVEL\_PRJ\_1_{it}$ in the regression controls for this effect. When the survey is carried out by projects leaders, endogeneity comes again into play, because developer *i* could have influenced the project leader's decision to create and diffuse a survey, or she can even be the project leader herself. As said, these cases should be only few. Moreover, even when this happens, the discussion on the opportunity to implement the survey and the questions constituting it are inevitably taken by a group of developers which is different from the set of respondents. Thus, endogeneity remains a problem only for the developers who participated in the production and diffusion of the surveys, and instead it takes precisely the causal direction I am interested in for those who received the survey. Since the number of the respondents who did not have any role in the construction of the survey should be much higher than the number of developers promoting it, the number of surveys answered by developer *i*'s colleagues should be basically independent of her influence.
Second, even if the peculiar construction of the regressors has excluded the direct influence of developer *i* on the social environment represented by the projects she belongs to, any possible residual influence of developer *i*'s change in her propensity to communicate on the environment has been ruled out introducing the variable $DEVEL\_MSG_{it}$, the number of messages sent by developer *i* to the forums of any project at time *t*.

3) *move*: the last source of endogeneity is due to the possibility that increasingly productive developers choose the most communicative projects.
To begin with, notice that if developers increase their contributions or launch new projects first, and then join other communicative projects (for example to advertise their own new projects) the lagged structure explained at point 1 assures that this is does not affect the estimates of the coefficients of $COLLEAGUE\_MSG_{it}$ and $COLLEAGUE\_SURV_{it}$. Our analysis, in fact, related *today* exposure to a communicative environment to *future* contributions.
Thus, we need to focus on the case in which developers whose propensity to contribute has begun to rise increase their *future* contribution thanks to the fact that in the *past* they have strategically chosen projects whose specific characteristics (related to communication) allowed them to "realize", to "put into practice", their augmented propensity to contribute. An efficient way to

control for this mechanism is to introduce the past levels of projects' characteristics that a) developers could consider "instruments" they can use to increase their contribution and that b) are correlated to the communicative level of the projects.
To identify the main controls to be introduced with this purpose, I can rely on the definition of developers' incentives described in table 1. Consider first that developers whose propensity to contribute is increases can search for more visible (i.e. central in the network), large and productive projects to give higher diffusion to their work and to send stronger and more visible signals into the *labor market*. Second, they could search for more productive projects because they look for collaborators to work on the software they need for their *own use*. They could also search for productive projects because they want to *learn* from other developers and *ask* questions on the specific problems they need to solve to increase their contribution. Eventually, their choices could be driven by more "*ideological*" reasons, so that they are willing to contribute mainly to projects which are committed to a specific view of the FLOSS phenomenon, or simply by their willingness to "have *fun*" programming with other productive developers.
As it is easy to see, all these mechanisms are positively correlated to the communicative level of the projects, and thus could be a possible source of endogeneity. Fortunately, the dataset is wide enough to account for all the characteristics of the projects upon which these mechanisms are based, as a comparison between table 5 and the previous description of the mechanisms can easily show. As illustrative examples, consider just the case of *learning* and *ideology*. Developers can search for productive projects because they want to learn form other developers. To control for this, I used the variables $PRJ\_FILE\_REL_{it}$, accounting for the productivity of the projects, $PRJ\_COLLEAGUES_{it}$, roughly counting the number members of the projects, and $DEVEL\_MSGS_{it}$, representing the number of messages the developer sends to the forums, which can also be considered a proxy for the developers' request of information. In the case of *ideology*, another control is introduced, PRJ_GPL_{it}, in order to use the number of projects the developer belongs to which have chosen as their first license the GPL as a proxy for the ideological level of the environment surrounding the developer.

To conclude, all the three sources of endogeneity detected at the beginning have been answered in the best possible way, so that the results can be consider reliable enough to actually verify or reject the hypothesis.

7. Results and robustness checks

Table 8 reports the results of the regressions in terms of incidence rate ratios (IRR) for the negative binomial regression models and odds ratios (OR) for the logistic regression models. All the results are larger than 1 (i.e. the effect is positive) and strongly significant. However, significance has to be judge not only in statistical terms, but also in economic terms.
To do that, consider first the nature of the phenomenon under analysis. The FLOSS production process is a self-organizing processes, so that it shares with the other systems in this class the characteristic of being *dissipative* (David and Rullani, 2006; Rullani, 2006). Such systems have to mobilize much more resources than what they really use to reach the outcome. This because the organization of the process is not centrally led, but each resource follows its own path through the process. Most of the resources are then lost during the process, which works as a "selection machinery" retaining only the resources which better fit the specificities of the process itself. Also the FLOSS production model can be conceived as a dissipative process because the community has to mobilize lots of developers to be able to involve just few of them. The huge amount of inactive projects and the skewness of the distributions of developers' productivity and participation in the discussions (Krishnamurthy, 2002) are clear manifestations of a system which needs to "burn" lots of resources (i.e. developers) to select in those who will eventually participate. In the context of the present paper this observation leads to the same conclusion as in David and Rullani (2006): the signals of activity have to be highlighted and magnified, because their magnitude is decreased by the large amount of resources lost in the process.

Keeping this in mind, consider now the Incidence Rate Ratios (IRR) and Odds Ratio (OR) shown in table 8. An increase in $COLLEAGUE\_MSG_{it}$ induces a change of 13.1% - 15.5% in the expected number of founded projects and of 3.9% - 8.2% in the odds of a contribution. A similar computation for the variable $COLLEAGUE\_SURV_{it}$ shows that the percentages reported above are 4.8% - 6.7% for $FOUNDED\_PRJ_{it}$ and 10.3% - 11% for $CONTRIBUT_{it}$. These estimated effects can then be considered fairly relevant. Interestingly enough, the two kinds of communication have different effects: while formal communication seems to be connected to a higher level of generic contribution, informal communication seems to be more effective in stimulating developers' willingness to create new projects.

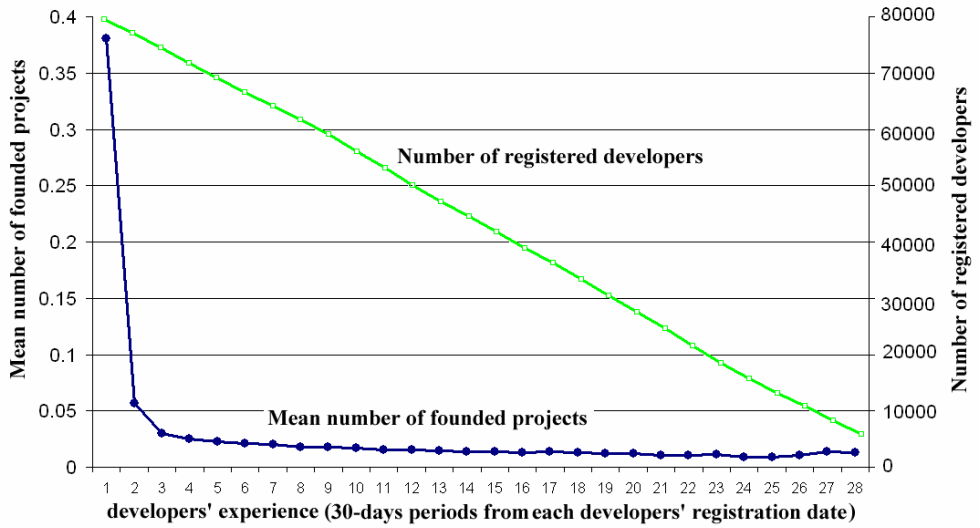
It is then possible to state that ***the exposure to the community social environment does increase developers' contribution*** *to the community activity* ***independently of their pre-determined preferences***.

Notice that this result is consistent with the positive relationship found by Kuk (2006) between knowledge sharing and the level of 'conversational interactivity', i.e. "The reciprocities, the interlinkages and penetrations among interdependent messages [posted on the KDE project developers' mailing list …]. The more diverse the views and ideas expressed in a discussion thread, the higher the level of conversational interactivity is" (Kuk, 2006, p. 1034). Thus, this construct can be directly linked to the richness of the community social environment studied here. The interesting result is that "conversational interactivity [is] positively related to knowledge sharing" (Kuk, 2006, p. 1034), confirming that a lively community environment increases developers' contribution.

A closer look at the models and the data allows a series of robustness checks.
As said, many different variables and models were used. Poisson, Negative binomial and zero-inflated negative binomial regression models, pooled regressions, random and fixed effects were combined and applied to a set of variables comprehending modifications of the ones described above. For example, $COLLEAGUE\_MSG_{it}$ has been substitute by the number of messages sent by *i*'s collogues to the projects *i* does not belong to, and by the number of messages received by the projects developer *i* belongs to.
In a previous version of the paper the effect of the last variable has been estimated also through OLS with a first difference structure and applying a wider set of controls. Using those variables, a logarithmic transformation of the regressors was also considered, in order to linearize the exponential structure of their effect on the number of founded projects.
The sample used to produce the results presented in table 8 has been selected considering the observations falling in the intersection between the sample needed to implement the negative binomial and the logistic regressions, both as panel estimations with fixed effect. In order to check for the robustness of the results, the same models were also estimated for the widest possible sample. For example, in the case of random effects the sample was enlarged to comprehend 69,474 individuals (968,999 observations), almost as wide as the whole population of 71,728 developers.
Other configurations of the models in table 8 were also checked. Projects' performance has been controlled using the number of CVS commits and downloads, and using a different definition of file releases. The results obtained without controlling for this dimensions were also taken into account. The definition of developers' contribution has been "enlarged" including all the trackers sent to the *SF.net* projects (including also 'Support Requests') into the variable $DEVEL\_TRACK_{it}$ and also "restricted" excluding $DEVEL\_JOBS_{it}$ and $DEVEL\_NEWS_{it}$ from $CONTRIBUT_{it}$ contemporarily and one at a time. Eventually, another series of regressions has been run considering different subsamples, i.e. including only developers' actions up to May 2002, or only those developers who registered after March 2001.
All the described robustness checks confirmed the hypothesis, and only in few cases the level of significance of some results was higher than 10% or the coefficient less relevant in economic terms.

In the last robustness checks described, the effect of the main regressors on the dependant variable was sometimes even stronger than the one reported in the paper.
A closer look to the behavior of the dependant variables created room for a further robustness check. Figure 3 reports the average number of founded projects by the 71,728 developers in the whole sample (left-hand axis) and the number of developers observed for each period (right-hand axis).

Figure 3. Developers' motivations at the beginning and now ordered by the difference between them.



As it possible to see, the first two periods of developers' experience in *SF.net* report a much higher number of founded projects. This means that there is a sort of "entry behavior". Participation as a contributor or a lurker (i.e. observer) is not a necessary condition to be registered. It is then likely that some developers decide to register to *SF.net* only when they decides to exploit *SF.net* facilities to create their own projects. After these two periods, the foundation activity becomes an almost constant activity (see David and Rullani, 2006; for a discussion of this process). In order not to induce any subjective bias into the analysis, the methodology applied to cope with this specific behavior of the data was introducing the time variable $DEVEL\_EXPER_{it}$ as a control into the regressions. A robustness check can then be performed excluding the first two periods form the analysis. Again, all the results are confirmed.

8. Discussion and conclusion

Science, Technology (Dasgupta and David, 1987, 1994), Hierarchies (Williamson, 1975) and Collective Inventions (Allen, 1983; Nuvolari, 2004, 2005) are provided with institutional structures able to solve the contradiction between the incentives to invest in knowledge production and its sequent diffusion (Arrow, 1962). On the contrary, there is no widespread agreement on the mechanisms at the basis of knowledge-based communities (David and Foray, 2003). The present paper has taken into account the institutional dimension of the Free/Libre/Open Source Software (FLOSS), one of the most interesting cases of knowledge-based communities, precisely to contribute to this debate.
In the literature the FLOSS community is perceived as able to solve the aforementioned contradiction mainly thanks to its members' exogenously determined preferences, counterbalancing the loss due to the "free" diffusion of their code. Moreover, these preferences have been usually considered as static. This creates room for the investigation of the dynamic side of developers' motivations. The first part of the paper tried to walk along this path, pointing out how the involved sets of incentives change their relative importance over time. The result is that a dynamic structure clearly emerged.
In the second part of the paper, this structure has been connected to the specificities of each

particular incentive. The aim was to find a possible principle running through all the incentives categories and sustaining developers' contribution *beyond* the level granted by their predetermined preferences. It has been found that the exposure to the social environment of the community activates a series of mechanisms which tie developers to the community activities, and enhances their contribution.
In the third part of the paper this hypothesis has been proved through a series of different regression models applied to data from SourceForge.net, where proxies for developers' contribution have been related to the level of the communication the community members where exposed to. Problems of endogeneity and robustness checks have also been taken into account. The results are in favor of the tested hypothesis.

This conclusion allows for a series of further observations.
First of all, the analysis has highlighted the necessity of conceiving developers' incentives as *dynamic* (Ghosh *et al*., 2002; Glott *et al*. 2004). The definition of the different "motivational fields" has to be translated according to their evolving structures. Career concerns and other monetary gains, own-use and learning, have to be imagined as exploration processes of *unexpected opportunities*. Developers gradually learn that their involvement in FLOSS production can generate a series of economic earnings. At the same time, the community stimulates their needs continuously, making them uncover new consumption opportunities and better specify the particular uses they can make of the knowledge and skills they acquire. This process of gradual involvement is particularly evident when social and psychological motivations are considered. Entering the community developers are engaged in the construction of shared identity, where social norms are discussed, challenged or internalized. Their very preferences become to a certain extent endogenous to this process, and their identity, vision of the world, and behavior change accordingly. This affects also their psychological dimension, and shapes the specific space where intrinsic motivations acts. *In nuce*, motivations have to be conceived as '*discovery processes*' relating the community to each one of its members and vice versa.

Second, the specific role of each motivation uncovered by the previous analysis sheds light on the mechanisms at work in different phases of developers' experience in the FLOSS environment. On the one hand, the results provided by the FLOSS surveys (Ghosh *et al*., 2002; David *et al*., 2003) have shown that the decision to *join* the FLOSS community is mainly due to learning, own-use and social and psychological incentives. On the other hand, the same data lead to the conclusion that developers' *retention* is mainly obtained through the gradual discovery of unexpected monetary gains and the acquisition of social norms, shared identity, community values, etc. brought about by the developers' exposure to the community social environment. Thus, the community itself is endowed with a mechanism (embodied in its social environment) enhancing cooperation and participation.

Eventually, combining the tested hypothesis with the previous discussion on motivations leads directly leads to the conclusion that the exposure to the community social environment is able to increase developers' participation because that environment is itself the *locus* where individuals discover unexpected extrinsic (consumption and monetary-related) or intrinsic (e.g. enjoyment) opportunities or participate in a social process combining mutual learning and the construction of a "thick" social space where identities, social ties and common rules becomes fundamental determinants of the behaviors. Thus, as long as the community is *alive*, it seems to have the capability to sustain developers' participation through the "supply" of new social spaces, challenges, experiments and opportunities. Keeping in mind that "self-maintenance of *organizational forms* and institutions […] are partly the result of directed (purposeful) action by the agents but also, partly, the unintentional outcome of the interplay of agent learning and collective interactions" (Dosi and Winter, 2000, p. 6), this result leads to a further confirmation of the

hypothesis affirming the *sustainability* of the FLOSS production process as a model for innovation (David and Rullani, 2006).

References

Allen R.C. (1983), *Collective invention*, Journal of Economic Behaviour and Organization 4 (1), 1–24.

Arrow K. (1962), *Economic Welfare and the Allocation of Resources for Invention*, in Nelson R.R. (Ed.), *The rate and direction of inventive activity. Economic and social factors*, Princeton University Press, Princeton.

Benussi L. (2005) *General Public License model of IPR: a localized technological knowledge approach*, FGA project – preliminary draft, University of Turin

Bitzer, J., Schrettl W., Schroder P. (2004), *Intrinsic Motivation in Software Development*, Free University of Berlin Discussion Paper 2004/19.

Bagozzi R.P, Dholakia U.M. (2006) *Open Source Software User Communities: A Study of Participation in Linux User Groups*, Management Science, Vol. 52, No. 7, July, p. 1099–1115.

Bonaccorsi A., Rossi C. (2006), *Comparing Motivations of Individual Programmers and Firms to Take Part in the Open Source Movement: From Community to Business,* Knowledge, Technology and Policy, Volume 18, Number 4, p. 40 - 64.

Cohendet P., Creplet F., Dupouët O. (2000), *Organisational innovation, communities of practice and epistemic communities: the case of Linux*, in Kirman A., Zimmermann J.B. (Ed), *Economics with Heterogeneous Interacting agents*. Springer, Berlin.

Comino S., Manenti F.M., Parisi M.L. (2005), *From Planning to Mature: on the Determinants of Open Source Take Off*, Discussion paper 2005-17, Università degli Studi di Trento.

Dalle J.-M., David P.A., (2005), *The Allocation of Software Development Resources in 'Open Source' Production Mode,* in J. Feller, et al. (eds), *Making Sense of the Bazaar: Perspectives on Open Source and Free Software*, Cambridge MA: MIT Press, 2005. Preprint available at: http://siepr.stanford.edu/papers/pdf/02-27.pdf.

Dalle J.-M., David P.A., Ghosh R.A., Wolak F.A. (2004), *Free & Open Source Software Developers and 'the Economy of Regard': Participation and Code-Signing in the Modules of the Linux Kernel*, presented at The Oxford Workshop on 'Libre Source' convened at the Oxford Internet Institute, 25-26th June.

Dasgupta P., David P.A. (1987), *Information Disclosure and the Economics of Science and Technology*. Ch. 16 in *Arrow and the Ascent of Modern Economic Theory*, in Feiwel G. (ed.), New York University Press, New York, p. 519-542.

Dasgupta P., David P.A. (1994), *Toward a new economics of science*, Research Policy, vol. 23 (5), p. 487-521.

David P.A., Foray D. (2003), *Economic fundamentals of the knowledge society*, Policy Futures in Education - An e-journal, Special Issue: "Education and the Knowledge Economy", 1 (1), January, at http://www-econ.stanford.edu/faculty/workp/swp02003.pdf.

David, P.A, Rullani F. (2006), *Launching Open Source software projects. How many entrepreneurs on the SourceForge.net platform?*, SIEPR Open Source Economics Project Working Paper, Stanford University (May). To be presented at 11th International J.A. Schumpeterian Society Conference "Innovation, competition and growth: Schumpeterian perspectives", 2006, June 22nd -24th, Sophia-Antipolis, France.

David P.A., Waterman A., Arora S. (2003), *The free/libre/open source software survey for 2003*, preliminary draft, September 2003, quoted with authors' permission, at http://www.stanford.edu/group/floss-us/report/FLOSS-US-Report.pdf.

Dosi, G. Winter, S. (2000), *Interpreting Economic Change: Evolution, Structures and Games*, LEM Working Paper 2000/08, July.

Edwards K. (2001), *Epistemic communities, situated learning and open source software development*, Department of Manufacturing Engineering and Management, Technical University of Denmark, at http://opensource.mit.edu/papers/kasperedwards-ec.pdf.

Franke N., von Hippel E. (2003), *Satisfying Heterogeneous User Needs via Innovation Toolkits: The Case of Apache Security Software*, Research Policy, Vol 32, No. 7, (July) p.1199-1215.

Frederiksen L. (2006), *User communication driving firm innovation: A communication patterns perspective on personal attributes and communication types in a user community. Lessons from the software music instrument industry*, presented at the DRUID Summer Conference 2006, June 18 - 20, Copenhagen, Denmark.

Gambardella, A. and Hall, B., 2006, *Proprietary vs Public Domain Licensing of Software and Research Products*, Research Policy, Vol. 35(*6*), pp.875-892

Ghosh R.A., Krieger B., Glott R., Robles G. (2002), *Free/Libre and Open Source Software. Part IV: Survey of Developers*, International Institute of Infonomics, Berlecom Research GmbH, at http://www.infonomics.nl/FLOSS/report/Final4.pdf

Ghosh R.A. and Glott R. (2005), *The Open Source Community as an environment for skills development and employment generation*, Proceedings of the European Academy of Management (EURAM) Conference, Munich, May 4-7. Related presentation available at: http://flossproject.org/papers/20051110/rishabGHOSH-icos05.pdf.

Ghosh R.A. (1998), *Cooking Pot Markets: An Economic Model for the Trade in Free Goods and Services on the Internet*, First Monday, vol.3 number 3, March, at http://www.firstmonday.dk/issues/issue3_3/ghosh/index.html;

Ghosh R.A. (2003a), *Licence fees and GDP per capita: The case for open source in developing countries*, First Monday, volume 8, number 12 (December 2003), at http://firstmonday.org/issues/issue8_12/ghosh/index.html

Ghosh R.A. (2003b), *Understanding Free Software Developers: Findings from the FLOSS Study*, presented at the Conference "New Models of Software Development", Harvard Business School/MIT Sloan Free/Open Source Software, June 19-20.

Giddens P. (1991), *Modernity and Self-identity*, Cambridge, Polity Press.

Giuri P., Ploner M., Rullani F., Torrisi S. (2005), *Skills, Division of Labor and performance in collective inventions. Evidence from the open source software*, paper presented at the EARIE Conference, Porto, Portugal, 1-4 September.

Giuri P., Rocchetti G., Torrisi S. (2002), *Open source software: from open science to new marketing models an enquiry into the economics and management of open source software*, LEM Working Paper 2002/23, July.

Glott, R., Ghosh R.A., Krieger B. (2004), *Motivations of free/libre and open source developers*, International Infonomics Institute Working Paper, University of Maastricht (May). See the presentation by R. Glott, *Towards integration of research approaches to FLOSS communities*, Oxford Workshop on Libre Software (OWLS), Oxford Internet Institute, June 25-26, 2004, at: www.oii.ox.ac.uk/fiveowlsgohoot/postevent/owls-presentation_r.glott.pdf.

Hertel G., Niedner S., Herrmann S. (2003). *Motivation of software developers in open source projects: an internet-based survey of contributors to the Linux kernel*, Research Policy, 32, p. 1159-1177.

Himanen P., Torvalds L., Castells M., (2001), *The hacker ethic and the spirit of the information age*, Secker & Warburg, London;

Howison, J., Crowston, K. (2004), *The perils and pitfalls of mining sourceforge*, in *Proceedings of Workshop on Mining Software Repositories at the International Conference on Software Engineering ICSE*.

Jeppesen, L.B. and Frederiksen, L. (2006), *Why firm-established user communities work for innovation? The personal attributes of innovative users in the case of computer-controlled music instruments*, Organization Science, 17(1), p.45-64.

Kirman A., Teschl (2006), *Searching for identity in the capability space*, Journal of Economic Methodology 13:3, 299–325, September.

Krishnamurthy S. (2002), *Cave or Community?: An Empirical Examination of 100 Mature Open Source Projects*, First Monday, volume 7, number 6, June, at http://firstmonday.org/issues/issue7_6/krishnamurthy/index.html

Kuk G. (2006), *Strategic Interaction and Knowledge Sharing in the KDE Developer Mailing List*, Management Science, Vol. 52, No. 7, July 2006, p. 1031–1042

Kuran T. (1989), *Sparks and prairie fires: A theory of unanticipated political revolution*, Public Choice 61(1), p. 41-74.

Kuran T. (1995), *The inevitability of future revolutionary surprises*, American Journal of Sociology, 100, May, p. 1528-51.

Kuran T. (1998) *Social mechanisms of dissonance reduction*, in Hedström P., Swedberg R. (eds.), *Social Mechanisms: An Analytical Approach to Social Theory*, Cambridge University Press, 1998, pp. 147-71.

Lakhani K.R., von Hippel E. (2003), *How open source software works: "free" developer-to-developer assistance*, Research Policy, 32, p. 923-943.

Lakhani K.R., Wolf R.G. (2005), *Why hackers do what they do: understanding motivations and effort in Free/Open Source Software Projects*, in Feller J., Fitzgerald B., Hissam S., Lakhani K.R., eds., *Perspectives on Free and Open Source Software*, MIT Press.

Lakhani K.R., Wolf R., Bates J., Di Bona C. (2002), *Hacker Survey (Release 0.73)*, The Boston Consulting Group, presented at the "O'Reilly Open Source Conference", July 24, 2002. at http://www.osdn.com/bcg/.

Lerner J., Tirole J. (2002), *Some simple economics of Open Source*, The Journal of Industrial Economics, vol. L number 2, p. 197-234.

Lerner J., Tirole J. (2005), *The Scope of Open Source Licensing*, Journal of Law, Economics, and Organization, 21, 20-56.

Lindgren, M. and N. Wåhlin, (2001), *Identity construction among boundary-crossing individuals*, Scandinavian Journal of Management, Vol 17:3, p 357-377.

Lin Y. (2004), *Contextualising knowledge–making in Linux developer groups*, First Monday, volume 9, number 11, November, at http://firstmonday.org/issues/issue9_11/lin/index.html.

Luthiger B. (2005), *Fun and software development*, in Scotto M., Succi G. (eds.), *Proceedings of the First International Conference on Open Source Systems*, Genova, 11-15 July, p. 273-278.

Mateos Garcia J., Steinmueller W. E. (2003), *The Open Source Way of Working: A New Paradigm for the Division of Labour in Software Development?*, Brighton: SPRU, Science and Technology Policy Research, Open Source Movement Research, INK, Working Paper number 1(January), at http://www.sussex.ac.uk/spru/publications/imprint/sewps/sewp92/sewp92.pdf.

Narduzzo A., Rossi A. (2004), *The Role of Modularity in Free/Open Source Software Development*, Ch. 10 in Koch S. (ed.), *Free/Open Source Software Development*. Hershey, Idea group.

Nuvolari A. (2004), *Collective invention during the British Industrial Revolution: the case of the Cornish pumping en*gine, Cambridge Journal of Economics, Oxford University Press, vol. 28(3), pages 347-363, May.

Nuvolari A. (2005), *Open Source Software Development: Some Historical Perspectives*, First Monday, volume 10, number 10 (October), at http://firstmonday.org/issues/issue10_10/nuvolari/index.html

Osterloh M., Rota S. G. (2004) *Open Source Software Development - Just Another Case of Collective Invention?*, March. Available from the SSRN at http://ssrn.com/abstract=561744.

Raymond E. (1998), *Homesteading the Noosphere*, First Monday, vol. 3, number 10, October, at http://www.firstmonday.dk/issues/issue3_10/raymond/#d12

Roberts J.A., Hann I.-H., Slaughter S.A. (2006), *Understanding the Motivations, Participation, and Performance of Open Source Software Developers: A Longitudinal Study of the Apache Projects*, Management Science, Vol. 52, No. 7, July, pp. 984–999

Rullani F. (2006), *The debate and the community. 'Reflexive identity' in the FLOSS community*, LEM Working Paper N. 2005/18, November 2006.

Shah S. (2006), *Motivation, Governance & the Viability of Hybrid Forms in Open Source Software Development*, Management Science, Vol. 52, No. 7, July, p. 1000-1014.

Torvalds L., Diamond D. (2001), *Just for Fun: The Story of an Accidental Revolutionary*, Texere, New York, NY.

von Hippel, E. (2002), *Open Source Projects as Horizontal Innovation Networks - By and For Users*, MIT Sloan Working Paper No. 4366-02, June.

von Hippel, E. von Krogh G. (2003), *Open Source Software and the 'Private-Collective' Innovation Model: Issues for Organization Science*, Organization Science, vol. 14, number 2, March–April.

von Krogh G., Haefliger S., Spaeth S. (2003a), *Collective Action and Communal Resources in Open Source Software Development: The Case of Freenet*, Presented at Academy of Management, 2003, Seattle.

von Krogh G., Spaeth S., Lakhani K. R. (2003b), *Community, joining, and specialization in open source software innovation: a case study*, Research Policy, 32, p. 1217-1241.

Weber S. (2000), *The political economy of open source software*, BRIE working paper n. 140, June.

Weber S. (2004), *The Success of Open Source*, Harvard University Press, Cambridge, USA.

Wenger E. (1998), *Community of practice; learning as a social system*, Systems Thinker, June.

Williamson O. (1975) *Markets and Hierarchies: Analysis and Antitrust Implications*. Free Press, New York, NY.

Zeitlyn D. (2003), *Gift economies in the development of open source software: Anthropological reflections*. Research Policy 32(7), p. 1287–1291.

Appendix

Table 6a. Descriptive Statistics.

Variable	*Aggregation level*	*Sample (71,728 developers)*					*Estimation Sample (14497 devel.)*				
		Min	*Max*	*Mean*	*St. dev.*	*Obs.*	*Min*	*Max*	*Mean*	*St. dev.*	*Obs.*
	overall (N)	0	14	0.045	0.230	1040727	0	14	0.092	0.328	244509
$FOUNDED\_PRJ_{it}$	*between (n)*				0.160	71728				0.116	14497
	within (T)				0.217					0.318	
	overall (N)	0	1	0.076	0.264	1040727	0	1	0.141	0.348	244509
$CONTRIBUT_{it}$	*between (n)*				0.177	71728				0.148	14497
	within (T)				0.238					0.323	
	overall (N)	0	358	0.143	1.934	1040727	0	356	0.213	2.185	244509
$DEVEL\_MSGS_{it}$	*between (n)*				1.362	71728				1.388	14497
	within (T)				1.496					1.821	
	overall (N)	0	38	0.039	0.349	1040727	0	38	0.088	0.530	244509
$DEVEL\_NEWS_{it}$	*between (n)*				0.232	71728				0.269	14497
	within (T)				0.302					0.470	
	overall (N)	0	53	0.961	0.807	1040727	0	31	1.152	1.180	244509
$DEVEL\_PRJ\_IN_{it}$	*between (n)*				0.643	71728				0.946	14497
	within (T)				0.408					0.646	
	overall (N)	0	1	0.000	0.013	1040727	0	1	0.000	0.016	244509
$DEVEL\_PRJ\_1_{it}$	*between (n)*				0.012	71728				0.014	14497
	within (T)				0.005					0.007	
	overall (N)	1	28	14.500	8.078	2008384	1	27	10.252	6.502	244509
$DEVEL\_EXPER_{it}$	*between (n)*				0.000	71728				3.335	14497
	within (T)				8.078					5.855	

Table 6b. Descriptive Statistics.

Variable	*Aggregation level*	*Sample (71,728 developers)*					*Estimation Sample (14,497 devel.)*				
		Min	*Max*	*Mean*	*St. dev.*	*Obs.*	*Min*	*Max*	*Mean*	*St. dev.*	*Obs.*
$COLLEAGUE\_MSG_{it}$	*overall (N)*	0	31	0.122	0.566	1040727	0	17	0.092	0.512	244509
	between (n)				0.458	71728				0.326	14497
	within (T)				0.436					0.399	
$COLLEAGUE\_SURV_{it}$	*overall (N)*	0	60	0.122	0.780	1040727	0	35	0.088	0.616	244509
	between (n)				0.474	71728				0.291	14497
	within (T)				0.709					0.555	
$PRJ\_COLLEAGUES_{it}$	*overall (N)*	0	382	4.205	9.430	1040727	0	190	2.653	8.225	244509
	between (n)				8.643	71728				6.618	14497
	within (T)				4.909					4.786	
PRJ_AGE_{it}	*overall (N)*	0	39	16.099	10.904	1040727	0	38	15.610	11.432	244509
	between (n)				11.109	71728				8.489	14497
	within (T)				5.926					8.698	
PRJ_GPL_{it}	*overall (N)*	0	40	0.679	0.731	1040727	0	20	0.818	0.976	244509
	between (n)				0.633	71728				0.816	14497
	within (T)				0.321					0.504	
PRJ_LINKS_{it}	*overall (N)*	0	27	1.155	0.959	1040727	0	27	1.162	1.110	244509
	between (n)				0.822	71728				0.909	14497
	within (T)				0.489					0.626	
$PRJ\_FILE\_REL_{it}$	*overall (N)*	0	118	0.151	0.868	1040727	0	118	0.124	0.702	244509
	between (n)				0.689	71728				0.346	14497
	within (T)				0.736					0.631	

Table 7a. Correlation Matrix.

#	Variable								
[1]	$FOUNDED\_PRJ_{it}$	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
[2]	$CONTRIBUT_{it}$	0.6890*	1						
[3]	$PRJ\_COLLEAGUES_{it}$	-0.0545*	-0.0117*	1					
[4]	PRJ_AGE_{it}	0.1192*	0.1368*	0.1106*	1				
[5]	PRJ_GPL_{it}	0.0705*	0.0858*	0.2036*	0.4104*	1			
[6]	PRJ_LINKS_{it}	0.0280*	0.0635*	0.3193*	0.4744*	0.5225*	1		
[7]	$PRJ\_FILE\_REL_{it}$	0.0481*	0.1162*	0.1584*	0.0819*	0.0394*	0.0883*	1	
[8]	$DEVEL\_MSGS_{it}$	0.0259*	0.1090*	0.0401*	0.0268*	0.0286*	0.0248*	0.0536*	1
[9]	$DEVEL\_NEWS_{it}$	0.1657*	0.3902*	-0.0015	0.0831*	0.0726*	0.0635*	0.1572*	0.1410*
[10]	$DEVEL\_PRJ\_IN_{it}$	0.0863*	0.1221*	0.3195*	0.5186*	0.7231*	0.7469*	0.0604*	0.0355*
[11]	$DEVEL\_PRJ\_1_{it}$	0.0024*	0.0040*	0.0343*	-0.0083*	-0.0003	0.0707*	-0.0015	0.0009
[12]	$DEVEL\_EXPER_{it}$	-0.1771*	-0.1605*	0.0448*	-0.0750*	0.1190*	0.1445*	-0.0528*	-0.0191*
[13]	$COLLEAGUE\_MSG_{it}$	-0.0110*	0.0234*	0.5074*	0.0622*	0.1361*	0.2404*	0.1167*	0.0664*
[14]	$COLLEAGUE\_SURV_{it}$	0.0202*	0.0454*	0.3379*	0.0817*	0.0859*	0.0972*	0.0966*	0.1007*

* *Significant at 5%.*

Table 7b. Correlation Matrix.

#	Variable					
[9]	$DEVEL\_NEWS_{it}$	[9]	[10]	[11]	[12]	[13]
[10]	$DEVEL\_PRJ\_IN_{it}$	0.0978*	1			
[11]	$DEVEL\_PRJ\_1_{it}$	0.0003	0.0242*	1		
[12]	$DEVEL\_EXPER_{it}$	-0.0441*	0.1629*	-0.0015	1	
[13]	$COLLEAGUE\_MSG_{it}$	0.0261*	0.1992*	0.0789*	-0.0176*	1
[14]	$COLLEAGUE\_SURV_{it}$	0.0451*	0.1103*	0.0186*	-0.0957*	0.3596*

* *Significant at 5%.*

Table 8. Negative Binomial and Logit regression models (fixed effect).

	Negative Binomial – FE			Logit – FE		
	FOUNDED PRJ_{it} (Incidence Rates Ratio)			*$CONTRIBUT_{it}$ (Odds Ratio)*		
$COLLEAGUE\_MSG_{i(t-1)}$	1.131 (0.024)***	1.155 (0.024)***		1.039 (0.017)**	1.082 (0.017)***	
$COLLEAGUE\_SURV_{i(t-1)}$	1.048 (0.013)***		1.067 (0.012)***	1.103 (0.011)***		1.110 (0.011)***
$DEVEL\_NEWS_{i(t-1)}$	1.085 (0.016)***	1.092 (0.016)***	1.087 (0.016)***			
$DEVEL\_MSGS_{i(t-1)}$	1.001 (0.004)	1.002 (0.004)	1.003 (0.004)	1.039 (0.003)***	1.042 (0.003)***	1.039 (0.003)***
$DEVEL\_PRJ\_IN_{i(t-1)}$	0.472 (0.015)***	0.470 (0.015)***	0.470 (0.015)***	0.763 (0.018)***	0.760 (0.018)***	0.761 (0.018)***
$DEVEL\_PRJ\_1_{i(t-1)}$	7.586 (6.534)**	7.152 (6.166)**	11.427 (9.804)***	28.243 (31.503)***	25.907 (28.891)***	32.152 (35.806)***
$DEVEL\_EXPER_{i(t-1)}$	1.052 (0.002)***	1.051 (0.002)***	1.051 (0.002)***	1.009 (0.001)***	1.008 (0.001)***	1.009 (0.001)***
$PRJ\_COLLEAGUES_{i(t-1)}$	1.024 (0.002)***	1.025 (0.002)***	1.029 (0.002)***	1.010 (0.001)***	1.012 (0.001)***	1.011 (0.001)***
$PRJ\_AGE_{i(t-1)}$	0.915 (0.001)***	0.915 (0.001)***	0.915 (0.001)***	0.969 (0.001)***	0.970 (0.001)***	0.969 (0.001)***
$PRJ\_GPL_{i(t-1)}$	0.802 (0.023)***	0.803 (0.023)***	0.802 (0.023)***	0.812 (0.018)***	0.816 (0.018)***	0.812 (0.018)***
$PRJ\_LINKS_{i(t-1)}$	1.196 (0.030)***	1.195 (0.030)***	1.206 (0.030)***	0.991 (0.019)	0.990 (0.019)	0.994 (0.019)
$PRJ\_FILE\_REL_{i(t-1)}$	1.061 (0.010)***	1.060 (0.010)***	1.061 (0.010)***	1.217 (0.011)***	1.221 (0.011)***	1.218 (0.011)***
Log Likelihood	-41371.808	-41378.421	-41386.907	-62409.652	-62457.345	-62412.383
Wald χ^2	11373.81	11363.56	11345.13			
LR χ^2				6135.07	6039.69	6129.61
Prob > $\chi 2$	0.000	0.000	0.000	0.000	0.000	0.000
Observations (groups)	244,509 (14,497)	244,509 (14,497)	244,509 (14,497)	244,509 (14,497)	244,509 (14,497)	244,509 (14,497)

* *Significant at 10%.*
** *Significant at 5%.*
*** *Significant at 1%.*