From Planning to Mature: on the Determinants of Open Source Take Off*

Stefano Comino† Fabio M. Manenti‡ Maria Laura Parisi§

July 2005

Abstract

In this paper we use data from SourceForge.net, the largest open source projects repository, to estimate the main determinants of the progress in the development of a stable and mature code of a software. We find that the less restrictive the licensing terms the larger the likelihood of reaching an advanced development status and that this effect is even stronger for newer projects. We also find that projects geared towards system administrators appear to be the more successful ones. The determinants of projects' development stage change with the age of the project in many dimensions, i.e. licensing terms, software audience and contents, thus supporting the common perception of opens source as a very dynamic phenomenon. The data seem to suggest that open source is evolving towards more commercial applications.

Keywords: software market, open source software, development status, intended audience, license

JEL classification: O38, L51, L63.

*We wish to thank Nicolas Garrido for the extremely helpful discussions and Riccardo Marcon for research assistance.

†Dipartimento di Scienze Economiche, Università di Trento, Via Inama 5, 38100 TRENTO (Italy), Tel. (39) 0461 882221, email: scomino@mail.economia.unitn.it

‡Corresponding author: Dipartimento di Scienze Economiche "M. Fanno", Università di Padova, Via del Santo 33, 35123 PADOVA (Italy), Tel. (39) 049 8274238, email: fabio.manenti@unipd.it

§Dipartimento di Scienze Economiche, Università di Brescia, Via San Faustino 74/b, 25122 BRESCIA (Italy), Tel. (39) 030 2988 826, email: parisi@eco.unibs.it

1 Introduction

Open Source Software (OSS) represents one of the most interesting and debated phenomena in the software industry. Various authors have scrutinized several aspects of OSS both from a theoretical and empirical perspective; why OSS exists, how a community of unpaid developers works and coordinates, why and how for-profit firms have decided to embrace OSS strategies are amongst the most puzzling arguments under scrutiny. In a nutshell, the large set of issues raised by the existence and the functioning of open source can be reduced to only one basic and simple question: what determines the success of an open source project?

According to Crowston et al. (2003), there are three main interrelated proxies that can be used to measure the extent to which an open source software succeeds: *i*) software creation, *ii*) software quality, and *iii*) software use. The first category includes several indicators of system success: from the size and level of activity of the members of a community of an OSS project, to the ability of the community to develop a stable and mature software. According to the second category, a software system is successful when its source code is of high quality, e.g. it is highly modular, correct and maintainable. Finally, the supporters of the third category consider a software as successful when it has been widely adopted by final users.[1]

Recently, various "case studies" contributed to a better understanding of the OSS phenomenon. Hertel et al. (2003) have tried to identify what determines the level of engagement of 141 developers into the large Linux project. Krogh et al. (2003) analyzed the strategic process by which new individuals join the community of developers of FreeNet, a peer-to-peer network of information distribution. In Lakhani and von Hippel (2003) the focus has been put on the nature and the functioning of the community of developers of the well known OSS Apache.

These studies shed new light on how large communities of developers arise, work and coordinate to obtain a successful software; the main limitation is that they focus on large and popular projects. While a closer look on such projects is crucial to understand better how communities work effectively, it may not be enough to explain the very basic nature of open source: large projects are only the "top of the iceberg" of the OSS world which is made of thousands of small, often individually run projects, which are almost unknown

[1] It is often difficult to measure OSS adoption; Krishnamurthy (2002) proxies it with total number of downloads.

outside the small circle of purist OSS developers. In this paper we look at the part of the OSS iceberg that is below the water line. We use data from SourceForge.net, the largest open source projects repository, to estimate the main determinants of the progress in the development of a stable and mature code of a software. As discussed in Crowston et al. (2003), the development stage of a project can be considered as a possible representation of the level of success of a software.

This measure of project's success moves our paper apart from the most recent literature: Fershtman and Gandal (2004) consider an alternative definition of system success, based on output per contributor; they examine how the type of license, the programming language, the audience to which the software is intended to and other factors affect the output per contributor in OSS projects. The empirical analysis is based on 71 projects observed during a period of eighteen months; their main result is that output per contributor of open source programs is significantly much higher when the scope of the license of the software is narrower, that is, when the terms for using, modifying and re-distributing the software are less restrictive.

The complex set of motivations that drives the choice of OSS license is at the center of Lerner and Tirole (2005); these authors point out that when choosing the terms under which her/his software is licensed, a project administrator has to balance opposing effects. On the one hand, by choosing a restrictive license that poses severe limitation to the possibilities of use and modification of the software the project administrator reduces the opportunity of commercial exploitation of the software, yet she/he is more likely to attract the contribution of the open source community, especially those programmers who are motivated by more idealistic, non-monetary rewards. On the other hand a less restrictive license (i.e. a license which is narrow in scope) potentially enlarges the monetary rewards that the administrator can obtain but, at the same time, it renders less likely the contribution to the software by the community (except for those who are motivated by the possible monetary rewards). These theoretical predictions are then tested by the two authors. The empirical analysis is performed using a compilation of nearly 40,000 open source projects hosted on SourceForge.net. According to their investigation, restrictive licenses are more likely to be chosen when software is geared toward end-users and/or for software developed in corporate settings.

Our analysis adds to these papers in many respects. Contrary to Fershtman and Gandal

(2004), who restrict their study to the most active projects and, strictu sensu, also to the most successful ones, we want to explore the determinants of the development status of open source projects focusing on the largest possible sample. Furthermore, we actually reverse the Lerner and Tirole's causal relationship; usually, the type of license according to which the software is developed and distributed is determined by the project leader at the time of launching the project: for this reason, we believe that it is more reasonable to measure how the choice of the license of a project affects its probability of reaching a stable and mature version, i.e. of being successful, rather than the opposite.

Another interesting characteristic of the open source movement that has been frequently pointed out refers to its evolving nature. For instance, Raymond (1998) observes that the historic evolution of OSS has come in different waves. During the seventies, most of the time and effort of programmers were devoted to develop toys and demos; in the eighties it has been the turn of Internet tools while in the nineties the interest shifted toward operating systems. The prediction for the future is that effort will be concentrated on "the last virgin territory" i.e. the development of programs/applications for non-techies.

The ever changing nature of OSS is also recognizable when looking at the "actors" of the community. While in the early days of the movement the community was based on a limited number of software experts devoting their spare time to program new artifacts, nowadays the for-profit, commercial world is heavily involved in the open source arena.

There are different ways for a commercial actor to enter the OSS world. One way is through the main entrance: many of the largest software and hardware producers have already launched or are under the process of launching open source projects. As discussed in West (2003), market leaders such as IBM, Apple and Sun have in various ways embraced open source strategies. In these cases, the choice of the licensing terms is crucial since a balance between openness, guaranteed by the "strict" open source licenses, aimed at reaching the wider possible audience, and control, aimed at keeping proprietary (and profitable) modifications may really determine the success of the project. Another way for profit oriented firms to enter the OSS world is to specialize into the supply of complement goods such as the creation of documentation, the provision of support services or the development of more user-friendly interfaces of open source packages.

It seems therefore apparent that since its origin open source has really experienced a drastic change in its nature and characteristics; we devote part of the paper to look for an

empirical evidence of this evolution by analyzing if and how the impact of licensing terms, of software audience and content on projects' success has changed through time.

The paper is organized as follows: Section 2 presents the dataset used for the estimations; Section 3 discusses more in details some conjectures about the main drivers of OSS development stage; these conjectures are then tested in Section 4 while Section 5 concludes.

2 The dataset

The dataset consists of all the open source projects that were hosted on SourceForge.net in December 2004.[2] SourceForge.net (SF hereafter) is the largest online platform which provides OS developers with useful tools to control and manage software development.[3] Project administrators register their software project on SF and provide most of the information which is then available on-line. Registration is for free and administrators are encouraged to register their projects as well as to maintain up-to-date all the relevant information.

For each registered project, SF provides the following information:

- *development status*: information regarding the development stage reached by the project at the data collection date. Each project can be classified into one (or more) of six different levels, from the earliest stage of production to a fully developed software: planning, pre-alpha, alpha, beta, production stable and mature. Even though the meaning of some levels is self-evident, no formal definition of these six stages is available on SF;

- *registration date*: date at which the project has been registered on SF;

- *number of developers* that have joined the project at the data collection date;

- *number of bugs, patches and feature requests*: developers and users may submit bug reports, feature requests and source code patches; for each project on SF, a dedicated

[2]We employ the dataset crawled by Dawid Weiss; data are available at public URL `www.cs.put.poznan.pl/dweiss/` and full documentation of the crawling system is in Weiss (2005).

[3]Freshmeat.net is the second largest repository with a number of hosted projects which is less than half of those on SF. Other minor repositories are GNU Savannah, CodeHaus, GridForge and CPAN.

tracker system automatically updates all these information about each project's activity;[4]

- *license*: the licensing terms under which the code is released, distributed and modified; projects listed on SF are grouped into more than 40 different licensing schemes. In our analysis, we have aggregated all these categories by using the scheme suggested in Lerner and Tirole (2005) and Välimäki (2005) where licenses are classified into three categories according to the degree of restrictions they impose on the use of the software: highly restrictive licenses (with copyleft and viral restrictions,[5] e.g. GPL), restrictive (only copyleft, e.g. LGPL) and unrestrictive licenses (e.g. BSD).

- *intended audience*: information about software audience. SF has adopted the following categories: end-users, developers, system administrators, information technology, customers, finance, education, manufacturing, research, telecommunications, and others. Many projects fall in more than one category, meaning that they may be of interest to more than one type of audience;

- *topic*: information about software content. A project may be dedicated to: communications, database, desktop environment, education, games-entertainment, Internet, multimedia, office-business, science-engineering, security, software development, system, terminals and text editors;

- *programming language* used to write the program and the *operating system* that supports it.

Since its beginning in 1999, SF requests each project administrator to provide all these information about his/her software; nevertheless, unlike for development status and topic which classification appears to be stable through time, SF has slightly modified the classification adopted to describe projects' intended audience. Up to 2002, SF has grouped projects into only three categories (a part from the residual one, called "others"): end users,

[4]The feature request tracker allows user to suggest or to require enhancements of the software. The patches are lines of code that can be submitted both by users as well as by developers.

[5]Copyleft is the short name for a legal framework to ensure that derivatives of a licensed work stay free/open. Copyleft licenses are sometimes referred to as viral copyright licenses when any works derived from a copylefted work must themselves be copylefted. In this case, the program cannot be compiled with proprietary programs

developers and system administrators. Since 2002, 7 new categories have been added to the list: information technology, customers, finance, education, manufacturing, research and telecommunications. In Section 4 we will discuss how we have dealt with this change in the classification of intended audience.

2.1 Caveat: the quality of the data

The dataset we employ is extremely large and it contains detailed information; nevertheless it presents some potential shortcomings that must be discussed before proceeding with the estimation.[6]

First of all, we need to observe that most of the available information is based on declarations of the project administrators rather than on objective measures. In some instances, in particular in the case of the development status of the project, these declarations depend on the subjective evaluation/perception of the project leader. Even though we are not able to verify the quality of these information, we believe that projects leaders do not have incentives to misrepresent them or not to keep them up-to-date. One of the aims of project leaders is to attract other members of the community and to persuade them to join the project and this inherently induces them to provide correct information about their activity; SF itself in various documentation encourages leaders and administrators to keep a correct behavior. As reported in Lerner and Tirole (2005) "undertaking a 'bait-and-switch' strategy at the time of recruiting new users - e.g., by making the project appear to be something other than what it really is - is unlikely to be a positive signal to prospective developers."

A frequently claimed limitation of SF data relates to the alleged large number of projects that have been registered but that are actually abandoned (and for which the project leaders do not care about cancellation from SF). A closer look to our dataset seems to confirm this observation: around 80% of the projects does not show any interaction within the community of developers, having recorded no bugs, patches nor feature requests since its registration. According to our view, the absence of any activity may accrue to two different explanations. First, it might be the case that there is no activity simply because the project is not able to attract the interest of other developers. Therefore, the project is either completely abandoned or it is carried on by the original developer (or developers, if more than one) with

[6]For a detailed discussion of the limitations of data obtained by crawling OSS repositories, see Crowston and Howison (2004).

no contribution from nor interaction with the rest of the community. In both instances, such a project is meaningful for our scopes: an abandoned project is a failure in the sense that it does not make any progress in the development stage; as such, it is crucial to include this project in our estimations that are aimed at evaluating the determinants of the progress (resp. the failure) of OS projects.

The second possible explanation for the absence of any activity recorded on SF is that the project has its own web page; that is, the project is listed on SF but it is actually hosted somewhere else.[7] For this type of projects the information that we possess may not be the correct one.[8] Unfortunately there is no way of disentangling between unattractive/abandoned projects and those that are hosted elsewhere; nevertheless, including in our regressions the projects that have their own web sites should not distort the results unless the vector of characteristics of these projects is biased in some way.[9]

2.2 Sample statistics

The sample has been crawled from SF web-site in December 2004 and it is made of 88192 observations.

Table 1 groups projects according to their age (measured as the difference between the data collection date and project's registration date). The oldest project was registered 5.12 years before the data collection date, that is in November 1999. The average projects' age is 2.05 years. The table highlights a dramatic increase in the number of registrations since the third year of activity of SF, with more than 15000 projects with an age between 3 and

[7]Just to have an idea of the magnitude of the "list but not hosted" phenomenon, we have randomly extracted 100 projects from SF; we found that only 8% of these projects has a clearly "independent" homepage; that is, a homepage that is hosted outside SF and that provides files downloading and bugs reporting facilities.

[8]In these cases, the information about projects' activity (bugs, patches and feature requests) is certainly uncorrect while we ignore whether the information concerning the number of developers and the development stage is kept up-to-date or not. On the contrary, there are no reasons to believe that the remaining information is not correct since it refers to declarations made at the time of the registration and this should not change over time.

[9]For example, we may have a problem if the projects with their own external web site are concentrated in the category of, let's say, end-users. In this case the estimations about the impact of intended audience on projects' development stage would be uncorrect. Nevertheless there are no a-priori reasons to believe that this indeed occurs for any specific characteristic.

4 years, and with more than 21000 projects that are steadily registered during each of the following years.

Table 1: **Age of the projects**
(difference between data collection date and registration date)

	> 5 years	**4 <years< 5**	**3 <years< 4**	**2 <years< 3**	**1 <years< 2**	**< 1 year**
N. of projects	2584	5371	15609	21238	22102	21288
Percentage	2.9 %	6.1%	17.7 %	24.1%	25.1 %	24.1%

Table 2 provides the distribution of projects in terms of the number of active developers that have joined a certain project. Quite surprisingly, the development of a large majority of projects rests on just one developer and projects with at most two developers account for more than the 80% of the whole sample. This figures are strongly in contrast with the common perception of open source as a phenomenon involving large and complex communities and confirm the observation in Krishnamurthy (2002) who first noticed that the vast majority of open source software are projected, written and distributed within very restricted circles of developers. In the sample, projects with large communities, for instance more than 16 developers, account for less than 1% of the entire population. The largest community is made of 274 developers.

Table 2: **Number of developers per project**

	One	**Two**	**Three or four**	**Five or six**	**Between 7 and 15**	**More than 16**
Number of projects	57406	13451	8583	3046	2728	586
Percentage	66.9%	15.7%	10.0 %	3.5%	3.2 %	0.7%

Table 3 provides some statistics on projects distribution according to development status, intended audience, topic and the terms of licensing. The table largely confirms the main characteristics of the open source movement found in previous studies:[10]

1. a part from mature ones which account of only 1.6% of the entire sample, projects are quite uniformly distributed across the various development stages;

[10]See Lerner and Tirole (2005).

2. more than 40% of the projects are geared towards developers;

3. the most popular OSS topics are games, projects related to the Internet, projects aimed at software and systems development (kernels, hardware, networking), software for communications and multimedia;

4. highly restrictive licenses (mainly GPL) represents by and large the most popular licensing scheme among projects leaders.

Table 3: **Percentage distribution of some characteristics**

Development Status		**Intended Audience**		**Topic**		**License**	
planning	20.8 %	end-users	25.1 %	communications	7.7 %	highly restrictive	66.5 %
pre-alpha	18.4 %	developers	44.3 %	database	2.5 %	restrictive	14 %
alpha	17.5 %	system ad	22.9 %	desktop	1.5 %	unrestrictive	17.1 %
beta	21.9 %	others	7.6 %	education	1.5 %	others	2.4 %
production stable	19.7 %			games	10.9 %		
mature	1.6 %			Internet	11.6 %		
				multimedia	8.6 %		
				office	4.2 %		
				science	7.5 %		
				security	1.8 %		
				software	22.4 %		
				system	15.6 %		
				terminals	0.8 %		
				text editors	3.2 %		

The aim of our econometric exercise is to determine the empirical regularities between the development progress of each project and its various characteristics. Clearly, one may be concerned by the presence of correlation between projects' characteristics: for example, it is likely that most of the projects geared towards system administrators may be system or security tools. In this case, intended audience and topic would be colinear thus biasing the estimations.

Table 4 shows the cross-correlation between topic and intended audience. A part from software tools that, as expected, are mainly intended for developers (89.47 %), it does not emerge a clear correlation between intended audience and project's topic. In particular, communication, system and Internet tools projects are at various extent distributed all across the three categories of intended audiences; games and multimedia tools are generally more geared towards end users and developers.

Table 4: **Cross-tabulation between Topic and Intended Audience**

	Intended Audience			
	Developers	End Users	System Ad	Total
Topic				
Communication	29.25	40.25	30.50	100
Games	36.12	58.52	5.36	100
Internet	42.48	24.26	33.06	100
Multimedia	44.31	50.38	5.31	100
Software	89.47	1.30	9.23	100
System	32.58	12.61	54.81	100
Full sample	48.07	26.77	25.15	100

3 Conjectures about the determinants of projects' development stage

OSS is an extremely complex phenomenon which cannot be easily explained; nevertheless, by looking at the most recent debate we can figure out some possible conjectures about what might induce an advancement in the development stage of a project, that is, in our meaning, what might induce an open source project to succeed.

First, there is a general consensus on the fact that the licensing terms under which the software is released and distributed are of crucial importance to determine its success; therefore the first conjecture that we pose is the following:

Conjecture 1. *The licensing terms have a significant impact on the development stage of the project.*

Nevertheless, there is no consensus on the "expected sign" of this variable. As discussed by Lerner and Tirole (2005) and West (2003), a more restrictive license, namely a license that narrowly circumscribes the use of the product and its future developments (typically, the GPL which imposes both viral and copyleft provisions), is more likely to attract the OSS community, especially the more "purist" programmers. According to this observation the following conjecture follows:

Conjecture 2. *Projects under more restrictive terms are more likely to attract the vast part of the OSS community of developers and therefore to reach an advanced stage of development.*

On the other hand, less restrictive licenses, namely those that to various extent allow for mingling of the source code with other software (i.e. LGPL) or that do not impose copyleft (typically, the BSD license), are more likely to attract programmers that prefer to be free to use the software as they wish without too many restrictions, including those that consider a possible commercial exploitation. From these arguments an alternative conjecture derives:

Conjecture 3. *Projects under less restrictive terms are those more likely to attract developers and therefore to reach an advanced stage of development.*

Various studies have pointed out that the vast majority of successful open source software is geared towards sophisticated/high-end users while it is hard to find successful OSS for the mass market segments where adopters are mainly unsophisticated users.[11] According to this observation, we might expect that:

Conjecture 4. *Projects geared towards sophisticated users are more likely to reach an advanced stage of development.*

Since Raymond's seminal paper[12] vast part of the literature has focused on the role of the community in the production process of OSS; the underlying idea is that the larger

[11] See Raymond (1999), Berlecon (2002) and Comino and Manenti (2005). According to Lerner and Tirole (2002) this is one of the main challenges faced by the open source community: a broader adoption of OS packages within the unsophisticated clientele may result only from the collaboration of the OS community with for-profit firms aimed at the provision of more user-friendly packages as well as of support services, documentation and other user interfaces.

[12] See Raymond (1999).

the community of developers (the "Bazaar", in Raymond's terminology), the greater the likelihood of reaching a mature and stable product.[13] A closer look at the data, shows that the presence of a large community does not seem to be a necessary condition for projects' development. We have already mentioned the paper by Krishnamurthy (2002), where the author, using a sample of 100 projects hosted on SF, has shown that projects developed within a small community have greater chances to succeed. Our sample statistics seem to support Krishnamurthy's observation: more than a half of stable and mature projects have been developed by a unique individual. It seem therefore of interest to test the following conjecture:

Conjecture 5. *The larger the community of developers the more likely that a project reaches an advanced stage of development.*

As briefly discussed in the introduction, one of the characteristics of the OSS movement is that it is continuously changing its nature in many respects. On the one hand, as West (2003) and Bonaccorsi and Rossi (2003a) among others have pointed out, the very basic nature of the OSS phenomenon based on altruistic or non-for-profit motivations to contribute, seems to be under pressure due to an ever more substantial presence of commercial actors on the open source stage. On a different theme, Raymond (1998) observes that both the content and the audiences of software projects are changing over time. Therefore, we might expect that:

Conjecture 6. *The determinants of development stage of older projects differ significantly from those of newer projects.*

4 Econometric framework

We use an ordered response model to represent our latent dependent variable which is the development stage of each project at the time of data collection. What we observe is development status self-declared by each project administrator. SF projects are classified into six discrete and successive categories: 1-planning, 2-pre-alpha, 3-alpha, 4-beta, 5-production stable and 6-mature.

[13] This idea is neatly summarized by the so-called "Linus' Law" according to which "given enough eyeballs, all bugs are shallow" (Raymond, 1999).

Let us indicate with S_i the unobserved development status of project i. This latent variable gets mapped into ordered multinomial variable as follows:

$$\begin{aligned} Y_i &= 1 \quad \text{if} \quad \alpha_0 < S_i \leq \alpha_1 \\ Y_i &= j \quad \text{if} \quad \alpha_{j-1} < S_i \leq \alpha_j \quad \text{for } j = 2, \ldots, 5 \\ Y_i &= 6 \quad \text{if} \quad \alpha_5 < S_i \leq \alpha_6 \end{aligned}$$

where α_0 to α_6 are unobserved thresholds. The regression equation can be written as

$$Y_i = \beta_1 AGE_i + \beta_2 DEV_i + \beta_3 ACTI_i + \beta_4 LIC_i + \beta_5 LIC_i * NEW_i + \gamma' \underline{D}_i + \delta' \underline{D}_i * NEW_i + \varepsilon_i \quad (1)$$

$$\varepsilon_i \sim iid\, N(0,1)$$

where:

- AGE_i is the age of the project measured as the difference between the data collection date and the project registration date;
- DEV_i is number of developers that are contributing to project i;
- $ACTI_i$ is activity index of project i defined as the sum of the bugs, feature requests and patches released.[14]
- LIC_i measures the level of restrictiveness of the licensing scheme of project i; to define this variable we have used the classification proposed in Lerner and Tirole (2005) and in Välimäki (2005) where licenses are classified into three categories according to the degree of restrictions they impose on the use of the software: highly restrictive licenses (with viral and copyleft restrictions, e.g. GPL), restrictive (only copyleft, e.g. LGPL) and unrestrictive licenses (e.g. BSD). This variable takes the value of 1 when the license is highly restrictive, 2 when it is restrictive and 3 when unrestricted.

and where $\underline{D}_i$ is a set of qualitative variables associated to project i to control for the following characteristics: intended audience, topic, programming language and operating system.

Finally, note that in order to account for the possible evolving nature of open source and to test for Conjecture 6 we have included among the regressors interactions of LIC_i and $\underline{D}_i$

[14] Our activity index differs from the index reported on SF: the measure that we employ is based on projects' total activity since the registration date and not, as in SF, on the last week amount of projects' activity.

with a dummy indicating whether the project is less than 2 years old, called NEW_i. This allows to perform Wald tests for the stability of the β_4 and γ coefficients.[15]

As we have already pointed out above, in 2002, SF has added seven new categories to the list of intended audience. This change in the classification system may pose a problem when estimating the impact of intended audience on projects' development status; problems may be particularly severe with respect to the interaction terms for newer projects.

In order to reconcile the new with the old classification, we should have been able to assign projects that now fall in one of the new categories to one of the old categories.[16] This is easy for those projects that belong both to one of the new as well as to one of the original categories: in this case we have simply assigned these projects to the latter (old) category. On the contrary, those projects that fall only in one or more of the new categories of intended audience have been dropped from the sample since there is no clear way to associate these observations to any of the old categories of intended audience.[17]

Given the nature of the dependent variable, we estimate the parameters through a (maximum-likelihood) ordered probit analysis. This is by now a standard maximization problem, for which Maddala (1983) derived the first order conditions. We calculate a set of tests including goodness of fit pseudo-R^2, log-likelihood ratio tests for subsets of parameters, and Wald tests for parameter stability.

[15]The choice of the threshold level for the time interaction is somehow arbitrary; we have chosen 2 years as the closest integer number to the average project's age which is 2.05.

[16]Note that the old categories of intended audience, that are end users, developers or system administrators, are also the most relevant ones since they account for roughly 90% of the whole sample.

[17]This decision has been taken after a detailed look at the data: by focusing on those projects falling both into one of the three original categories as well as in one of the new categories, we have tried to figure out any possible meaningful pattern. For instance, one might expect that projects directed to, let's say, customers may also fall into the category of end users. This would suggest a "rule" to classify all the projects intended for customers as projects intended indirectly for end users, thus reconciling the new with the old classification. Unfortunately, the data do not seem to support any clear association between the new and the original categories. For this reason we have preferred to drop the observations relative to the projects that were classified only into the new categories rather then arbitrarily assigning them to one of the original categories of intended audience.

4.1 Estimation results

Out of the 88192 observations of the whole sample, only 40512 have the full set of information required to estimate equation (1). Table 5 presents the results of our econometric model; in order to take into account of the concerns about the quality of the data, we have estimated two separate regressions: the first column shows the results obtained considering the entire sample while in the second column we have restricted the estimation by considering only those projects with at least two developers (15338 observations). The general presumption for including this second estimation is that the information on projects involving a minimal community of developers should be more reliable and accurate than for individually run projects.

Table 5: **The Determinants of Development Status**

Dependent: **Development Status**				
	full sample		restr. sample developers>1	
Age	.113	(.0074)***	.184	(.0114)***
Num. developers	.015	(.0073)**	.012	(.0043)***
Activity index	.001	(.0007)	.001	(.0002)**
License				
Licence	.019	(.0093)**	.039	(.0142)***
Licence_{new}	.030	(.0123)**	.033	(.0197)*
Intended Audience				
End users	-.017	(.0161)	-.019	(.0250)
End users_{new}	.089	(.0223)***	.078	(.0368)**
Developers	-.079	(.0163)***	-.034	(.0255)
Developers_{new}	.145	(.0237)***	.176	(.0176)***
System administrator	.112	(.0190)***	.111	(.0297)***
$\text{System administrator}_{new}$	.038	(.0283)	.075	(.0476)
Topic				
Communication	-.073	(.0210)***	-.057	(.0312)*
$\text{Communication}_{new}$	.054	(.0303)*	.024	(.0486)

Continued on next page...

... Table 5 continued

Games	-.245	(.0226)***	-.332	(.0337)***
Games_{new}	-.009	(.0320)	.041	(.495)
Internet	.097	(.0185)***	.102	(.0290)***
Internet_{new}	-.004	(.0261)	-.006	(.0435)
Multimedia	.158	(.0226)***	.147	(.0344)***
Multimedia_{new}	-.027	(.0318)	.019	(.0521)
Software	.220	(.0218)***	.229	(.0331)***
Software_{new}	-.092	(.0310)***	-.128	(.0511)**
System management	.026	(.0192)	.006	(.0300)
$\text{System management}_{new}$	-.028	(.0277)	.094	(.0461)**
α_0	-.800	(.0317)	-.539	(.0491)
α_1	-.215	(.0312)	.064	(.0486)
α_2	.257	(.0308)	.532	(.0485)
α_3	.923	(.0304)	1.197	(.0487)
α_4	2.324	(.0328)	2.651	(.0546)
pseudo-R^2	.0393		.0382	
Wald test (Licence)	5.95	[.014]	2.92	[.087]
Wald test (Intended Audience)	46.32	[.000]	23.8	[.000]
Wald test (Topic)	30.54	[.006]	26.25	[.024]
Num of obs.	40512		15338	

Regressions are conditioned also on programming language and operating system. Robust standard errors are in parentheses; *** 1%, ** 5% and * 10% refer to the significance levels for parameters' estimates. α_0, α_1, α_2, α_3 and α_4 are cutoff points. Wald tests are for the stability of the parameters between NEW and OLD projects; Prob $> \chi^2$ in square brackets.

From this table a series of interesting observations follows. First of all, it appears that there are no strong and significant differences between the full sample estimation and the restricted one; the signs of the various coefficients remain mostly unchanged and the significance of the parameters looks very similar.[18] This confirms the robustness of the full-sample

[18]The two most relevant differences between the two estimations is a change in the significance of the coefficients for the category "developers" of intended audience and the interaction term of the topic category "system management".

estimation and for this reason we comment on this.

1. We performed Wald tests for parameters stability across time. Quite interestingly the set dummies for intended audience, the license regressor and the control dummy for software topic show significant differences between old and new projects in their impact on development status. These tests substantially confirm Conjecture 6 and show that there has been a change in the nature of open source in the period 1999-2004. Obviously, some caution is needed when interpreting the result for the intended audience: the way that we have employed to neutralize the effect of the change in the SF classification system occurred in 2002, might not be sophisticated enough so that part of the change through time in the impact of these variables on the progress of the projects might be driven by the modification in the classification system.

2. Licence is positive and significant; this first confirms that the licensing terms are a key driver for the progress of a software project (Conjecture 1). The positive sign of the coefficient provides a support to Conjecture 3 while it rejects Conjecture 2: the less restrictive the licensing terms the larger the likelihood of reaching an advanced development status. Although obtained for a small subset of open source projects, a similar result is in Fershtman and Gandal (2004) where it is shown that output per-contributor is significantly higher under less restrictive licenses. Lerner and Tirole (2005) also find a negative correlation between the level of projects' activity and the restrictiveness of the licensing terms. We obtain our result considering the whole sample of projets hosted on SF thus reinforcing similar findings of previous authors.

3. The positive effect of a lower degree of restrictiveness is even stronger for newer projects. This is one of the more interesting results of our investigation; a possible explanation may accrue to the "commercialization" process of OSS: as highlighted by many authors (West, 2003; Bonaccorsi and Rossi, 2003b), for-profit actors are becoming more involved in the production of OSS platforms. A developer which is looking at an OS project as a source of possible revenues clearly prefers not to restrict too much the use and the exploitation of future developments of the project. Our finding may be a first signal of this recent evolution of open source.

4. Projects geared towards system administrators appear to be the more successful ones. This category of intended audience is usually made of highly sophisticated users, there-

fore this result seems to provide empirical support to Conjecture 4. Being geared towards end users and developers does not seem to have similar positive impact on the progress of a project; nevertheless things change when considering newer projects: in this case software directed to these two categories of final users have a larger likelihood of reaching stability of production. This further confirms Conjecture 6.

5. Conjecture 5 is supported by the data: the size of the community, here proxied with the number of developers, has a positive and significant coefficient. On the contrary, activity does not seem to play a large role in the advancement of the development status.

6. Consider now the topic category: while Internet, multimedia (e.g. audio/video software and graphics) and software (e.g. interpreters and compilers) tools have positive and significant parameters, projects aimed at developing games and, quite surprisingly, communication systems (e.g. chat and file sharing) have negative and significant coefficients. For this latter category, it is interesting to note that for newer projects the overall impact on the likelihood of reaching an advanced status of development becomes positive and significant. Again these may be considered as confirmations of the evolving nature of open source (Conjecture 6).

7. Finally, as expected, the coefficient of the variable AGE is positive and significant: the older the project the more advanced its development stage.

5 Conclusions

The existing empirical literature on open source software has focused almost exclusively on the analysis of "case studies". This kind of research allows for a very in-depth understanding of a certain project but it is very limited when trying to derive more general conclusions. This seems to be particularly relevant in the case of open source: the interest in studying how individuals interact and coordinate in distributed teams of programmers has biased previous works conducted in this field towards the analysis of very large and successful projects such as Linux, Apache and Freenet. However, even though these projects certainly represent the most challenging and interesting ones to be studied, they constitute only a minor part of the vast and complex open source world which, on the contrary, flourishes mainly within very

small circles of few developers: the data shows that two projects out of three are run by only one developer.

This paper is a first attempt to fill this gap: by employing a very large dataset crawled from SourceForge.net, we have investigated the main determinants of the progress in the development stage of open source projects; the question that we have tried to address is: "what characteristics/variables are more likely to affect the ability of an open source project to reach a stable and mature development stage of production?" In trying to answer this question we have tested some of the theoretical predictions that have been put forward in the literature on OSS.

Our analysis shows that the less restrictive the licensing terms of a project the larger the likelihood of reaching an advanced development status; this effect becomes even stronger for more recent projects. Moreover, we have found that projects geared towards sophisticated users (i.e. system administrators) have greater chances to succeed in term of making progress in the development stage. We have also found that the determinants of projects' developments stage change with the age of the project in many dimensions, i.e. licensing terms, software audience and content, thus supporting the common perception that open source is an evolving phenomenon.

References

Berlecon (2002). Free/Libre and Open Source Software (FLOSS): Survey and Study. International Institute of Infonomics, University of Maastricht and Berlecon Research.

Bonaccorsi, A. and Rossi, C. (2003a). Contributing to the common pool in open source software. a comparison between individuals and firms. Sant'Anna School of Advanced Studies, IIT Working Paper.

Bonaccorsi, A. and Rossi, C. (2003b). Why open source software can succeed. *Reserch Policy*, 32:1243–1258.

Comino, S. and Manenti, F. M. (2005). Government policies supporting open source software for the mass market. *Review of Industrial Organization*, 26(2):217–240.

Crowston, K., Annabi, H., and Howison, J. (2003). Defining Open Source Software Project Success. *Proceeding of International Conference on Information Systems*.

Crowston, K. and Howison, J. (2004). The Perils and Pitfalls of Mining SourceForge. Proc. of Workshop on Mining Software Repositories at the International Conference on Software Engineering ICSE.

Fershtman, C. and Gandal, N. (2004). The Determinants of Output per Contributor in Open Source Projects: An Empirical Examination. *CEPR Discussion Paper*, 4329.

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

Krishnamurthy, S. (2002). Cave or Community? An Empirical Examination of 100 Mature Open Source Projects. *First Monday*, 7.

Krogh, G. V., Spaeth, S., and Lakhani, K. R. (2003). Community, joining, and specialization in open source software innovation: a case study. *Reserch Policy*, 32:1217–1241.

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

Lerner, J. and Tirole, J. (2002). Some Simple Economics of Open Source. *Journal of Industrial Economics*, 52.

Lerner, J. and Tirole, J. (2005). The Scope of Open Source Licensing. *Journal of Law, Economics & Organization*, 21(1).

Raymond, E. (1998). Homesteading the Noosphere. *First Monday*, 3.

Raymond, E. (1999). *The Cathedral and the Bazaar: Musings on Linux and Open Source from an Accidental Revolutionary*. Sebastapol, CA: O'Reilly and Associates.

Välimäki, M. (2005). *The Rise of Open Source Licensing. A Challenge to the Use of Intellectual Property in the Software Industry*. Turre Publishing, Helsinki.

Weiss, D. (2005). A large crawl and quantitative analysis of open source projects hosted on sourceforge. Research report ra-001/05, Institute of Computing Science, Poznań University of Technology, Poland.

West, J. (2003). How open is open enough? melding proprietary and open source platform strategies. *Reserch Policy*, 32:1259–1285.