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

Skills, Division of Labor and Performance in Collective Inventions. Evidence from the Open Source Software

Paola GIURI*
Matteo PLONER*
Francesco RULLANI*
Salvatore TORRISI**

*LEM, Sant'Anna School of Advanced Studies, Pisa, Italy
**Department of Management, University of Bologna, Italy

2004/19 **July 2006**

SKILLS, DIVISION OF LABOR AND PERFORMANCE IN COLLECTIVE INVENTIONS. EVIDENCE FROM THE OPEN SOURCE SOFTWARE

Paola Giuri*, Matteo Ploner*, Francesco Rullani*, Salvatore Torrisi*°

* Laboratory of Economics and Management
Sant'Anna School of Advanced Studies
P.zza Martiri della Libertà, 33, 56127 Pisa, Italy
Tel. ++39-050-883359, Fax ++39-050-883344
e-mail: giuri@sssup.it, ploner@sssup.it, rullani@sssup.it

° *Corresponding author*: Department of Management
University of Bologna, Via Capo di Lucca, 34 40126 Bologna
Tel. ++39-051-2098061; Fax ++39-051-246411
e-mail: torrisi@unibo.it.

Abstract

This paper investigates the role of skills and the division of labor among participants in collective inventions. Our analysis draws on a large sample of projects registered at Sourceforge.net, the world's largest incubator of open source software activity. We test the hypothesis that the level of skills of participants and their skill variety are important for project performance. Skill heterogeneity across participants is in line with two fundamental organizational features of the open source development model: team work and modular design. We also explore the hypothesis whether the level of modularization of project activities is an important predictor of performance. Our econometric estimations show that both skill level and skill heterogeneity positively affect projects' survival and performances. However, the impact of skill diversity is non linear. Design modularity is also positively associated with the performance of the project.

Keywords: software, technological innovation, human capital, modularity
JEL classification: L86 O32, J24

Acknowledgements: we thank Gaia Rocchetti for her precious collaboration in the early stages of our research project and the administrators of SourceForge.net for providing us with their data on open source software projects and individual contributors. We also thank a sample of SourceForge developers for helping us to interpret some key variables in the dataset. We thank Bruno Cassiman, Paul David, Neil Gandal, Walter Garcia, Rishab Ghosh, and Gregorio Robles for useful comments to earlier drafts of the paper. We also thank the participants to workshops and conferences in Toulouse, Barcelona, Munich and Padova for comments and suggestions. We acknowledge the financial support of the project "Economic Change: the micro foundations of institutional and organisational change: EconChange", CE, DGXII, FP V.

1. Introduction

Collective inventions and informal know-how trade among profit-seeking individuals and organizations have become popular in the economic literature since the seminal paper of Robert Allen (1983) on the iron district of Cleveland in the XIX century. More recently, collective inventions have come to the forefront of economists' attention because of the diffusion of open source software (OSS hereafter). Most studies have attempted to explain why a growing number of independent developers ('hackers') voluntarily disclosure their inventions (Lerner and Tirole, 2002; Harhoff et al. 2003). Several theoretical works seek to understand the motivations for disclosure of the source code, the social norms and the patterns of collaboration among distributed developers, and the implications for efficiency and social welfare (e.g., Lerner and Tirole 2002; von Hippel 2001; Dalle and David 2005). To some scholars OSS is an example of 'private provision of public goods' (Johnson, 2002).

What are the incentives that lead hackers to freely reveal information in this particular case? Why free riding does not lead to under provision of the public good?

Moreover, recent empirical studies and evidence reported in this paper reveal that the distribution of the activity levels across OSS projects is very skewed but little is known about the determinants of the projects' survival and performances.

Two explanations are particularly useful in the context of this paper. The first is rooted into the team production theory. This theory predicts that complementarities in production and information asymmetry (i.e., the difficulty to observe the contribution to the total outcome of individual agents) induce free riding and adverse selection (Alchian and Demsetz, 1972). The problem of free-riding is moderated by the expectation of repeated interactions among participants and by peer pressure which can be particularly strong when team participants can monitor and punish their peers (shame and social punishment) (Kandel and Lazear, 1992). Moreover, team production requires collaborative skills, that is communication ability (people skills), leadership, and flexibility or ability to carry out multiple tasks. These skills provide teams with an additional input to traditional inputs (capital and specialized skills) and therefore have a positive impact on productivity. Collaborative skills also favor the discovery of 'new ways to assign, organize and perhaps alter tasks to produce more efficiently' (Hamilton et al., 2003, p. 470). Finally, heterogeneity among participants favours mutual learning and intra-team bargaining, creating opportunities for non monetary benefits such as a stimulating working environment, peer recognition and decisional authority (Hamilton et al., 2003).

The second line of explanation is associated with a key characteristic of the modern organization of production and design, i.e. modularity, which has been defined as a strategy for 'building a complex product or process from smaller subsystems that can be designed independently' (Baldwin and Clark, 1997). In modular production models not only the process but also the value generated by each module can be separated from the total outcome and therefore the marginal contribution of the subsystem can be measured. Software design, especially OSS design, relies on a modular approach (Baldwin and Clark, 2003). OSS projects share two important qualities. First, spatially dispersed participants contribute to a project on a voluntary basis. These projects are then an example of 'virtual' communities of practice or virtual innovative teams. Second, project teams make use of modularity and tools like CVS (concurrent versioning system) software and mailing lists to coordinate their efforts. This also increases peer monitoring and keeps opportunistic behaviour under control. Moreover, modularity allows for experimentation and innovation (Baldwin and Clark, 2003).

Our paper draws on these two streams of the literature to examine the activity of open source teams measured by different indicators – bugs and patches fixed, new feature requests completed, new file releases and CVS commits (changes made to the project's source code). This activity overall can be viewed as a cumulative, sequential innovation process carried out by different independent inventors.

In this setting we analyze the effects of members average skill level and skill heterogeneity on projects' survival and performance. As discussed later, skill heterogeneity across participants is in line with team production and modular design. Skill heterogeneity is expected to positively affect project performance in teams of open software developers for two reasons. First, individual participants have to be flexible as they must be prepared to carry out multiple tasks. Second, open source participants may have different levels of commitment to specific projects. In particular, we can distinguish core developers, who are highly committed and presumably highly experienced people, from the wider community of contributors, who occasionally participate in problem solving by supplying patches, and end users, who normally report bugs. It is likely then that the level and composition of skills varies across different categories of participants. This makes the analysis of skills heterogeneity relevant for our purposes.

Moreover, we posit that the level of design modularization or division of tasks at the project level is another important predictor of observable differences in performance across open source projects. As we discuss later, there are substantial differences across software projects in terms of size (number of registered participants). It is likely that larger projects

have a higher degree of division of labor among participants as compared with smaller projects. Moreover, there may be differences in the division of labor across projects that are independent from project size. Different core developers or project top members may have a different ability to identify problems and organize problem solving activities by separating a complex system into simpler tasks. Or, there may be different opportunities for task partitioning that depend on the nature of the problem to be solved, including complexity or the degree of interdependence between project modules. Therefore, we expect some degree of variance across projects which does not completely reflect their different size. We also expect that projects which, ceteris paribus, pursue higher levels of modularity outperform other projects. This is because, as mentioned before, modularity attenuates the problem of free riding and favors mutual learning between team members and innovation.

This paper provides a novel empirical contribution to the literature on the economics of collective inventions.

Our contribution is twofold. First, unlike many previous works that have focused on one or few open source projects, we provide an empirical investigation based on a large sample of OSS projects. To our knowledge, this is one of the first attempts to provide a systematic empirical overview of multiple dimensions of projects hosted by SF.net, one of the largest repository of OSS activity. Second, we explore the determinants of performance in a cross-section of source projects with the aim of understanding in particular the role of skill level and composition, and a key organizational dimension that is modularity.

The paper is organized as follows. Section 2 discusses the theoretical background. Section 3 presents the data while Section 4 illustrates the methodology for estimating the effects of skills and modularity on project performance. Section 5 analyzes the empirical results and Section 6 concludes.

2. Theoretical background

OSS is an example of collective invention. OSS developers participate in one or more projects with various degrees of commitment. For instance, members of the Linux community can download the source code, make modifications (further development of lines of code, debugging, etc.) and post it to the responsible of the project. The project leader, along with core developers, examines the proposed modifications and releases a new version of the program to the community. Open source products then evolve in a cumulative, incremental way and the marginal contribution of individual users-developers is evaluated

and acknowledged by project administrators. This process of collective, sequential innovation takes place within an institutional setting based on contractual rules that aim to protect intellectual property rights according to the 'copyleft' principle. The latter defines the right of early inventors to be cited in future inventions that draw on their contribution as well as the right of subsequent inventors to freely access to the source code of previous inventions (Di Bona et al., 1999; Raymond, 1999). Participants voluntarily reveal their inventions to the OSS community for various motivations examined by the literature (see, for instance, von Hippel 2001; Lakhani and von Hippel 2000; Lerner and Tirole 2002).

The active participation of expert users in inventive activity dates back to the beginnings of the computer industry. Users of early electronic computers in the 1940s and 1950s were mainly scientists and engineers with a good knowledge of computer systems and programming capabilities (Steinmueller, 1996). This community of software developers and users shared a common set of rules and visions of technological trajectories developed through continuous personal, informal contacts (Torrisi, 1998). However, over time a specialized software industry emerged and the vast majority of packaged software is now developed by software firms that distribute their products on the basis of licensing contracts that do not allow users to gain access to the program's source code.

On the contrary, OSS offers expert developers the opportunity to participate in an innovation network which shares several features with the communities of users of the early age of computing and other user-centered innovation networks such as those analyzed by Allen (1983) and von Hippel (1988).

Our paper focuses on two dimensions of collective inventions:

i) the coordination and division of labor among inventors;

ii) the skills of participants.

Coordination and modularity

Coordination is particularly difficult when participation in collective inventions, like OSS, takes place on a voluntary basis and participants are geographically dispersed.

A powerful coordination mechanism that reduces the problem of free-riding and monitoring costs is modularity and option-laden architectures (Baldwin and Clark, 2003). Modularity is a 'strategy for organizing complex products and processes efficiently'[1]. Modularity relies on system architectures that define the set of modules and their respective

[1] The implications of modularity for productivity in modern production systems have been analyzed formally by Milgrom and Roberts (1990 and 1995).

functions, the interfaces that allow modules to interact (compatibility) and the standards that are used to test the modules' compliance with the design rules (Baldwin and Clark, 1997). In the case of OSS the architecture of the system typically consists of a core structure (e.g., the kernel of the Linux operating system) and a series of modules that are developed independently of one another. Another feature that open source design shares with other creative activities is the option value of inventive efforts. The evolution of specific tasks/modules cannot be easily predicted since new features and problems are discovered during the process (Kaisla 2001). The final outcome of the collective effort is then uncertain and there is incentive for a developer to join a collective effort despite some degree of free-riding because she has free access to a multiplicity of complementary or substitute outcomes produced by other developers. The effort spent in developing a code is then the price of the real option that allows each developer to choose the best future outcome of the collective effort.

In this setting the early circulation of partially developed code attracts potential contributors who can evaluate the option value of the code and decide whether to contribute new code. Modular architecture and the real option nature of design together favor experimentation and make possible the private provision of a public good (Baldwin and Clark, 2003).

Product modularity and the use of coordination tools like CVS software, forums and mailing lists reduce transaction costs within the community of developers, allow a higher level of task partitioning and a lower level of explicit coordination/interaction among programmers.[2] In large projects like Linux kernel or Apache server, a significant level of modularity is achieved thanks to a clear distinction between the core product architecture and more 'external' features that are 'located in modules that can be selectively compiled and configured' (Mockus et al. 2000: 4). Formal organization of authority is adopted in large projects like Linux, where Linus Torvalds has the last word for any change to the source code that is officially released. By the same token, the Apache Group relies on a formal voting system for approval of changes to the source code. Even in these large projects, however, it is difficult to proceed with a sharp division of tasks among developers because participants are volunteers distributed across space and over time. Developers then contribute their work to a project or a specific task according to their preferences and time (generally they have a job

[2] This mode of coordination may result in excessive duplication of efforts. Open source projects try to moderate this problem by adopting different solutions. For instance, in the case of Apache, new developers focus on tasks that the 'code owner' is not working at (Mockus et al. 2000).

and contribute to open source in their free time). In many small projects it is likely that personal motivation, shared beliefs and values tie these virtual teams together rather than a formal organization (Elliot and Scacchi 2003). These informal, cultural dimensions interact with modularity in reducing transaction and communication costs among developers.

This discussion leads to the hypothesis that, *ceteris paribus*, a higher level of modularization of software design favors the performance of the projects. Notice that modularization may result in greater division of labor among developers, but, especially in the case of small projects, it is possible that few individuals carry out multiple tasks and are engaged in different module development at the same time. For instance, the Apache HTTP Server Project is a large project with over 15 core developers. However, the division of labor among these developers is blurred. All of them contribute code to various modules. Moreover, the role of release manager (a quite time-consuming and critical task) is rotated among them (Mockus et al. 2000).

Skills

The second dimension of collective inventions concerns the skills of participants. As noted by von Hippel (1988), lead users are the functional locus of innovation when they have higher expected benefits compared with other potential sources of innovation. The functional locus of innovation also depends on the distribution of skills among the parties potentially interested to an innovation. OSS is an example of user-led innovation network (Harhoff et al. 2003).

Although the average OSS contributor is an expert, it is likely that participants to a project comprise individuals with different types of skills and knowledge. We also expect substantial differences across projects in the average level of skills of participants and these differences should be reflected in project performance. The literature has analyzed the complementarity between skills and innovation. Human capital is found to be as an important complement to innovation in several empirical studies (e.g., Leiponen, 2005; Mohen and Roller, 2005). Skills are not only important to create new ideas but they are also important to use new technologies and absorb knowledge generated elsewhere. As Cohen and Levinthal (1989) have clearly pointed out, knowledge spillovers are in part endogenous because they cannot be recognized or exploited without investment in absorptive capacity.

The level of capabilities and skills is also particularly important for newly established firms. Well balanced founding teams (or highly skilled single founders) are able to attract financial resources, customers and collaborators (see, for instance, Bhidé 2000; Baron and

Hannah 2002). In the case of OSS, skilled core developers are more likely to attract new users because their software addresses relevant problems that are not met by commercial products or because it raises technical puzzles that are challenging to the community of developers. This in turn helps the project to evolve and become more productive. OSS projects then are not very different from the traditional entrepreneurial sector, where new ventures have to overcome the 'liability of newness' and must convince potential stakeholders to pour their resources to support new ideas and to grow[3].

Less explored by the literature is the role of skill heterogeneity in innovation. The economics and strategic management literature has examined different ways in which skill diversity may affect firm performance. First, skill heterogeneity implies that firms can experiment complex combinations of skills that are difficult to imitate (Lippman and Rumelt, 1982). Second, skill diversity allows a more flexible strategic adaptation to changing external environment (Galunic and Riordan, 1998). Skill heterogeneity provides firms with more comprehensive problem solving ability and creative conflict resolution (Galunic and Riordan, 1998; Sutton and Hargadon, 1997). The cognitive diversity resulting for the interaction among people with different perspectives makes it possible to identify and formulate a wider array of problems and to find a larger set of alternative solutions (Bantel and Jackson, 1989).

Finally, skill heterogeneity has a positive effect on team productivity because of mutual learning (higher skilled team members can transfer their knowledge to lower skilled partners) and intra-team bargaining. Higher skill workers have a higher exit option and a greater bargaining power. They can affect the work norm and induce a higher level of team productivity. High skill workers in team production systems may sacrifice some income in exchange for non-pecuniary benefits such as socialization and a more challenging working environment. Higher skilled participants are also likely to benefit most from team work because they are rewarded by a higher social status and greater decisional authority among peers. Mutual learning and intra-team bargaining power may overcome the problem of free riding and adverse selection raised by the literature on teams (Hamilton et al. 2003).

Various empirical papers examine the benefits of heterogeneous workforce using firm-level and worker-level data. For instance, Bantel and Jackson (1989) have analyzed the top management teams of a sample of US banks and showed that more innovative banks have a more diversified set of top manager expertise. Similarly, Hamilton et al. (2003) have

[3] Obviously, there are dynamic feedbacks between performance and skills. The performance observed at any given time results from the past interaction between initial skill endowment, performance outcomes, and new

analyzed the individual and team productivity in a US garment firm in a three-year period during which the workers could voluntarily switch from traditional production lines to flexible work teams based on a module production system, U-shaped work place and multitasking. They find that skill heterogeneity of workers has a positive effect on team productivity. Laursen et al. (2005) have examined the performance of engineering consulting firms in Denmark and found less clear-cut results. More precisely, they report a non-linear relationship between skill diversity and performance in large firms whereas small firms do not seem to benefit from skill diversity at all. Laursen et al. (2005) claim that these results reflect the cost of skills diversity. Communication costs, misunderstanding and inefficient bargaining lead to negative productivity outcomes that counterbalance the productivity gains arising from the creativity and flexibility advantages discussed above. Carbonell and Rodriguez (2006) also find a curvilinear (inverted U-shape) relation between functional diversity in product development teams and speed of innovation in a sample of Spanish firms in different sectors. In the first portion of the curve the speed of innovation increases with the functional diversity of team members, while with high levels of diversity it is more difficult to share the same objectives with implied loss of efficiency and speed in the development process.

These contrasting results raise the question as to whether skill diversity is always good for team productivity. We will address this issue in the context of OSS projects where, as mentioned before, there are coordination costs due to the spatial dispersion of contributors. These costs, however, are moderated by shared beliefs and visions among participants as well as the use of a modular design model.

3. Data and variables

Our empirical analysis draws on a unique dataset containing information on OSS projects and individuals (registered users). The dataset has been built from data provided by the SourceForge.net website (http://sourceforge.net, *SF.net* henceforth) for the period November 3rd 1999 to January 10th 2003.

The number of projects registered to *SF.net* has increased quite rapidly since its foundation. In May 2005 the number of registered projects was about 100,000 and the number of registered users-developers was more than 1,000,000. Other websites hosting open

human capital formation. Skills at time 0 may yield a given performance outcome at time 1 which in turn may attract new skilled members at time 2 and so forth.

source projects are much smaller. For example, in May 2005 Savannah hosted 2,384 projects and 35,709 registered users-developers (http://savannah.gnu.org).

Even though none of these datasets is representative of the entire open source community, *SF.net* remains the world largest pool of projects and developers and one of the most widely used by earlier empirical papers (e.g., Krishnamurthy 2002; Fershtman and Gandal 2004). To our knowledge, however, only few works have gained access to large samples of projects hosted in *SF.net* (for example, Lerner and Tirole 2005; Comino *et al*., 2005; Madey *et al*., 2004).

For the purposes of this paper, we selected a list of relevant variables. In order to understand the meaning of some key variables and the functioning of open source projects hosted by SF.net we first analyzed a random selection of e-mail messages between project contributors extracted from 'project forums' of SF.net. We then surveyed a random sample of 58 SF.net users and asked questions about the procedure adopted for admitting a new contributor to a registered project, the criteria for assigning tasks and roles to projects' members and those for reviewing the inputs submitted by various categories of contributors. We also asked information about the most appropriate measures of individual contributions (bugs, new feature requests, lines of code etc.) and projects' performance (new releases, fixed bugs or patches, CVS commits). We received responses by 13 users and checked this information through informal face-to-face discussions with experts and SF.net users.

The *SF.net* dataset includes a great deal of information at two different levels of aggregation: the project or team and the single user-developer. A "project" is carried out by a group of users-developers working on the same software. A "user-developer" is an individual who contributes to one or more projects. Our version of the dataset consists of 65,535 projects and 544,669 individuals. The latter include both people registered with a project (i.e., participants who are supposed to be 'active' contributors) and non registered participants, i.e., individuals who registered with the SF.net website but are not associated with any specific project. The latter include several individuals who download software and signal bugs or post questions to the project forums. The number of people ever registered with a project is 90,255, only a tiny fraction of all individuals registered with the SF.net. Of these only 26,314 provide data on skills.

The sample we used in this paper includes data on 49,422 individuals who registered with 24,954 projects. The sample is smaller than the population of individuals registered with at least one project (90,255) for a number of reasons. First, to avoid contemporaneous correlation between dependent and independent variables we focus on observations at the end

of the period (October-December 2002) for our dependent variables and draw on lagged observations of regressors (November 1999- September 2002). The three-month period trade offs between two opposite aims. On the one hand, as shown in the next Section, skills, one of our key regressors, are observed at the end of the period (January 2003). Therefore, we want to make sure that they do not change significantly during the period selected for the dependent variables because of changes in the team composition and learning by individual participants. It is unlikely that skills change substantially over three months because of learning. They may change, however, because of modifications in team composition. To account for this effect we considered individuals who have registered with a project before September 2002. On the other hand, we want to allow our dependent variables to change over time. A three-month period represents a good compromise between these two objectives. A shorter period does not guarantee a satisfying level of variability of the dependent variables while longer periods may reduce the reliability of the data on skills. To be sure, we performed robustness checks of our estimations by adopting different time windows for the dependent and explanatory variables.

The procedure described above implies the exclusion of projects registered after September 2002 and this leads to a sample of 58,165 projects. We have also excluded projects with missing information on control variables, such as natural language and programming language, and ended up with a final sample of 24,954 projects and 49,422 individuals. Only 17,675 of these participants (registered with 13,427 projects) declared their skills.

A possible drawback of this dataset is that it covers many small projects. Some large and popular projects, like Linux and Apache, are excluded. However, large projects like Freenet or BZflag are well represented in SF.net, as well as popular projects like phpMyadmin, which is listed among the top 20 popular projects by Freshmeat.net (http://freshmeat.net).

Recent works have pointed out other potential weaknesses that should be considered when using the SF.net data. First, a large number of projects hosted in SF.net is inactive (Rainer and Gale, 2005). This is not a major drawback to our purposes because our analysis is precisely aimed at identifying the determinants of projects' activity. Second, some project administrators register their projects to SF.net only to acquire visibility in the open source community and carry out software development activity outside the SF.net (Howison and Crowston, 2004). However, Comino et al. (2005) show that this phenomenon is quite rare. Finally, some variables concerning the contribution of individuals to project activity may be affected by measurement errors. We devoted a considerable effort to reduce these errors. For

example, we checked and corrected manually any error in the affiliation of each member to SF.net projects. As mentioned before, phone and face-to-face interviews with SF.net expert developers also helped to select variables and interpret results.

3.1. Dependent variables

Our choice of performance indicators was driven by the literature and interviews with users-developers. A number of studies have tried to assess the performance of open source projects relative to proprietary software. For instance, Kuan (2001) has analyzed the rate of bugs' resolution in three open source projects – Apache, FreeBSD and Gnome – as a proxy for product improvement or quality. Similarly, Mockus el al. (2000) have studied different measures of quality for the Apache Server such as defect density (defects per thousand lines of code added) and response time to problems reported by users.

Crowston *et al*. (2004) have reviewed several alternative indicators of open source project success and popularity such as the speed of response to bugs reports. They have tested the validity of these measures by using expert interviews and SourceForge data. Moreover, they have compared bug-fixing time with the number of core developers and users who send bugs reports, the level of activity provided directly by SF.net and the number of downloads. Crowston *et al*. (2004) conclude that a combination of different indicators should be preferred to single proxies for project success.

Very few studies have explored indicators of performance that are more familiar to economists like productivity. For instance, Fershtman and Gandal (2004) have analyzed a sample of 71 open source projects hosted at the SF.net website between 2002 and 2003 and used the lines of source code per contributor (developer) as a proxy for productivity.

We look at different indicators of project performance by exploiting the rich set of information offered by the SF dataset about various types of project activities - bugs, patches, new features and support requests (trackers), CVS commits and new file releases.

More precisely, our measures of performance are as follows: ACTIVE, CVS and FILE_RELEASE.

ACTIVE is a dummy variable that takes value 1 if at least one of the following events occurred between October 1, 2002 and December 31, 2002: a fixed bug, a fixed patch, a new feature request fulfilled, a new product release, a new CVS commit. This variable denotes that a project is alive and active in SF.net. The choice of a composite measure is motivated by the need to account for a representative set of activities that project members typically carry out within a project. In this sense we use it as a measure of survival of the project.

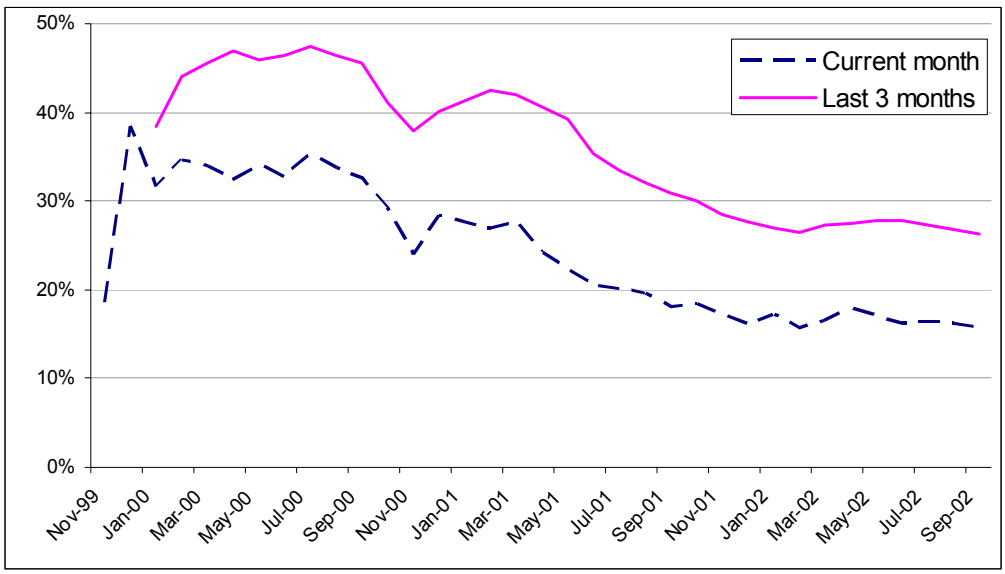
About 20% of the groups in our sample were active in the last three months of the period (see the descriptive statistics in Table 1). This low share indicates that a very large number of projects do not produce any output, or that the activity in the project is very discontinuous, producing only rare events over time.

Figure 1 shows the share of active projects over total projects registered over the sample period. We compute for our sample the share of active projects in 1-month and 3-month time windows. As expected the share of active projects is higher when a three-months time window is considered, suggesting that there is discontinuity in the project activity. The decline in the share over time suggests that increasing number of registered projects has brought about the entry of many unproductive projects.

We also examined the persistence of ACTIVE status of each projects over time and found out that three months is a large enough period for defining our dependent variables. Figure 2 shows the share of projects active in each trimester which are also active in the previous trimester. Notice that this share increases over time suggesting that at the end of the sample time the set of active groups tend to stabilize. This moderates the problem of right censoring in our data – i.e., projects that are active (inactive) in the last trimester of the sample may become inactive (active) afterwards.

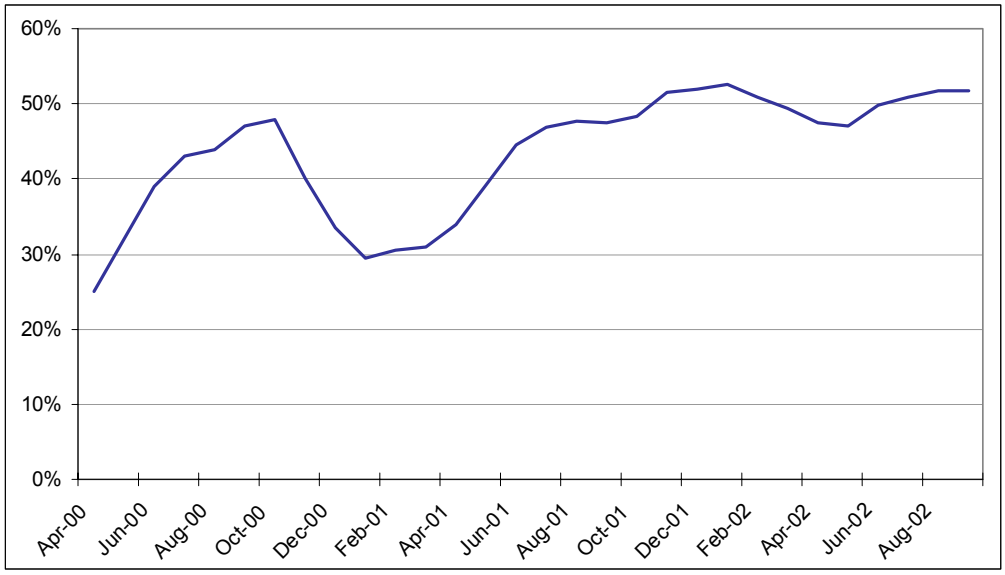
As a robustness check, we repeated our estimations with different time spans of the dependent variables (i.e., 1 month and 6 months) and obtained qualitatively similar results. The same checks have been carried out with the other dependent variables described below and confirmed the results shown in this paper.

Figure 1. Share of active projects over total projects registered



Note: Total number of projects at September 2002= 24,954.

Figure 2. Share of projects active in two subsequent trimesters



CVS is the number of CVS commits submitted to the project's archive from October 2002 to December 2002. We use it as a proxy for the level of productivity of each project. CVS is a tool which has two main features. First, it is the repository of the project, where the lines of code offered by developers are "assembled". Second, it synchronizes the users' local version of the software with the online one, with the purpose of facilitating the cooperation and avoiding the proliferation of different versions of the software. When a developer modifies the project code and uploads the changes on the CVS repository, the system records a "commit". In our sample the average number of CVS commits in the last three months is 28.21 per project, with a minimum of zero and a maximum of more than 17,000.

Our data do not provide information on the lines of source code per contributor and, indeed, it would be arduous to obtain this data for a large sample of projects like ours. Moreover, we are interested in understanding the level of project activity, i.e. the level of active participation of project's members in design and problem solving, rather than code writing as such. The CVS repository provides a wide range of information which can be used to infer the developers' coding activity. Each CVS commit represents a change introduced by a developer to the project code. Thus, the number of CVS commits can be viewed as a reasonably good proxy for the productivity at the project level and earlier works have used it as a measure of developers' coding activity (see, for example, Lopez-Fernandez et al., 2004; Gonzalez-Barahona *et al.*, 2004). Moreover, working with the code does not necessarily mean adding new lines to the existing code since a developer can change the existing code or even erase redundant lines of code. From this perspective, the number of CVS commits

seems to be a better measure of members' participation and project activity than the simple count of the lines of code[4].

FILE_RELEASE. This measure of project performance is obtained by counting the number of project's file releases from October 2002 to December 2002. A new file release represents the launch of a new version of the software produced by the project. Since a file has to be stable enough to be released, file releases represent a milestone in the production process. This highlights an important difference between file releases and CVS commits. The latter measure the input offered by people involved in project's development activities whereas the former are a proxy for the output of these activities that is potentially accessible to a wider community of OSS participants – including non expert users.

Finally, compared to other measures of performance used in previous works, like the rate of bugs resolution, the submission of patches, or the development stage of the projects, both CVS and FILE_RELEASE can be considered as different steps of the development process. For example, a bug resolution can be integrated in a CVS commit and, ultimately, in a file release.

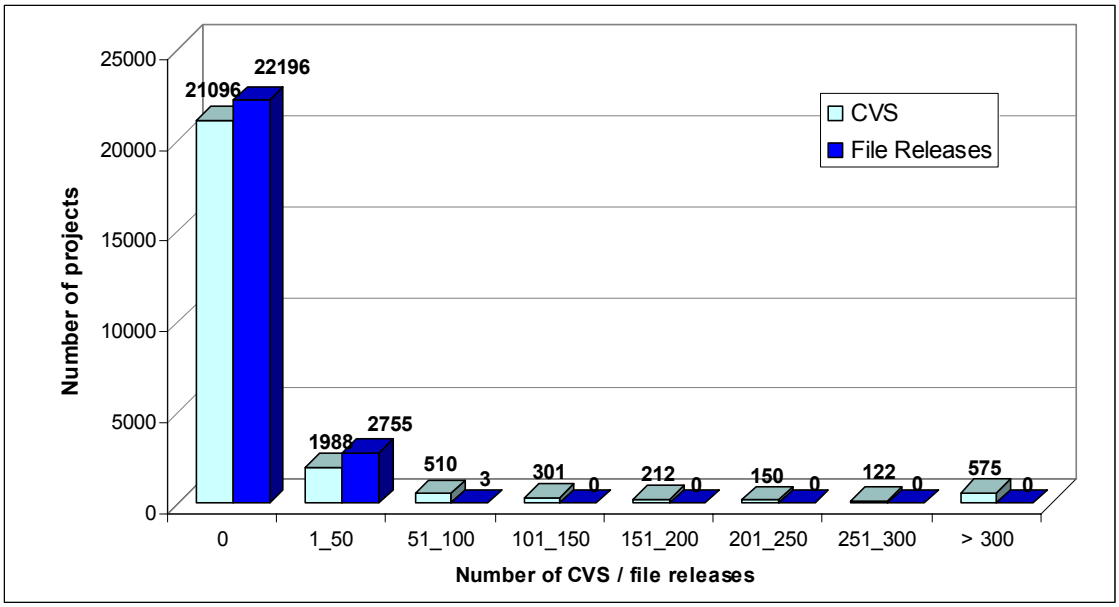
It is worth noting that the distribution of projects by number of CVS and file releases is very skewed (Figure 3). In the period October-December 2002, 21,096 projects have not produced any CVS commit and 22,196 projects have not released any new version of the program. Most projects which have produced a positive number of CVS and file releases have only produced less than 50. Smaller shares of projects have instead produced more than 50 CVS commits, with 575 projects with more than 300.

A potential weakness of our measures of project performance arises from the fact that SF.net may function as an incubator. After the start-up stage, a successful project may decide to continue its activity outside SF.net for the reason that it does not need this infrastructure anymore. This would entail a truncation problem on our measures of performance. To account for this potential bias we include size and age among our controls.[5]

[4] Additional details on the tools used by OSS developers can be found in Ghosh and David (2003).

[5] Another problem that could bias our estimates of performance is that a project may stop its activity not because of a lack of success but because it has reached its goals. This category of projects cannot be easily identified because most OSS projects have no time constraints.

Figure 3. Distribution of projects by number of CVS commits and file releases



3.2. Main explanatory variables

Skill level and diversity

Data on individual skills are usually very difficult to obtain. However, the SF.net dataset provides detailed information on 33 types of different skills which can be grouped into three main areas of expertise: programming languages (e.g., C/C++ and Python), application-specific skills (e.g., networking, security etc.) and 'people' skills proxied by the knowledge of spoken languages[6]. This information is provided by users at the time of registration at *SF.net,* when they are asked to report which skills they possess among the 33 categories mentioned above and, for each category, to self-assess the level and experience. Although these measures might be affected by self-evaluation biases, we believe that this is not a serious problem in our case because the information supplied by developers can be made public to other developers who can check its reliability. Since, as noted by Lerner and Tirole (2002), expected delayed benefits arising from signalling represent an important incentive to contribute to the OSS community, individuals who register with SF.net have strong reasons to provide information as much as possible close to the reality. A similar mechanism has been highlighted by Comino *et al*. (2005) and Lerner and Tirole (2005) with respect to projects

self-declared characteristics. Moreover, there are no reasons to believe that a potential bias due to self-reporting should affect particular types of developers or projects.

It is also worth noting that users-developers can update the information relative to their skills at any time, so that in every moment we have in principle a good approximation of each developer's skills set. In order to account for potential endogeneity, we rely on skills updated until September 2002, i.e. the month before the time span of the dependent variables. We retrieved the information about the project membership in September 2002 and built the skill measures only for people belonging to the projects up to that time. We excluded individuals who have joined the project over the last three months. Although skills are observed at the end of the sample (January 10th 2003), we can reasonably assume that individuals do not update their skills in the SF.net website in such a short time windows. Neither we expect substantial learning effects over three months.

Drawing on this information, we build the following measures of skills at the project level.

EXPERIENCE. This variable is computed as the average level of experience of the project's members. The level of experience of each member is measured on a five point Likert scale (i.e. 1= less than 6 months; 2) 6 months-2 years; 3) 2-5years; 4) 5-10 years; 5) more than 10 years.

EXPERIENCE_SD. We use the standard deviation of the project members' experience in order to take into account the presence of people with heterogeneous levels of skills (for example a few core developers and other members carrying out lower level activities).

EXPERIENCE_SD2. We use the square term of the standard deviation of the skill level of project members to capture potential curvilinear relations between skill diversity and performance.

N_SKILLS. This variable indicates the number of different skills mastered by the project's members. It is a proxy for the variety of skills in the project.

N_SKILLS_SD. We use the standard deviation of the number of skills mastered by each member of the project to differentiate projects composed by members with similar skill profiles from projects characterised by diversity in the skill profile of people, i.e. projects with both generalist and specialist people.

[6] This classification is in line with those adopted in the literature on software development. For instance, Faraj and Sproull (2000) point out three dimensions of development expertise: 1) technical expertise defined as knowledge of specialised technologies and tools; 2) design expertise that refers to the knowledge of software design principles and architecture; 3) domain expertise (knowledge about the application domain and customers operations) (p. 1559).

N_SKILLS_SD2. The square term of the standard deviation of the number of skills mastered by the project members is used to capture potential curvilinear relations between diversity in the skill profile of project members and performance.

HERF_MIN. This variable measures the level of skill diversification of the most diversified project member. To build our measure, we first define the share of each skill *i* on the total skills experience of the individual *j*:

$$s_{ij} = S_{ij} * e_{ij} / \sum_{i=1}^{n} S_{ij} * e_{ij}$$

where S_{ij}=1 when individual *j* reports skill *i*, and 0 otherwise, and e_{ij} represents the experience of the skill *i* of the individual *j*. Than we computed the following Herfindahl index of skills for each individual: $Herf_j = \sum_{i=1}^{n} s_{ij}^2$.

HERF_MIN represents the minimum Herfindahl index among all project members. It denotes the presence in the project of at least one member with a generalist skill profile.

HERF_SKILLS. At the project level we also compute the Herfindahl index by aggregating the information about skills level and diversity across all members. For any project *k* we calculated the share of skill *i* (by summing up the experience in skill *i* of all *j* members) on the total level of experience in all skills:

$$sen_{jk} = \sum_{j} S_{ijk} * e_{ijk} / \sum_{i,j} S_{ijk} * e_{ijk}$$

where S_{ijk} is equal to 1 when individual *j* belonging to project *k* claims some experience in skill *i*, and 0 otherwise. We than computed the Herfindahl index at the group level: $HERF\_SKILLS = \sum_{i=1}^{n} sen_{ik}^2$

For a more intuitive interpretation of the variables based on the Herfindahl indexes in our analysis we use the *1-HERF_MIN* and *1-HERF_SKILLS.*

We also computed entropy measures of skills at the individual and project levels as alternative proxies for the diversification of skills. In unreported regressions we used these variables in place of the Herfindahl indexes and the results are very similar.

Modularity

A key hypothesis of this paper is that project performance is affected by the level of modularity or the division of tasks at the project level. As mentioned before, we have no

access to the source code of the project and therefore we cannot analyze the architecture of the system from a purely technical perspective. However, we know whether the project has been divided into subprojects. Our measure of project modularity is precisely the cumulated number of subprojects opened by the project until September 2002 (*NSUB_PROJECTS*).

As suggested by the literature, the architecture of the system is made of the main project, where the basic code is developed, and a series of subprojects that corresponds to the modules discussed before (Clark and Baldwin, 2003). The latter focus on specific tasks such as the construction of a web site, the production of documentation, the translation of the original code into other programming languages, and so on.[7]

Table A.1 in the Appendix illustrates the architecture of a typical project in our dataset. As suggested by the list of subprojects reported in the Table, it is evident that the activity of some of these subprojects involves the production of code for specific modules (e.g., code generators for translation in other programming languages and graphical interfaces) that at some point will be integrated into the whole system, while other subprojects focus on administrative or organizational tasks. Our measure of modularity then captures the organizational architecture of the system rather than its technical one. Notice that we considered only subprojects that have opened at least one task, i.e. have initiated an activity to carry out during the project life. Inactive subprojects then have been excluded from the analysis.

In some regressions we also include some interaction terms in order to account for potential complementarities between the two key dimensions of team production or problem solving activities – skill diversity and modularity.[8] We use the following variables:

- *NSUB_PROJ_X_EXPER_SD.* Number of subprojects multiplied by the standard deviation of the level of experience of the project members.
- *NSUB_PROJ_X_N_SKILLS_SD.* Number of subprojects multiplied by the standard deviation of the number of skills of the project members.
- *NSUB_PROJ_X_ 1-HERF_SKILLS.* Number of subprojects multiplied by the average Herfindahl index of the project members.
- *NSUB_PROJ_X_1- HERF_MIN.* Number of subprojects multiplied by the minimum Herfindahl index among the project members.

[7] Our definition of modularity is in line with the literature on software engineering (e.g., Kiczales, 1996, and Parnas, 1972).

3.3. Other regressors and controls

SIZE. Our measure of project size is the number of members registered with the project by September 2002.[9] Notice that only 19.5 per cent of the 24,954 projects in our sample have more than two members against over 63 per cent with only one member registered.

ENTRY_DATE. To control for the age of the project we use a variable indicating the month of registration with SF.net, from November 1999.

USERS_EXT_LINKS. The average number of projects participated by the project's members up to September 2002 is used as a proxy for project's links with other projects.

FORUM. This variable was obtained by counting the messages sent to the project's forums until September 2002.

AUDIENCE_dummies. We accounted for the intended audiences addressed by the project. Each group can declare more than one intended audience. Our dummy variables identify the projects declaring intended audiences among the most popular ones, i.e. End Users/Desktop, Developers and System Administrators (*AUDIENCE_MAIN*), the projects declaring only intended audiences outside the set of the principal ones, i.e. Customer Service, Education, Financial and Insurance Industry, Healthcare Industry, Information Technology, Legal Industry, Manufacturing, Religion, Science/Research, Telecommunications Industry, Other Audience (*AUDIENCE_OTHER*), the projects declaring both type of audiences (*AUDIENCE_BOTH*), and the projects for what no audience was defined (*AUDIENCE_NO*). Addressing popular intended audiences may yields two contrasting effects on project performance. First, they can leverage their innovations on a larger 'market' with many potential feedbacks that help the product to evolve more quickly over time. Second, addressing very popular audiences may produce negative externalities due to congestion and competition. Although there is no rivalry in the use of software, it is possible that in popular 'markets' like those addressing the needs of system administrators some rivalry in the use of input takes place since several projects 'fish' in the same 'pool' of skilled people. Moreover,

[8] We should note that estimates of the interaction term is far from being a proper test for complementarity, which is outside the scope of this paper.

[9] In January 2003 SF.net hosts 65,535. 11,262 of them (17.18%) have no members, 36,389 (55.53%) have just one member, and the rest (17,884) has size greater than 1. Among all the projects with at least one member (54,273), 67.05% have only one registered developer. We measured size with the number of members on August 21st 2002 while our dependent variables are observed from October 1st 2002. During the period August 21-October 1, 1436 new projects have entered the *SF.net* dataset. We have included these projects in our regressions since it is unlikely that new members (apart from the founders) have joined the project because of project's performance in such a short time window. Our results do not change when these projects are dropped from the sample.

in these fields it is likely that network externalities, reputation effects or social group agglomeration processes attract users towards few large projects (e.g., Apache or Python) and leave other projects on the margins of the OSS community.

LI_Dummies. Dummies for the main license adopted by the project (GPL, LGPL, BSD, Public Domain, Artistic license, Apache license, MIT license, Others). In our sample the majority of projects (72.91%) are distributed under a GPL license, 8% under a LGPL license, 6.4% under a BSD license and the remaining 12.69% of projects are distributed across other types of licenses. In our estimations GPL is used as baseline dummy.

NL_Dummies. Dummies for the project's official spoken language (English, German, French, Spanish, Russian, Japanese, Italian, Dutch, Portuguese-Brazilian, Polish, Swedish, Chinese, Others). In our estimations English is used as reference group.

PL_Dummies. Dummies for the programming language of the project (C/C++, Java, Php, Perl, Python, Visual Basic, Unix Shell, Others). In our estimations C/C++ is used as reference group.

4. Econometric method

The goal of our econometric analysis is to study the effects of skills and modularity on project survival and performance. To this purpose, we have estimated two sets of equations.

First, we estimated the probability of projects' survival with a logistic regression. The dependent variable is ACTIVE, which is equal to 1 when at least one of the following events has occurred during the period October 2002 to December 2002: a fixed bug, a fixed patch, a fulfilled feature request, a CVS commit message sent to the CVS repository, a new file release. The probability of positive counts

$$\Pr ob(Y = 1) = \frac{e^{\beta' x}}{1 + e^{\beta' x}}$$

is a function of covariates x that include measures of skills, modularity and other regressors.

Second, we analyse the determinants of projects' performance by estimating two zero-inflated negative binomial equations. Our dependent variables are counts of respectively file releases and CVS commits during the last 3 months of the sample period.

The choice of negative binomial is motivated by the nature of the dependent variable. In general, empirical models of innovation are affected by unobservable permanent heterogeneity that leads to overdispersion and larger numbers of zero counts than the

frequency predicted under the standard null Poisson and negative binomial models (Blundell et alt., 1995). The literature suggests several goodness-of-fit tests to evaluate alternative models to the Poisson specification. The Poisson regression model assumes the equality of the conditional mean and variance: $E[Y_i]=\exp[\beta X_i]=\text{Var}\ [Y_i]$. Cross-section heterogeneity implies overdispersion, i.e., $\text{Var}\ [Y_i] > E[Y_i]$ ('mixing spreads out the distribution towards the tails') (Mullahy, 1997:338). A simple test for overdispersion consist in comparing the null hypothesis Ho: $\text{Var}\ [Y_i] = E[Y_i]$ against the alternative H_1: $\text{Var}\ [Y_i] = E[Y_i] + \alpha\ f(E[Y_i])$. In this setting, the test of the null hypothesis is conducted by running a t test on the hypothesis whether the coefficient α is significantly different from 0 (Greene, 1997).

Alternative specifications, such as the negative binomial, allow for overdispersion but, like the Poisson, tend to under-predict the true population probability of zero outcomes of the dependent variable. One solution is the adoption of zero-inflated or positive count models. A goodness-of-fit test for zero-inflated models against the standard Poisson or negative binomial models in presence of excess zero counts has been developed by Vuong (1989). We use this test to compare the standard negative binomial model with a zero-inflated negative binomial model. Zero-inflated estimators are two steps ML estimators. A binary probability equation (inflation equation) is first estimated to predict the membership in the 'always zero'. Then a (truncated) negative binomial model describing the distribution of the 'not always zero' cases (non negative outcomes) is estimated.

Panel data count models (Hausman et al, 1984) are used in alternative to zero-inflated estimations when information on the dependent variable for each observation is repeated. In our case, we may observe different counts of CVS commits or file releases for the same project. These models allow for fixed effects but make the assumption of strict exogeneity of explanatory variables which is unlikely to be the case with our data because of dynamic feedbacks – past outcomes (e.g., file releases) are likely to affect the probability of current outcomes. The same problem has been addressed by Blundell et al. (1995) in a panel of firm level innovations. They have followed two alternative estimation strategies. First, a GMM panel data estimator for non-linear count data models has been used to account for fixed effects and weakly exogenous (or predetermined) regressors. Second, they used the pre-sample history of the dependent variable (innovations) as a measure of firm unobservable fixed effects. The application of these techniques is bounded by the length of the panel. For example, Blundell et al (1995) estimate the probability of firm innovation between 1972 and 1982 by using its innovations in the period between 1947 and 1971.

The shortness of our panel does not allow using these estimators. We can only moderate the problem of endogeneity by relying on predetermined regressors. Our main regressors may be endogenous because skilled individuals may decide to enter the project at time **t** on the basis of earlier project performance. To moderate the endogeneity of skill-related variables we used lagged values of these variables. We know the history of users-developers over the sample period and therefore we can see whether a particular user-developer leaves a project to join another one. The history of users-developers allows us to predetermine the main regressors. It is worth to recall that the team composition of the sample projects changes very slowly during the sample period. To this aim we have analyzed the users' registration status every 30 days starting from September 1st, 2000. The total number of periods is then 28 (after the exclusion of the last period which lasts only 20 days).

Focusing on individuals who provided information about skills registered after September 1, 2000 (N=41,637), we found that the average number of projects participated by registered users is about 1.51 (SD=1.16).[10] On average these individuals remain in the same project 11.43 months (SD=7.00). The pattern of project participation for the population of individuals registered after September 1 2000 is very similar to that described above.

The low level of individual mobility across projects that we found in the data suggests that team composition changes very slowly over the time windows analysed here. It is unlikely then that the composition of the projects observed before the last three months changes during the last period and affects our dependent variables.

The intended audience, the language and the license are also pre-determined. More precisely, the variables are observed at the time of project registration and do not change much over time. For instance, Fershtman and Gandal (2004) noted that these variables changed only rarely in a sample of 71 SF.net projects and license models in particular never changed.

5. Results

5.1. Descriptive statistics

Table 1 summarizes the variables used in the empirical analysis and provides some descriptive statistics for the sample of 24,954 projects. For about 46% of these projects (11,527) data on skills are missing or not reported. Given the importance of this variable in our analysis, we compared these two categories of projects and found that there are not

significant differences between them in the percentage of active projects and the number of file releases.

Instead, we found significant differences in the average size, the number of CVS commits and the number of subprojects. For instance, the average number of CVS is 20 (SD=205.05) for projects with missing skills against 35 (SD=197.14) for projects which report data on skills. Moreover, the average number of subprojects is 0.166 (SD=0.68) in projects with missing skills against 0.52 (SD=1.43) of the other category of projects. Overall, projects with missing data on skills are smaller, less productive in terms of CVS commits and have a limited number of subprojects as compared with projects with data on skills. To avoid that coefficients of skill-related variables are biased we have introduced a dummy variables for observations with missing data (see Hall and Ziedonis, 2001 for a similar treatment of potential selection on key regressors).

Table 2 provides the bivariate relationship between the regressors. Obviously, Pearson correlation coefficients between different indicators of skill diversity are positive and significant. The correlation between project size and other important regressors is also strong. For instance, size is strongly correlated with the number of different skills (ρ=0.43) and its standard deviation (ρ=0.445), and with the number of subprojects (ρ=0.275).

As expected, there is a quite strong correlation between CVS commits and file releases (not shown in Table 2). We should also remind that while CVS commits measure the continuous flow of inputs to the development process provided by different contributors, file releases represent the output of the same collective development process. Therefore, the latter is a much more discontinuous variable. This important difference motivates the estimation of two separate equations for these two performance indicators.

[10] Note that the difference with the total number of individuals for which data on skills are available is due to the fact that information on individual mobility across projects is not available before September 2000. We have also excluded individuals registered after September 2002

Table 1. Definition of variables and descriptive statistics – Main variables

Variable	Description	Mean	Std.	Median	Min	Max
ACTIVE	Dummy variable that takes value 1 if at least one of these events has occurred between October 1 2002 and December 31 2002: a fixed bug, a fixed patch, a new feature request fulfilled, a new product release, a new CVS commit	0.202	0.401	0	0	1
CVS	Number of concurrent versions system commits produced by the project in the period October - December 2002	28.213	201.055	0	0	17724
FILE_RELEASE	Number of files released by the project in the period October - December 2002	0.330	1.798	0	0	139
NSUB_PROJECTS	Cumulated number of subprojects with at least one activated task opened by the project until September 2002	0.361	1.159	0	0	45
EXPERIENCE*	Project members' average experience in the skills they declared (5 classes of years)	2.770	0.655	2.73	1	5
EXPERIENCE_SD*	Standard deviation of project members' experience in the skills they declared (5 classes of years)	0.906	0.396	0.93	0	2.82
EXPERIENCE_SD2*	Squared term of the standard deviation of project members' experience in the skills they declared	0.977	0.732	0.86	0	7.95
N_SKILLS*	Total number of skills declared by project members (avoiding double counting)	8.381	4.677	8	1	30
N_SKILLS_SD*	Standard deviation of the number of skills declared by the project's members	0.746	1.598	0	0	13.43
N_SKILLS_SD2*	Squared term of the standard deviation of the number of skills declared by the project's members	3.111	9.566	0	0	180.36
HERF_SKILLS*	Herfindahl index of the skills at the project level: skills weighted both for the number of individuals having that skill in the project and for the experience level of the project	0.203	0.181	0.14	.03	1
HERF_MIN*	Minimum individual Herfindahl index relative to skill experience of project's members	0.206	0.180	0.15	0	1
NO_SKILLS	A dummy variable taking value 1 if none of the users belonging to the project has declared her/his skills set	0.462	0.499	0	0	1
SIZE	Number of members of the project at September 2002	2.054	2.651	1	1	82
ENTRY_DATE	The registration period of the project measured in months from November 1999	23.284	7.721	24	1	35
USERS_EXT_LINKS	Average number of projects per user in September 2002	1.871	1.679	1.2857	0	24
FORUM	Number of messages sent to the project's forums before October 2002	12.329	129.256	3	0	11355
AUDIENCE_MAIN	A dummy variable taking value 1 if the project declared at least one only intended audiences among the principal ones (i.e. system administrators, desktop/end users and developers).	0.811	0.392	1	0	1
AUDIENCE_OTHER	A dummy variable taking value 1 if the project declared only intended audiences outside the set of the principal ones.	0.138	0.345	0	0	1
AUDIENCE_BOTH	A dummy variable taking value 1 if a project declared intended audiences both among the principal ones and outside the set of the principal ones.	0.023	0.149	0	0	1
AUDIENCE_NO	A dummy variable taking value 1 if a project did not declare any intended audience.	0.028	0.165	0	0	1

Note: Number of observations =24,954. * For the variables using information about skills the number of observations is 13,427.

Table 1 (cont.). Definition of variables and descriptive statistics – Control variables (n=24954)

Variable	Description	Mean	Std.	Median	Min	Max
LI_GPL	Project's license: GPL	0.729	0.444	1	0	1
LI_LGPL	Project's license: LGPL	0.080	0.271	0	0	1
LI_BSD	Project's license: BSD	0.064	0.244	0	0	1
LI_PUBDOM	Project's license: Public Domain	0.018	0.133	0	0	1
LI_ARTLIC	Project's license: Artistic license	0.022	0.147	0	0	1
LI_APACHE	Project's license: Apache license	0.014	0.117	0	0	1
LI_MIT	Project's license: MIT license	0.016	0.126	0	0	1
LI_OTHER	Project's license: Others	0.057	0.233	0	0	1
NL_ENGL	Project's official spoken language: English	0.957	0.203	1	0	1
NL_GERMAN	Project's official spoken language: German	0.095	0.293	0	0	1
NL_FRENCH	Project's official spoken language: French	0.058	0.233	0	0	1
NL_SPANISH	Project's official spoken language: Spanish	0.034	0.180	0	0	1
NL_RUSSIAN	Project's official spoken language: Russian	0.016	0.124	0	0	1
NL_JAPANESE	Project's official spoken language: Japanese	0.010	0.098	0	0	1
NL_ITALIAN	Project's official spoken language: Italian	0.003	0.054	0	0	1
NL_DUTCH	Project's official spoken language: Dutch	0.002	0.041	0	0	1
NL_P_BRAZ	Project's official spoken language: Portuguese-Brazilian	0.002	0.041	0	0	1
NL_POLISH	Project's official spoken language: Polish	0.001	0.025	0	0	1
NL_SWEDISH	Project's official spoken language: Swedish	0.001	0.028	0	0	1
NL_CHINESE	Project's official spoken language: Chinese	0.001	0.029	0	0	1
NL_OTHER	Project's official spoken language: Others	0.001	0.028	0	0	1
PL_C	Project's programming language: C or C++	0.270	0.444	0	0	1
PL_JAVA	Project's programming language: Java	0.217	0.412	0	0	1
PL_PHP	Project's programming language: Php	0.170	0.376	0	0	1
PL_PERL	Project's programming language: Perl	0.117	0.321	0	0	1
PL_PYTHON	Project's programming language: Python	0.057	0.231	0	0	1
PL_VISBASIC	Project's programming language: Visual Basic	0.037	0.188	0	0	1
PL_UNIXSHELL	Project's programming language: Unix Shell	0.028	0.165	0	0	1
PL_OTHERS	Project's programming language: Others	0.196	0.397	0	0	1

Table 2. Correlations between main variables

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)
(1) NSUB_PROJECTS	-													
(2) EXPERIENCE	-0.003	-												
(3) EXPERIENCE_SD	0.065*	0.071*	-											
(4) N_SKILLS	0.218*	-0.012	0.281*	-										
(5) N_SKILLS_SD	0.215*	-0.016	0.150*	0.551*	-									
(6) 1-HERF_SKILLS	0.101*	-0.013	0.360*	0.681*	0.261*	-								
(7) 1-HERF_MIN	0.095*	-0.012	0.354*	0.669*	0.254*	0.994*	-							
(8) USERS_EXT _LINKS	-0.033*	0.034*	0.026*	0.132*	0.010	0.110*	0.111*	-						
(9) FORUM	0.091*	0.021*	0.029*	0.118*	0.110*	0.043*	0.039*	-0.019*	-					
(10) AUDIENCE _MAIN	-0.038*	0.028*	-0.010	-0.028*	-0.012	-0.009	-0.008	0.018*	-0.008	-				
(11) AUDIENCE _OTHER	0.009	-0.028*	-0.006	-0.012	-0.007	-0.016	-0.017	-0.004	-0.010	-0.303*	-			
(12) ENTRY_DATE	-0.094*	-0.033*	-0.055*	-0.096*	-0.103*	-0.058*	-0.054*	-0.067*	-0.070*	-0.007	-0.021*	-		
(13) SIZE	0.275*	0.030*	0.109*	0.430*	0.445*	0.152*	0.133*	-0.048*	0.223*	-0.020*	-0.007	-0.137*	-	
(14) LI_GPL	-0.011	-0.094*	0.020*	-0.063*	-0.022*	-0.057*	-0.056*	-0.070*	-0.008	-0.035*	-0.022*	-0.012	-0.040*	-

*p <0.05.

5.2. Estimation results

Table 3 reports the results of logit estimates of project activity as a function of skills, the number of subprojects and other regressors. To check for the robustness of results, we tried different specifications of the logit equation. In our baseline specification (model 1), we have entered our key regressors without the control variables. In model 2 we add the interaction term between the number of sub-projects and the standard deviation of experience. This specification also contains the full set of project-specific controls, such as size, age, type of license model, type of intended audiences and the average number of projects participated by the project members. The remaining specifications contain alternative measures of project skills and skill diversity with and without controls: the total number of different skills, its standard deviation and the square term of the standard deviation (models 3 and 4), the Herfindahl index of skills at the project level (models 5 and 6) and the minimum Herfindahl index among project members (models 7 and 8).

Tables 4 and 5 report the results of zero-inflated negative binomial estimation of CVS commits and file releases respectively. These estimates were obtained by using the same set of regressors used to estimate the logit models above. The same set of explanatory variables was used to estimate the first stage (inflation) equations.

We should note that the equation showed in Table 3 is different from the selection equation used to estimate the zero-inflated negative binomial model reported in Tables 4 and 5. The former relies on a wide set of 'soft' and 'hard' signals of project activity - from bug fixed to new software releases. The latter only focuses on a 'hard' set of activities – respectively CVS commits and new file releases. As mentioned before, these are tangible indicators of innovative activity.

Our estimates show that skill EXPERIENCE has a strong positive effect on both project survival and performance. We also computed the marginal effects of the regressors at their sample mean[11]. An increase in the level of experience from the sample mean of 1.489 to 2.95 (one standard deviation above the mean) increases the likelihood of survival of about 14.7% points, irrespective of model specification.

[11] Marginal effects and predicted values of the dependent variable at different levels of the regressor of interest are calculated with STATA post-estimation commands by taking the values of the other regressors at the sample mean. All marginal effects are calculated with model specifications with control variables and without interactions and squared terms. As Ai and Norton (2003) have demonstrated, the marginal effect of changes in interacted variables in nonlinear models is different from the marginal effect of the interaction term. The same applies to quadratic terms. To our knowledge, STATA programme computes interaction effects in logit and probit models but not in negative-binomial or zero-inflated NB models. The tables reporting marginal effects are not shown in the paper and are available from the authors.

As illustrated by Table 4, experience has a positive effect on CVS, although the significance of its coefficient fades away when the full set of controls is introduced into the equation. The sign of the coefficient is always negative and significant in the inflation model estimating the "always zero" observations, confirming the results obtained in Table 3 also when focusing only on the production of CVS commits. The predicted CVS commits increases by 20.4 to 21.5 units, according to the model specification, when experience is augmented of one standard deviation above the sample mean.

As Table 3 clearly shows, all measures of skill diversity have positive effects on project survival. The significance of some variables decreases when controls are introduced in the model. The measures of skill diversity which remain significant with controls are the standard deviation of the level of experience and the standard deviation of the number of skills of project members. Note that they are both measures of skill diversity across the members of the project. An increase in the standard deviation of the number of skills from 0.40 (the sample mean) to 1.63 (one SD above the mean) increases the likelihood of survival by 1.5% points. By the same token, when this variable increases by one standard deviation above the sample mean the predicted number of CVS commits increases by 3.9 units.

The quadratic terms EXPERIENCE_SD2 and N_SKILLS2 in the logit estimations (models 1 and 3) are negative and significant. This suggests that, beyond a threshold level of diversity, the benefits of skill diversity tend to decrease. The non-linear effects of EXPERIENCE_SD2 disappear in the zero-inflated negative binomial estimation of CVS commits, although they remain significant in the logit inflation model[12].

NSUB_PROJECTS, our proxy for modularity, has also a positive and significant effect on project survival, as shown by all models in Table 3. An increase in the number of subprojects from the sample mean (0.36) of one standard deviation above the mean (1.52) increases the likelihood of survival of 1.1 to 1.2% points, according to the model specification.

Modularity has also a positive effect on CVS commits which is robust to alternative specifications of the model (see Table 4). An increase in the number of subprojects of one standard deviation yields an increase in the predicted number of CVS commits in the range of 3.1 to 4.8 units, according to model specification.

Finally, the interaction between various measures of skill diversity and the number of subprojects yields negative effects on survival. It is important to note that these negative

[12] It is also worth noting that in unreported regressions we exclude the quadratic variables and the sign and significance levels of the coefficients of the other regressors do not change.

interaction effects appear to be strong also in the CVS estimations. As Table 4 shows, the interaction between the number of subprojects and EXPERIENCE_SD (model 2) is negative and significant as well as the interaction between number of subprojects and N_SKILLS_SD in model 4. Similarly, the interactions with the Herfindahl indexes are negative (model 6 and 8), showing that high levels of diversification at the project level and high level of modularity yield negative effects on performance.

Apparently, this contrasts the view that flexible, modular design systems call for high levels of skill diversity. Our results indicate that projects with a large number of modules and homogeneous skill sets (as well as projects with few modules and heterogeneous skills) outperform highly modularized projects with heterogeneous skill sets. Why do projects that have adopted a more decentralized design approach, signalled by a large number of presumably independent subprojects, benefit from limited skill differences among participants?

To answer this question we should remind that OSS is a collective invention based on modular product architectures. However, there are differences among projects in terms of product complexity (i.e., the strength of interdependencies among modules) or the design strategy pursued by project leaders[13]. The division of the collective invention labour is then affected by product complexity. A higher level of complexity reduces the possibility to divide the work into separate tasks (subprojects) and leads to a more centralized division of labour. In a centralized setting coordination costs among different interconnected tasks and contributors are quite limited. The benefits arising from differences in skill experience among participants (e.g., mutual learning and creativity) then can be easily translated into higher levels of productivity with limited effects on coordination costs and the control of system interdependencies (among tasks and modules). The productivity gains of this division of collective labour then draws on the benefits typical of a workshop-like knowledge production system. The flexibility of the system does not rely much on organizational modularity but depends on skill diversity, imitation and knowledge transfer among participants.

Projects with a large division of problem-solving labour among subprojects probably focus on less complex product architectures, i.e., a high level of modularity and finer problem decomposition.[14] In this organizational setting, corresponding to a 'factory-like' production approach, problem solving is more decentralized and knowledge is highly partitioned. This

[13] Complexity is in part endogenous to the problem-solver strategy as discussed by von Hippel (1990) and Arora and Gambardella (1990)

can bring about higher level of skill diversity among participants. Marked differences among participants (e.g., in skill experience) imply mutual learning, creativity and many potential inputs. However, the variance in the quality of inputs tends to increase with the level of heterogeneity of contributors and this imposes high monitoring costs on project core developers. Our results suggest that the monitoring costs associated with high skill variety tend to overcome the benefits when problem solving is decentralized (i.e., modularity is high).

It is possible that the number of subprojects and the division of labour among developers are affected by the size of the project, but we control for project size in our regressions and the coefficients of subprojects and the interaction terms remain significant.

These findings overall suggest that the interaction between skill heterogeneity and the division of problem-solving in open source projects yields quite ambiguous effects on performance that are largely independent from the size of the project. These results then call for a finer-grained, qualitative analysis of the mechanisms underlying the development process and relationships among participants.

Other coefficient estimates are worthy to be briefly discussed. As expected the SIZE of the project is positively related both on survival and performance. Interestingly, late entrants are more likely to survive (see the coefficient of ENTRY_DATE in Table 3), and are less likely to produce zero CVS commits (Table 4, inflation model). However, when projects are active, older projects perform better in terms of number of CVS produced (Table 4, main regression). This is coherent with an environment in which projects enter with some initial level of activity, but only a subset of these projects consolidate over time and become more productive. These results are confirmed in all model specifications.

Project members' external linkages (USERS_EXT_LINKS) have ambiguous effects. In the logit estimations the coefficient is always non significant, while in the negative binomial regressions the coefficient is negative in the inflation model estimating the zero observations, and negative in the main equation, suggesting that participants highly committed with one or few projects are more productive in each project than those who participate in many projects. This does not necessarily imply that participation in several projects has a negative effect on users' ability and experience. Being part of a large network of projects may produce some private benefits to participants but apparently it does not yield significant externalities to the

[14] For a formal treatment of the division of labor in collective problem-solving under different levels of complexity see Marengo and Dosi (2005).

projects. One reason is probably that participation in different projects may yield over-commitment and reduce the attention and motivations towards a specific project. Moreover, the positive coefficient of FORUM would suggest that exposure of the project to the rest of the community is important for productivity, even though the coefficient is almost zero and the nature of messages sent to the forum is very heterogeneous.

We also find that the probability of survival is higher when projects address only less popular audiences or when they address both popular and niche audiences, compared to the baseline case of addressing only principal audiences. Projects that do not address any audience are instead less likely to be active. These results are in line with the hypothesis that congestion effects imply competition for developers time and efforts and the probability of survival becomes lower. However, the coefficient for these dummies are almost never significant in the estimations of the number of CVS.

Finally, we find that projects adopting a General Public License (GPL) model, *ceteris paribus*, are more likely to be active than projects relying on other license schemes. In all our estimations of survival the dummies for LGPL, BSD, Apache, MIT and other licenses (LI_LGPL, LI_BSD, LI_APACHE, LI_MIT, LI_OTHER) are all positive and significant, with the only exception of the public domain license and the artistic license (LI_PUBDOM and the LI_ARTISTIC). These results are also confirmed in our negative binomial estimates, with the exception of the coefficient of LI_BSD that remains positive but non significant.

This amounts to say that adopting a GPL license model (LI_GPL) (the baseline dummy in our regressions) yields a negative effect on performance as compared with other license models. Our findings are consistent with the results obtained by Fershtman and Gandal (2004), who show that restrictive licenses like the GPL are associated with less output per contributor than less restrictive licenses such as BSD. They are also in line with Lerner and Tirole (2005) who reveal that projects running commercial operating systems are less likely to be registered with a restrictive license like the GPL.

The analysis of new file releases confirms in part the results of the CVS estimates (see Table 5). The signs of the main regressors in zero-inflated estimations have the expected signs but their significance is low especially when controls are introduced in the specification.

An important difference with the results of CVS equations is worth to note. In the case of CVS commits, the coefficients of the main regressors remain significant when controls are introduced in the main equation. This reflects the differences between CVS commits and new

file releases discussed before. In particular, file releases are much less frequent events than CVS commits and this explains why it is so difficult to explain the variance in the number of file releases in a short time window. Notice that coefficients of EXPERIENCE in the inflation equation remain significant because the dependent variable is forced to be binary.

To conclude, we have tried different robustness checks. First, we have run separate regressions for groups of projects that differs by their size. In particular we performed separate estimates for small projects (up to 10 members or 98.8 per cent of sample projects) and large projects (with more than 10 members), and the results appear to be similar to those of the pooled regression. The signs of coefficients do not change although their significance fall below the conventional levels, especially for large projects, because of the small number of observations. We have also excluded project with less than two or three members from the analysis. By and large, the results appear to be robust to these checks.

Other robustness checks also yield similar results than those reported in the tables. For instance, we estimated our equations by excluding some outliers (projects with a number of subprojects above the median and the 75^{th} percentile).

We have also used alternative measure of skill heterogeneity, such as entropy, and obtained results similar to those reported above.

Table 3 Logit estimations. Dependent variable: ACTIVE

	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7	Model 8
NO_SKILLS	2.016 (0.134)***	1.673 (0.121)***	1.921 (0.115)***	1.475 (0.119)***	1.415 (0.102)***	1.493 (0.109)***	1.421 (0.102)***	1.494 (0.109)***
NSUB_PROJECTS	0.150 (0.013)***	0.127 (0.028)***	0.106 (0.013)***	0.106 (0.018)***	0.151 (0.013)***	0.300 (0.098)***	0.152 (0.013)***	0.260 (0.097)***
EXPERIENCE	0.566 (0.033)***	0.541 (0.035)***	0.600 (0.034)***	0.544 (0.035)***	0.568 (0.033)***	0.544 (0.035)***	0.565 (0.033)***	0.544 (0.035)***
EXPERIENCE_SD	0.975 (0.175)***	0.195 (0.060)***						
EXPERIENCE_SD2	-0.412 (0.093)***							
NSUB_PROJ_X_EXPER_SD		-0.080 (0.029)***						
N_SKILLS			0.016 (0.006)***	-0.013 (0.006)**				
N_SKILLS_SD			0.325 (0.033)***	0.094 (0.017)***				
N_SKILLS_SD2			-0.030 (0.005)***					
NSUB_PROJ_X_N_SKILLS_SD				-0.025 (0.006)***				
1-HERF_SKILLS					0.607 (0.130)***	0.211 (0.137)		
NSUB_PROJ_X_ 1-HERF_SKILLS						-0.271 (0.110)**		
1-HERF_MIN							0.525 (0.129)***	0.187 (0.137)
NSUB_PROJ_X_ 1-HERF_MIN								-0.227 (0.109)**
Control variables								
SIZE	No	0.175 (0.008)***	No	0.166 (0.008)***	No	0.175 (0.008)***	No	0.175 (0.008)***
ENTRY_DATE	No	0.055 (0.002)***	No	0.056 (0.002)***	No	0.056 (0.002)***	No	0.056 (0.002)***
FORUM	No	0.005 (0.000)***	No	0.005 (0.000)***	No	0.005 (0.000)***	No	0.005 (0.000)***
USERSLINKS	No	-0.007 (0.011)	No	-0.003 (0.011)	No	-0.007 (0.011)	No	-0.007 (0.011)
AUDIENCE_dummies	No	Yes	No	Yes	No	Yes	No	Yes
LI_dummies	No	Yes	No	Yes	No	Yes	No	Yes
PL_dummies	No	Yes	No	Yes	No	Yes	No	Yes
NL_dummies	No	Yes	No	Yes	No	Yes	No	Yes
Constant	-3.496 (0.132)***	-4.861 (0.142)***	-3.392 (0.112)***	-4.659 (0.137)***	-3.501 (0.151)***	-4.870 (0.175)***	-3.426 (0.150)***	-4.847 (0.175)***
Observations	24954	24954	24954	24954	24954	24954	24954	24954
Log Likelihood	-12298.393	-11342.782	-12189.465	-11332.18	-12307.306	-11346.823	-12310.169	-11347.79
Prob > chi2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Pseudo R2	0.020	0.096	0.028	0.097	0.019	0.096	0.019	0.096

* p<0.10, **p<0.05, ***p<0.01. Standard errors in parenthesis.

Table 4. Zero inflated negative binomial. Dependent variable: CVS commit. NEGATIVE BINOMIAL REGRESSION

	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7	Model 8
NO_SKILLS	0.719 (0.268)***	0.518 (0.251)**	0.454 (0.216)**	0.175 (0.244)	0.156 (0.212)	0.503 (0.230)**	0.158 (0.213)	0.499 (0.229)**
NSUB_PROJECTS	0.138 (0.021)***	0.161 (0.047)***	0.121 (0.021)***	0.097 (0.026)***	0.135 (0.021)***	0.360 (0.146)	0.135 (0.021)***	0.336 (0.139)**
EXPERIENCE	0.158 (0.067)**	0.110 (0.071)	0.129 (0.064)**	0.085 (0.071)	0.148 (0.069)**	0.100 (0.073)	0.146 (0.069)**	0.101 (0.073)
EXPERIENCE_SD	0.762 (0.308)**	-0.106 (0.124)						
EXPERIENCE_SD2	-0.261 (0.265)							
NSUB_PROJ_X_EXPER_SD		-0.096 (0.045)**						
N_SKILLS			-0.000 (0.009)	-0.036 (0.010)***				
N_SKILLS_SD			0.320 (0.051)***	0.097 (0.029)***				
N_SKILLS_SD2			-0.031 (0.006)***					
NSUB_PROJ_X_N_SKILLS_SD				-0.008 (0.007)				
1-HERF_SKILLS					0.387 (0.219)*	-0.156 (0.315)		
NSUB_PROJ_X_ 1-HERF_SKILLS						-0.323 (0.159)**		
1-HERF_MIN							0.340 (0.223)	-0.113 (0.308)
NSUB_PROJ_X_ 1-HERF_MIN								-0.298 (0.153)*
Control variables								
SIZE	No	0.080 (0.009)***	No	0.083 (0.010)***	No	0.084 (0.009)***	No	0.084 (0.009)***
ENTRY_DATE	No	-0.011 (0.004)**	No	-0.011 (0.004)**	No	-0.011 (0.004)***	No	-0.011 (0.004)**
FORUM	No	0.000 (0.000)**	No	0.000 (0.000)*	No	0.000 (0.000)**	No	0.000 (0.000)**
USERSLINKS	No	-0.055 (0.022)**	No	-0.044 (0.021)**	No	-0.047 (0.022)**	No	-0.048 (0.022)**
AUDIENCE_dummies	No	Yes	No	Yes	No	Yes	No	Yes
LI_dummies	No	Yes	No	Yes	No	Yes	No	Yes
PL_dummies	No	Yes	No	Yes	No	Yes	No	Yes
NL_dummies	No	Yes	No	Yes	No	Yes	No	Yes
Constant	3.981 (0.265)***	4.167 (0.291)***	4.270 (0.212)***	4.555 (0.267)***	4.155 (0.293)***	4.348 (0.381)***	4.201 (0.296)***	4.311 (0.377)***

Table 4 (cont). Zero inflated negative binomial: CVS commit – INFLATION MODEL: LOGIT

	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7	Model 8
NO_SKILLS	-2.186 (0.163)***	-1.878 (0.164)***	-2.025 (0.138)***	-1.630 (0.158)***	-1.392 (0.123)***	-1.656 (0.152)***	-1.398 (0.123)***	-1.655 (0.152)***
NSUB_PROJECTS	-0.197 (0.017)***	-0.192 (0.040)***	-0.143 (0.017)***	-0.146 (0.026)***	-0.198 (0.017)***	-0.319 (0.160)***	-0.200 (0.017)***	-0.258 (0.152)*
EXPERIENCE	-0.632 (0.040)***	-0.604 (0.048)***	-0.671 (0.041)***	-0.605 (0.047)***	-0.639 (0.040)***	-0.617 (0.049)***	-0.635 (0.040)***	-0.617 (0.049)***
EXPERIENCE_SD	-1.290 (0.212)***	-0.272 (0.080)***						
EXPERIENCE_SD2	0.552 (0.113)***							
NSUB_PROJ_X_EXPER_SD		0.121 (0.047)**						
N_SKILLS			-0.022 (0.006)***	0.007 (0.008)				
N_SKILLS_SD			-0.352 (0.038)***	-0.075 (0.023)***				
N_SKILLS_SD2			-0.032 (0.006)***					
NSUB_PROJ_X_N_SKILLS_SD				0.044 (0.009)***				
1-HERF_SKILLS					-0.892 (0.158)***	-0.247 (0.188)		
NSUB_PROJ_X_ 1-HERF_SKILLS						0.234 (0.178)		
1-HERF_MIN							-0.794 (0.157)***	-0.232 (0.188)
NSUB_PROJ_X_ 1-HERF_MIN								0.166 (0.170)
Control variables								
SIZE	No	-0.315 (0.022)***	No	-0.293 (0.020)***	No	-0.325 (0.022)***	No	-0.325 (0.022)***
ENTRY_DATE	No	-0.063 (0.004)***	No	-0.061 (0.004)***	No	-0.064 (0.004)***	No	-0.064 (0.004)***
FORUM	No	-0.023 (0.007)***	No	-0.019 (0.004)***	No	-0.028 (0.008)***	No	-0.028 (0.008)***
USERSLINKS	No	-0.037 (0.014)***	No	-0.038 (0.014)***	No	-0.039 (0.014)***	No	-0.039 (0.014)***
AUDIENCE, LI, PL, NL_dummies	No	Yes	No	Yes	No	Yes	No	Yes
Constant	3.803 (0.160)***	5.600 (0.212)***	3.652 (0.135)***	5.299 (0.192)***	3.898 (0.185)***	5.647 (0.256)***	3.807 (0.183)***	5.631 (0.255)***
/lnalpha	1.439 (0.046)***	1.662 (0.094)***	1.393 (0.045)***	1.589 (0.067)***	1.444 (0.047)***	1.718 (0.086)***	1.445 (0.047)***	1.719 (0.086)***
Alpha	4.216 (0.196)	5.270 (0.495)	4.026 (0.180)	4.900 (0.330)	4.240 (0.198)	5.572 (0.481)	4.243 (0.198)	5.580 (0.481)
Vuong test (zinb vs. nb)	5.70 (0.000)***	13.92 (0.000) ***	6.67 (0.000)***	13.92 (0.000) ***	5.65 (0.000)***	13.97 (0.000) ***	5.60(0.000)	13.97 (0.000) ***
Observations	24954	24954	24954	24954	24954	24954	24954	24954
Nonzero Obs.	3858	3858	3858	3858	3858	3858	3858	3858
Log Likelihood	-32713.4	-31630.2	-32564.6	-31623.8	-32726.2	-31635.7	-32730.4	-31636.9
Prob > chi2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

* p<0.10, **p<0.05, ***p<0.01. Standard errors in parenthesis.

Table 5. Zero Inflated Negative Binomial: File releases

	Model 1	Model 2
Negative binomial regression		
NO_SKILLS	0.045 (0.254)	0.199 (0.239)
NSUB_PROJECTS	0.066 (0.023)***	0.042 (0.044)
EXPERIENCE	0.035 (0.073)	0.084 (0.071)
EXPERIENCE_SD	-0.027 (0.144)	-0.119 (0.136)
NSUB_PROJ_X_EXPER_SD		-0.030 (0.045)
CONTROLS	No	Yes
Constant	-0.715 (0.290)**	-0.631 (0.281)*
Inflation model: logit		
NO_SKILLS	-3.741 (0.505)***	-2.328 (0.295)***
NSUB_PROJECTS	-0.002 (0.073)	-0.054 (0.068)
EXPERIENCE	-1.144 (0.185)***	-0.682 (0.095)***
EXPERIENCE_SD	-0.284 (0.250)	-0.294 (0.164)*
NSUB_PROJ_X_EXPER_SD	-0.247 (0.118)**	-0.023 (0.078)
CONTROLS	No	Yes
Constant	3.142 (0.379)***	5.601 (0.331)***
/lnalpha	2.176 (0.111)***	1.557 (0.079)***
Alpha	8.815 (0.980)	4.746 (0.374)
Observations	24954	24954
Nonzero Obs.	2758	2758
Log Likelihood	-13640.3	-13208.8
Prob > chi2	0.029	0.000

6. Conclusions

Our analysis provides novel empirical evidence on an important example of collective inventions. The analysis has been driven by two streams of the literature. First, the theory of teams has submitted different contrasting hypotheses about the effect of team composition on team performance. In particular, this paper has addressed the role of skill level and heterogeneity as determinants of performance. Second, the economics of modern production has explored the importance of modular production and design for performance and innovation. Modularity is a key technological characteristic of OSS development. Our analysis has focused on a measure of organizational modularity that is the number of distinct subprojects.

Our empirical results provide support to the hypothesis that skills are important for project performance. To this purpose, we have identified two different measures of project activity. First, as a proxy for project survival we assumed that at least one of the following events must have occurred in the last three moths of the dataset: a fixed bug, a fixed patch, a new feature request fulfilled, a new product release, a new CVS commit.

Second, we used two proxies for performance: new product releases and additions to the project code archive (CVS commits). The former is a proxy for the output of the collective inventive activity whereas the latter measures the inputs supplied by different contributors to the same activity.

Our analysis shows that the level of members' skills (average experience in all skills) have a positive and significant effect on project survival and performance. Similarly, all measures of skill diversity adopted are positively and significantly associated with project survival. Although some effects are not robust to controls, the standard deviation of the level of experience and of the number of skills across project members remains significant when controls are included in the equation. This suggests that OSS projects benefit from an internal division of labour between 'specialists' and 'generalists' and between "high-skilled" and "low-skilled" members. The effect of diversity is however non linear, confirming the hypothesis that after a threshold very high level of diversity produce negative effects on survival and performance.

Our analysis also offers a strong support to the hypothesis that modularity is good for project activity. Although system modularity is a basic characteristic of open source products in general, there may be substantial differences across projects in the level of product and design modularity. Our measure of modularity is based on the number of different subprojects identified by the project team. Clearly, this is an organizational dimension that reflects the development strategy of the project team or the complexity of the product, i.e. the strength of interdependences among its modules or components. Projects that are similar on many respects, like size and the average skill level of the team, may differ in the degree of process modularity for reasons associated to the type of software developed, the type of applications/users addressed or the unobserved ability of project leaders. The estimates of modularity that we observe are robust to several controls.

Finally, we find that the interaction between skill diversity and modularity at the project level yields a negative effect on project performance. This result suggests that there are two organizational approaches to software development in the SF.net environment. The first approach recalls a workshop-like organization since it relies on members with diversified skills who tend to carry out multiple tasks. The other approach is found in projects with a high level of modularity and a low level of skill diversity. This approach reminds us of software factory-like organizations.

We argue that these two models of OSS organization highlight a classical trade-off between 'creativity' and 'control'. High levels of skill diversification are good for 'creativity'

but impose substantial management costs. A high level of skill diversity yields greater variance in the quality of contributions and calls for more control and coordination efforts. When the number of subprojects increases the costs of skill diversity tend to overcome the benefits arising from creativity.

To our knowledge this is one of the first attempts at exploring a large cross-section of OSS projects. Moreover, unlike earlier works this paper does not try to compare OSS with proprietary software. The limitations of the data do not allow using dynamic panel data models or other techniques to treat unobservable fixed effects. Our main regressors, however, are predetermined to moderate the problem of endogeneity.

References

Ai, C.R. and Norton, E.C.(2003) Interation terms in logit and probit models, Economics Letters 80(1):123-129.

Alchian, A.A., Demsetz, H. (1972), "Production, Information Costs, and Economic Organisation", *American Economic Review*, vol. 62, dic., pp. 777-95.

Allen, R. C. (1983) "Collective Invention", *Journal of Economic Behaviour and Organization*, 4, pp. 1-24.

Arora, A., Gambardella, A., (1990), "Complementarity and External Linkages: The Strategies of the Large Firms in Biotechnology", *The Journal of Industrial Economics*, 38 (4), pp. 361-379.

Baldwin, C.Y., Clark, K.B. (2003) "The Architecture of Cooperation: How Code Architecture Mitigates Free Riding in the Open Source Development Model", Harvard Business School, mimeo, Cambridge, Mass.

Baldwin, C.Y., Clark, K.B. (1997) "Managing in an Age of Modularity", *Harvard Business Review*, September-October, 84-93.

Bantel, K. , Jackson, S.E. (1989) Top management and innovations in bankings: does the composition of the top team make a difference?, *Strategic Management Journal*, 10, 107-124.

Baron, J.N., Hannah. M.T. (2002). Organizational Blueprints for Success in High-Tech Start-ups. *California Management Review*, 44(3) 8-36.

Bhidé, A.V. (2000). *The Origin and Evolution of New Business*. Oxford University Press, Oxford.

Blundell R., Griffith, R., Van Reenen, J. (1995), Dynamic count data models of technological innovation, *Economic Journal*, 105, pp. 333-344.

Carbonell P., Rodriguez A.I. (2006), Designing teams for speedy product development: The moderating effect of technological complexity, *Journal of Business Research*, 59, 225-232.

Cohen, W., Levinthal, D. (1989), 'Innovation and Learning: The Two Faces of R&D', *The Economic Journal*, 99, pp. 569–96.

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.

Crowston, K., Annabi, H., Howison, J., and Masango, C. (2004). *Towards a portfolio of FLOSS project success measures*. Paper presented at the Workshop on Open Source Software Engineering, 26th International Conference on Software Engineering, Edinburgh.

Dalle, J.-M. and P.A. David (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*. MIT Press: Cambridge MA. Preprint available at: http://siepr.stanford.edu/papers/pdf/02-27.pdf.

Di Bona, C., S. Ockman, M. Stone, eds. (1999). *Open Sources: Voices from the Open Source Revolution*. O'Reilly, Sebastopol, CA.

Elliot, M.S., Scacchi, W. (2003), Free software: a case study of software development in a virtual organizational culture, Institute for Software Research, University of California, Irvine, CA, mimeo, April.

Faraj, S., Sproull. L. (2000). Coordinating Expertise in Software Development Teams. *Management Science*, 46(12) 1554-1568.

Fershtman, C. and N. Gandal (2004), 'The Determinants of Output Per Contributor in Open Source Projects: An Empirical Examination,' Discussion paper, No. 4329, CEPR, London.

Galunic, D.C., Riordan, S. (1998) Resource re-combinations in the firm: knowledge structures and the potential for Schumpeterian innovation, *Strategic Management Journal*, 19, 1193-1201.

Ghosh, R.A and P.A. David (2003), 'The nature and composition of the Linux kernel developer community: a dynamic analysis,' Working Paper, Project NOSTRA, SIEPR. Draft version available at http://dxm.org/papers/licks1/.

Gonzalez-Barahona J.M., Lopez L., Robles G. (2004) *Community structure of modules in the apache project*, in *Proceedings of the 4th Workshop on Open Source Software Engineering. 26th International Conference on Software Engineering*, Edinburgh, May.

Greene, W. H. (1997), *Econometric Analysis, Prentice-Hall International*. Upper Saddle River, NJ.

Hall, B.H., Ziedonis, R.H. (2001), "The Patent Paradox Revisited: and Empirical Study of Patenting in the US semiconductor Industry 1979-1995", *The RAND Journal of Economics*, 32 (1), pp. 101-128.

Hamilton, B.H., Nickerson, J.A., Owan, H. (2003) "Team incentives and worker heterogeneity: an empirical analysis of the impact of teams on productivity and participation", *Journal of Political Economy*, 1111(3): 465-497.

Harhoff, D., Henkel, J., von Hippel, E. (2003) "Profiting from Voluntary Information Spillovers: how Users Benefit by Freely Revealing their Innovations", *Research Policy*, 32, pp. 1753-1769.

Hausman, J., Hall, B.H., Griliches, Z. (1984), "Econometric Models for Count Data with an Application to the Patents-R & D Relationship", *Econometrica*, 52, (4), pp. 909-938

Howison, J., Conklin, M. S., Crowston, K. (2005), *OSSmole: A collaborative repository for FLOSS research data and analysis*, in *Proceedings of the 1st International Conference on Open Source Systems (OSS 2005)*, Genova, Italy.

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*.

Johnson, J. P. (2002) "Open Source Software: Private Provision of a Public Good" *Journal of Economics and Management Strategy*, 2(4): 637-662.

Kaisla, J. (2001), Constitutional dynamics of the open source software development, Dept. of Industrial Economics and Strategy, Copenhagen Business School, mimeo, May.

Kandel, E., Lazear, E.P. (1992) "Peer Pressure and Partnership", *Journal of Political Economy*, 100: 801-17.

Kiczales, G. (1996) Beyond the Black Box: Open Implementation, *IEEE Software*, January, pp. 8-10.

Krishnamurthy, S. 2002. Cave or Community? An Empirical Examination of 100 Mature Open Source Projects. First Monday 7(6). http://firstmonday.org/issues/issue7_6/.

Kuan, J. (2001) open source software as consumer integration into production, Haas School of Business, University of California at Berkeley, CA, mimeo, January.

Lakhani, K., von Hippel E. (2000), *How open Source software works: 'Free' user-to-user assistance*. Working paper 4117, MIT Sloan school of Management.

Laursen, K., Mahnke, V., Vejrup-Hansen, P. (2005) Do differences make a difference? The impact of human capital diversity, experience and compensation on firm performance in engineering consulting, Druid Working Paper No. 05-04, Danish Research Unit for Industrial Dynamics, Copenhagen.

Leiponen A. (2005), Skills and innovation, *International Journal of Industrial Organization*, 232, 303-323.

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

Lerner, J., Tirole, J. 2005, "The Scope of Open Source Licensing" . *Journal of Law, Economics, and Organization*, Vol. 21, No. 1, pp. 20-56,

Lippman, S.A., Rumelt, R.O. (1982) Uncertain imitability: An analysis of interfirm differences in efficiency under competition, *Bell Journal of Economics*. 13, 418-438.

Lopez-Fernandez L., Robles G., Gonzalez-Barahona J.M. (2004), *Applying Social Network Analysis to the Information in CVS* Repositories, in *Proceedings of 1st International Workshop on Mining Software Repositories* (MSR2004), pp. 101–105.

Madey G., Freeh V., Tynan R. (2004), *Modeling the free/open source software community: A quantitative investigation*, in Koch S. (ed.), *Free/Open Source Software Development*, p. 203-220, Idea Group Publishing, Hershey, PA.

Marengo, L, Dosi, G. (2005). "Division of labor, organizational coordination and market mechanisms in collective problem-solving," *Journal of Economic Behavior and Organization*, Elsevier, vol. 58(2), pp. 303-326.

Milgrom, P., Roberts, J. (1990) "The Economics of Modern Manufacturing", *American Economic Review*, 80 (3), pp. 511-28.

Milgrom, P., Roberts, J. (1995) "Complementarities and fit. Strategy, structure, and organizational change in modern manufacturing, *Journal of Accounting and Economics*, 19, 179-208.

Mockus, A., R.T. Fielding, J. Herbsleb. (2000), A Case Study of Open Source Software Development: The Apache Server. *Proc. of the Twenty-Second Internat. Conf. on Software Engineering* 263 –272.

Mohen, P. , Roller, L.H. (2005) "Complementarity in innovation policy" *European Economic Review* 49(5), 1431-1450.

Mullahay, J. (1997) Heterogeneity, excess zeros, and the structure of count data models', *Journal of Applied Econometrics*, 12, pp. 337-350.

Parnas, D.L. (1972) On the Criteria to be Used in Decomposing Systems into Modules, *Communications of the ACM*, Vol. 15, No. 12, pp. 1053-1058.

Rainer A., Gale S. (2005), *Evaluating the Quality and the Quantity of Data on open Source Software Projects*, in *Proceedings of the 1st International Conference on Open Source Systems (OSS 2005)*, Genova, Italy.

Raymond, E. S. 1999. Linux and Open-Source Success. *IEEE Software* 16(1) 85-89.

Steinmueller W.E. (1996), 'The U.S. Software Industry: An Analysis and Interpretative History', in Mowery, D. (ed.), pp. 15–52.

Sutton R.I, Hargadon, A. (1997), Brainstorming groups in context: effectiveness in a product design firm, *Administrative Science Quarterly*, 42, 685-718.

Torrisi, S. (1998) Industrial Organization and Innovation. An International Study of the Software Industry, Edward Elgar, Cheltenham.

von Hippel, E. (1988), *The Sources of Innovation,* Oxford University Press, New York.

von Hippel, E. (1990), 'Task Partitioning: An Innovation Process Variable', *Research Policy*, 19, pp. 407–18.

von Hippel, E. (2001), Learning from Open Source Software. *Sloan Management Review* 42(4) 82-86.

Vuong, Q. (1989) Likelihood ratio tests for model selection and non-nested hypotheses' *Econometrica*, 57, 307-334.

Appendix

Table A.1. The architecture of the Joose Project

group_id	***project_name***	***description***
65952	Symbolic Executors	
65952	Web site	The home page for the tool
65952	Theorem Proving	Theorem Proving techniques and associated modules
65952	IDE	The Graphical IDE for the tool
65952	Parsers	The Parsers for the tool
65952	AST	Abstract Syntax Trees
65952	Code Generators	Generate code in other languages e.g. Java
65952	Theory	Theoretical stuff with no direct code related to it
65952	SF project admin	Administration of the SourceForge.net project