International Journal of Industrial Organization 28 (2010) 54–68

Contents lists available at ScienceDirect

International Journal of Industrial Organization

journal homepage: www.elsevier.com/locate/ijio

Skills, division of labor and performance in collective inventions: Evidence from open source software

Paola Giuri [a], Matteo Ploner [b], Francesco Rullani [c], Salvatore Torrisi [a,d,*]

[a] Department of Management, University of Bologna, Via Capo di Lucca, 34, 40126 Bologna, Italy
[b] Department of Economics, University of Trento, Via Inama, 5, 38100 Trento, Italy
[c] Department of Innovation and Organizational Economics, Copenhagen Business School, Kilevej 14A, 2000 Frederiksberg, Denmark
[d] CESPRI-KITES, University L. Bocconi, Milano, Italy

ARTICLE INFO

Article history:
Received 24 September 2006
Received in revised form 6 February 2009
Accepted 2 July 2009
Available online 19 July 2009

JEL classification:
O31
O32
L86

Keywords:
Software
Technological innovation
Human capital
Modularity

ABSTRACT

This paper investigates the 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 skill variety of participants is associated with project performance. We also explore whether the level of modularization of project activities is correlated with performance. Our econometric estimations show that skill heterogeneity is associated with project survival and performance. However, the relationship between skill diversity and performance is non-monotonic. Design modularity is also positively associated with the performance of the project. Finally, the interaction between skill heterogeneity and modularity is negatively associated with performance.

1. Introduction

Collective inventions among profit-seeking individuals and organizations have become popular in the economics literature since the seminal paper of Robert Allen (1983) on the iron district of Cleveland in the nineteenth century. More recently, collective inventions have come to the forefront of economists' attention because of the diffusion of open source software (OSS hereafter). OSS can be viewed as a 'virtual' community of practice made up of inventors who voluntarily contribute to multiple collective inventions. OSS offers expert developers the opportunity to participate in innovation networks which are, to some extent, reminiscent of the communities of users in the early age of computing (Steinmueller, 1996; Torrisi, 1998) or other user-centered innovation processes such as those analyzed by von Hippel (1988).

Most studies have attempted to explain why a growing number of independent developers ('hackers') voluntarily disclose their inventions. Several theoretical works seek to understand not only the motivations for disclosure of the source code, but also the social norms and the patterns of collaboration among distributed developers, and the implications for efficiency and social welfare (e.g. Raymond, 1999; von Hippel, 2001; Lerner and Tirole, 2002; Johnson, 2002; Harhoff et al., 2003; Dalle and David, 2005).

Empirical studies (e.g. Lakhani and von Hippel, 2003; Hertel et al., 2003; Lakhani and Wolf, 2005) also ask why hackers freely reveal information and what is the contribution of single participants to the productivity of specific OSS projects. However, little is known about the determinants of OSS projects' performance on a larger scale.

Our paper uses a large sample of OSS teams to study the association between a project's performance (measured by bugs and patches fixed, new feature requests completed, new file releases and changes made to the project's source code) and two important dimensions of team production — skill composition and the level of modularity of project activities.

Our analysis draws on two streams of the literature. The first one is rooted into team production theory. Team production requires collaborative skills, i.e. communication ability (people skills), leadership, and the ability to carry out multiple tasks. These skills add to specialized technical skills, thereby expanding production possibilities. Collaborative skills also favor the "discovery of ways to assign, organize, and perhaps alter tasks to produce more efficiently" (Hamilton et al., 2003: p. 470). Moreover, most importantly for this paper, heterogeneity among team members favors mutual learning and intra-team bargaining, creating opportunities for nonmonetary benefits such as a stimulating working environment, peer recognition and decisional authority (Hamilton et al., 2003).

* Corresponding author. Department of Management, University of Bologna, Via Capo di Lucca, 34 40126 Bologna, Italy. Tel.: +39 051 2098061; fax: +39 051 246411.
E-mail address: torrisi@unibo.it (S. Torrisi).

0167-7187/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.ijindorg.2009.07.004

In this setting we analyze the association between project performance and skill heterogeneity of its members. We expect that skill heterogeneity is positively associated with project performance in teams of open source software developers (Galunic and Riordan, 1998; Sutton and Hargadon, 1997). We focus on two dimensions of skill heterogeneity. First, individual participants must be prepared to carry out multiple tasks whose fulfillment requires a variety of skills. Second, open source participants may have different levels of commitment to single projects. In particular, we can distinguish core developers, who are highly committed and presumably highly experienced people, from the varied community of contributors, who occasionally participate in problem solving by supplying patches, reporting bugs or asking for assistance. Then, it is likely that the level and composition of skills vary across different categories of participants.

The second research line is associated with a key characteristic of the modern organization design that is modularity (Milgrom and Roberts, 1990, 1995). Modularity in design and production 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, p. 84). In modular production, the value generated by each module can be separated from the total outcome. Moreover, modularity allows for experimentation and innovation, increases the efficiency of design activities, favors mutual learning between team members, and stimulates innovation (Baldwin and Clark, 1997; Langlois, 2002; Pil and Cohen, 2006). Thus, we posit that the level of design modularization or division of tasks at the project level is correlated with observable differences in performance across open source software projects.

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 a few open source software projects, we provide an empirical investigation based on a large sample of OSS projects hosted by the SourceForge.net website, one of the largest repositories of OSS activity. To our knowledge, this is one of the few attempts to provide a systematic empirical analysis of multiple dimensions of OSS projects. Second, we focus on a crucial economic issue and examine open source projects with the aim of understanding the association between performance and key project characteristics — team members' skill composition and design modularity.

This paper is organized as follows. Section 2 discusses the theoretical background. Section 3 presents the data. Section 4 illustrates the methodology for estimating the relationship between skills and modularity and project performance. Section 5 analyzes the empirical results. Section 6 concludes.

2. Theoretical background

Our paper focuses on two dimensions of collective inventions: (i) the diversity of skills of team members and (ii) modularity.

2.1. Diversity of skills

The economics literature has analyzed the association between skills and innovation. Human capital is found to be an important input to innovative activity in several empirical studies (e.g. Leiponen, 2005; Mohen and Roller, 2005). Skills are not only important for creating new ideas, but also for using new technologies and absorbing knowledge generated elsewhere. A vast body of the literature on productivity growth has demonstrated the complementarity between skills and investments in new technologies (e.g. Bresnahan et al., 2002).

The implications of skill heterogeneity for productivity and innovation have been less explored in the literature. First, skill heterogeneity implies that firms can experiment with 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 environments (Galunic and Riordan, 1998). Skill heterogeneity provides firms with more comprehensive problem-solving ability and creative conflict resolution (Sutton and Hargadon, 1997; Galunic and Riordan, 1998). The cognitive diversity resulting from 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. Even if the participation decision is beyond the scope of our paper, we should recall briefly the reasons why heterogeneous individuals decide to participate in the same team. This is important to our purposes because, as Hamilton et al. (2003) have noted, "the productivity level achieved by the team is limited by the productivity of the highest-ability worker on the team, and this worker will not join a team without an additional source of surplus from team production" (p. 472). While it is quite obvious why a low-ability individual joins a team, the participation incentives of high-ability individuals are much less clear. Higher-skilled workers have a higher outside option and a greater bargaining power; therefore, they can affect the work norm and induce a higher level of team productivity. Moreover, highest-ability workers in team production systems may sacrifice some income in exchange for non-pecuniary benefits, such as socialization, a higher social status, greater decisional authority among peers, and a more challenging working environment (Hamilton et al., 2003).

Various empirical papers examine the benefits of heterogeneous workforce using firm-level and worker-level data. For example, Bantel and Jackson (1989) have analyzed the top management teams of a sample of U.S. banks and showed that more innovative banks have a more diversified set of top manager expertise. Similarly, Hamilton et al. (2003) have analyzed individual and team productivity in a U.S. garment firm over a three-year period, during which the workers could voluntarily switch from traditional production lines to flexible work teams based on a modular production system, U-shaped workplace, and multitasking. They found 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-monotonic 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 skill diversity. Communication costs and misunderstandings lead to negative productivity outcomes that counterbalance the productivity gains arising from the creativity and flexibility advantages discussed above. The negative productivity effects of communication costs have also been noted by Lazear (1999), who has made the point that without a common language, intercultural, global teams cannot gain from diversity. To take advantage of their complementary skills, team members have to reduce communication costs by sharing a common language. This line of reasoning can be extended to OSS teams where coordination costs are primarily due to the spatial dispersion of contributors. Although these costs are moderated by shared beliefs and visions among participants, Lazear's theory of multi-cultural teams suggests that the members of OSS teams should have some overlapping skills to communicate and coordinate their efforts.[1]

The discussion above leads to the following hypotheses. First, we expect a positive relationship between diversity in the skill level and profile of team members and project performance, controlling for the average level of skills. Second, when diversity increases beyond some

[1] We thank an anonymous referee for raising the point of overlapping skills and knowledge as a moderator of communication costs.

given threshold, coordination among team members becomes more difficult and the performance of the team decreases. We therefore expect to find results consistent with an inverted U-shaped relationship between skill heterogeneity and project performance. Finally, some degree of overlapping skills and knowledge is positively associated with project performance because it reduces the cost of coordination among team members.

2.2. Modularity

Coordination is particularly difficult when participation in collective inventions, like OSS, takes place on a voluntary basis and participants are geographically dispersed. Modularity is a powerful mechanism to reduce coordination costs (Baldwin and Clark, 2006). The implications of modularity for productivity in modern production systems have been analyzed formally by Milgrom and Roberts (1990, 1995). Modularity has been defined as "a strategy for organizing complex products and processes efficiently" (Baldwin and Clark, 1997, p. 86). Modularity relies on system architectures that define the set of modules and their respective 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). The implications of modularity for productivity and innovation are quite clear. Once the overall architectural requirements are met, module designers are free to experiment with different approaches. Baldwin and Clark have noticed, for example, that in the computer industry "the more experiments and the more flexibility each designer has to develop and test the experimental modules, the faster the industry is able to arrive at improved versions" (Baldwin and Clark, 1997, p. 85). Modularity is also important in other industries, such as chemical engineering and financial services where "the intrinsic modularity of financial instruments has been an enormous boost in innovation … for example, designers can split up securities into smaller units that can then be reconfigured into derivative financial products" (Baldwin and Clark, 1997, p. 88). More generally, various empirical works have found that modularity in product design increases the ability to recognize new technological opportunities and to adapt rapidly to changes in market opportunities through innovation and recombination (Pil and Cohen, 2006).[2]

Software design, especially OSS, relies on a modular approach (Baldwin and Clark, 2006). In the case of OSS, the architecture of the system typically consists of a core structure (e.g. the kernel of the GNU/Linux operating system) and a series of modules that are developed independently of one another.

Product modularity – along with coordination tools like CVS (concurrent versioning system) software,[3] forums, and mailing lists – reduces transaction costs and allows for a higher level of task partitioning and a lower level of explicit coordination/interaction among developers. 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 "located in modules that can be selectively compiled and configured" (Mockus et al., 2000, p. 266). Formal organization of authority is adopted in large projects like Linux, where Linus Torvalds has the last word on 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. In many small projects it is likely that, rather than a formal organization, personal motivation, shared beliefs and values tie these virtual teams together (Elliot and Scacchi, 2003). These informal, cultural dimensions reduce communication and coordination costs among developers and increase the gains of modularity.

This discussion leads to the hypothesis that, *ceteris paribus*, a higher level of modularization in the design and production process is associated with a higher performance of the project. 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 at the same time.

2.3. Performance

A number of studies have tried to assess the performance of open source projects relative to proprietary software. For instance, Kuan (2001) 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) 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. (2006) reviewed several alternative indicators of open source project success and popularity such as the speed of response to bug reports. They tested the validity of these measures by using expert interviews and SourceForge data. Moreover, they compared bug-fixing time with the number of core developers and users who sent bug reports, the level of activity provided directly by the SourceForge.net website (http://sourceforge.net, SF.net henceforth), and the number of downloads. Crowston et al. (2006) concluded that a combination of different indicators should be preferred to single proxies for project success. Comino et al. (2007) used the probability of progressing from the initial phases of product development to the maturity and stability stages as a measure of project performance.

Very few studies have explored indicators of performance, like productivity, that are more familiar to economists. For instance, Fershtman and Gandal (2004) 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.

Our choice of performance indicators was driven by the literature and interviews with contributors. We look at different indicators of project performance by exploiting the rich set of information offered by the SF.net dataset about various types of project activities — bugs, patches, new features (trackers), CVS commits, and new product releases. As we explain later in more detail, the CVS is the repository where the lines of code filed by developers are accumulated. Each CVS commit represents an addition or modification to the project code. A new file release represents the launch of a new version of the project code. These variables account for different dimensions of project performance, from minor product changes such as a CVS commit – which modifies or corrects existing lines of code – to more significant innovations embodied in new file releases.

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 SF.net over the period November 3rd 1999 to January 10th 2003.

The number of projects registered at SF.net has increased quite rapidly since its foundation. In May 2008, the number of registered projects was about 178,000 and the number of registered contributors was more than 1,800,000. Other websites hosting open source projects are much smaller. For example, in May 2008 Savannah hosted 2901 active projects and 57,156 registered contributors (http://savannah.gnu.org).

Even though none of these datasets is representative of the entire open source software community, SF.net remains the world's largest pool of projects and developers and one of the most widely used by

[2] Examples of modularity in chemical engineering and other industries are reported by Arora and Gambardella (1994) and Brusoni and Prencipe (2001).

[3] As it will be described in the next Section, CVS is the repository where new or modified lines of code filed by developers are assembled and stocked.

earlier empirical papers (e.g. Krishnamurthy, 2002; Fershtman and Gandal, 2004). To our knowledge, however, only a few works have gained access to the whole sample of projects hosted in SF.net (for example, Madey et al., 2004; Lerner and Tirole, 2005; Comino et al., 2007).

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 OSS 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 reviewing the inputs submitted by various categories of contributors and the most appropriate measures of individual contributions (bugs, new feature requests, lines of code, etc.). From this background information we tried to infer the most appropriate measures of project performance — new releases, fixed bugs or patches, and changes of the project source code (CVS commits). We received responses from 13 users and checked this information through informal face-to-face discussions with experts and SF.net users. From this investigation, CVS commits and file releases emerged as the most reliable quantitative measures of output. The validity of these measures is also confirmed by their use in previous works (e.g. Robles et al., 2006). Furthermore, Crowston et al. (2006) compared different measures of OSS activity and found that file releases and CVS check-ins are valid measures of activity.

The SF.net dataset includes a great deal of information at two different levels of aggregation: the project or team and the single contributor. A project is carried out by a group of contributors working on the same software. A contributor is an individual who contributes to one or more projects. Our version of the dataset consists of 65,535 projects and 544,669 individuals. These individuals 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 are 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 highest number of people ever registered with a project is 90,255, which is only a tiny fraction of all individuals registered with the SF.net.[4] Of these only 26,314 provide data on skills.

The sample we used in this paper includes data on 29,633 individuals who registered with 9076 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 is a trade-off 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 or learning. It is unlikely that skills change substantially over 3 months because of learning. They may change, however, because of modifications in team composition. To account for this effect we considered individuals who 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 would not guarantee a satisfactory 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 all dependent variables.

The procedure described above implies the exclusion of projects registered after September 2002, and this leads to a sample of 58,166 projects. We also excluded projects with missing information on control variables, such as natural language and programming language, and limited the sample to 24,846 projects and 39,533 individuals. Only 17,675 of these participants (registered with 13,354 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 in May 2008 was listed among the top 20 popular projects by Freshmeat.net (http://freshmeat.net). To mitigate this problem, we focused on projects composed of at least two participants. Our final sample is thus composed of 9076 projects and 29,633 individuals; 10,585 individuals belonging to 6704 projects declared their skills.

Our analysis is based on a cross-section of OSS projects because time-series data were available for only a few variables (e.g. projects' members and CVS commits), whereas for other variables (e.g. file releases or number of subprojects) data were more difficult to properly track over time.

3.1. Project performance

We built the following measures of performance: ACTIVE, CVS, and FILE_RELEASE.

ACTIVE is a dummy variable that takes the value of 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 or 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 28% of the projects in our sample were active in the last 3 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.

Fig. 1 illustrates how the share of active projects changes over time in our sample. More precisely, it shows the number of projects that are active in month (trimester) t as a share of sample projects registered with SF.net by month (trimester) t. For example, more than 40% of all projects registered until January 2000 were active in the trimester November 1999–January 2000, whereas about 35% of all projects registered by September, 2002, were active during July–September 2002. The decline in the share of active projects suggests that, over time, an increasing proportion of unproductive projects register with SF.net. As expected, the share of active projects is higher when we pass from a one-month to a three-month time window because the probability that a project undertakes some activity increases with the length of the time duration.

Fig. 2 shows the share of sample projects active in trimester t that were also active in the previous trimester. These data indicate that the share of persistently active projects increases over time and tends to stabilize at the end of the sample period. 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) afterward. The persistence of each project in the ACTIVE status suggests that 3 months is a large enough period for defining our dependent variables.

As a robustness check, we repeated our estimations with different time spans of the dependent variables (i.e. 1 year and

[4] We have underestimated the real number of project members because it is not possible to retrieve information on project membership for the period November 1999–September 2000.

Table 1
Definition of variables and descriptive statistics: main variables.

Variable	Description	Mean	S.D.	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	0.280	0.449	0.000	0.000	1.000
CVS	Number of CVS commits produced by the project in the period October–December 2002	54.482	301.645	0.000	0.000	17,724.000
FILE_RELEASE	Number of files released by the project in the period October–December 2002	0.438	1.666	0.000	0.000	37.000
NSUB_PROJECTS	Cumulated number of subprojects with at least one activated task opened by the project up until September 2002	0.712	1.668	0.000	0.000	45.000
EXPERIENCE[a]	Project members' average experience in the skills they declared (five classes of years)	2.766	0.614	2.720	1.000	5.000
EXPERIENCE_CV[a]	Coefficient of variation of project members' average experience in the skills they declared (five classes of years) obtained dividing the standard deviation of project members' average experience by the project members' average experience	0.100	0.134	0.007	0.000	0.940
EXPERIENCE_CV1[a]	Coefficient of variation of project members' average experience, considering only the first group used in the spline procedure	0.086	0.103	0.007	0.000	0.260
EXPERIENCE_CV2[a]	Coefficient of variation of project members' average experience, considering only the second group used in the spline procedure	0.014	0.050	0.000	0.000	0.680
N_SKILLS_MEAN	Average number of skills declared by the project's members	6.883	3.107	6.660	1.000	23.000
N_SKILLS_CV[a]	Coefficient of variation of project members' number of declared skills (five classes of years) obtained dividing the standard deviation of project members' number of skills by the project members' average number of skills	0.224	0.293	0.000	0.000	1.265
N_SKILLS_CV1[a]	Coefficient of variation of project members' number of declared skills, considering only the first group used in the spline procedure	0.197	0.235	0.000	0.000	0.592
N_SKILLS_ CV 2[a]	Coefficient of variation of project members' number of declared skills, considering only the second group used in the spline procedure	0.028	0.092	0.000	0.000	0.673
DISTANCE[a]	The average Euclidean distance in experience of each project member from all the other members in the project	4.069	4.274	3.606	0.000	16.703
OVERLAP[a]	The measure of the overlap between the project members' skills	0.600	1.403	0.000	0.000	40.810
NO_SKILLS	A dummy variable taking value 1 if none of the users belonging to the project has declared her/his skills set	0.261	0.439	0.000	0.000	1.000
ENTRY_TIME	The registration period of the project measured in months from November, 1999	21.565	7.676	22.000	1.000	34.000
SIZE	Number of members of the project as of September 2002	3.883	3.741	3.000	2.000	82.000
LI_LGPL	Project's license: LGPL	0.082	0.275	0	0	1
LI_BSD	Project's license: BSD	0.067	0.250	0	0	1
LI_PUBDOM	Project's license: Public Domain	0.016	0.127	0	0	1
LI_ARTLIC	Project's license: Artistic license	0.025	0.156	0	0	1
LI_APACHE	Project's license: Apache license	0.015	0.120	0	0	1
LI_MIT	Project's license: MIT license	0.016	0.125	0	0	1
LI_OTHER	Project's license: Others	0.069	0.253	0	0	1
NL_GERMAN	Project's official spoken language: German	0.107	0.309	0	0	1
NL_FRENCH	Project's official spoken language: French	0.063	0.244	0	0	1
NL_SPANISH	Project's official spoken language: Spanish	0.034	0.181	0	0	1
NL_RUSSIAN	Project's official spoken language: Russian	0.015	0.120	0	0	1
NL_JAPANESE	Project's official spoken language: Japanese	0.010	0.101	0	0	1
NL_ITALIAN	Project's official spoken language: Italian	0.002	0.049	0	0	1
NL_DUTCH	Project's official spoken language: Dutch	0.001	0.031	0	0	1
NL_P_BRAZ	Project's official spoken language: Portuguese-Brazilian	0.001	0.035	0	0	1
NL_POLISH	Project's official spoken language: Polish	0.001	0.023	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	0.026	0	0	1
NL_OTHER	Project's official spoken language: Others	0.004	0.060	0	0	1
PL_JAVA	Project's programming language: Java	0.245	0.430	0	0	1
PL_PHP	Project's programming language: Php	0.162	0.369	0	0	1
PL_PERL	Project's programming language: Perl	0.120	0.325	0	0	1
PL_PYTHON	Project's programming language: Python	0.058	0.234	0	0	1
PL_VISBASIC	Project's programming language: Visual Basic	0.033	0.179	0	0	1
PL_UNIXSHELL	Project's programming language: Unix Shell	0.026	0.159	0	0	1
PL_OTHERS	Project's programming language: Others	0.200	0.400	0	0	1

Note: Number of observations = 9076.
[a] For the variables using information about skills the number of observations is 6704.

6 months) and obtained qualitatively similar results. The same checks were also carried out with the other dependent variables described below, and most of the results shown in this paper are confirmed.

CVS is the number of CVS commits submitted to the project archive between October and December 2002. CVS software is used to avoid unnecessary duplication of efforts and to integrate different lines of code supplied by different contributors working at the same time on the same module of the program. CVS is also the repository where the lines of code filed by developers are assembled and stocked. When a developer submits a new file to the CVS repository by using a "commit" command, other developers can see this modification of the project source code. Our data do not provide information on the lines of source code per contributor and, indeed, it would be arduous to obtain these 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 members in design and problem solving, rather than code writing as such. The CVS repository offers a wide range of information that can be used to infer the developers' coding activity. Since a CVS commit represents a change to the project code, the number of CVS commits can be viewed as a reasonably good proxy for productivity at the project level. In particular, the number of CVS commits seems to be a better measure of project activity than the simple count of the lines of code, as 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 some lines of

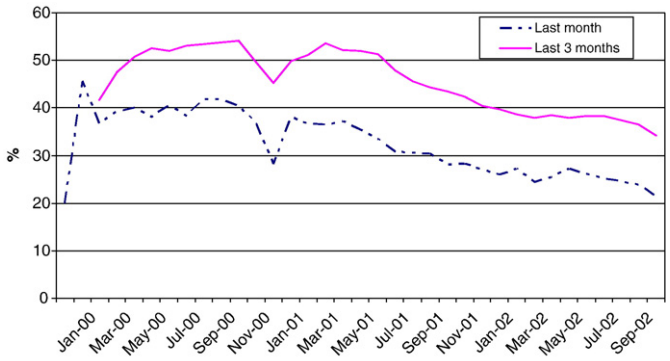

Fig. 1. Share of active projects over projects registered by month (trimester) *t*.

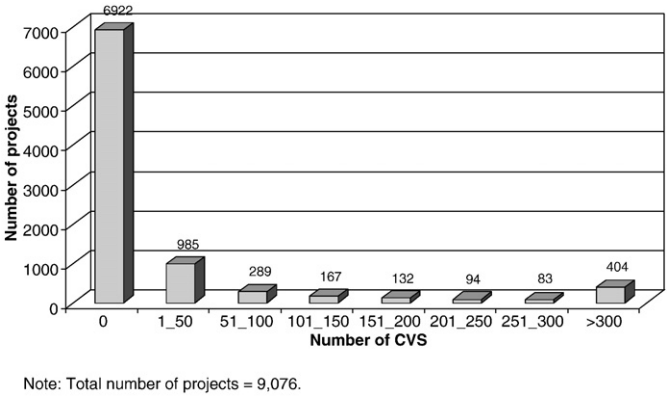

Fig. 3. Distribution of projects by number of CVS commits.

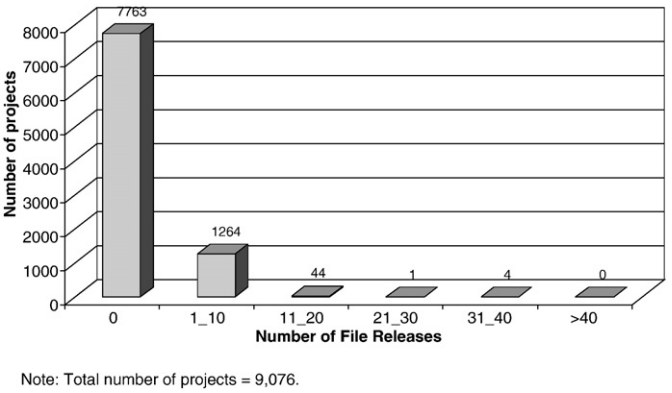

Fig. 4. Distribution of projects by number of file releases.

code. Indeed, earlier works have used it as a measure of developers' coding activity (see, for example, Lopez-Fernandez et al., 2004).

In our sample, the average number of CVS commits in the last 3 months is 54.48 per project, with a minimum of zero and a maximum of more than 17,000 (see Table 1).

FILE_RELEASE. This measure of project performance is obtained by counting the number of the project's file releases between October and December 2002.

A new file release represents the launch of a new version of the software produced by the project. Since the program has to be stable enough to be released, file releases represent a milestone in the development process. This highlights an important difference between file releases and CVS commits. The latter measure the input offered by people involved in project 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 bug 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 (Figs. 3 and 4). In the period October–December 2002, 6922 projects did not produce any CVS commits and 7763 projects did not release any new version of the program. Most projects that produced a positive number of CVS and file releases only produced, respectively, less than 50 CVS commits and less than 10 file releases. Instead, less than 2000 projects have produced more than 50 CVS commits and 404 projects more than 300 CVS commits.

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.

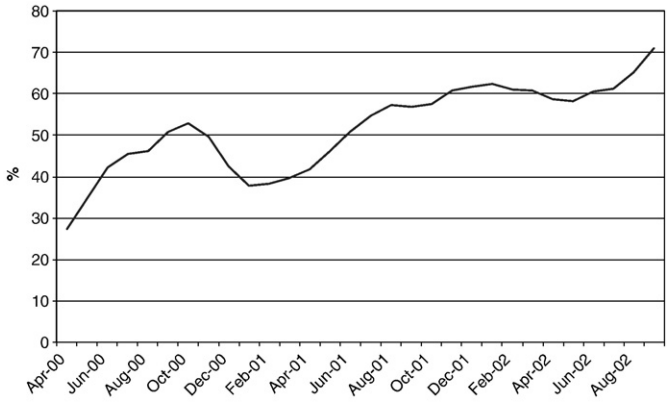

Fig. 2. Share of projects active in the last trimester that were active also in the earlier trimester.

3.2. Main co-variates

3.2.1. Skill level and diversity

In general, data on individual skills are very difficult to obtain. However, the SF.net dataset provides detailed information on 33 types of different skills that 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.[5] This information is provided by users at the time of registration on SF.net, when they are asked to report which skills they possess among the 33 categories mentioned and, for each category, to self-assess the level and experience. Skill profiles can then be updated over time. 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

[5] 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 specialized technologies and tools; 2) design expertise that refers to the knowledge of software design principles and architecture; and 3) domain expertise (knowledge about the application domain and customers operations) (p. 1559).

supplied by developers can be made public to other developers who can check its reliability.[6]

For reasons explained above, we retrieved the information about project membership before September, 2002, i.e. before the time span of project performance and built skill measures only for people belonging to the projects up to that time.

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

EXPERIENCE_MEAN. This variable is computed as the average level of experience of the project 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 to 2 years; 3 = 2 to 5 years; 4 = 5 to 10 years; and 5 = more than 10 years.

EXPERIENCE_CV. We use the coefficient of variation of the project members' experience in order to account for differences in the level of skills across members (for example, a few core developers and other members carrying out lower-level activities). The coefficient is given by the ratio between the standard deviation and the average of the project members' experience.

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

N_SKILLS_CV. We use the coefficient of variation of the number of skills mastered by each member of the project in order to differentiate projects composed by members with a similar skill scope (e.g., all members are specialists in a field) from projects characterized by a diversified skill scope, i.e. projects with both generalists and specialists. The coefficient is given by the ratio between the standard deviation and the average of the number of skills mastered by each project member.

To capture the potential non-linear impact of EXPERIENCE_CV and N_SKILLS_CV on the dependent variable, we employ a piecewise linear transformation (spline) of the two explanatory variables and test the equality of the resulting coefficients by a Wald test. A knot at the 90th percentile was imposed for both variables. This specification was chosen after having explored alternative specifications and having considered the nature of the two variables, both characterized by a high number of observations at the lower end of the distribution.[7] Given the specification employed, the effect of explanatory variable is assumed to be piecewise linear in the two segments generated, e.g. EXPERIENCE_CV1 and EXPERIENCE_CV2.

DISTANCE. To measure diversity in skill sets among project members we calculate the average Euclidean distance of skill experience among them. To this purpose, we first constructed the vector of skills possessed by project member i $(e_{i1}, e_{i2}, e_{ik}\ldots, e_{in})$, where e_{ik} is the experience of project member i in skill k, and calculated the Euclidean distance between member i and member j as follows:

$$D = \sqrt{\sum_{k=1}^{n} (e_{ik} - e_{jk})^2}$$

where e_{jk} is project member j's experience in skill k.

To obtain the variable *DISTANCE,* we computed D for each project member against all other project members and took the average of these measures.[8]

OVERLAP. To capture the overlap between project members' skills, we rely on *beta diversity*, an index developed by biologists to measure "the proportion by which a region…is richer than the average locality…within it" (Harrison et al., 1992, p. 152; see also Whittaker, 1960). In our setting, *beta diversity* (β) is defined as the ratio between the number of different skills in the project (without duplications) over the average number of skills mastered by each project member. For instance, suppose there are two individuals in a project and the skill vectors are $(e_{i1},e_{i2},e_{i5},e_{i8})$ and (e_{j1},e_{j2},e_{j10}) for individuals i and j respectively. β is equal to $5/3.5 = 1.42$. The minimum of β is 1 when the project members have identical skill sets. β increases with the number of non-overlapping skills, and its maximum equals the number of project members. In our dataset its maximum value is 4.96.

Our measure of overlapping skills among project members is inversely correlated with beta diversity

$$OVERLAP = n - \beta = n - \frac{N_{pr}}{\left(\sum_{i}^{n} N_i\right)/n} = n\left(1 - \frac{N_{pr}}{\sum_{i}^{n} N_i}\right)$$

where N_{pr} is the number of different skills in the project (without duplications), N_i is the number of skills mastered by member i, and n is the number of members that declared their skills. OVERLAP is equal to zero when member i's skill vector and member j's skill vector are disjoint for all $i \neq j$. OVERLAP increases when some of the skills of project members overlap and decreases with the number of skills mastered by the project as a whole (N_{pr}).

3.2.2. Modularity

A key hypothesis of this paper is that project performance is correlated with the level of modularity of the project. Our measure of project modularity is precisely the cumulated number of subprojects opened by the project until September 2002 (*NSUB_PROJECTS*).

As mentioned before, the architecture of the system is made up of the main project, where the basic code is developed, and a series of subprojects corresponding to specific modules. The latter focus on specific tasks such as the construction of a website, the production of documentation, the translation of the original code into other programming languages and so on.[9]

Table A.1 in the Appendix illustrates the architecture of a typical project (Joose) in our dataset. The list of subprojects reported in the Table shows that the activity of some of these subprojects involves the production of code for specific modules (e.g. code generators for translation into 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 reflects both the technical architecture of the product and the organization of the development process. Notice that we considered only subprojects that have opened up at least one task, i.e. have initiated an activity to carry out during the subproject life.

In some regressions we also include interaction terms in order to account for potential complementarities between skill diversity and modularity.[10] We use the following variables:

- *NSUB_PROJ_X_EXPER_CV.* Number of subprojects multiplied by the coefficient of variation of the level of experience of the project members.
- *NSUB_PROJ_X_N_SKILLS_CV.* Number of subprojects multiplied by the coefficient of variation of the number of skills of the project members.

[6] Earlier studies based on large samples of the SF.net dataset have also relied on self-reported characteristics of OSS projects (e.g. Lerner and Tirole, 2005).

[7] Estimates based on the 70th percentile show that there is no evidence of nonlinearity for either EXPERIENCE_CV or N_SKILLS_CV, while the coefficients on other regressors remain similar to those reported in the paper.

[8] We also computed Herfindahl and 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 DISTANCE and the results are very similar to those reported here.

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

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

— *NSUB_PROJ_X_ DISTANCE.* Number of subprojects multiplied by the average Euclidean distance of skill experience among project members.
— *NSUB_PROJ_X_OVERLAP.* Number of subprojects multiplied by the index capturing the overlapping of the project members' skills.

3.3. Other regressors and controls

SIZE. Our measure of project size is the number of members registered with the project by September 2002.[11] Note that 53.3% of the 9076 projects in our sample have more than two members and only 16% have more than five members registered.

It is likely that larger projects have a higher degree of division of labor among participants as compared with smaller projects. However, there are differences in the division of labor across projects that may be independent of project size. For example, different core developers or project top members have a different ability for identifying problems and organizing problem-solving activities by separating a complex system into simpler tasks. Our dataset does not allow accounting for these characteristics, but we can observe other dimensions of the project that are described below.

ENTRY_TIME. To control for the age of the project we use a variable indicating the month of registration with SF.net, from November 1999. ENTRY_TIME varies between 1 (for projects entered in the first month) to 39 (for projects entered in the last month).

NO_SKILLS. Dummy equal to 1 if none of project's members provides information on her/his skills.

LI_Dummies. Dummies for the main license initially chosen by the project's founder (GPL, LGPL, BSD, Public Domain, Artistic license, Apache license, MIT license and Others). In our sample the majority of projects (71.01%) are distributed under a general public license (GPL), 8.24% under a lesser general public license (LGPL), 6.71% under a BSD license and the remaining 14.04% of projects are distributed across other types of licenses. In our estimations GPL is used as a baseline dummy.

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

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

4. Econometric method

To study how skills and modularity are associated with project survival and performance we have estimated three sets of equations.

First, we estimated the probability of project 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 change to the project source code (a CVS commit sent to the CVS repository) or a new file release. Our covariates include measures of skills, modularity and controls.

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

The choice of zero-inflated 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 al., 1995). The literature suggests several goodness-of-fit tests to evaluate alternative models to the Poisson specification. 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.[12] Zero-inflated estimators are two-step ML estimators. A binary probability equation (inflation equation) is first estimated to predict the membership in the "always zero" cases. Then a (truncated) negative binomial model describing the distribution of the "not always zero" cases (non-negative outcomes) is estimated.

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 problem of dynamic feedbacks between performance and skills we used lagged values of skill-related variables.[13] We know the history of contributors over the sample period and, therefore, we can see whether a particular contributor leaves a project to join another one. The history of contributors allows us to predetermine the main regressors. To our purposes, it is important to know how rapidly the team composition of the sample projects changes during the sample period. To this aim we analyzed the users' registration status every 30 days starting from December 1, 2001. The following conservative measurement strategy for users' participation was adopted. We focused on the sample of users who registered in Sourceforge between September 1, 2000, and December 31, 2000, and provided information about their skills ($N = 3550$). These users were tracked in the 24 periods between January, 2001, and December, 2002, and their participation status was collected for each of the projects to which they registered. We found that, on average, each user participates to about 1.86 (SD = 1.5) projects and that, on average, the participation extends over 16.8 periods (SD = 7.6), i.e. 70% of the total number of periods. Thus, users' participation is clearly persistent over the time horizon considered.

The small individual mobility across projects that we found in the data suggests that team composition changes very slowly over the time windows analyzed here. It is unlikely then that the composition of the projects observed before the last 3 months changes during the last period and affects our dependent variables.

However, we must notice that using lagged values of regressors does not solve the problem of endogeneity. Our estimates may be biased by self-selection. The skill composition of project teams is likely to reflect characteristics of the projects other than performance such as its size, age and license model.[14]

In order to understand the likely magnitude of the endogeneity problem, in unreported regressions we estimate how our measures of skill heterogeneity vary with observable characteristics of the project — i.e. size, registration time, type of license, official spoken language and programming language. Size represents the number of members of the project team, so that it can be considered predetermined with respect to the skills mastered by its members. It can be viewed as a proxy for the complexity of the project. The project license is chosen by the founder at the time of registration with the SF.net website and is then predetermined. The programming and natural languages are also predetermined because they are normally set at the time of

[11] In January 2003 SF.net hosts 65,535 projects — 11,262 of them (17.18%) have no members, 36,389 (55.53%) have just one member, and the rest (17,884) more than 1. Among all the projects with at least one member (54,273), 67.05% have only one registered member.

[12] A goodness-of-fit test for zero-inflated models against the standard Poisson or negative binomial models in the 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.

[13] A similar strategy is often adopted in studies aiming at analyzing the effect of endogenous variables. For example, Caroli and Van Reenen (2001) have used lagged values of organizational change in a variable cost share equation. Similarly, Bresnahan et al. (2002) use lagged proxies for organizational change, ICT investment and skills in a production function estimation.

[14] We thank one of the referees for making this point clearly.

Table 2
Correlations between main variables.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
(1) NSUB_PROJECTS	1								
(2) EXPERIENCE	−0.0009	1							
(3) EXPERIENCE_CV	0.1364*	−0.0940*	1						
(4) N_SKILLS_MEAN	0.0591*	−0.0237	−0.0486*	1					
(5) N_SKILLS_CV	0.1281*	−0.0241*	0.6166*	−0.0634*	1				
(6) OVERLAP	0.1994*	−0.0234	0.3469*	0.0470*	0.3192*	1			
(7) DISTANCE	0.1615*	0.0791*	0.6792*	0.1138*	0.7365*	0.4024*	1		
(8) ENTRY_TIME	−0.0831*	−0.0715*	−0.0768*	−0.0238	−0.0783*	−0.0916*	−0.1129*	1	
(9) SIZE	0.2282*	0.0528*	0.2882*	−0.0224	0.2941*	0.7564*	0.3248*	−0.1279*	1
(10) LI_LGPL	−0.0107	0.0374*	−0.0193	0.0449*	−0.0113	−0.0108	−0.0102	0.0082	−0.0073

For the variables using information about skills the number of observations is 6704.
* $p<0.05$. Note: Number of observations = 9076.

project registration and do not change much over time. Fershtman and Gandal (2004) noted that these variables changed only rarely in a sample of 71 SF.net projects. Larger, more challenging projects are likely to attract contributors endowed with different skill sets and have then a higher probability to outperform smaller, less attractive projects.

The linkage between modularity and performance is affected by similar problems of self-selection. For this reason we have examined how modularity varies with the same exogenous set of a project's characteristics used in skill equations. The results of these estimations, reported later, indicate that we cannot solve the endogeneity problem because the set of project characteristics that affect skill composition and modularity (size, registration time, programming language, etc.) directly affect project performance and cannot be used then as instruments for our regressors. This leads us to say that we cannot make inferences about causality between skill heterogeneity and modularity and project performance. We can instead view our findings as multiple correlations among project performance, team composition and modularity, controlling for a series of exogenous project level variables. In the concluding section we explain why our findings are interesting even if we cannot establish causal relationships among these variables.

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 9076 projects. For about 26% of these projects (2372) 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 checked for statistically significant differences in the two distributions. About 29.13% of sample projects that report data on skills are active while the same percentage is equal to 24.83% among projects for which data on skills are not available. Moreover, 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 59 (SD = 258.46) for projects with missing skills as against 40 (SD = 398.94) for projects that report data on skills. Projects reporting skill data are also characterized by more file releases (mean = 0.476, SD = 1.768) than those that do not report data on skills (mean = 0.328, SD = 1.333). Finally, projects reporting skill data are bigger in size (mean = 4.26, SD = 4.166 vs. mean = 2.82, SD = 1.719) and have more subprojects (mean = 0.83, SD = 1.818 vs. mean = 0.37, SD = 1.067). A statistically significant difference between the two subsamples is registered in each of the dimensions above reported (Pearson's chi-squared test, p-value < 0.01 for all comparisons). Overall, projects with missing data on skills are less likely to be active, are smaller in size, are less productive in terms of CVS commits and file released and have a limited number of subprojects when compared to projects with data on skills. To avoid coefficients of skill-related variables being biased we introduced dummy variables for observations with missing data (see Hall and Ziedonis, 2001 for a similar treatment of potential selection on key regressors).

Table 2 reports the correlations 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 correlated with the coefficient of variation of the project members' experience (ρ=0.29) and with the coefficient of variation of the number of different skills (ρ=0.29), its standard deviation (ρ=0.445), and with the number of subprojects (ρ=0.228).

The correlation between CVS commits and file releases is significant and positive, but quite weak (ρ=0.267). This is because 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 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.

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 subprojects and the coefficient of variation of experience. This specification also contains the full set of project-specific controls such as size, age, type of license model, natural language and programming language of the project. The remaining specifications contain alternative measures of project skills and skill diversity with and without controls: the total number of different skills and the coefficient of variation of the number of skills (Models 3 and 4), the Euclidean distance of skills at the project level (Models 5 and 6) and the overlap of skills among project members (Models 7 and 8).

Tables 4a,4b and 5a,5b report zero-inflated negative binomial estimation results for CVS commits and file releases respectively. These results 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.[15]

[15] We should note that the equation shown 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 signals of project activity — from bug fixed to new software releases. The latter only focuses on CVS commits and new file releases.

Table 3
Logit estimates.

	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7	Model 8
NO_SKILLS	2.124*** (0.150)	1.863*** (0.154)	2.174*** (0.166)	1.977*** (0.171)	2.043*** (0.146)	1.929*** (0.152)	2.095*** (0.146)	1.827*** (0.151)
NSUB_PROJECTS	0.075*** (0.014)	0.057*** (0.021)	0.071*** (0.014)	0.059*** (0.022)	0.073*** (0.014)	0.069*** (0.023)	0.058*** (0.015)	0.056*** (0.017)
EXPERIENCE	0.762*** (0.046)	0.681*** (0.048)	0.764*** (0.046)	0.694*** (0.048)	0.721*** (0.046)	0.686*** (0.048)	0.761*** (0.046)	0.672*** (0.048)
EXPERIENCE_CV1	2.025*** (0.304)							
EXPERIENCE_CV2	−1.760*** (0.668)							
EXPERIENCE_CV		−0.004 (0.254)						
NSUB_PROJ_X_EXPER_CV		−0.155 (0.105)						
N_SKILLS_MEAN			0.000 (0.009)	0.004 (0.009)				
N_SKILLS_CV1			1.059*** (0.135)					
N_SKILLS_CV2			−0.622* (0.335)					
N_SKILLS_CV				0.261** (0.111)				
NSUB_PROJ_X_N_SKILLS_CV				−0.084* (0.051)				
DISTANCE					0.045*** (0.006)	0.015** (0.007)		
NSUB_PROJ_X_DISTANCE						−0.006** (0.003)		
OVERLAP							0.219*** (0.024)	−0.069* (0.038)
NSUB_PROJ_X_OVERLAP								−0.014* (0.007)
ENTRY_TIME		0.018*** (0.003)		0.018*** (0.003)		0.018*** (0.003)		0.018*** (0.003)
SIZE		0.136*** (0.009)		0.130*** (0.009)		0.131*** (0.009)		0.154*** (0.011)
LI_dummies	No	Yes	No	Yes	No	Yes	No	Yes
NL_dummies	No	Yes	No	Yes	No	Yes	No	Yes
PL_dummies	No	Yes	No	Yes	No	Yes	No	Yes
Constant	−3.261*** (0.142)	−3.814*** (0.174)	−3.308*** (0.159)	−3.917*** (0.189)	−3.179*** (0.138)	−3.874*** (0.172)	−3.225*** (0.138)	−3.830*** (0.170)
Observations	9076	9076	9076	9076	9076	9076	9076	9076
Log Likelihood	−5198.89	−4918.031	−5187.559	−4916.228	−5196.556	−4916.454	−5172.402	−4913.048
Prob>chi2	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001
Pseudo R2	0.0341	0.0863	0.0362	0.0866	0.0345	0.0866	0.0390	0.0872

Dependent variable: ACTIVE.
*p<0.10, **p<0.05, ***p<0.01. Standard errors in parentheses.

Our estimates of the logit equation (Table 3) show that EXPERIENCE_MEAN is associated with project activity. As illustrated by Table 4a, experience is also positively correlated with the number of CVS commits, although it is not significant. Moreover, EXPERIENCE_MEAN does not enter significantly in the file release estimates (Table 5a).

Table 4a
Zero inflated negative binomial estimates.

	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7	Model 8
NO_SKILLS	0.385 (0.244)	0.226 (0.239)	0.139 (0.277)	0.137 (0.271)	0.282 (0.241)	0.216 (0.235)	0.326 (0.238)	0.118 (0.235)
NSUB_PROJECTS	0.095*** (0.021)	0.124*** (0.027)	0.096*** (0.021)	0.102*** (0.028)	0.092*** (0.021)	0.114*** (0.034)	0.083*** (0.021)	0.113*** (0.025)
EXPERIENCE	0.092 (0.074)	0.046 (0.072)	0.054 (0.075)	0.043 (0.073)	0.054 (0.074)	0.046 (0.071)	0.092 (0.074)	0.026 (0.072)
EXPERIENCE_CV1	2.413*** (0.483)							
EXPERIENCE_CV2	−1.128 (1.077)							
EXPERIENCE_CV		0.587 (0.368)						
NSUB_PROJ_X_EXPER_CV		−0.377*** (0.126)						
N_SKILLS_MEAN			−0.029** (0.014)	−0.017 (0.013)				
N_SKILLS_CV1			1.199*** (0.198)					
N_SKILLS_CV2			−0.499 (0.490)					
N_SKILLS_CV				0.356** (0.159)				
NSUB_PROJ_X_N_SKILLS_CV				−0.08 (0.055)				
DISTANCE					0.048*** (0.010)	0.009 (0.011)		
NSUB_PROJ_X_DISTANCE						−0.006 (0.004)		
OVERLAP							0.160*** (0.030)	−0.104** (0.043)
NSUB_PROJ_X_OVERLAP								−0.019** (0.008)
ENTRY_TIME		−0.011** (0.004)		−0.012*** (0.004)		−0.011** (0.005)		−0.012*** (0.004)
SIZE		0.083*** (0.009)		0.078*** (0.009)		0.084*** (0.009)		0.114*** (0.013)
LI_dummies	No	Yes	No	Yes	No	Yes	No	Yes
NL_dummies	No	Yes	No	Yes	No	Yes	No	Yes
PL_dummies	No	Yes	No	Yes	No	Yes	No	Yes
Constant	4.650*** (0.231)	4.657*** (0.251)	4.901*** (0.266)	4.782*** (0.283)	4.753*** (0.228)	4.661*** (0.248)	4.719*** (0.225)	4.701*** (0.249)

Dependent variable: CVS commits.
*p<0.10, **p<0.05, ***p<0.01. Standard errors in parentheses.

Table 4b
Zero inflated negative binomial: CVS commits — INFLATION MODEL: LOGIT.

	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7	Model 8
NO_SKILLS	−2.385*** (0.178)	−2.025*** (0.179)	−2.458*** (0.196)	−2.170*** (0.198)	−2.272*** (0.174)	−2.084*** (0.176)	−2.350*** (0.174)	−2.003*** (0.176)
NSUB_PROJECTS	−0.081*** (0.018)	−0.043* (0.024)	−0.076*** (0.018)	−0.044* (0.025)	−0.081*** (0.018)	−0.064** (0.027)	−0.057*** (0.018)	−0.048** (0.020)
EXPERIENCE	−0.861*** (0.056)	−0.745*** (0.056)	−0.862*** (0.056)	−0.757*** (0.056)	−0.812*** (0.056)	−0.747*** (0.056)	−0.855*** (0.056)	−0.739*** (0.056)
EXPERIENCE_CV1	−2.446*** (0.347)							
EXPERIENCE_CV2	2.168*** (0.759)							
EXPERIENCE_CV		−0.021 (0.290)						
NSUB_PROJ_X_EXPER_CV		0.098 (0.122)						
N_SKILLS_MEAN			−0.006 (0.010)	−0.01 (0.011)				
N_SKILLS_CV1			−1.184*** (0.153)					
N_SKILLS_CV2			0.732* (0.380)					
N_SKILLS_CV				−0.248** (0.126)				
NSUB_PROJ_X_N_SKILLS_CV				0.061 (0.058)				
DISTANCE					−0.051*** (0.007)	−0.017** (0.009)		
NSUB_PROJ_X_DISTANCE						0.006* (0.003)		
OVERLAP							−0.296*** (0.033)	0.025 (0.047)
NSUB_PROJ_X_OVERLAP								0.015* (0.008)
ENTRY_TIME		−0.013*** (0.004)		−0.013*** (0.004)		−0.014*** (0.004)		−0.013*** (0.004)
SIZE		−0.159*** (0.012)		−0.153*** (0.012)		−0.153*** (0.012)		−0.169*** (0.015)
LI_dummies	No	Yes	No	Yes	No	Yes	No	Yes
NL_dummies	No	Yes	No	Yes	No	Yes	No	Yes
PL_dummies	No	Yes	No	Yes	No	Yes	No	Yes
Constant	3.546*** (0.169)	3.988*** (0.202)	3.625*** (0.187)	4.122*** (0.219)	3.431*** (0.164)	4.043*** (0.200)	3.505*** (0.165)	4.001*** (0.199)
/lnalpha	1.281*** (0.053)	1.122*** (0.050)	1.262*** (0.052)	1.119*** (0.050)	1.287*** (0.054)	1.123*** (0.050)	1.275*** (0.053)	1.114*** (0.050)
Alpha	3.599 (0.191)	3.072 (0.153)	3.532 (0.185)	3.062 (0.152)	3.623 (0.194)	3.074 (0.153)	(3.578) (0.190)	3.046 (0.151)
Vuong test (zinb vs. nb)	4.12***	6.73***	4.12***	6.79***	4.00***	6.84***	4.26***	6.81***
Observations	9076	9076	9076	9076	9076	9076	9076	9076
Nonzero Obs.	2154	2154	2154	2154	2154	2154	2154	2154
Log Likelihood	−17,785.31	−17,408.32	−17,766.83	−17,406.61	−17,788.83	−17,408.71	−17,744.17	−17,400.57
Prob>chi^2	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001

*$p<0.10$, **$p<0.05$, ***$p<0.01$. Standard errors in parentheses.

As Table 3 shows, the coefficient of variation of the level of experience (EXPERIENCE_CV) is negative and non-significant (Model 2), while the coefficient of variation of the number of project members' skills (NSKILLS_CV) is positive and significant (Model 4), suggesting that project activity is positively associated with diversity in the profile of skills of project members.

Table 5a
Zero inflated negative binomial: estimates.

	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7	Model 8
NO_SKILLS	−0.394 (0.298)	−0.448 (0.295)	−0.512 (0.314)	−0.611* (0.318)	−0.529* (0.283)	−0.506* (0.29)	−0.502* (0.284)	−0.620** (0.293)
NSUB_PROJECTS	0.044** (0.022)	0.049* (0.028)	0.040* (0.022)	0.054* (0.030)	0.040* (0.022)	0.060* (0.033)	0.035 (0.021)	0.007 (0.024)
EXPERIENCE	−0.081 (0.089)	−0.078 (0.090)	−0.114 (0.086)	−0.08 (0.091)	−0.127 (0.087)	−0.096 (0.090)	−0.078 (0.088)	−0.108 (0.091)
EXPERIENCE_CV1	1.748*** (0.625)							
EXPERIENCE_CV2	−0.202 (1.479)							
EXPERIENCE_CV		0.970** (0.422)						
NSUB_PROJ_X_EXPER_CV		−0.263 (0.166)						
N_SKILLS_MEAN			−0.006 (0.016)	−0.023 (0.017)				
N_SKILLS_CV1			0.825*** (0.250)					
N_SKILLS_CV2			−0.435 (0.616)					
N_SKILLS_CV				0.361* (0.185)				
NSUB_PROJ_X_N_SKILLS_CV				−0.125** (0.063)				
DISTANCE					0.034*** (0.012)	0.021* (0.013)		
NSUB_PROJ_X_DISTANCE						−0.008* (0.004)		
OVERLAP							0.054* (0.028)	−0.023 (0.046)
NSUB_PROJ_X_OVERLAP								0.004 (0.008)
ENTRY_TIME		0.006 (0.005)		0.005 (0.005)		0.006 (0.005)		0.005 (0.005)
SIZE		0.015* (0.008)		0.015* (0.008)		0.015* (0.008)		0.022* (0.013)
LI_dummies	No	Yes	No	Yes	No	Yes	No	Yes
NL_dummies	No	Yes	No	Yes	No	Yes	No	Yes
PL_dummies	No	Yes	No	Yes	No	Yes	No	Yes
Constant	−0.051 (0.353)	0.006 (0.331)	0.171 (0.351)	0.169 (0.351)	0.167 (0.325)	0.062 (0.327)	0.255 (0.305)	0.179 (0.328)

Dependent variable: File releases.
*$p<0.1$, **$p<0.05$, ***$p<0.01$. Standard errors in parentheses.

Table 5b
Zero inflated negative binomial: file releases — INFLATION MODEL: LOGIT.

	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7	Model 8
NO_SKILLS	−4.065*** (0.669)	−2.509*** (0.376)	−3.821*** (0.592)	−2.844*** (0.409)	−3.737*** (0.574)	−2.649*** (0.372)	−3.324*** (0.453)	−2.626*** (0.385)
NSUB_PROJECTS	−0.119*** (0.043)	−0.102** (0.043)	−0.104*** (0.038)	−0.075 (0.047)	−0.109*** (0.039)	−0.081 (0.054)	−0.076** (0.035)	−0.103** (0.042)
EXPERIENCE	−1.491*** (0.265)	−0.861*** (0.126)	−1.340*** (0.216)	−0.879*** (0.127)	−1.318*** (0.220)	−0.888*** (0.127)	−1.141*** (0.161)	−0.884*** (0.130)
EXPERIENCE_CV1	0.022 (1.002)							
EXPERIENCE_CV2	0.919 (1.727)							
EXPERIENCE_CV		0.908* (0.512)						
NSUB_PROJ_X_EXPER_CV		0.265 (0.273)						
N_SKILLS_MEAN			−0.006 (0.024)	−0.032 (0.021)				
N_SKILLS_CV1			−0.569 (0.353)					
N_SKILLS_CV2			0.752 (0.787)					
N_SKILLS_CV				0.064 (0.228)				
NSUB_PROJ_X_N_SKILLS_CV				0.013 (0.111)				
DISTANCE					−0.025 (0.019)	0.004 (0.017)		
NSUB_PROJ_X_DISTANCE						0.002 (0.008)		
OVERLAP							−0.279*** (0.072)	0.172* (0.099)
NSUB_PROJ_X_OVERLAP								0.015 (0.018)
ENTRY_TIME		−0.034*** (0.007)		−0.034*** (0.007)		−0.034*** (0.007)		−0.034*** (0.007)
SIZE		−0.210*** (0.029)		−0.200*** (0.029)		−0.199*** (0.030)		−0.240*** (0.036)
LI_dummies	No	Yes	No	Yes	No	Yes	No	Yes
NL_dummies	No	Yes	No	Yes	No	Yes	No	Yes
PL_dummies	No	Yes	No	Yes	No	Yes	No	Yes
Constant	4.107*** (0.485)	4.491*** (0.401)	4.055*** (0.450)	4.804*** (0.436)	3.933*** (0.425)	4.610*** (0.402)	3.708*** (0.365)	4.676*** (0.408)
/lnalpha	1.587*** (0.179)	1.012*** (0.128)	1.451*** (0.183)	1.024*** (0.134)	1.479*** (0.178)	1.018*** (0.133)	1.326*** (0.162)	1.047*** (0.131)
Alpha	4.89 (0.877)	2.751 (0.353)	4.266 (0.781)	2.785 (0.372)	4.388 (0.783)	2.768 (0.369)	3.764 (0.610)	2.849 (0.372)
Vuong test (zinb vs. nb)	5.31***	8.46***	5.28***	8.44***	5.21***	8.33***	5.33***	8.31***
Observations	9076	9076	9076	9076	9076	9076	9076	9076
Nonzero Obs.	1313	1313	1313	1313	1313	1313	1313	1313
Log Likelihood	−6126.861	−5946.054	−6119.268	−5947.533	−6123.902	−5949.668	−6114.715	−5949.485
Prob>chi^2	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001

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

The piecewise linear transformation (spline) of EXPERIENCE_CV (1 and 2) and NSKILLS_CV (1 and 2) in the logit estimation (Models 1 and 3) show that, beyond a threshold level of diversity, the benefits of skill diversity tend to decrease.

The equations relative to CVS commits confirm the non-linear effect, although the results are less significant. While the significance and sign of the coefficients for NSKILLS_CV1 and EXPERIENCE_CV1 still persist, the coefficients for NSKILLS_CV2 and EXPERIENCE_CV2 remain negative but are not significant.

We also find that the coefficient of DISTANCE is positive and significant in the logit estimation of ACTIVE. This variable also enters significantly in the new file releases equation (Tables 5a and 5b), whereas its coefficient is not significantly different from zero in the CVS commits equation (Tables 4a and 4b).

Contrary to our expectation, the association between OVERLAP and our measures of performance is ambiguous. The coefficient on OVERLAP is always positive and significant in the estimations of ACTIVE, CVS and file releases (Model 7 without controls and interaction terms) but its sign becomes negative and significant when controls and interaction terms are introduced. These results apparently contradict the hypothesis that some degree of skill overlap is important for the communication among project members to take place (Lazear, 1999). We believe that one reason why this is not the case is that the majority of our projects are composed of individuals with a common technical background (e.g. programming and information systems administration). Moreover, the typical OSS project members have the same values and a common vision of software development. They are part of the same virtual community of practice and as such they share the same language. Moreover, the small size of the average OSS project reduces the need for skill overlap. For these reasons it is likely that high levels of overlap do not yield any productivity gain and could even limit the division of labor among project members.

NSUB_PROJECTS, our proxy for modularity, is positively correlated with project activity, as shown in Table 3. Modularity is also positively associated with CVS commits (Tables 4a and 4b). This result is robust to alternative specifications of the model and suggests that the division of labor in different modules is typical of active and more performing projects. The coefficient on NSUB_PROJECTS is positive also in our estimations of file releases, although it is not always significant (Tables 5a and 5b).

The interaction between various measures of skill diversity and the number of subprojects is negatively associated with survival. It is important to note that the association appears to be strong also in the CVS estimations. As Tables 4a and 4b show, the coefficient on the interaction between the number of subprojects and EXPERIENCE_CV (Model 2) is negative and significant. Similarly, the interactions of NSUB_PROJECTS with DISTANCE and OVERLAP are negative (Models 6 and 8), showing that high levels of skill diversification and overlapping skills coexist with high levels of modularity mostly in low-performance projects.

Estimate results for new file releases are similar to those for CVS commits, and also in this case the coefficients on interaction terms are not always significant (see Models 4 and 6 in Tables 5a and 5b). Apparently, this contrasts with 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, signaled by a large number of presumably independent subprojects, benefit from limited skill differences among participants?

To answer this question we should remember that there are differences among projects in terms of product complexity (i.e. the strength of interdependencies among modules), which are likely to be reflected into the division of labor among team members.

High levels of complexity reduce the possibility of dividing the work into separate tasks (subprojects) and lead to a more centralized division of labor that resembles the typical workshop-like production system. In this type of system there are significant information flows among activities and contributors, which increases the signaling effect of high-skill participants and the learning benefits of low-skill participants. The productivity of this system then depends on imitation and knowledge transfer among participants endowed with different skills.

By contrast, projects focusing on less complex product architectures are more likely to experience a large division of problem-solving labor among subprojects, i.e. a high level of modularity and finer problem decomposition.[16] In this organizational setting, corresponding to a "factory-like" production approach, problem solving is more decentralized and knowledge is highly partitioned. These organizations rely on less intensive flows of information among activities and individuals. Each contributor specializes in specific modules or tasks and this reduces the need for multi-skilled participants. The number of different skills in projects with a high level of modularity then is likely to be limited. [17]

As the number of subprojects may be affected by the size of the project, we control for project size in our regressions and the coefficients of subprojects and the interaction terms remain significant.

Overall, these findings suggest that the interaction between skill heterogeneity and the division of labor in open source projects yields quite ambiguous effects on performance. These results call for a finer-grained, qualitative analysis of the mechanisms underlying the development process and relationships among participants.

Other coefficient estimates are worth discussing briefly. As expected, the SIZE of the project is positively related to both survival and performance. Interestingly, late entrants are more likely to be active (see the coefficient of ENTRY_TIME in Table 3). This is consistent with the fact that several projects enter the OSS community, produce some activity in the early stages of their life and become unproductive afterward. These projects remain in the SF.net dataset because the cost of staying is virtually zero. However, the negative coefficient on ENTRY_TIME in Table 4a shows that, among active projects, older projects perform better in terms of number of CVS commits produced. This finding is coherent with an environment in which projects enter with some initial level of activity, but only a subset of these projects consolidates over time and becomes more productive. These results are confirmed in all model specifications. However, ENTRY_TIME is not important for new file releases (Tables 5a and 5b).

Finally, we find that projects adopting a GPL model are more likely, *ceteris paribus*, to be inactive 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 exceptions being the public domain license (LI_PUBDOM) and the artistic license (LI_ARTISTIC). These results are only partially confirmed in our negative binomial estimates where only LGPL and MIT remain significant. The coefficients on other license schemes remain positive but not significant at the conventional levels.

This amounts to saying that adopting a GPL model (the baseline dummy in our regressions) is negatively associated with 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 more likely to be associated with less output per contributor than less restrictive licenses such as BSD. In line with what suggested by Lerner and Tirole (2005) and Fershtman and Gandal (2004), our finding may be due to the fact that projects with less restrictive licenses have greater commercial potential and are more likely to be participated by profit-oriented actors which may foster project performance.

5.3. Robustness checks and endogeneity

The differences in the estimation of file releases and CVS commits discussed before may reflect the fact that file releases are much less frequent events than CVS commits. To see how these differences affect our results we estimated the file release equations with file releases counted over a longer time window (i.e. 6 months and 1 year). The results are by and large similar to those reported in Tables 5a and 5b, especially when file releases are counted over a one-year time window (even if convergence is not always achieved). Moreover, estimation results for CVS commits are confirmed in particular when a six-months' time window is adopted.[18]

As discussed before, our results may be biased by self-selection problems. To assess the importance of endogeneity in our data, we have analyzed the effect of observable project characteristics on skill indicators and modularity. To this end, as explained before, we run separate estimation of different measures of project skills (skill level and skill diversity) and modularity against project size, time of registration, programming language, spoken language and license type. Size and registration time positively affect both the project's skill composition and modularity.[19]

These results suggest that size (a proxy for project complexity) drives the process of skill diversification, probably because large, more challenging projects attract new contributors endowed with different skill sets. Size also drives modularity, suggesting that larger projects need a more efficient division of labor as compared with smaller ones. Moreover, the time of project entry into SF.net is strongly associated with both skill and modularity, while the age of a project does not seem to influence the overlap between project members' skills. This finding suggests that, compared with later entrants, early entrants attract more highly skilled and heterogeneous individuals. Moreover, they rely on modularity to a greater extent than later entrants. This may be due to the fact that early projects have had more time to experiment and learn how to take advantage of modularity.

These results clearly show that there is a self-selection problem in our performance estimations. However, as we already observed, size, registration time and other variables have also a direct effect on project performance. Therefore they cannot be used as instruments for skill and modularity and, unfortunately, the dataset does not offer any alternative instruments.

Although our findings cannot be interpreted as a proper test for causality between skill diversity, modularity and project performance, we believe that our analysis offers interesting insights for the organization of innovation in a representative sample of OSS projects. First, even if both modularity and skill diversity are important for innovation in this context, their combination is not obvious at the project level. As mentioned before, highly productive projects can be found among projects with high levels of modularity and low levels of

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

[17] In unreported regressions we entered a new variable obtained by interacting *N_SKILLS_MEAN* and *NSUB_PROJECTS*. While the interaction terms reported in the paper are based on measures of skill diversity across project members, *N_SKILLS_MEAN* measures the importance of multi-skilled individuals in the project. This new interaction term enters negatively and significantly in the file releases equation, whereas it is not significant in the active and CVS commits equations. This evidence is consistent with the idea that multi-skilled, versatile individuals are important for the productivity of complex, low-modularity systems.

[18] These estimation results are available upon request from the authors.

[19] These regressions are not reported for reasons of space, but the results are available from the authors.

skill heterogeneity as well as among those with high levels of skill heterogeneity and low levels of modularity. Second, size (a proxy for project complexity) and project's age (a proxy for experience) drive the use of more heterogeneous skills and a more modular organization. Even if we cannot use size and experience as instruments for skills and modularity, these variables add another important dimension to our picture: projects that are more modular and endowed with a more heterogeneous skill set are larger and older than the average OSS project.

6. Discussion and conclusions

Our analysis provides novel empirical evidence on an important example of collective inventions. The analysis draws on 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 association between skill heterogeneity and performance. Second, the economics of modern production has explored the importance of modular design and production for performance and innovation. Modularity is a key technological characteristic of OSS development. Our analysis has focused on a measure of organizational modularity, which is the number of distinct subprojects.

Our empirical results provide support to the hypothesis that skill diversity and modularity are positively associated with project performance. To this end, we have identified two different measures of project activity. First, as a proxy for project activity we assumed that at least one of the following events must have occurred in the last 3 months of the dataset: a fixed bug, a fixed patch, a new feature request fulfilled, a new product release or a new CVS commit.

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

Our analysis shows that the level of members' skills (average experience in all skills) is positively associated with project survival. All measures of skill diversity adopted in our estimations are positively and significantly associated with project activity. Although some results are not robust to controls, our different measures of skill diversity overall remain significant when controls are included in the equation. This suggests that OSS projects benefit from an internal division of labor between “specialists” and “generalists” and between ‘high-skilled’ and ‘low-skilled’ members. The relationship between diversity and performance is however non-linear, confirming the hypothesis that beyond a threshold very high levels of diversity produce negative effects on survival and performance. By the same token, we find that high levels of overlapping skills in this setting are negatively associated with project performance. This can be explained by the fact that, unlike other settings such as multi-cultural and transnational organizations, OSS projects on average are made of members who share a similar language and vision of the innovation process; thus, high levels of skill overlap are not required to facilitate communication. On the contrary, our results suggest that significant skill overlap reduces the opportunities for division of labor and productivity at the project level.

Our analysis also offers a strong support of the hypothesis that modularity is an important characteristic of highly productive projects. 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 and the complexity of the product, i.e. the strength of interdependences among its modules or components. Our estimates of modularity are robust to several controls.

Finally, we find that the interaction between skill diversity and modularity at the project level is negatively associated with project performance. The substitutability between skill diversity and modularity suggests that there are two organizational approaches to software development in the OSS environment. The first approach recalls a workshop-like organization that relies on members with different skill profiles, which does not use much modularity in its developing activities. 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. Both types of projects have above-average productivity.

An important conclusion of our analysis is that there exists a variety of efficient forms of organization of OSS each combining two key dimensions of modern design and manufacturing – skills and modularity – in different ways. We also believe that this evidence raises new issues for further theoretical and empirical work on team production and human capital.

As previously mentioned, the dataset does not provide satisfactory instrumental variables and the limitations of the data do not allow the use of other techniques to deal with unobservable fixed effects. This makes it difficult to effectively deal with the endogeneity of skills and modularity. However, as we have already noted, the results remain appealing even if we cannot conduct a proper test for the causality relationships among the relevant variables.

Acknowledgements

We thank Gaia Rocchetti for her valuable 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 Fontes, Joachim Henkel and Gregorio Robles for useful comments to earlier drafts of the paper. We also thank the participants of workshops and conferences in Toulouse, Barcelona, Munich, Milan and Padua for comments and suggestions. We acknowledge the financial support of the project “Economic Change: the micro foundations of institutional and organisational change: EconChange,” EC, DGXII, FP V and the international mobility program of Fondazione IRI.

Appendix A

Table A.1
The architecture of the Joose project.

Project_id	*Subproject_name*	*Description*
65952	Symbolic executors	
65952	Website	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

References

Allen, R.C., 1983. Collective invention. Journal of Economic Behavior and Organization 4, 1–24.

Arora, A., Gambardella, A., 1994. The changing technology of technological change: general and abstract knowledge and the division of innovative labour'. Research Policy 23 (5), 523–532.

Baldwin, C.Y., Clark, K.B., 2006. The architecture of participation: does code architecture mitigate free riding in the open source development model? Management Science 52 (7), 1116–1127.

Baldwin, C.Y., Clark, K.B., 1997. Managing in an age of modularity. Harvard Business Review 75, 84–93 September–October.

Bantel, K.A., Jackson, S.E., 1989. Top management and innovations in banking: does the composition of the top team make a difference? Strategic Management Journal 10, 107–124 Summer Special Issue.
Blundell, R., Griffith, R., Van Reenan, J., 1995. Dynamic count data models of technological innovation. Economic Journal 105 (429), 333–344.
Bresnahan, T.F., Brynjolfsson, E., Hitt, L.M., 2002. Information technology, workplace organization and the demand for skilled labor: firm-level evidence. Quarterly Journal of Economics 117 (1), 339–376.
Brusoni, S., Prencipe, A., 2001. Unpacking the black box of modularity: technologies, products and organizations. Industrial and Corporate Change 10, 179–205.
Caroli, E., Van Reenen, J., 2001. Skill-biased organizational change? Evidence from a panel of British and French establishments. Quarterly Journal of Economics 116, 1449–1492.
Comino, S., Manenti, F.M., Parisi, M.L., 2007. From planning to mature: on the success of open source projects. Research Policy 36 (10), 1575–1586.
Crowston, K., Howison, J., Annabi, H., 2006. Information systems success in free and open source software development: theory and measures. Software Process: improvement and practice 11 (Special Issue on Free/Open Source Software Processes), pp. 123–148.
Dalle, J.M., David, P.A., 2005. The allocation of software development resources in open source production mode. In: Fitzgerald, B., Hissam, S., Lakhani, K.R., Feller, J (Eds.), Perspectives on Open Source and Free Software. MIT Press, Cambridge, MA, pp. 297–328.
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.
Faraj, S., Sproull, L., 2000. Coordinating expertise in software development teams. Management Science 46 (12), 1554–1568.
Fershtman, C., Gandal, N., 2004. The determinants of output per contributor in open source projects: an empirical examination. CEPR discussion paper, No. 4329, London.
Galunic, D.C., Riordan, S., 1998. Resource recombinations in the firm: knowledge structures and the potential for Schumpeterian innovation. Strategic Management Journal 19 (12), 1193–1201.
Hall, B.H., Ziedonis, R.H., 2001. The patent paradox revisited: an empirical study of patenting in the US semiconductor industry 1979–1995. RAND Journal of Economics 32 (1), 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 1 (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, 1753–1769.
Harrison, S., Ross, S.J., Lawson, J.H., 1992. Beta diversity on geographic gradients in Britain. Journal of Animal Ecology 61 (1), 151–158.
Hertel, G., Niedner, S., Herrmann, S., 2003. Motivation of software developers in open source projects: an Internet-based survey of contributors to the Linux kernel. Research Policy 32 (7), 1159–1177.
Johnson, J.P., 2002. Open source software: private provision of a public good. Journal of Economics and Management Strategy 11 (4), 637–662.
Kiczales, G., 1996. Beyond the black box: open implementation. IEEE Software 13 (1), 8–11.
Krishnamurthy, S., 2002. Cave or community? An empirical examination of 100 mature open source projects. First Monday 7 (6) http://outreach.lib.uic.edu/www/issues/issue7_6/krishnamurthy/.
Kuan, J., 2001. Open source software as consumer integration into production. Mimeo, Haas School of Business, University of California, Berkeley, CA.
Lakhani, K.R., Wolf, R.G., 2005. Why hackers do what they do: understanding motivation and effort in free/open source software projects. In: Fitzgerald, B., Hissam, S., Lakhani, K.R., Feller, J. (Eds.), Perspectives on Free and Open Source Software. MIT Press, Cambridge, MA, pp. 3–21.
Lakhani, K.R., von Hippel, E., 2003. How open source software works: "free" developer-to-developer assistance. Research Policy 32, 923–943.
Langlois, R.N., 2002. Modularity in technology and organization. Journal of Economic Behavior and Organization 49, 19–37.
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, Copenhagen.
Lazear, E.P., 1999. Globalization and the market for team-mates. Economic Journal 109 (454), 15–40.
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. Journal of Industrial Economics 50 (2), 197–234.
Lerner, J., Tirole, J., 2005. The scope of open source licensing. Journal of Law, Economics, and Organization 21 (1), 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. 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. Idea Group Publishing, Hershey PA, pp. 203–220.
Marengo, L., Dosi, G., 2005. Division of labor, organizational coordination and market mechanisms in collective problem-solving. Journal of Economic Behavior and Organization 58 (2), 303–326.
Milgrom, P., Roberts, J., 1990. The economics of modern manufacturing. American Economic Review 80 (3), 511–528.
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., Fielding, R.T., Herbsleb, J., 2000. A case study of open source software development: the Apache server. Proceedings of the Twenty-Second International Conference on Software Engineering, pp. 263–272.
Mohen, P., Roller, L.H., 2005. Complementarity in innovation policy. European Economic Review 49 (5), 1431–1450.
Parnas, D.L., 1972. On the criteria to be used in decomposing systems into modules. Communications of the ACM 15 (12), 1053–1058.
Pil, F.K., Cohen, S.K., 2006. Modularity: implications for imitation, innovation, and sustained advantage. Academy of Management Review 31 (4), 995–1011.
Raymond, E.S., 1999. Linux and open-source success. IEEE Software 16 (1), 85–89.
Robles, G., Gonzalez-Barahona, J.M., Merelo, J.J., 2006. Beyond source code: the importance of other artifacts in software development (a case study). Journal of Systems and Software 79, 1233–1248.
Steinmueller, W.E., 1996. The U.S. software industry: an analysis and interpretative history. In: Mowery, D. (Ed.), The International Computer Software Industry. A Comparative Study of Industry Evolution and Structure. Oxford University Press, New York, 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., 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.
Whittaker, R.H., 1960. Vegetation of the Siskiyou Mountains, Oregon and California. Ecological Monographs 30 (4), 279–338.