UNIVERSITÀ DEGLI STUDI DI PADOVA

Dipartimento di Scienze Economiche "Marco Fanno"

FROM PLANNING TO MATURE: ON THE DETERMINANTS OF OPEN SOURCE TAKE OFF

STEFANO COMINO
Università di Trento

FABIO M. MANENTI
Università di Padova

MARIA LAURA PARISI
Università di Brescia

January 2007

"MARCO FANNO" WORKING PAPER N.35

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

Stefano Comino[*] Fabio M. Manenti[†] Maria Laura Parisi[‡]

January 2007

Abstract

Open source is an example of *user-centric innovation* initiated by an individual or group of users to satisfy their specific needs; the more a software evolves towards a stable release able to address the requirements of its developers, the more successful the project. In this paper we use a large data-set obtained from SourceForge.net to estimate the impact of observed project characteristics on the evolution of the source code from a preliminary release to its mature version. We show that while projects distributed under highly restrictive licensing terms (GPL) have a significantly smaller probability of reaching a stable release, applications towards sophisticated users have a larger probability of evolving in the development status. Interestingly, we find that the size of the "community of developers" increases the chances of progress but this effect decreases as the community gets larger, a signal of possible coordination problems. Finally, we show that the determinants of the development stage of older projects differ significantly from those of newer projects, thus supporting the common perception of open source as an extremely dynamic phenomenon.

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

JEL classification: O38, L51, L63.

[*]Dipartimento di Scienze Economiche, Università di Trento, Via Inama 5, 38100 TRENTO (Italy), Tel. (39) 0461 882221, email: stefano.comino@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, Fax. (39) 049 8274211, 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

In recent years, the traditional paradigm of software innovation based on intellectual property rights has been challenged by the emergence of the Open Source (OS hereafter) phenomenon. The effectiveness and the general appeal of this alternative mode of software production have attracted considerable attention. An increasing number of studies and public debates have been devoted to understanding its underlying structure and to assessing the potential benefits of a more widespread diffusion of open source.

According to von Hippel (2005), OS is an example of *user-centric innovation* where an individual or group of users initiate a project because they expect to benefit from *using* the software they develop rather than from *selling* it. From an economic point of view, OS software is a *privately provided public good*; it is the result of private efforts of developers who devote their time to writing the source code and it is non-excludable and non-rival.[1] Non-excludability is guaranteed by OS licences which ensure that the source code is kept in the public domain; OS licenses are so crucial that a software is defined as open source only when it is distributed under a licensing scheme that satisfies the criteria set by the Open Source Initiative (see the OSI web site, `www.opensource.org`). Furthermore, being an information good, software is non-rival since it does not depreciate with the number of users who download it.

In this paper, we conduct an empirical analysis aimed at evaluating the impact of a series of different characteristics of an OS project on its probability of success. Our analysis differs from the existing studies on the same topic in two main respects. First, we define project's success in terms of the evolution of the source code released to the community: a project is "fully" successful when its code reaches a stable or mature release. Our measure seems natural given the user-centric characteristic of OS projects: the more a software project evolves towards its final release, the better it addresses the specific needs of its developers. Second, we employ an extremely large data-set extracted from SourceForge.net (SF hereafter), the most extensive available repository of OS projects. In this way, we are able to exploit a very comprehensive set of information on the OS phenomenon, thus notably reducing the risk of a sample selection bias.

We find that the licensing terms under which a code is released, together with the project's

[1] For comprehensive surveys of the literature on the economic issues related to OS software see Lerner and Tirole (2002), Bonaccorsi and Rossi (2003b) and Rossi (2004).

characteristics such as the age of the project, the audience the software is intended for or the content of the software, are key drivers for project evolution. More specifically, we show that a project distributed under a licensing scheme that places severe limitations on derivative works of the software has a significantly smaller probability of evolving towards a stable release. We show that applications for sophisticated users have greater chances of reaching an advanced development stage, while applications for less sophisticated users are less likely to evolve towards a mature release. Interestingly, the size of the "community of developers" has a positive but non-linear effect on the evolution of software code. Finally, we find that the impact of projects characteristics on the likelihood of success differs substantially between older and newer projects. In particular, we find that for newer projects the type of license seems to play a less relevant role than for older ones.

The paper is organized as follows: in Section 2, we discuss how the issue of OS software success has been studied in the literature and how our contribution adds to this literature; we formulate various testable hypotheses on the determinants of the evolution of OS software code. Section 3 presents the data and the methodology used for the estimations while in Section 4 we provide and discuss the results of the estimation. In Section 5 we check the robustness of our results and Section 6 concludes.

2 Theory and hypotheses

2.1 What is OS success? A survey of the literature

Different measures to define the success of open source projects have been proposed in the literature. Moving from the more recent contributions, these can be classified into three categories: *i*) software use, *ii*) size of the community and/or its level of activity and *iii*) technical achievements of the project.

According to the first two categories, an OS software is successful when it is widely adopted among users or when a large and active community of developers contributes to its production. In Fershtman and Gandal (2004) software success is measured in terms of output per contributor, that is the average number of source code lines written by each contributor. The analysis is based on a sample of 71 open source projects observed during a period of eighteen months, and it shows that output per contributor is significantly higher when the software is distributed under non-restrictive licensing terms. A similar result is obtained by Stewart et al. (2005) in a study on 138 open source projects registered on Freshmeat. They measure project success based on the number of subscribers associated with a project, and find that this number is larger when the software is released under non-restrictive schemes and sponsored by non-market organizations (e.g. Universities).

The third category of software success includes several possible indicators. Crowston et al. (2003) consider a software successful when its source code is of high quality, e.g. it is highly modular, correct and maintainable.[2] Grewal et al. (2006) focus on the commits to the Concurrent Versioning System (CVS), that is, they measure the technical achievements in terms of the number of new releases of the code that the manager uploads to the CVS. The authors study the relationship between organizational issues and software success; they focus on how projects are, to various extents, interconnected, the so-called "network embeddedness"; based on a sample of 108 projects, they find that network embeddedness positively affects software success.

2.2 Working definition of OS success and hypotheses

Open source is a typical example of user-driven innovation, usually initiated by an individual or a group of users looking for new or additional software functionalities. After a planning phase, the development of a software proceeds through several stages: from the release of intermediate beta-versions up to the latest completion of a mature software. Since the project originates from a specific need of an individual or group of developers, its evolution towards a final/stable version can be considered a natural measure of success, i.e. when the needs of the initiators are effectively satisfied. Therefore, we measure project success in terms of the development stage that it has reached. Following the above classification, this definition of success belongs to the third category of "technical achievements of the project" and it is in line with the one used in Grewal et al. (2006).

Various characteristics of an OS project are relevant in determining the likelihood of its success; in what follows, we present a series of testable hypotheses that can be figured out from the most recent debate on OS.

Licences are recognized as one of the fundamentals of open source; currently, according to the

[2]It should be noted that quality in software is a very controversial matter. Software quality is measured in terms of conformance to certain requirements; the set of requirements is very wide and it may go from mere aesthetic issues, such as the elegance of the code, to more practical evaluations such as readability, fault-tolerance and/or absence of bugs.

OSI definition, more than 40 different schemes are defined as open source licences. OS licences are usually classified according to the restrictions that they impose on derivative works (Bonaccorsi and Rossi, 2003a; Rosen, 2001; Lerner and Tirole, 2005). The most restrictive licence is the General Public License (GPL); according to this scheme, a software derived from other GPLed software or simply including lines of code released under GPL, must be distributed under the same licensing terms (inheritance or copyleft property). The GPL is by and large the most popular OS software licensing scheme; nevertheless, the strong restrictions it imposes have induced the OS community to develop other, less restrictive, licences. The Lesser GPL (LGPL) and the Berkeley Software Distribution (BSD) licences are the two most popular alternatives to the GPL. The LGPL has mainly the same characteristics as the GPL apart from the fact that it allows for the use of libraries developed under LGPL in other software programs, without modifying the licence terms of these latter. The BSD licence is even less restrictive since it allows the licensee to modify the source code and even to create proprietary versions of the software (non-copyleft licence).

In principle, the degree of restrictiveness may have either positive or negative effects on the success of a project. As discussed by Lerner and Tirole (2005) and West (2003), a more restrictive license might be more likely to attract the OS community, especially the more "idealistic" programmers. On the other hand, less restrictive licenses potentially increase the monetary rewards that can be obtained, thus attracting profit-motivated developers. Previous empirical studies suggest that the negative effect of restrictiveness on project technical achievements dominates (Lerner and Tirole, 2005; Fershtman and Gandal, 2004; Stewart et al., 2005); therefore, we test for the following hypothesis:

Hypothesis 1. *Projects distributed under highly restrictive terms are less likely to reach an advanced stage of development.*

Various studies have pointed out that the vast majority of successful open source software are applications for sophisticated/high-end users while it is hard to find successful OS applications for less sophisticated users (Raymond, 1999; Berlecon, 2002; Comino and Manenti, 2005); notably, Healy and Schussman (2003) show that while end user applications are the most downloaded OS software, developers' efforts mainly focus on "behind the scenes system applications, programming environments or utilities providing basic functionalities to an operating system". According to this observation, we might expect that:

Hypothesis 2. *Applications for sophisticated users are more likely to reach an advanced stage of development.*

Since Raymond's seminal paper "The Cathedral and the Bazaar", a large part of the literature has focused on the role of the community in the production process of OS software; the underlying idea is that the larger the community of developers, the greater the likelihood that the project will succeed. However, a closer look at the data shows that the presence of a large community is not a necessary condition for project development. Healy and Schussman (2003) note that the distribution of projects according to the number of developers is highly skewed, with a median number of one. Similar findings are also in Gosh and Prakash (2000) and in Krishnamurthy (2002). It seems therefore of interest to test for the following hypothesis:

Hypothesis 3. *The larger the community of developers, the more likely it is that a project will reach an advanced stage of development.*

Another interesting characteristic of the open source movement that has been frequently pointed out refers to its evolving nature. Although it is extremely difficult to figure out a unique pattern in the evolution of the OS phenomenon, various influential authors have suggested some possible evolving paths. For instance, Raymond (1998) observes that the historic evolution of OS software has come in different waves. During the seventies, most of the time and effort of programmers were devoted to developing games and demos; in the eighties it was the turn of Internet tools while in the nineties the interest shifted towards operating systems. The prediction for the future is that efforts will be concentrated on "the last virgin territory" i.e. the development of programs/applications for non-techies.

The ever changing nature of open source 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 programming new artifacts, nowadays the for-profit, commercial world is heavily involved in the open source arena. According to Hecker (1999), many commercial firms have entered the software industry using OS models of production and profiting from related products; on the same line, West (2003) and Bonaccorsi and Rossi (2003b) have pointed out that the very basic nature of the OS phenomenon based on altruistic motivations or motivations other than profit seems to be under pressure due to an ever more substantial presence of commercial actors on the open source stage.[3] It seems therefore apparent that since its origin, the OS phenomenon has experienced dramatic changes in its nature and characteristics, hence we might expect that even in a shorter/medium run:

Hypothesis 4. *The determinants of the development stage of older projects differ significantly from those of newer projects.*

3 Data and Methodology

The dataset we employ in our analysis consists of all OS projects that were hosted on SourceForge.net in December 2004.[4] SourceForge.net is the largest existing online platform providing OS developers with useful tools to control and manage software development. Project administrators register their software project on SF and provide the required information which is then available on-line. Registration is for free and administrators are encouraged to register their projects as well as to maintain all the relevant information up-to-date.

For each registered project, SF provides the following information:

- *development status*: information regarding the current development stage reached by the project. Each project can be classified into one 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 was registered on SF;

- *number of developers* that are currently contributing to the development of the project;

- *number of bugs, patches and feature requests*:[5] developers and users may submit bug reports, feature requests and source code patches; for each project on SF, a dedicated tracker system

[3]For a recent analysis of the strategies adopted by profit-oriented firms in the OS world, see Bonaccorsi et al. (2006).

[4]We employ the dataset crawled by Dawid Weiss; data are publicly available on www.cs.put.poznan.pl/dweiss/ and Weiss (2005) provides full documentation of the crawling system.

[5]The feature request tracker allows the 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.

automatically updates these data so that information about the total number of bugs, patches and feature requests submitted since the inception of the project is available;

- *license*: the licensing terms under which the code is released and distributed;

- *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 into 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 management, terminals and text editors;

- *programming language* used to write the program and the *operating system* that supports it (i.e. the operating system which is compatible with the software).

Since its beginning in 1999, SF requests each project administrator to provide all this information about her/his software; nevertheless, unlike development status and topic for which classification appears to be stable through time, SF has slightly modified the classification adopted to describe the project's intended audience. Up to 2002, SF grouped projects into only three categories (apart from the residual one, called "others"): end users, 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 3.2 we will discuss how we have dealt with this change in the classification of intended audience.

The overall sample is composed of 88192 projects. In Figure 1 we group projects according to their age (measured as the difference between the data collection date and the project registration date). The oldest project was registered in November 1999, 5.12 years before the data collection date; the average project age is 2.05 years. The figure highlights a dramatic increase in the number of registrations since the third year of activity of SF, with 15609 projects with an age between 3 and 4 years, and with more than 21000 projects steadily registered during each of the following years.

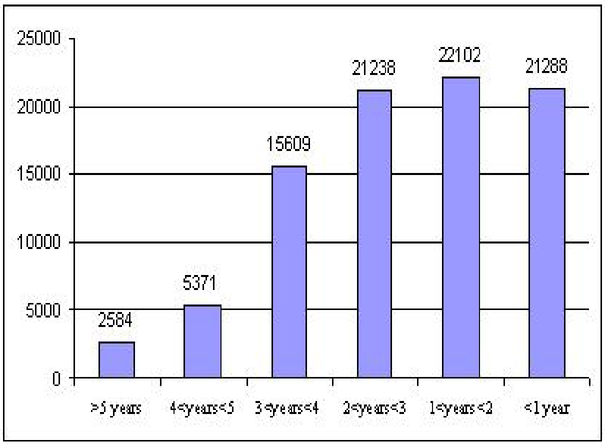

Fig 1: Age of the projects

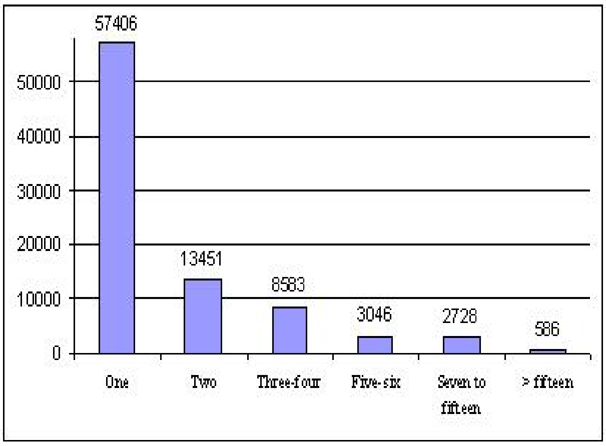

Fig 2: Number of developers per project

Figure 2 shows that the distribution of projects in terms of the number of developers is highly skewed. Projects with just one developer account for 66.9% of the whole sample, and more than 80% of the projects have at most two developers. These figures confirm the findings of Gosh and Prakash (2000), Krishnamurthy (2002) and Healy and Schussman (2003) who have already noticed that the vast majority of open source software is designed and written within very restricted circles of developers. In our 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 1 provides project distribution percentages according to development status, intended audience, topic and licensing terms. The table largely confirms the main characteristics of the open source movement found in previous studies (Lerner and Tirole, 2005):

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

2. projects geared towards developers represent more than 40% of the whole sample and this is the most common intended audience;

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

4. highly restrictive licenses (i.e. GPL) represent by and large the most popular licensing schemes among project leaders.[6]

Table 1: **Percentage distribution of projects 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 %		
				sw tools	22.4 %		
				system	15.6 %		
				terminals	0.8 %		
				text editors	3.2 %		

The next two tables show percentage conditional distributions of categories of intended audience. Table 2 shows the percentage distributions of intended audience conditional to the most common categories of software topics. Apart from software tools that are extremely skewed towards developers (89.47%), such skewness is not so prominent for the rest of conditioning topics. In particular, communication, system and Internet tool projects give to various extents rather uniform distributions to the three categories of intended audiences; as far as games, multimedia tools and science is concerned, end users and developers are the mode.

Table 3 analogously shows that while intended audience is rather uniformly distributed conditional to highly restrictive licences (mainly the GPL), categories distributed under less restrictive licences are quite skewed towards developers.

[6]Following Lerner and Tirole (2005), in Table 1 licenses have been classified into highly restrictive licenses (GPL like), restrictive (LGPL like) and unrestrictive licenses (BSD like).

Table 2: **Conditional to Topic distributions of Intended Audience**

	Intended Audience			
	Developers	End Users	System Ad	Total
Topic				
Communication	29.7	40.0	30.3	100
Games	36.3	58.0	5.7	100
Internet	42.4	23.9	33.7	100
Multimedia	44.4	50.3	5.3	100
Sw tools	89.1	1.4	9.5	100
System	32.8	12.6	54.6	100
Office	34.9	47.6	17.5	100
Science	65.7	29.6	4.7	100
Others	42.9	36.0	21.1	100

3.1 Caveat: the quality of the data

The dataset we employ is extremely large and contains detailed information; nevertheless it presents some potential shortcomings that must be discussed before proceeding with the estimation (Crowston and Howison, 2004). 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 project development status, these declarations depend on the subjective evaluation/perception of the project leader. Even though we are not able to verify the quality of this information, we believe that project leaders do not have incentives to misrepresent them or not keep them up-to-date. Project leaders wish to persuade other developers to contribute to the software code; the strategy of making the project appear to be something different from what it is might attract the interest of developers initially but it is unlikely to induce them to contribute: as soon as they get involved, developers will inevitably discover the project's real conditions (Lerner and Tirole, 2005). SF itself in various documentation encourages leaders and administrators to maintain a correct behavior.

Second, 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 which the project leaders do not bother to cancel from SF). A closer look at our dataset seems to confirm this observation: around

Table 3: **Conditional to License distributions of Intended Audience**

	Intended Audience			
	Developers	End Users	System Ad	Total
Highly restrictive	38.9	33.6	27.5	100
Restrictive	74.7	11.7	17.6	100
Unrestrictive	57.5	17.7	24.6	100
Others	58.0	17.2	24.8	100

80% of the projects do not show any interaction within the community of developers, having recorded no bugs, patches or feature requests since their registration. In our view, the absence of any activity has two possible explanations. First, it may be that 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 no contribution from or interaction with the rest of the community. In both instances, such a project is meaningful for our purposes; in particular, an abandoned project is a failure in the sense that it does not make any progress in the development stage and, as such, it is crucial to include it in our empirical exercise.

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 actually hosted somewhere else.[7] For this type of project the information we have gathered on the stage of development may not be the actual one. Unfortunately there is no way of disentangling unattractive/abandoned projects from those that are hosted elsewhere; nevertheless, the projects that have their own web site should not distort the estimates unless the vector of characteristics of these projects is biased in some way.[8]

[7] To have an idea of the magnitude of the "listed but not hosted" phenomenon, we randomly extracted 100 projects from SF; we found that only 8% of these projects has its own homepage outside SF that provides file downloading and bug reporting facilities. However, in most cases there is a clear link between SF and the outside homepage, suggesting that the information available on SF is maintained up-to-date.

[8] 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 our estimations about the impact of intended audience on project development stage would be incorrect. Nevertheless there are no a-priori reasons to believe that this indeed occurs for any specific characteristic.

In section 5 we discuss a series of control estimations to check the robustness of our main results.

3.2 Econometric specification

We measure project success in terms of the development of its software code from a preliminary to a mature release. The econometric model is built around the following latent regression:

$$Y_i^* = \beta' \underline{x}_i + \gamma' \underline{D}_i + \delta' \underline{D}_i \, NEW_i + \varepsilon_i \tag{1}$$

$$\varepsilon_i \sim iid \, N(0, \sigma^2)$$

where Y_i^* is the unobserved continuous degree of development reached by project i, and $\underline{x}_i$ is a vector including the following project characteristics:

- AGE_i is the age of the project measured as the difference between the data collection date and the project registration date;
- DEV_i and DEV_i^2 are the number of developers contributing to project i and its square (controlling for potential non-linearities);

$\underline{D}_i$ is a set of dummy variables such as:

- $HIREST_i$ takes value 1 when project i is distributed under highly restrictive licensing terms, that is under the GPL, and it is 0 otherwise;[9]
- the audience for which the software is intended (see Table 1);
- the software topic (see Table 1);
- the programming language used to write the software (C++, C, Java, PHP, PERL, Python, Javascript, C#, Visual Basic, Delphi, Unix Shell and Assembly);

[9] We treat the licensing scheme as an exogenous variable that is chosen at the beginning of the project. Even though changing the licensing terms in due course is in principle possible, in practice, to the best of our knowledge, this seems to be a very unfrequent event. In fact, as discussed in Lerner and Tirole (2005) the decision to switch the license type is extremely complicated, and it also raises legal concerns.

- the operating system compatible with the software (Posix, MS Windows, OS independent, PDA and Apple-MAC.[10])

Moreover, we interact the vector $\underline{D}_i$ with a dummy variable NEW_i that takes value 1 when project i is less than two years old.[11] The test on the δ' coefficients allow us to check whether the impact of the dummies $\underline{D}_i$ on project success differs in a significant way between older and newer projects. Note also that for newer projects the coefficients of the dummies are given by $\gamma' + \delta'$.

The observed dependent variable in our data is the development stage of the project classified into six discrete and successive categories: 1-planning, 2-pre-alpha, 3-alpha, 4-beta, 5-production stable and 6-mature.[12] Therefore, what we observe is Y_i such that:

$$
\begin{aligned}
Y_i &= 1 \quad \text{if} \quad Y_i^* \leq \alpha_0 \\
Y_i &= j \quad \text{if} \quad \alpha_{j-1} < Y_i^* \leq \alpha_j \quad \text{for}\, j = 1, \ldots, 4 \\
Y_i &= 6 \quad \text{if} \quad Y_i^* > \alpha_4
\end{aligned}
$$

where α_0 to α_4 are unobserved thresholds delimiting the steps needed by a project to jump to the next degree of development. They are estimated together with β', γ', and δ'.

Given the nature of the dependent variable, discrete and ordinal, we estimate a univariate ordered probit model. This is by now a standard maximization problem, for which Maddala (1983) derived the first order conditions.

As pointed out above, in 2002 SF added seven new categories to the list of intended audiences. This change in the classification may pose a problem when estimating the impact of intended audience on project development status. In order to reconcile the new with the old classification,

[10]The Posix category groups: Linux, UCLinux, all the BSD platforms (i.e. FreeBSD/NetBSD/OpenBSD), Solaris HP-UX, BeOS, SGI IRIX, SCO and IBM AIX. The MS Windows category includes: MS-DOS, MS Windows 3.X, 32-bit MS Windows (i.e. 95/98/NT/2000/XP), WinXP, Win2K, WinME, Cigwin, MS Windows Server 2003, WinE and MinGW/MSYS. The PDA category groups: SymbianOS, WinCE and Palm OS.

[11]We chose 2 years as the closest integer number to the average project age which is 2.05. Note that qualitatively identical results are obtained by setting this threshold level at values in the neighborhood of 2. The results are available upon request from the authors.

[12]Note that in only a few cases administrators classified their projects into more than one development status category. As a rule of thumb, in the estimations we used only those projects that have been classified into at most three consecutive categories. For projects classified into two or three consecutive development stages we considered the most advanced category as the actual stage of development reached by the project.

we should be able to assign projects that now fall into one of the new categories to one of the old categories. This is easy for those projects belonging 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 falling only into new categories have been dropped from the sample, because there is no clear way of associating these observations with any of the old categories of intended audience.

4 Results

The aim of our empirical exercise is to estimate the impact of observed OS project characteristics on its development status; Table 4 presents the estimated coefficients of interest in our model specification. The regression includes other control variables like programming languages and operating systems.

The bottom of the table shows a goodness of fit χ-square test and a Link specification test (Pregibon, 1980). The χ-square tests whether the predicted probabilities in the six categories are equal to the true ones; the value of this test is 10.69 with p-value equal to 0.06, thus indicating that the model fits the data reasonably well. The Link specification test indicates that the model is correctly specified.

Table 4: **The Determinants of Development Status**

Dependent: **Development Status**		
AGE	.1339***	(.0065)
DEV	.0357***	(.0019)
DEV^2	-.0001***	(.00002)
License		
HIREST	-.1156***	(.0153)
$HIREST_{new}$	.1076***	(.0210)
Topic		
Communication	-.0640***	(.0197)
$Communication_{new}$	.0574**	(.0285)
Games	-.2352***	(.0213)
$Games_{new}$	-.0479	(.0304)
Internet	.1049***	(.0174)
$Internet_{new}$	-.0074	(.0246)

Continued on next page...

... Table 4 continued

Multimedia	.1448***	(.0209)
Multimedia$_{new}$	-.0112	(.0298)
Sw tools	.2134***	(.0200)
Sw tools$_{new}$	-.0633**	(.0282)
System management	.0396**	(.0181)
System management$_{new}$	.0122	(.0262)
Office	-.1180***	(.0336)
Office$_{new}$	.0523	(.0472)
Science	.0900***	(.0242)
Science$_{new}$	.0009	(.0362)
Intended Audience		
End users	.0023	(.0154)
End users$_{new}$	.0690***	(.0225)
Developers	-.0830***	(.0151)
Developers$_{new}$	.1732***	(.0214)
System administrator	.1159***	(.0178)
System administrator$_{new}$	.0591**	(.0268)
α_0	-.7986	(.0268)
α_1	-.2176	(.0265)
α_2	.2517	(.0265)
α_3	.9101	(.0267)
α_4	2.3243	(.0300)
Wald test (Licence)	26.22	[0.000]
Wald test (Intended Audience)	70.93	[0.000]
Wald test (Topic)	31.05	[0.005]
pseudo-R^2	0.0241	
χ-square test	10.69	[0.06]
link-test	-.0055	[0.843]
Num of obs.	45100	

Regressions are conditioned also on programming language and operating system. Robust standard errors 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; p-values in brackets.

From this table, the following considerations can be derived:

1. The dummy variable controlling for highly restrictive licenses, HIREST, has a negative and significant coefficient; this result confirms Hypothesis 1 and is in line with previous studies

(Fershtman and Gandal, 2004; Lerner and Tirole, 2005; Stewart et al., 2005).[13] It is worth noting that the test of significance on the parameter $HIREST_{new}$ indicates that new and old projects do not have the same coefficient estimate for the highly restrictive license dummy (value of the test: 26.22, p-value: 0.00); for newer projects the coefficient of the license dummy is -0.008, i.e. the sum of $HIREST$ and $HIREST_{new}$ coefficients. A simple t-test shows that this coefficient is not significantly different from zero thus suggesting that for newer projects the type of license does not have a significant impact on projects success;[14]

2. Hypothesis 2 suggests that applications directed towards sophisticated users are generally the more successful ones. From the SourceForge list of software topics it is not always clear which projects are directed towards sophisticated or unsophisticated users. The categories of "Software tools" (i.e. tools for software development) and "System management" certainly identify applications intended for sophisticated users, while "Games" and "Office" refer to software suited for unsophisticated users; the other topic categories cannot be clearly associated with a given type of user. The results shown in Table 4 seems to support Hypothesis 2: the dummies controlling for Software and System management applications have positive and significant coefficients, while the coefficients of Games and Office are negative and significant.

3. Hypothesis 3 is supported by the data: the size of the community, here measured by the number of developers, has a positive impact on project development status. It is interesting to note that the size of the community affects in a non-linear way the likelihood of reaching an advanced stage of development. Indeed, the coefficient of DEV^2 is negative, even though it has a very small magnitude. This suggests that the contribution of an additional developer to the productivity of the community of developers working on a certain project decreases with the size of the community; a possible interpretation of this result relies on coordination issues:

[13] According to Lerner and Tirole (2005) a more restrictive licence may be chosen when the project has a limited community appeal (e.g. when the community distrusts the project leader); if this is indeed the case, our estimations about the impact of license restrictiveness should be considered with caution due to a problem of self- selection. A way to test for self-selection is to check whether GPLed and non-GPLed projects follow different patterns in their code evolution. We ran separate regressions for GPLed and non-GPLed projects and we found that the sign and significance of the parameters are largely the same; this seems to suggest that self-selection does not have a statistically significant effect.

[14] The standard error of the license dummy for newer projects is 0.016.

the larger the community of developers, the more complex its organization and governance.

4. In order to control for the evolving nature of OS, we have included the interaction of the regressors with the dummy variable NEW_i in the estimation. The Wald test for parameter stability suggests that, overall, the impact of the licence terms, of topic and of intended audience on project development stage is significantly different between older and newer projects. This result supports Hypothesis 4 and suggests that the OS phenomenon has changed through time. However, looking at the sign and significance of these interactions it is difficult to pinpoint a clear direction in the evolution of the OS.

5. As far as the intended audience characteristics are concerned, projects geared towards system administrators have greater chances of reaching an advanced development stage.

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

5 Robustness

In order to control for the robustness of our results and to deal with the concerns related to the quality of the data, we ran two complementary estimations: equation (1) was estimated *a*) by dropping from the sample the 5%-largest projects measured in terms of the number of developers, and *b*) by aggregating the development status of the projects into four broader categories: 1 if the project is classified as planning, 2 if the project is classified as pre-alpha or alpha, 3 if it is classified as beta and 4 when the project falls into either the production stable or the mature category.

As discussed in Section 3, one of the main concerns about the use of the data collected on SF is the so- called "listed-but-not-hosted" phenomenon: projects may decide to create their own web sites, omitting to update the information on their development stage reported on SF. Although there is no clear relationship between the project's characteristics and the fact that it has its own web page, two patterns seem reasonable: projects that have reached an advanced development stage or projects that have a large community of developers may need more sophisticated and specific on-line facilities than those provided by SF. Therefore, the "listed-but-not-hosted" concern may be a real issue, if any, in relation to projects that have reached an advanced development stage or that are characterized by a large community of developers.

In regression *a)* of Table 5 we dropped from the sample the top 5% projects in terms of the number of developers; we therefore replicated the estimation of equation (1) focusing on projects with a "small community of developers", that is, those projects for which the "listed-but-not-hosted" should be a less relevant issue.

Regression *b)* was included in order to deal with the second pattern according to which a project creates its own web page. Consider a "production stable" project that creates its own web page and that, at the same time, omits to update the information on SF. In this case, if the project goes forward to the next stage, i.e. it becomes mature, then information on SF would not be correct: the project is observed as "production stable" while it is actually "mature". By aggregating the more advanced stages of development we reduce these possible shortcomings of the database, even though we base our estimation on a rougher classification for the development stage.

Employing broader categories of development status is also helpful to control for another potential pitfall of the database. As discussed above, it is not easy to pinpoint what makes a project belong to two adjacent categories of development stage: by aggregating successive categories we also control for this possible source of distortion.

The results reported in Table 5 are in line with those obtained in our full-sample regression of Table 4: the signs and the statistical significance of all the coefficients do not show relevant differences in the three estimations.[15] The stability of results in the three sets of regressions strongly supports our full-sample estimation and is a confirmation of the quality of the dataset; therefore we can claim that the "listed-but-not-hosted" phenomenon either is not an issue (i.e. even those projects that have their own web page continue to keep the information on SF up-to-date) or it does not distort the results of our analysis.[16]

[15] The main difference is in the impact of the number of developers on development status: in regression *a*), DEV^2 positively affects the likelihood of reaching an advanced development status, rather than negatively as in Table 4, thus suggesting an increasing marginal impact of an additional developer. This is not surprising: when the community is not big enough, coordination is not a problem and having one more developer working on a project may have a more than proportional impact on the probability of its success.

[16] We have also re-run our main regression removing from the sample the "autistic" projects i.e. those projects that have only one developer and that do not show any interaction with the community. For these projects the available information might be less reliable since it is not filtered by the community. Again, the sign and significance of the parameters of interest do not change with respect to our main regression.

Table 5: **Robustness**

Dependent: **Development Status**				
	a) Restricted sample		*b*) Aggregated develop. status	
AGE	.1270***	(.0067)	.1328***	(.0067)
DEV	.0088	(.0174)	.0396***	(.0021)
DEV^2	.0053*	(.0030)	-.0001***	(.00002)
License				
HIREST	-.1071***	(.0159)	-.1062***	(.0157)
$HIREST_{new}$	.0956***	(.0216)	.0995***	(.0215)
Topic				
Communication	-.0749***	(.0204)	-.0755***	(.0200)
$Communication_{new}$	.0683**	(.0293)	.0683**	(.0291)
Games	-.2257***	(.0221)	-.2343***	(.0212)
$Games_{new}$	-.0430	(.0314)	-.0526*	(.0305)
Internet	.1087***	(.0180)	.1043***	(.0180)
$Internet_{new}$	-.0228	(.0252)	-.0070	(.0255)
Multimedia	.1415***	(.0219)	.1394***	(.0213)
$Multimedia_{new}$	-.0158	(.0308)	-.0226	(.0306)
Sw tools	.2120***	(.0208)	.2082***	(.0206)
Sw $tools_{new}$	-.0662**	(.0290)	-.0603**	(.0290)
System management	.0339*	(.0187)	.0460**	(.0188)
System $management_{new}$	.0142	(.0269)	.0016	(.0272)
Office	-.1247***	(.0350)	-.1266***	(.0343)
$Office_{new}$	.0547	(.0487)	.0647	(.0484)
Science	.0819***	(.0254)	.0763***	(.0247)
$Science_{new}$	.0125	(.0375)	.0051	(.0372)
Intended Audience				
End users	.0010	(.0160)	.0045	(.0158)
End $users_{new}$	.0733***	(.0230)	.0700***	(.0230)
Developers	-.0886***	(.0156)	-.0909***	(.0156)
$Developers_{new}$	.1795***	(.0219)	.1806***	(.0220)
System administrator	.1076***	(.0184)	.1203***	(.0184)
System $administrator_{new}$	.0679**	(.0274)	.0668**	(.0278)
α_0	-.8590	(.0320)	-.7913	(.0274)
α_1	-.2754	(.0317)	.2559	(.0272)
α_2	.1970	(.0316)	.9147	(.0273)
α_3	.8590	(.0318)		
α_4	2.2555	(.0346)		
Wald test (Licence)	19.61	[0.000]	21.38	[0.000]

Continued on next page...

... Table 5 continued

Wald test (Intended Audience)	73.83	[0.000]	73.39	[0.000]
Wald test (Topic)	34.80	[0.002]	29.93	[0.008]
pseudo-R^2	0.0208		0.0281	
χ-square test	6.01	[0.305]	4.00	[0.261]
link-test	0.0556	[0.188]	-.0149	[0.617]
Num of obs.	42814		45100	

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; p-values in brackets.

6 Conclusions

In this paper, we have evaluated the impact of various characteristics of an OS project on its probability of success. Our analysis adds to the existing literature in several respects. We measure the success of a project in terms of the development stage it has reached. This seems to be an appropriate measure of success given the user-centric innovation characteristic of OS projects, and it is in line with the definition adopted in other studies that focus on the technical achievements of a software. In the empirical exercise we employ a comprehensive dataset, consisting of all the projects hosted on SourceForge.net in December 2004. Previous studies have usually focussed on a more limited number of projects and, typically, on projects that have already reached some level of success; by using such a large dataset, we notably reduce the risk of sample selection.

We find that projects distributed under restrictive licensing terms have a lower probability of reaching an advanced development stage; this effect disappears for more recent projects. Applications for sophisticated users have higher chances of evolving towards a stable release. Interestingly, we find that the size of the "community of developers" has a non-linear impact on the probability of success of a project; the negative and significant sign of the coefficient of DEV^2 might signal possible problems of coordination when the community of developers gets larger. Finally, we find that the impact of project characteristics on the likelihood of success significantly differs between older and more recent projects; nevertheless, we are unable to characterize a clear dynamic pattern in the evolution of the OS phenomenon.

Our analysis has some limitations too. By employing a large set of data we cannot control for some important determinants of software success. In particular, we do not consider different organizational aspects of the projects that have been shown to be relevant by other studies, e.g. the degree of modularity or the level of embeddedness of a project (see, for instance, Grewal et al., 2006). Nonetheless, we believe that this paper may help to have a better understanding of a complex phenomenon such as that of OS.

Acknowledgements: Paper presented at the 2006 EARIE Conference (Amsterdam) and at the 2006 Jornadas de Economia Industrial (Barcelona). We wish to thank Nicolas Garrido for the extremely helpful discussions and Riccardo Marcon for research assistance. Financial support from the University of Padua through the "Progetto di Ateneo" research programme is gratefully acknowledged. The usual disclaimer applies.

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., Giannangeli, S., and Rossi, C. (2006). Entry Strategies under Competing Standards: Hybrid Business Models in the Open Source Software Industry. *Management Science*, 52(7):1085–98.

Bonaccorsi, A. and Rossi, C. (2003a). Licensing Schemes in the Production and Distribution of Open Source Software. An Empirical Investigation. Sant'Anna School of Advanced Studies, Pisa.

Bonaccorsi, A. and Rossi, C. (2003b). Why Open Source Software Can Succeed. *Research 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.

Gosh, R. and Prakash, V. (2000). The Orbiten Free Software Survey. *First Monday*, 5(7):1–8.

Grewal, R., Lilien, G., and Mallapragada, G. (2006). Location, Location, Location: How Network Embeddedness Affects Project Success in Open Source Systems. *Management Science*, 52(7):1043–56.

Healy, K. and Schussman, A. (2003). The Ecology of Open-Source Software Development. University of Arizona, *mimeo*.

Hecker, F. (1999). Setting Up Shop: the Business of Open-Source Software. *IEEE Software*, 16(1):45–51.

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

Lerner, J. and Tirole, J. (2002). Some Simple Economics of Open Source. *Journal of Industrial Economics*, 50(2):197–234.

Lerner, J. and Tirole, J. (2005). The Scope of Open Source Licensing. *Journal of Law, Economics & Organization*, 21(1):20–56.

Maddala, G. (1983). *Limited-Dependent and Qualitative Variables in Econometrics.* Econometric Society Monographs in Quantitative Economics. Cambridge, Cambridge University Press.

Pregibon, D. (1980). Goodness of Link Tests for Generalized Linear Models. *Applied Statistics*, 29:15–24.

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.

Rosen, L. (2001). Which Open Source License Should I Use for My Software? Open Source Initiative.

Rossi, A. (2004). Decoding the Free/Open Source (F/OSS) Software Puzzle: a Survey of Theoretical and Empirical Contributions. Quaderni - Università di Siena, n. 424.

Stewart, K., Ammeter, A., and Maruping, L. (2005). Impact of Licence Choice and Orgnizational Sponsorship on Success in Open Source Software Development. University of Maryland, mimeo.

von Hippel, E. (2005). *Democratizing Innovation*. The MIT Press, Cambridge Mass., London.

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? Melting Proprietary and Open Source Platform Strategies. *Research Policy*, 32:1259–1285.