

Skills and Openness of OSS Projects: implications for performance

Paola Giuri*, Matteo Ploner*, Francesco Rullani*, Salvatore Torrisi*^o

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

^o *Corresponding author*: Department of Management
University of Bologna, Piazza Scaravilli, 1 40126 Bologna
Tel. ++39-051-2098064 (segr. ++39-051-2098051)
Fax: ++39-051-2098074, e-mail: torrisi@unibo.it.

DRAFT: January 2005

Abstract

This paper is about open source software projects' activity and the characteristics of different categories of contributors. Our empirical analysis draws on a large sample of OSS projects registered at the Sourceforge website. For each project we have information about individual contributors such as skills, roles, and tasks assigned. Key variables at the project level are the number of project members or internal contributors (i.e., people who have subscribed to the project), the number of external contributors (project openness), the overall skill combination of contributors, the number of different intended audiences (e.g., developers and end users), and various measures of activity (e.g., number of file releases, bugs and patches closed over time).

We conducted a multinomial logit analysis to see whether skills' level, experience and variety of project members predict their role played in the project (e.g., developer or project manager). We have also carried out an econometric analysis to estimate the contribution of skills and openness to projects' survival and activity.

Keywords: software, technological innovation, human capital

JEL classification: L86 O32, J24

Acknowledgements: we thank Gaia Rocchetti for her precious collaboration in the early stages of our research project and the administrators of SourceForge.net for providing us with their data on open source software projects and individual contributors. We also acknowledge the financial support of the project "Economic Change: the microfoundations of institutional and organisational change (EconChange)", CE, DGXII, FP V.

1. Introduction

This paper is about skills and roles played by members of open source software (OSS) projects. We also investigate the importance of internal contributors (members) and external contributors for the performance of the projects (survival and level of activity).

Most theoretical works on OSS try to understand the motivations for disclosure of the source code, the social norms and the patterns of collaboration among distributed developers, and the implications for dynamic efficiency and social welfare (e.g., Lerner and Tirole 2002b; von Hippel 2001; Dalle and Jullien 2001; Dalle and David 2003).

Several empirical technical papers ask how OSS performs compared with proprietary software in terms of quality, time of response to problems encountered by users, stability or security (e.g., Wheeler 2002).

Most of the empirical analysis focuses on one or few OSS projects with the aim of testing hypotheses drawn from the software engineering, economics or management literature.

The contribution of this paper to the economics and management literature is twofold. First, we provide an empirical investigation of individuals and project characteristics based on a large sample of OSS projects. To our knowledge, this is the first attempt at providing a wide empirical overview of some fundamental characteristics concerning individual contributors and projects like skills, roles, project activity by internal members and outside contributors. Second, unlike many earlier works this paper does not try to compare OSS software with proprietary software. Instead, we compare different OSS projects to understand: a) skills, experience, and roles of contributors; b) the distribution of projects by size and contributors characteristics (skills and roles); c) the determinants of survival and activity (bugs fixed, patches etc.) of the projects.

2. Theoretical background

An increasing number of papers in the economics and management literature have analyzed the fundamental characteristics of the OSS organization and the motivations of participants (see, for

instance, von Hippel 2001; Lakhani and von Hippel 2000; Lerner and Tirole 2002b).

A vast body of works have been produced by lead software developers or scholars who are strong supporters of OSS (See Di Bona et al. 1999, for a collection of contributions). This stream of the literature tries to demonstrate the superior performance of OSS vs. proprietary software. The most popular example of these studies is the “The Cathedral and the Bazaar” by Eric Raymond. He draws on the case of Linux to show the strengths of the OSS development paradigm (the “Bazaar”) compared to the traditional software engineering approach (the “Cathedral”). A key characteristic of the Linux development process - as described by Raymond - is the role played by various contributors (developers and users). The Linux community members (both users and developers) can download the source code, make modifications (further development of lines of code, debugging, etc.) and post it to the responsible of the project (i.e., Linus Torvalds). The project leader, along with core developers, examines the proposed modifications and releases a new version of the program to the community. The product evolves rapidly thanks to the continuous feedbacks and improvements from different users and developers.

The active participation of users is not a novelty if compared to the Unix community’s tradition for which the free exchange of code was a diffused practice. However, as argued by Raymond, the distinctive feature of the Linux process is that new releases including modifications, fixed bugs and improvements are distributed quite rapidly, and it is often the case that new releases still contain bugs (“release early, release often”). This practice, which provides strong incentives to the community members, is common to other OSS projects. Raymond compares the development process of Linux (and OSS in general) to the traditional software engineering process, characterised by a rigid hierarchical organisation in which each group of tasks is associated to a team of developers.

He argues that the OSS development process is more efficient than the traditional one, and yields a higher level of product quality. He also argues that open source developers are self-

selected, so that only the most talented people participate in the process. Furthermore, according to Raymond, OSS developers are free to express their creativity and are highly motivated by fun, while developers working in traditional groups perform their task mainly because of economic incentives. Finally, the intense activity of debugging makes software more reliable and its pace of evolution more rapid (the essence of what Raymond calls the ‘Linux’s law’ is that “given enough eyeballs, all bugs are shallow”, Raymond 1999). A critical assessment of the comparative performance of OSS as discussed by Raymond and other OSS advocates is beyond the scope of this work. However, it is worth to recall that some of these scholars have recognized that OSS has important weaknesses. For example, the lack of an explicit design can imply that the project’s quality is limited and constrain the long term performance of open source products (see Giuri et al. 2002, for a survey).

In the following part of this section we focus on some issues about OSS that are relevant to the purposes of our empirical analysis.

The distribution of developers and skills across projects. As mentioned before, OSS is viewed by many observers as a radically new development model (the Bazaar), based on a large community that share values (generalized reciprocity and meritocracy), conduct rules and institutions (such as priority, peer review principles, and the refuse of formal development methodologies and management systems). These social norms and institutions conform to the concepts of self-organization and gift economy rather than the principles of engineering and market economy (e.g., Raymond 1999 and 2001; Di Bona et al. 1999).

As a matter of fact, an increasingly large number of programmers offer their voluntary contribution to several OSS projects. However, several studies show that a large number of projects are participated by a small number of active, highly committed programmers who contribute to only one or few projects. With few exceptions, the literature focuses on specific OSS projects, especially large ones. For instance, Koch and Schneider (2002) have analyzed the CVS (concurrent versions system) of the GNOME project (an OSS project dedicated to a

desktop environment for users and an application framework for developers) and found that only a small number of programmers work together on the same file. The number of co-developers increases with the size of the file and more active programmers work more for large files compared with less active programmers. Similarly, Mockus et al. (2000) found that in the case of the Apache server project the top 15 developers contributed more than 83% of changes to Apache source code during the period 1995-1999¹. Moreover, only 25 developers submitted changes on a regular basis. Mockus et al. (2000) also show that a wider community of contributors participated in bug reporting and repairing.

In the same research line, Ghosh and David (2003) have analyzed three Linux kernel versions and found that that a large proportion of modules have been developed by less than five developers while a significant number of modules have only one developer. They also found that over 70% of authors contribute to only one or two projects or packages (large, self-contained, integrated modules). However, they also found that the vast majority of developers have contributed to modules with a large number of co-developers and virtually all developers have at least a co-developer. As Gosh and David argue, the fact that most developers contribute to only one project suggests that developers tend to join large projects, with large numbers of developers. Reputation effects or social group agglomeration processes may explain this pattern of collaboration among developers. Even if they do not mention possible scale effects, one might also ask whether productivity increases with the number of participants.

It is unclear whether these results are specific to large, successful projects. What do we know about the majority of smaller OSS projects? Krishnamurthy (2002) has analyzed the top 100 mature projects in Sourceforge and found that the median number of co-developers was only 4. Moreover, the vast majority of these projects have generated a small number of messages in their

¹ Modification requests are changes submitted to the Apache CVS archive. This archive makes it possible to identify the developers who submitted the change (or patch), the files involved, the number of lines of code added or deleted, and a description of the change (Mockus et al. 2000: 3). Changes associated to problems (bugs) can be distinguished from changes concerning new features added to the code. Modification requests and problem reports are measures of development of new functionality and maintenance of existing features.

forums and about one third of them had no messages at all. According to Krishnamurthy these data suggest that the development stage of OSS products does not conform to the Bazaar model. Instead, according to Krishnamurthy, these projects are in line with the 'cave' (lone producers) model. If we consider that less than 2% of all Sourceforge projects reached the maturity stage in 2004, 18 % reached the production/stable stage and a large share (22 %) were still in their planning stage, the data collected by Krishnamurthy suggest that the majority of OSS are small and generate only limited informal exchange among users.

The evidence discussed before suggests that a strong selection of projects takes place in the OSS community, with many small projects remaining isolated from the rest of the community and never reaching the maturity stage. In a world characterized by network externalities and strong social ties among participants this strong selection is not surprising. A typical outcome of selection under these conditions is that few successful projects attract the majority of good developers and discussion while the majority of projects below a given threshold are marginalized from the main information and knowledge exchange network. For instance, Linux has become the indisputable leader among the OSS operating systems like the BSD Unix. It has then attracted development resources and users away from alternative systems. This 'winner-takes-all' outcome indicates the strength of network externalities. Another factor that drives the selection mechanism is represented by the signaling incentives of programmers. Similar to the academic world, where popular research agenda attract many researchers and provides a large audience, good programmers prefer large, successful projects because these offer a larger audience and higher reputation opportunities (Lerner and Tirole 2002a, Dalle and David 2003).

The division of labor at the project level. The romantic view of OSS as a large community of skilled users freely contributing to a process of collective invention appears to be somewhat in contrast with the evidence of a quite well defined division of labor between groups of 'core developers' who control the evolution of the base code and a wider 'periphery' of contributors who provide feedbacks that are critical for product quality improvement. For instance, in the

case of the Apache HTTP Server Project, the ‘core’ corresponds approximately to the Apache Group, the organization responsible for the management of the Project. The maintainers of this project (about 15 people) are primarily devoted to developing or reviewing new functionalities to the base code and, to a lesser extent, to fixing defects. The division of labor among core developers is blurred. All of them contribute code to various modules. Moreover, the role of release manager (a quite time-consuming and critical task) is rotated among them (Mockus et al. 2000). The periphery of less active contributors is made of non core developers (about 250 people during the time window of Mockus et al.’s analysis) which, relative to core developers, is more active in bugs or problem-related changes (patches). The most external part of the Apache server’s periphery is made of a wider community of over 3,000 users who only report bugs.

This apparently complex organization relies on two important infrastructures: modular design and the use of Internet. Internet (email, newsgroups, forums etc.) reduces transaction and communication costs among developers and therefore provides a fundamental infrastructure for distributed development across space and over time.

Product modularity reduces the systemic interdependencies between different files of the same product and therefore allows a higher level of task partitioning and a lower level of explicit coordination and interaction among programmers. In large projects like Linux kernel or Apache server, a significant level of modularity is achieved thanks to a clear division of labor between the core product architecture and more ‘external’ features that are ‘located in modules that can be selectively compiled and configured’ (Mockus et al. 2000: 4). In the case of Apache, new developers try to avoid duplication of efforts by focusing on task that the ‘code owner’ is not working at (Mockus et al. *ibid.*). Even in these large projects, however, it is difficult to proceed with a sharp division of tasks among teams or individual developers because participants are volunteers distributed across space and over time. Developers then contribute their work to a project or a specific task according to their preferences and time (generally they have a job and contribute to OSS in their free time) (Elliot and Scacchi 2003). Moreover, the evolution of

specific tasks/modules cannot be easily predicted since new features and problems are discovered during the process (Kaisla 2001). According to some scholars, then, there is no formal organization in OSS projects and participants self-organize without the use of timelines or roadmaps. Personal motivation, shared beliefs and values tie these virtual organization together (Elliot and Scacchi 2003).

As Lerner and Tirole (2002a) pointed out, the leader has to carry out some critical tasks : a) to provide a 'vision' that is provided through a critical mass of code that demonstrates her expertise and credibility; b) to attract new programmers by posing challenging issues and, at the same time, leaving to potential contributors significant opportunities for future improvements to the initial code; c) to ensure an efficient division of the project into modules and to allow contributors to perform their tasks independently from the rest of the contributors; d) to avoid that conflicting views and approaches among participants lead to dropouts and forking (p. 21).

Leadership can rely on important technologies like the Internet and concurrent versions systems (CVS), which represents an important instrument for the coordination of different contributors. But cultural norms and organizational technology are also critical to keep the level of duplication and conflict under control, therefore supporting the task of leaders or project maintainers.

Organization theory suggests a rich menu of organizational structures that fit different levels of complexity in the division of tasks (Minzberg, 1979). As mentioned before, modular design and standard interfaces represent an important organizational technology adopted in the OSS environment. Formal organization of authority is also adopted in the largest projects like Linux, where Linus Torvald has the last word for any change to the source code that is officially released. By the same token, the Apache Group relies on a formal voting system for approval of changes to the source code.

Even if success and large size put a strong pressure on project managers and call for formal organization, according to some observers, even large projects still rely on 'low-level coordination' mechanisms (e.g., CVS, bugs tracking systems and mailing lists) while they lack

higher-level coordination such as group decision making, knowledge management and task scheduling (Cubranic and Booth 1999).

Project performance. A rising number of studies have tried to assess the performance of OSS projects, especially in comparison with proprietary software. For instance, Kuan (2001) has analyzed the rate of bugs resolution in three OSS projects – Apache, FreeBSD and Gnome. The rate of bug resolution is used as a proxy for product improvement or quality. The quality outcomes of these three products are compared with those of equivalent proprietary software with results suggesting a high relative performance of OSS vs. proprietary software. These results may be explained by the fact that users of OSS have access to the source code and therefore they are not purely bug reporters. As such, they can conduct a deeper analysis of the problems and, on some occasions, fix the bugs by themselves and submit the patches to the project maintainers (Hecker 1999).

Similarly, Mockus et al. (2000) have studied different measures of quality for the Apache Server such as defect density (defects per thousand lines of code added) and response time to problems reported by users. This study also shows that Apache software performs relatively well compared with proprietary software.

These results suggest that a strong core of developers is necessary but not sufficient for the success of OSS projects. Highly skilled core developers may lead to many new functionalities. However, if a wider set of peripheral contributors does not join the core developers, the project cannot survive because it will lack the resource needed to find and repair bugs (Mockus et al. 2000). This is a relevant issue because, as mentioned before, only few projects reach the maturity stage and very few are integrated into OSS packages that are distributed through traditional retail channels.

Other studies suggest other indicators of performance such as the ‘popularity score’ (an index based on a combination of record hits, URL hits and subscriptions) and the ‘vitality score’ (calculated from the announcements of official releases) (www.FreshMeat.net).

More recently, a few studies have started to explore more conventional economic indicators of performance like the contributor productivity measured with source lines of code per contributor (developer). For instance, Fershtman and Gandal (2004) have analyzed a sample of 71 OSS projects hosted at the SourceForge website between 2002 and 2003 and found that productivity is lower for projects with more restrictive licenses such as the GPL.

From an economics and managerial perspective it is important to understand what are the project characteristics that help to attract the interest of outsiders, i.e. skilled users that have not participated in the foundation of the project but are interested to contribute by pointing out problems and participating to the discussion about the evolution of the product.

A potentially important characteristic is related to the capabilities of core developers (technical, communication or people skills). Good core developers are more likely to attract new users because their software addresses relevant problems that are not met by commercial products or because it raises technical puzzles that are challenging to the community of developers. In the strategic management literature the capabilities and skills of the founding or core team are considered as a key determinant for the success of new ventures. Well balanced founding teams (or highly skilled single founders) are able to attract financial resources, customers and collaborators (see, for instance, Bhidé 2000; Baron and Hannah 2002) and are more likely to enter new ventures (Lazear 2002). The OSS world in this respect is not very different from the traditional entrepreneurial sector, where new ventures have to overcome the 'liability of newness' and must convince potential stakeholders to pour their resources to support new ideas. The different performance across projects then can be predicted by the different human capital endowment.²

The variety of intended audience may also impact on the performance of a project. In theory, projects that have diversified intended audiences can leverage their innovations on a wider

potential user community and therefore have more feedbacks that help the product to evolve more quickly over time.

2.2. Hypotheses

Drawing on the earlier discussion of the characteristics and performance of OSS development we submit the following hypotheses that will be tested in subsequent sections.

Hypothesis 1. The division of labor at the project level leads to the emergence of leaders which have skill profiles different from those of other contributors such as pure developers.

Hypothesis 2. OSS projects with a high level of different skills generate more new functionalities (new patches and new features) and have a higher likelihood of survival compared with other projects.

Hypothesis 3. Project performance (survival and activity) is also determined by the ability to attract users beyond the set of core contributors. External contributors represent a fundamental resource to improve the quality of the base code.

3. Methods

3.1. Sample

We use a unique dataset containing information on projects and users of the Sourceforge.net website (*SF.net*) from November 3rd 1999 to January 10th 2003. *SF.net* is the world largest repository of OSS projects. The number of projects registered to *Sf.net* has increased quite rapidly since its foundation. Currently, the number of registered projects is over 86,000 and the number of registered users is about 905,000. Other websites hosting open source projects are much smaller than SF.net. For example, Savannah hosts 2,048 GNU and non-GNU projects and 29,639 users have registered to this website (savannah.gnu.org and savannah.nongnu.org). Each

² Obviously, the relationship between performance and skills over a long time is bidirectional. The performance observed at any given time results from the past interaction between initial skill endowment, performance

one of these virtual spaces aims at facilitating the development of OSS emerged from a particular background. For example, while SF.net is a proprietary infrastructure built-up by VA Linux, Savannah is a totally free space created by the Free Software Foundation. So, none of these spaces represents the entire OSS community. Nevertheless, as said, *SF.net* is one of the most diffused tool among developers and one of the most analyzed spaces where development takes place, so that it can be fruitfully used to give a picture wide enough of the OSS phenomenon.

As noted in the previous sections, some authors have used data on small samples of projects. Some of these studies have drawn their information from the SF website (Krishnamurthy 2002; Fershtman and Gandal 2004). But, to our knowledge, only few authors have been able to use a large sample of projects hosted in *Sf.net* (Lerner and Tirole 2002b, for an analysis of the OSS licences; Newby et al. 2002, for an account of the Lotka's Law in understanding software development productivity).

We have obtained from the Sourceforge staff the complete dataset updated at January 2003. Even if the SF dataset does not include a few large and popular projects like Linux, Apache, and Sendmail, several other large and widespread projects like Postgresql, phpMyadmin, Gaim, Python are registered to *SF.net*. Some of the SF projects are normally present in the list of top 20 popular projects of the Freshmeat.net website, hosting the largest index of software with an open source license.

The SF dataset includes several information at different levels of aggregation: the single project, the user, the skills of each user, the single contribution to the project. For the purposes of our analysis we use data on 65,535 projects³ and 544,669 users. It is worth to note that only 83,119 users (15.26% of the sample) are registered to a project.

A “project” is carried out by a group of users working on specific tasks, such as bug fixing, patch developing and reviewing, or broader tasks, such as web site maintenance and release

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

management. A “user” is an individual who contributes to one or more projects. We classified users as *internal contributors* of a given project if they have registered with that project; otherwise they are classified as *external contributors*.⁴

3.2. Measures and descriptive statistics

Our analysis draws on the data about individual users and projects. As far as users are concerned our critical variables are skills and experience, and the role played in the project. For what concerns the projects, the key variables are the level of activity, survival and openness (measured with the participation of external contributors). In this section we describes these measures and show the descriptive statistics of the main regressors used in the econometric analysis to test our hypotheses.

Skills. Data on individual skills are normally very difficult to obtain. The most typical information provided by survey data is about educational background or working experience. The SF dataset provides detailed information on 33 types of skills which can be grouped into three categories: a) Technical expertise (programming languages); b) analysis and design expertise; c) domain or application skills (e.g., networking, security and databases) and c) knowledge of spoken languages.⁵ This information can be provided by users at the time of registration at *Sf.net*, when they are requested to self-assess the level and the experience with each skill declared. Information about skills is available for 51,023 users, 24,563 of which are registered as members of one or more projects (29,446 are the projects with at least one member who declared its skills)

³ In the SF dataset, what is commonly know as “project” is labelled as “group”. Throughout the paper we follow the website notation because it is the most commonly used in the literature and by practitioners.

⁴ External contributors include individuals who have not registered to SF.net and individuals who have registered to SF.net but have not registered to any specific project.

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

while 26,460 are external contributors⁶. Most of our empirical analysis is based on these restricted samples.

Another indirect indicator of skills and ability is the number of projects to which each user contributes.

Drawing on these data we calculated the following measures of skills at the individual level.

- *Days*: number of days since the registration to the Sf.Net website.
- *N_skills*: the number of skills mastered by the user.
- *Skill_level*: the average level of the individuals' skills measured on a five point Likert scale⁷.
- *Skill_experience*: the average experience of skills per user measured on a five point Likert scale⁸.
- *Skill_index*: a synthetic measure which accounts for the number of different skills, the level of each skill, and the experience. The skill index SI for the individual j is calculated as follows: $SI_j = \sum_i (S_{ij})$, where $S_{ij} = (L_{ij} * E_{ij})$ is the skill level and experience of skill i for user j ; L_{ij} = level of skill i for user j ; E_{ij} = experience of skill i for user j .
- *Herf_skill*: the Herfindahl index is used as an indicator of skills diversification and is calculated as follows: $HS_j = \sum_i (S_{ij} / SI_j)^2$.
- *N_projects*: the number of projects to which each individual contributes.

As Table 1 clearly shows, internal contributors (project members), on average, are more skilled than external contributors, as indicated by the number of skills, the skill level, the skill experience and the summary skill index. Moreover, internal contributors have more diversified skills compared with outside contributors.

⁶ The number of users is lower than the number of projects because some individuals participate to more than one project.

⁷ The range of level of skills is the following: 1) Want to learn; 2) Competent; 3) Wizard; 4) Wrote the book; 5) Wrote it.

⁸ The range of experience of skills is the following: 1) <6 months; 2) 6mo-2yr; 3) 2yr-5yr; 4) 5yr-10yr; 5) > 10yr.

As far as project participation is concerned, the average number of projects to which each user is registered is 1.696 with a maximum number of 24 projects for a single user (Table 1). Users for which information about skills is available are registered to a larger number of projects than the average user⁹.

At the project level we computed the following measures based on the skills of their internal members:

- *Size*. Number of project's members.
- *N_skill_project*: number of skills possessed by the members of the project.
- *Skill_level_project*: Average skill level of the project members.
- *Skill_exper_project*: Average skill experience of the project members.
- *Skill_index_project*: Average skill index of the project members.
- *Herf_skill_project*: Average Herfindahl index of the project members.
- *Min_Herf_project*: Minimum Herfindahl index of the project members.

The descriptive statistics of the skills of the projects in Table 1 show variability of the skill level, experience and diversification across different projects.

[Table 1 about here]

Members' role. As mentioned before, the dataset allows to distinguish if the user is an internal member of the project or an external contributor. For internal members the role in the project is specified (developer, project manager, web designer, content manager, all hands persons and other roles)¹⁰. Information about role is useful for studying the organization of the projects and

⁹ We compared the distribution of number of projects per users in the whole sample of users registered to a project and in the restricted sample of users with information with skills. The Mann-Whitney and the Kolmogorov-Smirnov tests of equality of means and distributions of the number of groups in the two samples reject the null hypothesis of equality.

¹⁰ For internal members it is also possible to know if they registered as Master administrator. This role is responsible for the overall project while project managers are responsible for project's modules. A high share of members is registered as master administrators (77.08%) and this depends on the large number of projects with only one member. In these cases obviously the member is also the master administrator.

the skills of individuals with different roles. It is however worth to remind that roles are normally assigned after registration to the project while skills are declared at the time of registration.

In our dataset the largest share of members who provided information about skills works as developer (33.73%). The share of project managers is 18.19%. This share is larger in the sample with skills than in the total sample of users registered to a project, suggesting that people with a key position in the project are more willing to reveal their skills. 52.05% of members cover other roles. Among these 3% are All hands persons and each other role is covered by no more than 1% of the internal members.

Project activity. The dataset provides information about bugs, patches, new features and support requests, tasks¹¹ and CVS that have been submitted and fixed in the period covered by the dataset. The information on the number of file released is also available. Out of 29446 projects in our sample, 4176 have closed at least one bug in the observed period, 1087 at least one patch, 2027 at least one feature request and 10922 have produced at least one file release. From these numbers it is very clear that a very large number of projects does not produce any output, and the activity is concentrated on a small share of projects. On average in each project internal members fix about 8 bugs while external contributors about 11. These numbers are respectively 5.9 and 4.7 for patches, 6.5 and 6.7 for feature requests and 2 and s for support requests. Relying on these data we study the level of activity and performance of individual contributors and projects.

From the dataset it is also possible to know if the project is active, deleted, pending or hold. In our sample 26254 projects are Active (about the 89%) and 3181 are Deleted (11%). Only 11 projects are classified in the remaining categories. These data are important to study project survival and its determinants.

Project openness. As indicators of openness we rely on two sets of variables: a) the share of external contributors to total contributors and the resolution rates of various activities submitted by

internal and external contributors respectively; b) the use of communication channels like surveys, forums and newsgroups which are typical in the OSS community.

We used the SF data at the project level and at the single activity level to generate the following set of variables:

- *N_users*: the number of total contributors to the project.
- *IN_contributors*: the number of internal members who contributed.
- *OUT_countributors*: the number of external contributors.
- *OUT/IN*: share of external to internal contributors of activities (bugs, patches and feature requests) whose resolution state is closed. For computing this indicator we do not include support requests because they are mainly presented by external contributors.
- *Use_forum*, *Use_survey*, *Use_news*, *Use_cvs*: Dummies for the existence of communication tools with the community for each project. This information is provided for each project at the time of registration to SF. In our sample, 84.8% of the projects have a forum toolbox, 91.6% have the newsgroup, 75.7% use surveys and 90.8% use the CVS.

Table 2 shows that on average each project is composed of 1.4 internal contributors¹². 2083 projects in our sample have external contributors. On average 5 external individuals contribute to these projects and the share of external to internal is around 2. This suggests that on average for each internal contributors there are two external contributors to the project.

[Table 2 about here]

CONTROLS. The SF dataset classifies each project according to the following characteristics: intended audience, type of license, programming language, natural language of projects and

¹¹ The definition of “task” is not straightforward because each group uses different policies to manage this instrument. Roughly, it can be defined as a specific objective the group decides to tackle in the next period.

¹² We compared the distribution of number of users per project in the sample of projects with users who provided information about skills and in the total sample of projects. This number is lower for the total sample and the Mann-Whitney and the Kolmogorov-Smirnov tests of equality of means and distributions of the number of users in the two samples reject the null hypothesis of equality.

users, project's date of registration and members' date of registration. At the individual level we also control if the member works in a company through her e-mail address.

4. Results

Before we start with the analysis of the research hypotheses, we present means, standard deviations and correlations among the main explanatory variables used in the empirical models. Most correlations reported in Table 3 are below 0.50 and this indicates that we do not have serious multicollinearity problems.

[Table 3 about here]

Our Hypothesis 1 claims that the division of labor at the project level leads to the emergence of leaders who have skill profiles different from those of other contributors such as occasional developers and users. For our purposes here, we distinguish between three main categories of project members. First, pure project managers (PM) are members who are responsible for specific project tasks such as modules or tasks. They are not significantly involved in development activities although, according to our dataset, they are assigned a considerable number of bugs fixing and new feature requests. Second, pure developers (DV) are primarily involved in technical tasks and are not involved in managerial tasks. It is worth to note that, on average, these are assigned a limited number of tasks such as bugs fixing compared with project managers. The Kolmogorov-Smirnov test for equality of distribution functions rejects the null hypothesis (0.0317, $p=0.00$).

The third category includes members who combine development and project management roles (MPD). These carry out a larger number of technical tasks, such as bugs fixing, as compared with DVs and MPs (K-S: 0.2889; $p=0.000$).

The division of labor among these three categories of members then is not primarily based on a sharp distinction between technical vs. managerial tasks. Instead, it points out different degrees of involvement in the overall project activities, with PMDs being involved in a wide set of

activities as compared with other categories. Finally, members who are not comprised in any of these categories have very limited levels of involvement. This category includes a variety of roles such as master administrators, who are responsible for the project website. The latter are not neatly distinguishable from other categories of members because many projects have only one or two members. In these cases the same person plays the role of master administrator and developer.

As Table 4 illustrates, the three categories of members are also different in terms of skill profiles. PMDs have a higher skill level and a more diversified range of skills as compared with PMs and DVs. It is worth to note that all these three categories of users have higher skill levels and a more diversified skill sets when compared with the rest of project members. Finally, they contribute to a larger number of projects than other members.

[Table 4 about here]

A closer look at the main categories of internal contributors introduced before is made possible by the relatively rich set of data about individuals. To explain which individual characteristics predict the role played in OSS projects we conducted a multinomial logit analysis. Our dependent variable takes on four values corresponding to the four roles described above (PM, DV, DPM and other roles). Our main regressors include the time of registration at *SF.net*, the number of projects subscribed by the individual, various measures of skills (average skill level, skill experience), the average number of members in projects subscribed by the individual, a set of tasks assigned (bugs, feature requests and support requests). Our controls include a dummy for affiliation to a commercial organization and the natural language spoken by the individual.

After running a series of logit regressions for separate categories of roles, we carried out two multinomial logits. The results of the multinomial logit analysis are shown in Table 5. The advantage of multinomial analysis is that it allows pair-wise comparisons between different types of roles. For the sake of simplicity we only show the results of multinomial logit analysis. Table 5

shows two models. In the first model the fourth, residual group of members is used as the comparison group while in the second model PMD is the comparison group.

When compared with the residual group, both DV and PM are more frequent among individuals with more diversified skill sets (lower Herfindhal index). The level of skills has ambiguous or insignificant effects on the likelihood to participate in projects as PM, DV or PMD. The average size of projects participated by our sample individuals instead is an important predictor of their role, especially for DV and PMD. This is in line with the theory that larger projects offer higher opportunities for division of labour between specialists and more general-purpose tasks.

[Table 5 about here]

It is important to note that the effect of skill diversification is particularly strong in the case of PMD. This supports the hypothesis that individuals with a managerial (entrepreneurial) role have more balanced skill sets compared with individuals who play more specialised tasks (Lazear, 2002). Finally, all three categories (DV, PM and PMD) are predicted by the number of bugs assigned and, to a lesser extent, by other categories of technical tasks assigned. As mentioned before, this shows that the division of tasks among these three types of roles is not primarily based on a distinction between purely technical vs. purely managerial activities.

The use of PMD as a reference group is interesting because it allows to compare specialist roles (PM and DV) to more 'horizontal' ones. The first important point to note is that, compared with PMD, both PM and DV have a narrower skill set (larger Herfindahl indexes increase the probability to be in the PM or DV category, compared with PMD). This provides further evidence about the importance of balanced skill sets for people who carry out complex, multi-task jobs. The level of skills is a less important predictor, except for the case of pure DV. Compared with PMD, DV are more likely to be found among less skilled people. Finally,

compared with PMD, other specialized roles are less likely to be observed among members of large projects.¹³

According to hypothesis 2, the skills of project members are important to explain the performance of the project. Our analysis relies on two measures of performance. The first measure is the survival of the project. Our dataset includes 3,181 projects that have been cancelled from SF (11% of total sample). Second, we counted the number of four different types of activity during the last month of the project life (December 2002): bugs closed, patches closed, new feature requests closed and software releases. The choice of this time window is driven by the aim of minimizing the potential endogeneity of some regressors.

Table 7 shows the logit estimation of the determinants of the projects' survival. The minimum project's Herfindahl index (a measure of skill diversification of the most diversified member of the project) has always a negative effect on the probability of survival. This suggests that projects with more varied skill sets have better chances to remain active. The coefficient of skill level is insignificant because of correlation with the Herfindahl index and other regressors. The average number of projects participated by the user is another proxy for users' expertise. Skilled users have probably more chances to be invited to join new projects as compared with other users. The negative effect of this variable on the likelihood of survival indicates that the potential benefits arising from the participation of skilled members may be offset by their over commitment with too many projects.

[Table 7 about here]

To further explore the implications of member skills for project performance we estimated four independent project activity equations. Estimates were obtained by maximum-likelihood zero-inflated Negative Binomial regression. This specification was preferred to alternative ones because of the nature of the dependent variables (nonnegative count data). In particular, we

¹³ The Hausman test of independence of irrelevant alternatives (IIA) yields mixed results. We have also run single logit estimations which suggest that the different roles represent different individual profiles (types of tasks assigned, skills etc.).

opted for the negative zero-inflation binomial model to deal with cross-sectional heterogeneity and the large number of zeroes in the dataset (see Greene, 1997).

The estimation results are reported in Table 8. By and large, a diversified mix of skills yields positive effects on the number of closed bugs, patches, and feature requests. It also has a positive influence on the likelihood of project releases. The effect of skill level is more ambiguous and often not significant at the conventional levels. Instead, with the exception of the feature requests equation, skill experience has always a positive effect on project activity. Like in the survival equation, the average number of projects participated by the members has a negative effect on the project activity.

[Table 8 about here]

Our hypothesis 3 claims that external contributors or, more generally, the strength of the links with the OSS community, are important for the project performance. The results of the survival equation reported in Table 7 show that the share of external contributors (users who have not subscribed with the project but have filed at least one contribution over the sample period) has a positive additional effect beyond that of internal members' skills. The positive effect of the dummy that indicates the use of forums by the project also points out the importance of the channels with the OSS community for survival.¹⁴ By the same token, a large number of diverse audiences addressed by the project increases the opportunity of contacts with a wide portion of the OSS community and this explains the positive effect on survival, although the estimates are not significant at the conventional levels.

When we look at the estimates of activity equations we find that, by and large, the share of external contributors have a positive and significant effect on different types of activities (Table 8). Similarly, the number of different audiences addressed has a positive influence on the number of closed artifacts, except for the closed patches equation where the coefficient is estimated with

a low precision.

The skills of core developers are not enough to survive. Projects that do not attract users beyond the set of core contributors will not survive because they will lack the resources needed to improve the base code by finding and repairing bugs.

These results then provide substantial support to our hypotheses. For what concerns the activity equations, as Table 8 shows, the large number of zero observations considerably reduces the sample size. We then must warn against the generalization of our results to the rest of the OSS population. However, these findings are interesting for the following reasons. First, they provide further evidence in favor of our earlier results on survival. Second, they show that there exists a quite limited number of projects that are really active in the population of OSS projects. Third, for this subset of projects we have collected evidence that points out the importance of both internal determinants of performance (member skills) and external factors (that is the links with the community which are accounted for by the share of external contributors and the variety of audiences addressed).

Finally, in future research we shall replicate our estimates by trying different time windows for project activities. As mentioned before, the present choice of activities conducted during the last month of the sample was induced by the need to deal with the potential endogeneity of regressors.

5. Discussion and conclusions

The main purposes of this paper were: (i) to provide novel evidence about the characteristics of OSS project leaders compared with other contributors; (ii) to study the main determinants of projects' survival and activity. To study these topics we have relied on a large sample of OSS project registered with the Sourceforge dataset.

¹⁴ We have also analyzed the effects of other communication channels adopted by the projects, such as the use of surveys and newsgroups. Since these variables (including use of forums) are highly correlated with each other, we opted for the use of forum because it appears to be a quite popular communication channel in the OSS community.

To address the first issue we have classified project members by their roles as reported in the *Sf.net* dataset. More precisely, we have distinguished among three categories of members: pure developers, pure project managers and multi-task contributors who carry out both development and management activities at the project level. Moreover, for each individual contributor we collected information about the date of registration with the project, the number of different skills declared at the time of registration, the level of expertise in each skill, the number of different projects she contributes, and the number of artifacts contributed (e.g., bugs fixing and patches assigned). Our multinomial logit estimations show that there exist marked differences between ‘specialists’ (pure project managers and pure developers) on the one side and ‘multiple task’ members on the other. It is important to recall that multiple task people carry out a larger number of different technical tasks (such as bug fixing) as compared with ‘specialists’ (including ‘pure developers’). These individuals correspond to the ‘core’ of highly committed programmers studied in earlier works (e.g., Koch and Schneider, 2002; Mockus et al., 2000). Moreover, these individuals have more diversified skills as compared with ‘specialists’. It is worth to recall that the skills considered in our study include three distinct categories of expertise: technical, application or domain-related and people or communication skills (proxied by the knowledge of foreign languages). In this respect, then, our analysis contributes to the literature on the economics and management of OSS by offering a quantitative measure of different skill and task profiles at the project level. In particular, we identify a set of individual characteristics that the literature has pointed out as critical capabilities that leaders must possess to manage efficiently geographically dispersed development teams. Previous studies have made the point that leaders have to be credible, by convincing the community of their technical skills and programmers have to trust the leadership. Trust results from experience and the leaders’ deliberate action. As Lerner and Tirole put it ‘a good leadership should also clearly communicate its goals and evaluation procedures’ (p. 24). Different software experts point out the importance of ‘people’ skills relative to technical skills. Some authors claim that coordinators do not necessarily need strong

design expertise; instead, they have to identify good design ideas of other developers (Sanders 1998). Social and communication skills are important to attract and retain good developers. Finally, leaders have to select developers and the team leaders (Hecker 1999).

To deal with the second issue we distinguished internal factors (the skill characteristics of internal contributors or project members) from external ones (the participation of external contributors and the number of intended audience of the project). As previous works have pointed out, the skill of internal contributors are not enough for project survival. The lack of external contributions reduces the level of project activity and may lead to its failure (Mockus et al., 2000). For the purposes here, we started with the analysis of project survival and found out that a diversified skill set and skill experience have significant positive effects on the likelihood of survival. Moreover, the share of external contributors over total contributors, the use of project forums, and the number of different intended audiences (e.g., end users, developers and system administrators) have a positive effect on project survival. To understand the effect of project's skills and external contributors on project activity we have we have counted the number of closed bugs, feature requests and patches, and the new product releases over the last month of activity in the dataset. Our results show that skill diversification and skill experience have also a positive effect on project activity. The share of external contributors and the number of different intended audiences have a significant effect beyond that of member skills. These findings provides new evidence that supports previous research on the performance of OSS projects. Many earlier works have emphasized the importance of external contributions as a key distinctive characteristic of the OSS development paradigm as compared with the traditional software development model (Raymond, 1999; von Hippel, 2001; Lerner and Tirole, 2002b). Moreover, unlike earlier works that have focused on specific dimensions of performance (e.g., Kuan, 2001; and Fershtman and Gandal, 2004), our analysis provides insights into several dimensions of performance (from survival to bugs fixing and new releases). Finally, to the best

of our knowledge, this is the first work which analyzes the joint action of ‘internal’ and ‘external’ determinants of project performance.

The large cross-sectional dataset at the level of single projects and users which we used for our analysis offers new opportunities for exploring important characteristics of the OSS development community that have remained unexplored so far because of lack of data. This study shows the usefulness of large datasets for the analysis of the population ecology of OSS projects. Our analysis has various limitations, some of which arising from the nature of data used. First, we relied on mere counts of tasks assigned to individual programmers but we had no access to data on lines of codes produced. Future research should try to integrate the information about different types of tasks with more precise measures of programming effort. Second, our analysis provides a very incomplete picture of the linkages between projects. We found out that external contributors have positive effects on the project performance but we have not examined which projects are joined by whom. A deeper analysis of the matching process between individuals and projects could provide additional information about the internal composition of project teams. It can also help to get a finer grained picture of knowledge flows across projects that take place through programmers’ mobility and joint participation in different projects.

Third, additional information should be collected at the level of single contributors to better understand their background (e.g., level of education and working experience) and affiliations.

References

- Baron, J.N., M.T. Hanna. 2002. Organizational Blueprints for Success in High-Tech Start-ups. *California Management Review*, **44**(3) 8-36.
- Bhidé, A.V. 2000. *The Origin and Evolution of New Business*. Oxford University Press, Oxford.

- Cübranić D., Booth K. (1999), “Coordinating Open Source Software Development”, IEEE Proceedings, 8th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, (WET ICE '99), pp. 61–66.
- Dalle, J., N. Jullien. 2001. *Open-source vs. Proprietary Software*. Working paper. http://opensource.mit.edu/online_papers.php
- Dalle, J.M., P.A. David. 2003. *The allocation of software development resources in ‘open source’ production mode*. <http://opensource.mit.edu/papers/dalledavid.pdf>.
- Di Bona, C., S. Ockman, M. Stone, eds. 1999. *Open Sources: Voices from the Open Source Revolution*. O’Reilly, Sebastopol, CA.
- Elliot, M. S. and Scacchi, W. 2003, free software: a case study of software development in a virtual organizational culture, Institute for Software Research, University of California, Irvine, CA, mimeo, April.
- Kaisla, J. 2001 Constitutional dynamics of the open source software development, Dept. of Industrial Economics and Strategy, Copenhagen Business School, mimeo, May.
- Minzberg, H. 1979, the structuring of organization , Prentice Hall, Englewood Cliffs, NJ.
- Faraj, S., L. Sproull. 2000. Coordinating Expertise in Software Development Teams. *Management Science*, 46(12) 1554-1568.
- Fershtman C. and Gandal, N. (2004) The determinants of output per contributor in open source projects: an empirical examination, mimeo, Tel Aviv University, March 1st.
- Ghosh, R. A., P.A. David 2003. *The nature and composition of the Linux kernel developer community: a dynamic analysis*. Working Paper SIEPR-Project NOSTRA, <http://dxm.org/papers/licks1/>.
- Giuri P., G. Rocchetti, S. Torrisi 2002. *Open source software: from open science to new marketing models an enquiry into the economics and management of open source software*. Working Paper 23, LEM.
- Greene, W. H. 1997. *Econometric Analysis*, Prentice-Hall International. Upper Saddle River, NJ.
- Koch S. and Schneider, G. (2002) Effort, co-operation and co-ordination in an open source software project: GNOME, *Info Systems J.* 12: 27-42.

- Krishnamurthy, S. 2002. Cave or Community? An Empirical Examination of 100 Mature Open Source Projects. *First Monday* 7(6). http://firstmonday.org/issues/issue7_6/.
- Kuan, J. (2001) open source software as consumer integration into production, Haas School of Business, University of California at Berkeley, CA, mimeo, January.
- Hecker F. (1999), "Setting Up Shop: The Business of Open-Source Software", IEEE Software, Jan/February, pp. 45-51.
- Lakhani, K., E. von Hippel 2000. *How open Source software works: 'Free' user-to-user assistance*. Working paper 4117, MIT Sloan school of Management.
- Lazear, E. P. 2002. *Entrepreneurship*, Working paper 9109, NBER.
- Lerner J., J. Tirole. 2002a. Some simple economics of Open Source. *The Journal of Industrial Economics* **L**(2) 197-234.
- Lerner, J., J. Tirole. 2002b. *Working Paper Series The Scope Of Open Source Licensing*. Working Paper 9363, NBER. <http://www.nber.org/papers/w9363>
- Mockus, A., R.T. Fielding, J. Herbsleb. 2000. A Case Study of Open Source Software Development: The Apache Server. *Proc. of the Twenty-Second Internat. Conf. on Software Engineering* 263 –272.
- Newby, G. B., J. Greenberg, P. Jones. 2002. Open Source Software Development and Lotka's Law: Bibliometric Patterns in Programming. *Journal Of The American Society For Information Science And Technology*. **54**(2) 169-178.
- Raymond, E. S. 1999. Linux and Open-Source Success. *IEEE Software* **16**(1) 85-89.
- Raymond, E. S. 2001. *The Cathedral and the Bazaar. Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly, Sebastopol, CA.
- Von Hippel, E. 2001. Learning from Open Source Software. *Sloan Management Review* **42**(4) 82-86.
- Wheeler, D. 2002. *Why Open Source Software/Free Software (OSS/FS)? Look at the Numbers!*. http://www.dwheeler.com/oss_fs_why.html.

Table 1. Users' and projects' skills

	N	Min	Max	Median	Mean	SD
Internal contributors who provide information about skills						
N_skills	24563	1	29	6	6.570	3.607
Skill level	24563	1	5	2.18	2.243	.487
Skill experience	24563	1	5	2.75	2.765	.736
Skill index	24563	1	650	38	44.217	30.721
Herfindahl skills	24563	.03	1	.21	.279	.218
N_projects	24563	1	24	1	1.696	1.343
External contributors who provide information about skills						
N_skills	26460	1	33	5	5.888	3.495
Skill level	26460	1	5	2	2.166	.513
Skill experience	26460	1	5	2.71	2.747	.808
Skill index	26460	1	375	32	38.302	28.639
Herfindahl skills	26460	.04	1	.23	.322	.249
Projects with internal contributors who provide information about skills						
N_skill_project	29446	1	31	7	8.001	4.620
Skill_level_project	29446	1	5	2.21	2.271	.463
Skill_exper_project	29446	1	5	2.75	2.776	.687
Skill_index_project	29446	1	650	42	46.829	30.750
Herf_skill_project	29446	0.03	1	0.2	0.266	0.201
Herf_skill_min	29446	0.03	1	0.18	0.246	0.201

Table 2. Projects' contributors

	N	Min	Max	Median	Mean	SD
IN_contributors	29446	1	43	1	1.415	1.281
OUT_contributors	2083	1	1138	1	4.979	28.479
OUT/IN	1592	0.0435	162.571	1	2.193	6.704

Table 3. Correlation matrix of main regressors at the individual level

	n_projects	Avgsizeothe rproj.	herf_index	avg_sk_lev	avg_sk_exp	bugs_as.	patches_as.	Feat_req_as	supp_req_a.
n_projects	1.0000								
avgsizeotherproj	-0.0279***	1.0000							
herf_index	-0.0852***	-0.0061	1.0000						
avg_skill_lev	0.0457***	-0.0179***	-0.0294***	1.0000					
avg_skill_exp	0.0233***	0.0570***	-0.0422***	0.4342***	1.0000				
bugs_assigned	0.1611***	0.0437***	-0.0212***	0.0180***	0.0373***	1.0000			
patches_assigned	0.1292***	0.0312***	-0.0137***	0.0108**	0.0316***	0.4679***	1.0000		
feat_req_assigned	0.0869***	0.0170***	-0.0235***	0.0125***	0.0267***	0.3963***	0.1096***	1.0000	
supp_req_assigned	0.0414***	-0.0000	0.0015	0.0098**	0.0052	0.0704***	0.0083***	0.2087***	1.0000

Table 4. Skill index in different roles

	Min	Max	Median	Mean	St dev	N
<i>Skill index</i>						
Project Manager	1	279	42	48.060	31.870	3494
Developer	1	315	38	43.920	29.720	7311
Project Manager & Developer	2	220	48	53.648	33.030	973
Other	1	650	36	42.617	30.570	12784
<i>Diversification index</i>						
Project Manager	.05	1	.20	.260	.199	3494
Developer	.04	1	.20	.270	.212	7311
Project Manager & Developer	.06	1	.17	.270	.212	973
Other	.03	1	.21	.293	.228	12784
<i>Number of projects</i>						
Project Manager	1	24	1	1.596	1.208	8618
Developer	1	17	1	1.504	1.116	24726
Project Manager & Developer	2	33	3	3.781	2.474	1723
Other	1	5	1	1.268	0.765	48032

Table 5. Multinomial logistic regression: Members' role

	Multinomial logit (Base=Others)			Multinomial logit (Base=PM+Dev)		
	Developer	PM	Dev+PM	Other	Dev	PM
days	0.001 (0.000)***	0.001 (0.000)***	0.001 (0.000)***	-0.001 (0.000)***	0.001 (0.000)***	0.000 (0.000)
n_projects	0.454 (0.016)***	0.391 (0.019)***	0.847 (0.022)***	-0.847 (0.022)***	-0.393 (0.018)***	-0.456 (0.021)***
Avg_sizeotherproj	0.084 (0.003)***	-0.003 (0.005)	0.079 (0.005)***	-0.079 (0.005)***	0.005 (0.004)	-0.082 (0.006)***
herf_index	-0.433 (0.073)***	-0.656 (0.096)***	-1.206 (0.212)***	1.206 (0.212)***	0.773 (0.213)***	0.550 (0.223)**
avg_skill_lev	-0.171 (0.036)***	0.048 (0.043)	0.056 (0.085)	-0.056 (0.085)	-0.227 (0.085)***	-0.008 (0.089)
avg_skill_exp	0.031 (0.023)	0.089 (0.029)***	0.016 (0.056)	-0.016 (0.056)	0.016 (0.057)	0.073 (0.060)
bugs_assigned	0.012 (0.003)***	0.014 (0.003)***	0.013 (0.003)***	-0.013 (0.003)***	-0.001 (0.002)	0.001 (0.002)
patches_assigned	-0.003 (0.014)	0.016 (0.014)	0.006 (0.015)	-0.006 (0.015)	-0.009 (0.010)	0.010 (0.007)
feat_req_assigned	0.018 (0.009)**	0.038 (0.008)***	0.046 (0.010)***	-0.046 (0.010)***	-0.028 (0.009)***	-0.009 (0.008)
supp_req_assigned	-0.002 (0.004)	-0.001 (0.000)***	-0.001 (0.001)	0.001 (0.001)	-0.001 (0.004)	0.000 (0.001)
Constant	-2.078 (0.119)	-2.549 (0.148)***	-4.965 (0.287)***	4.965 (0.287)***	2.887 (0.288)***	2.416 (0.303)***
Log likelihood	-24797.788			-24797.788		
Number of obs.	24546			24546		
LR chi2 (d.f.)	4700.67 (51)			4700.67 (51)		
Prob > chi2	0			0		
Pseudo R2	0.0866			0.0866		

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

Controls: Dummies for natural language (English, Spanish, German, French, Italian, Russian), Dummy for e-mail address.com).

Table 6. Correlation matrix of main regressors at the project level

	avg_n_pr.	Min_herf.	avg_sk_lev	avg_sk_exp	Out/in_con.	use_forum	use_survey	use_cvcs	use_news	N_audience
avg_n_projects_of_members	1.0000									
Min_herf_ind	-0.1093***	1.0000								
avg_skill_lev	0.0513***	-0.0012	1.0000							
avg_skill_exp	0.0412***	-0.0603***	0.3888***	1.0000						
Out/in_contributors	0.0213***	-0.0242***	0.0068	0.0316***	1.0000					
use_forum	-0.1180***	0.0566***	-0.0006	-0.0488***	-0.0407***	1.0000				
use_survey	-0.1000***	0.0606***	0.0061	-0.0382***	-0.0605***	0.6540***	1.0000			
use_cvcs	-0.0260***	-0.0042	-0.0146**	0.0131**	-0.0221***	0.3643***	0.4192***	1.0000		
use_news	-0.0761***	0.0226***	0.0004	-0.0303***	-0.0220***	0.5621***	0.5196***	0.3852***	1.0000	
N_audience	-0.0069	-0.0528***	0.0419***	-0.0183***	-0.0023	0.0220***	0.0201***	-0.0112**	0.0269***	1.0000

Table 7. Logistic regression: Project Survival

	Active (no controls)	Active (with controls)	Active (with controls + use_forum)
avg_n_projectsbymembers	-0.114 (0.015)***	-0.092 (0.021)***	-0.080 (0.021)***
Min_herfindahl_index	-0.634 (0.223)***	-0.481 (0.293)	-0.490 (0.292)*
avg_skill_level	-0.234 (0.110)**	-0.172 (0.138)	-0.171 (0.138)
avg_skill_experience	0.323 (0.078)***	0.221 (0.098)**	0.219 (0.099)**
Out/in_contributors	1.165 (0.325)***	0.761 (0.305)**	0.842 (0.314)***
use_forum			0.650 (0.132)***
N_audience	0.105 (0.061)*	0.148 (0.145)	0.138 (0.145)
Constant	3.657 (0.286)***	3.426 (0.596)***	2.968 (0.605)***
CONTROLS			
Size		Yes	Yes
Dummies for entry cohorts		Yes	Yes
Dummies for Intended Audience		Yes	Yes
Dummies for type of licence		Yes	Yes
Dummies for programming language		Yes	Yes
Dummies for spoken language		Yes	Yes
Log likelihood	-2174.99	-1399.93	-1388.8478
Number of obs.	20774	15805	15805
LR chi2 (d.f.)	110.55 (6)	204.49 (32)	226.66 (33)
Prob > chi2	0	0	0
Pseudo R2	0.0248	0.0681	0.0754

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

Controls: Dummies for entry cohorts: old, young; Dummies for Intended Audience: developers, users, system administrators; Dummies for type of licence: GPL, LGPL, BSD, Public domain, Artistic licence, Apache; Dummies for Programming language: C, C++; Java, Php, Perl, Python, Visual Basic, Unixshell; Dummies for natural language: English, Spanish, German, French, Russian, Dummy for e-mail address.com, time of registration at SF.net).

Table 8. Zero-Inflated Negative Binomial Regression (inflation model: Logit): Project Activity

	Bugs ⁺ (no controls)	Bugs (with controls)	Patches (no controls)	Patches (with controls)	Feature requests (no controls)	Feature requests (with controls)	File releases (no controls)	File releases (with controls)
avg_n_projectsbymembers	-	-0.104 (0.034)***	0.018 (0.047)	0.017 (0.059)	-0.220 (0.043)***	-0.083 (0.047)*	-0.060 (0.019)***	-0.056 (0.021)***
Min_herfindahl_index	-	-1.017 (0.372)***	-3.060 (0.823)***	-1.438 (0.897)	-2.420 (0.508)***	-1.370 (0.565)**	-0.747 (0.206)***	-0.426 (0.223)*
avg_skill_level	-	0.103 (0.163)	-0.612 (0.327)*	-0.542 (0.398)	-0.334 (0.215)	-0.100 (0.235)	-0.248 (0.095)***	-0.324 (0.101)***
avg_skill_experience	-	0.247 (0.106)**	1.070 (0.191)***	0.773 (0.226)***	0.189 (0.126)	0.172 (0.148)	0.344 (0.063)***	0.380 (0.067)***
Out/in_contributors	-	0.597 (0.089)***	1.128 (0.123)***	0.278 (0.132)**	0.863 (0.084)***	0.428 (0.103)***	0.255 (0.039)***	0.212 (0.049)***
use_forum	-	-0.853 (0.146)***	-0.295 (0.249)	-0.288 (0.313)	-1.397 (0.163)***	-1.310 (0.202)***	-0.369 (0.090)***	-0.494 (0.100)***
N_audience	-	0.356 (0.098)***	0.079 (0.131)	0.159 (0.225)	0.060 (0.075)	0.358 (0.130)***	0.149 (0.040)***	0.217 (0.064)***
Constant	-	-4.726 (0.616)***	-6.140 (0.831)***	-7.176 (1.547)***	-1.843 (0.542)***	-4.843 (0.900)***	-2.133 (0.240)***	-2.518 (0.361)***
CONTROLS								
Size		Yes		Yes		Yes		Yes
Dummies for entry cohorts		Yes		Yes		Yes		Yes
Dummies for Intended Audience		Yes		Yes		Yes		Yes
Dummies for type of licence		Yes		Yes		Yes		Yes
Dummies for programming language		Yes		Yes		Yes		Yes
Dummies for spoken language		Yes		Yes		Yes		Yes
Log likelihood		-3304.275	-1038.129	-694.5862	-2145.686	-1634.06	-6844.047	-5457.56
Number of obs.		15805	20774	15805	20774	15805	20774	15805
Number of non-zero obs.		563	151	106	334	270	1334	1125
LR chi2 (d.f.)		1034.11 (33)	332.96 (7)	297.02 (33)	574.2 (7)	641.9 (33)	189.81 (7)	531.98 (33)
Prob > chi2		0	0	0	0	0	0	0

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

+ The regression relative to Bugs without controls does not converge.

Controls: Dummies for entry cohorts: old, young; Dummies for Intended Audience: developers, users, system administrators; Dummies for type of licence: GPL, LGPL, BSD, Public domain, Artistic licence, Apache; Dummies for Programming language: C, C++; Java, Php, Perl, Python, Visual Basic, Unixshell; Dummies for natural language: English, Spanish, German, French, Russian, Dummy for e-mail address.com, time of registration at SF.net).