Information and Software Technology 52 (2010) 902–922

Contents lists available at ScienceDirect

Information and Software Technology

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

Survival analysis on the duration of open source projects

Ioannis Samoladas *, Lefteris Angelis, Ioannis Stamelos

Department of Informatics, Aristotle University of Thessaloniki, 541 24, Thessaloniki, Greece

A R T I C L E I N F O

Article history:
Received 28 August 2009
Received in revised form 30 April 2010
Accepted 2 May 2010
Available online 11 May 2010

Keywords:
Open source software
Survival analysis
Prediction

A B S T R A C T

Context: Open source (FLOSS) project survivability is an important piece of information for many open source stakeholders. Coordinators of open source projects would like to know the chances for the survival of the projects they coordinate. Companies are also interested in knowing how viable a project is in order to either participate or invest in it, and volunteers want to contribute to vivid projects.
Objective: The purpose of this article is the application of survival analysis techniques for estimating the future development of a FLOSS project.
Method: In order to apply such approach, duration data regarding FLOSS projects from the FLOSSMETRICS (This work was partially supported by the European Community's Sixth Framework Program under the Contract FP6-033982) database were collected. Such database contains metadata for thousands of FLOSS projects, derived from various forges. Subsequently, survival analysis methods were employed to predict the survivability of the projects, i.e. their probability of continuation in the future, by examining their duration, combined with other project characteristics such as their application domain and number of committers.
Results: It was shown how probability of termination or continuation may be calculated and how a prediction model may be built to upraise project future. In addition, the benefit of adding more committers to FLOSS projects was quantified.
Conclusion: Analysis results demonstrate the usefulness of the proposed framework for assessing the survival probability of a FLOSS project.

1. Introduction

Open source software has been transformed from a model of voluntary participation in code activities into a viable software production alternative, keeping its volunteer nature. Products like the Apache web server, the kernel of the Linux operating system, the Mozilla browser or the MySQL relational database management system have been deployed and widely used both by industry and desktop users. Apart from these successful open source projects, there is a vast amount of software that is not publicly known or has started and terminated as it has been shown by various studies.

Project survivability is useful information for many open source stakeholders. Coordinators of open source projects would like to know the chances for the survivability of the projects they coordinate, and in particular whether their project is doomed to failure. Companies interested in either adopting an open source product in their information technology infrastructure or participating in a project are also interested in knowing how viable a project is. The probability of the continuation or inversely the probability of "short life" of a project is a factor affecting participation in a project. Volunteer programmers might be more interested in entering a project that has high chances to evolve than to fail and be abandoned [9,17]. Finally, open source survivability would be beneficial for educational purposes. In recent years there are many universities that have incorporated participation in open source projects in their software engineering curriculum. Instructors supervising such courses are highly interested in their students participating in a successful, to be continued, project [24].

The purpose of this article is to present a novel framework, based on statistical methods, for the analysis of the survivability of open source projects. In order to test our framework we applied the method proposed on a set of open source projects available in the FLOSSMetrics (a European Union funded project) database.

The rest of the paper is organized as follows: first in Section 2, we present and discuss previous work done regarding open source project viability, and then Section 3 presents our approach to study the problem of open source projects survival. Section 4 presents discussion of the results, Section 5 provides threats to validity

* Corresponding author. Tel.: +30 6976624421.
E-mail addresses: ioansam@csd.auth.gr (I. Samoladas), lef@csd.auth.gr (L. Angelis), stamelos@csd.auth.gr (I. Stamelos).

0950-5849/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2010.05.001

and finally Section 6 concludes the paper, along with future research.

2. Related work

The definition and the study of success and failure in the context of open source projects is a matter of interest for many researchers. In traditional software development success means the completion of a software project in time and without exceeding its budget. On the contrary, in open source software development, time and money is not the case. Crowston et al. [4,5] tried to define what success in the context of open source software means by adapting existing models of success in information systems development, into the volunteer nature of open source development. They suggested that success has multiple aspects since it may depend on a researcher's or stakeholder's specific interest. In their study, they split their success indicators into three success measure aspects namely: (1) Project output, consisting of indicators such as developer's satisfaction and development progress according to identified goals, (2) process, consisting of indicators such as the number of developers, level of activity and bug resolution time and (3) outcomes for project members, consisting of indicators such as knowledge creation and individual reputation. In order to test the validity of their measures they conducted an empirical study, applying their model to a sample of projects from Sourceforge.net and what they concluded was that the majority of them could be characterized as successful.

Rainer and Gale [20] studied Sourceforge.net and concluded that the majority of the projects hosted there should be considered as failures, which, according to them, is the absence of activity in code development, mailing lists and bug reporting. Weiss [26] defined success of a FLOSS project in correlation with its popularity on the web. He proposed a set of metrics that utilize data from web search engines in order to define how successful a project is. In a more recent study, English and Schweik [6] proposed a model consisting of six categories, in order to define success and failure in FLOSS development. The latter classifies a FLOSS project into these categories according to the number of its releases in accordance with the time between these releases. The application of their classification model on the projects hosted at Sourceforge.net resulted in the majority of the projects being abandoned or in an early stage. The same authors [21] have also conducted a test using logistic regression and data from Sourceforge.net to reach the conclusion that adding more programmers to an open source project raises the chances of this project being successful. Subramaniam et al. [25] investigated FLOSS success using longitudinal data from FLOSS projects hosted at Sourceforge.net. Their success measures are developer interest, user interest and project activity levels. In addition they determined two categories of factors that affect those measures: time invariant factors (license, operating system and programming language) and time variant factors (project status, project activity, user interest and developer interest). Their findings indicate that these three measures are inter-related, affecting each other. For example they demonstrated that developer interest as a factor has a positive impact on developer activity.

Recently Beecher et al. [2] conducted a study concerning the evolutionary stages in open source projects that are hosted in different kind of repositories. Authors examined large repositories of open source projects, namely Sourceforge.net, KDE, GNOME, Rubyforge and Savannah and found evidence that they – the repositories – play an important role in a project's evolution. Repositories like Sourceforge.net, Savannah and Rubyforge have more "relaxed" preconditions in hosting a project (in fact anyone can start a project there) and contain projects with low activity. Authors showed that when an open source project enters a more "controlled" repository, like the KDE or GNOME, it has better chances to evolve into a successful project, which means – according to them – growing in size and gaining attention from developers.

Survival analysis has also been used to study the duration of both closed software and open source software projects. In our prior work [1,22,23], we proposed a methodology for duration analysis in closed software projects using the ISBSG[1] repository and showed that duration is affected by factors like customer participation, requirements and complexity. Recently, in parallel with our study, Evangelopoulos et al. [7] also presented a statistical methodology based on a combination of fitted logistic regression and discrete event simulation for determining the "death" of a process based activity based on lack of progress. They exemplified their methodology on software process by selecting 300 projects from the Sourceforge hosting service for a 76 months period and by calculating "resurgence probability estimates". They suggest that the average monthly death rate is 1.87%, but they argue that they have to study this more.

A more recent study elaborating on survival analysis is that of Ortega and Izquierdo [18]. In that study, authors are interested in the lifetime of volunteer contributors and its impact on the continuity of an open source project. They perform a similar comparison regarding contributors in Wikipedia, the open content, volunteer-based encyclopedia, and compared the results. This study also makes use of data from the FLOSSMETRICS project, the same project that our data come from, with the exception that authors include only projects with more than 5 years history. Their duration analysis involved the time distance of commits for each committer of the projects under study and the time distance between the first and the last contribution of Wikipedia contributors. They found that the Median Survival Time (the median time in days) regarding committers was between 500 and 1000, whereas for Wikipedia contributors this was restricted to 200 and 400. Without going into further details, authors suggest that their methodology would be valuable to open source project coordinators as an indicator of their project's vitality and attractiveness.

We should also mention here an interesting application of survival methods in modeling fault proneness in FLOSS projects which can be found in Koru et al. [14]. The authors proposed a Cox hazard modeling method and they found that the size of a software module, in particular the size of C++ classes, has a significant effect on defect proneness. They claim that using their approach into FLOSS development can help developers to prevent bugs and perform preemptive fixes.

This study contributes to FLOSS research by adding value to the previous ones in the following ways:

- It is the first study that applies survival analysis to a large scale of FLOSS projects and to a full extent by exploiting all survival analysis tools, such as Cox regression models and Kaplan–Meier non parametric approximation of a survival curve. In particular, we provide a regression model for predicting the chances of a FLOSS project survival.
- It is the first time that projects coming from multiple forges are analyzed for survivability.
- Data from 1147 projects are used, while the maximum number of projects previously analyzed is 300. This fact permits the investigation of project survivability across many different application domains.

We have to note here that it is rather difficult to define and measure in an objective manner the notions of success and failure. There are several factors that have to be accounted for while some

[1] ISBSG Data Disk, Release 7, http://www.isbsg.org.au.

of them are sometimes subjective and controversial. For example, a project coordinator may actually have in mind to run a short life project. Also, it is probably not so fair to classify projects in a strictly binary manner, such as "success" or "failure", but rather to allow intermediate states of successfulness. In spite of all these difficulties and the subjectivity in defining what is really successful, we believe that the duration of projects and the factors that affect it, is particularly interesting and should be taken into account in any attempt to determine success.

3. Methodology

In order to perform our study we used data from the FLOSSMetrics project. FLOSSMetrics is a European Union funded project, run by a consortium that is formed by four universities and three enterprises. The main purpose of FLOSSMetrics is to construct a large database containing information and metrics about open source software development coming from several thousands of software projects. The data cover the main three open source development parameters: source code management systems (code repositories), mailing lists and bug tracking systems. For the purposes of our study we used data coming only from source code repositories.

After data collection we applied statistical methods that have been extensively used in medical and engineering research for studying the survival time of patients or the reliability of devices. They have also been applied in marketing, criminology, epidemiology, and even social and behavioural sciences. As mentioned above, such methods have been used for the duration of software projects by our research team [23]. Survival analysis is particularly useful for studying FLOSS projects for two main reasons, (a) it analyses activities over time defined by one starting and one terminating event and (b) it takes into consideration cases that are still in progress, which is true for thousands of FLOSS projects that are active simultaneously during any project duration study.

3.1. Data collection

The data we used originate from the FLOSSMETRICS database. The time of data collection was March 2009. At that time the database contained 1147 projects from various areas. Some of the projects were included in the study randomly among a set of projects that fulfilled certain criteria regarding their size from the Debian distribution and the Sourceforge.net hosting site. Some others were included specifically because they are considered large projects. From the Debian distribution the FLOSSMETRICS team considered about 150 of the largest projects according to their source lines of code. Projects that are part of larger meta-projects (collections of other projects), like Apache and KDE or GNOME are also included in the database (e.g. a music player part of the GNOME project). Projects from Sourceforge.net were chosen randomly. For each of these projects, their source code management hosting URL was located and with the help of appropriate tools, their development history was stored in a unified way in the FLOSSMETRICS database.

It is important here to emphasize the fact that the FLOSSMETRICS database contains historical data from the source code management system present at the time of the analysis. The database is maintained in a regular basis. Although we used a snapshot of the database during March 2009, we decided to use a cut off date in March 2008 for the duration of our historical data. The choice was based on the fact that up to that date the database contained full historical data for all 1147 projects and the decision to exclude projects that started after that date. This means that the data cover the period for which the code repository is valid, i.e. on line, at the time of study. We have not investigated whether a project was hosted elsewhere before or during our study has been moved. This is a common limitation in open source research, since project coordinators may choose at some time to move to a different hosting service, the project itself might fork, i.e. continue with a different name, or to switch to another source code management system. The latter case was frequently observed during the previous years because of the transition from CVS to subversion source code management system. This case is considered in our model and will be discussed in the following sections. In addition we have also categorized the projects into 12 categories according to their intended audience, in order to study potential differences in project survivability across diverse application domains. Concerning categorization, we used the same categories as those of Sourceforge.net. In case a project was already categorized by Sourceforge.net we adopted this category, otherwise we placed the project into one of these categories ourselves, by inspecting its characteristics. These categories are:

1. Communications,
2. Database,
3. Desktop,
4. Education,
5. Games and entertainment,
6. Internet,
7. Multimedia,
8. Office,
9. Science,
10. Security,
11. Software development,
12. System.

After collecting raw data from the FLOSSMetrics database we performed a second level analysis in order to perform our study. The time slot we chose for our statistical analysis covers 160 months (we consider a month to be four weeks), each month considered to consist of four weeks. The actual dates are from midnight (00:00:01) of January 1st of 1996 to midnight (23:59:59) of March 12th of 2008. The methodology described in the next sub section, studies the survival time which is defined as the time to the occurrence of a predefined, terminal event. This terminal event, for the purposes of our study will be discussed later. This event has a critical meaning depending on the context of the study (death, failure, response to treatment, etc.). In our study, we use the term duration time or simply duration to refer to the time from *birth month* to *the death month*.

With the term *birth month* we refer to the very first instance of activity in the project's repository as it is stored in the FLOSSMetrics database. We assume that *birth month* is the first revision in the stored repository URL. Particularly for projects that already existed on the initial date of our study, we assume that their *birth month* is this date. The 12% of our sample projects belong in this category. The usage of subjects that were already "alive" at the start of the study period is typical in survival analysis. They convey useful information and the ability to handle them distinguishes survival analysis from other similar approaches. We discuss this issue specifically in the validity threats section.

Death month refers to the month that the terminal event occurred. In our study, the terminal event can be of two types:

a. The project is considered not active and thus it is abandoned.
b. The project has suddenly no code activity.

In order to define (a) we have made the assumption that a project is considered to be inactive if it has less than two commits per month. If this happens then the project is considered inactive for this month. If for the next month the project is inactive again, then it is considered *abandoned* and thus terminated. In their study, Evangelopoulos et al. [7], consider a project dead if there is no activity at all. For (b), we considered a project to be *lost to follow up* if suddenly it has no commits at all. For this case we assume that the stored repository URL in the FLOSSMetrics database is no longer valid, because for some reason the coordinators of the specific project have decided to move either to another location or to another source code management system. The distribution of commits in time, although not in our study plans, did not show any particular kind of pattern. The majority of the projects followed patterns similar to that of [8,10,15], i.e. a lot of activity in certain time slots near releases and less otherwise.

The duration data include apart from the measurement of duration other variables characterising the projects like the type of the project, the language used and the number of committers. The study of duration data is focused on estimation of the probabilities of duration, comparisons between distributions of different project types and identification of variables correlated to duration.

3.2. Analysis

As mentioned earlier, for our analysis purposes we used statistical methods suitable for duration data, known as survival analysis. The peculiarity of duration data is that the precise duration times of some projects may not be known. This happens when the terminal event has not yet occurred for some projects in the dataset. These cases in a survival dataset are generally called *censored times* but since our data concern software projects we can refer to them as *active* projects. In our study we have four possible cases for projects, according to their duration: The active ones, those being *inactive for 1 month*, those being *inactive for 2 months* and the ones which are *lost to follow up*. Projects characterized as *inactive for 1* month and *inactive for 2 months* are very few in our sample. They exist because they have been characterized as such during the last couple of month in our analysis time slot. Should our analysis go beyond the time mentioned, some of these would be abandoned, some would become active and other projects would enter these two states, too. In our analysis we will consider

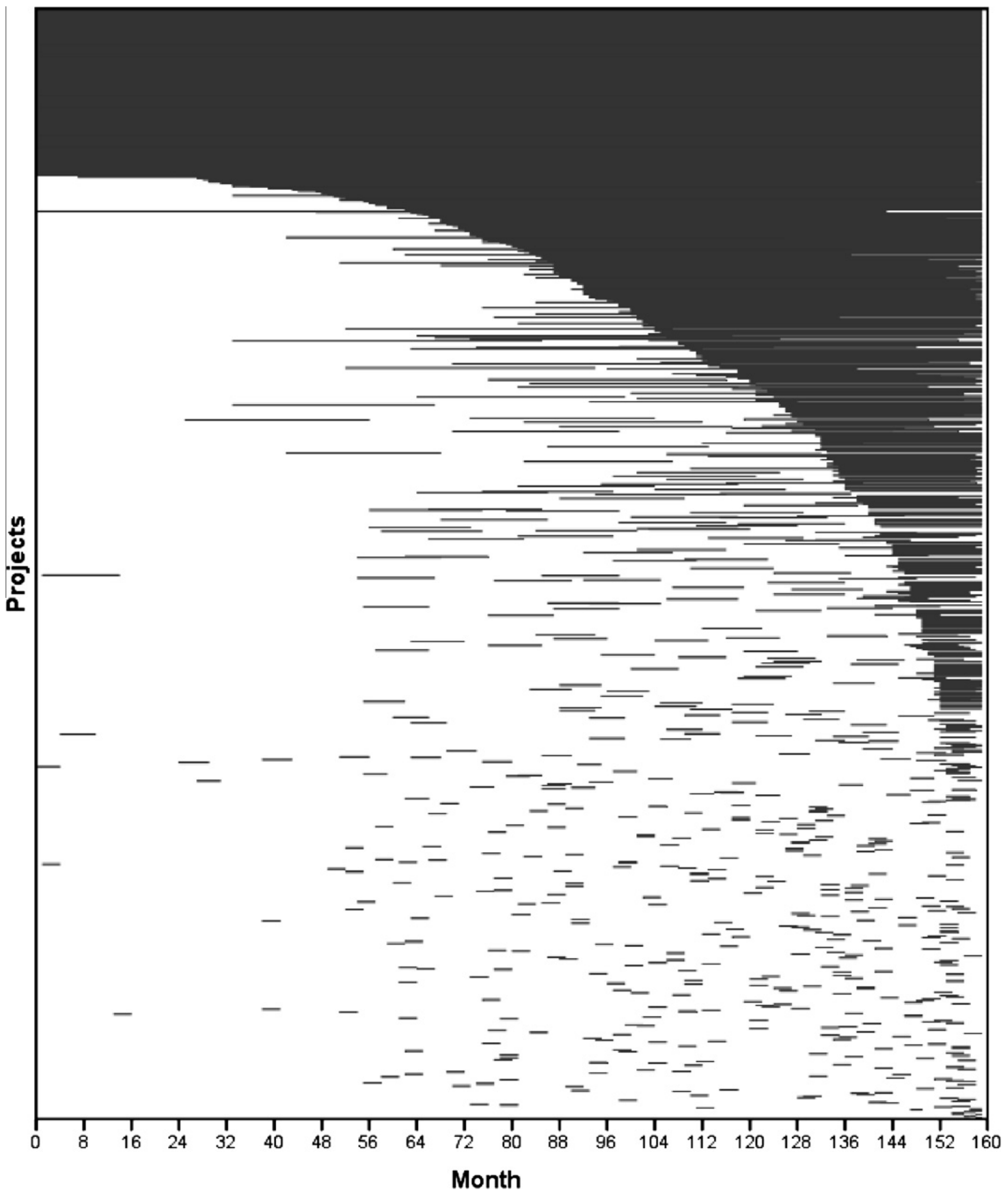

Fig. 1. Duration of all projects.

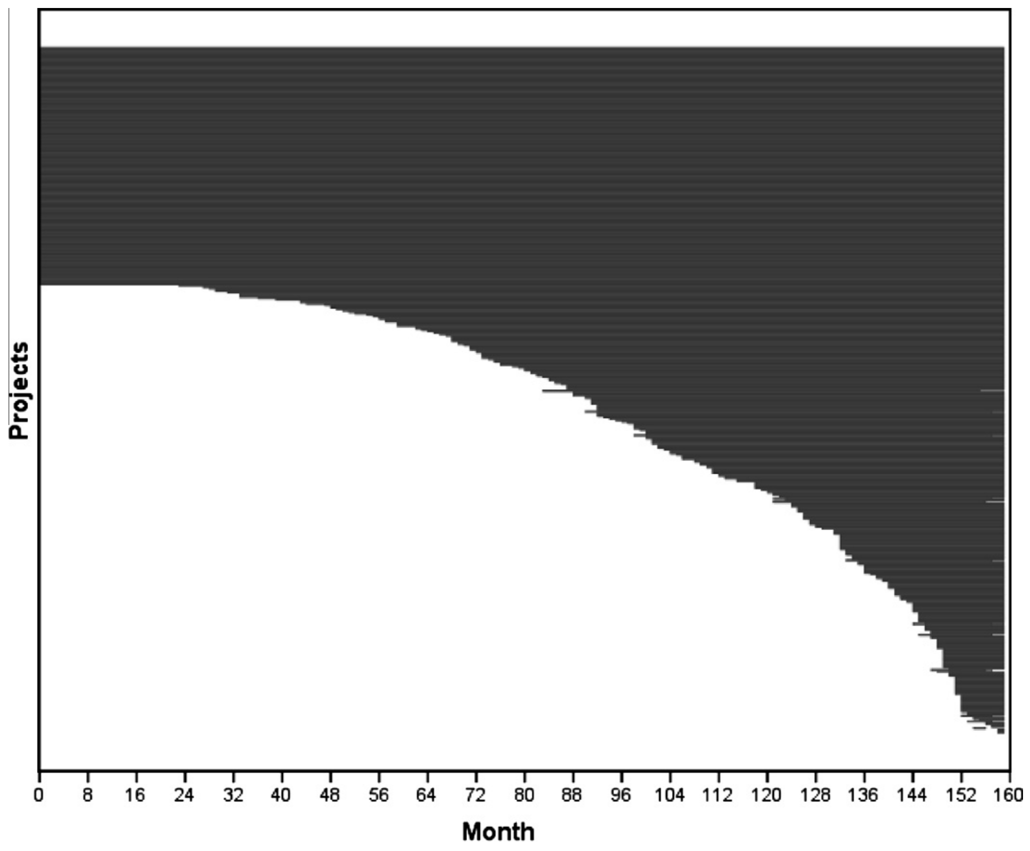

Fig. 2. Duration of active projects.

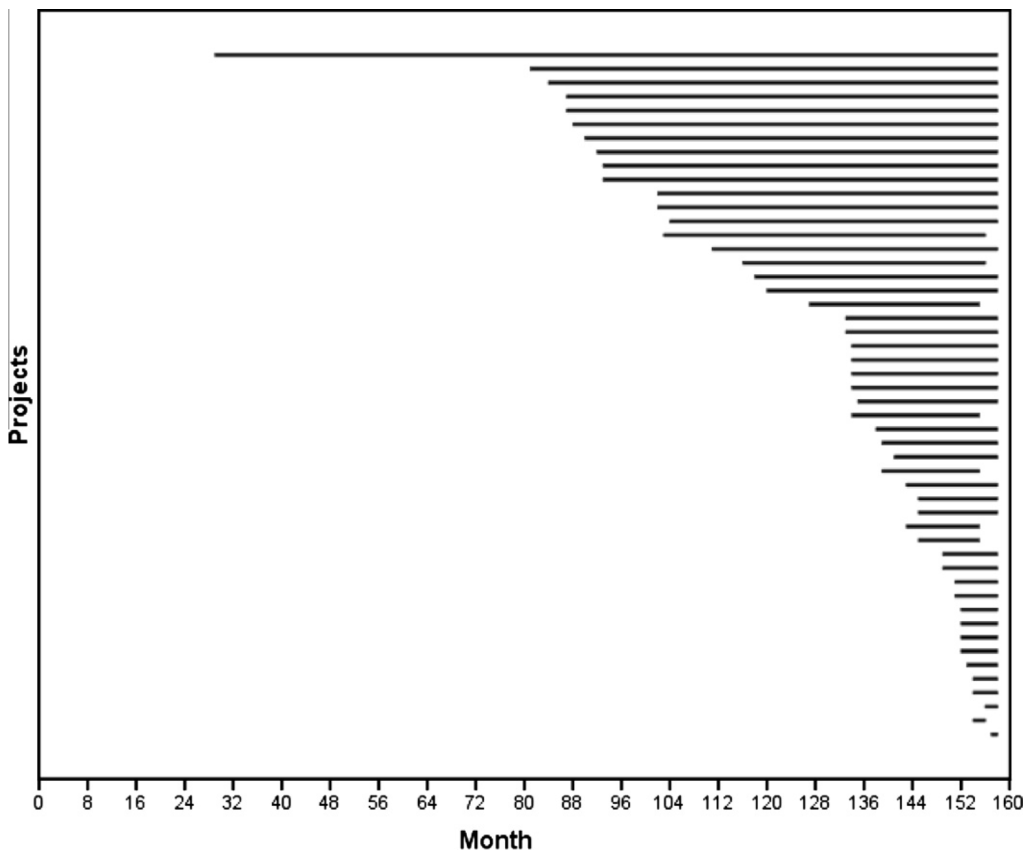

Fig. 3. Duration of projects characterized as ''Inactive 1st month''.

the most pessimistic view, i.e. only the active projects will be considered censored, while all the others will be considered terminated, even though *lost to follow up* projects can be still active.

Since the starting month of the research is set to 0 and the cut-off month at 159, we give in the following plots (the numbers of the list refer to the figures accordingly) of all durations:

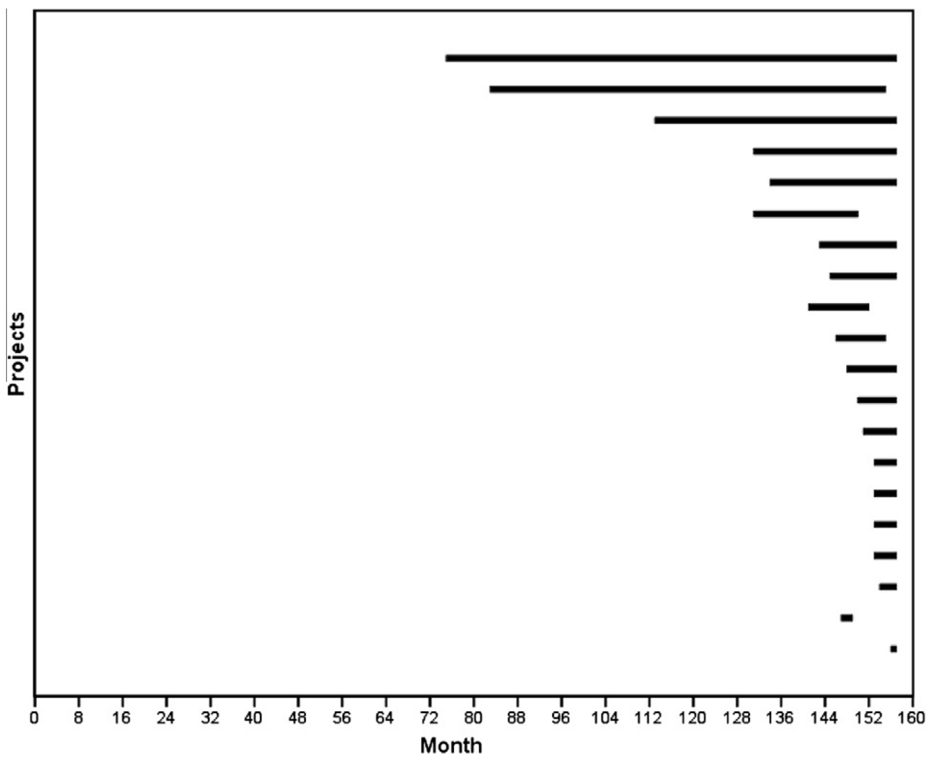

Fig. 4. Duration of projects characterized as "Inactive 2nd month".

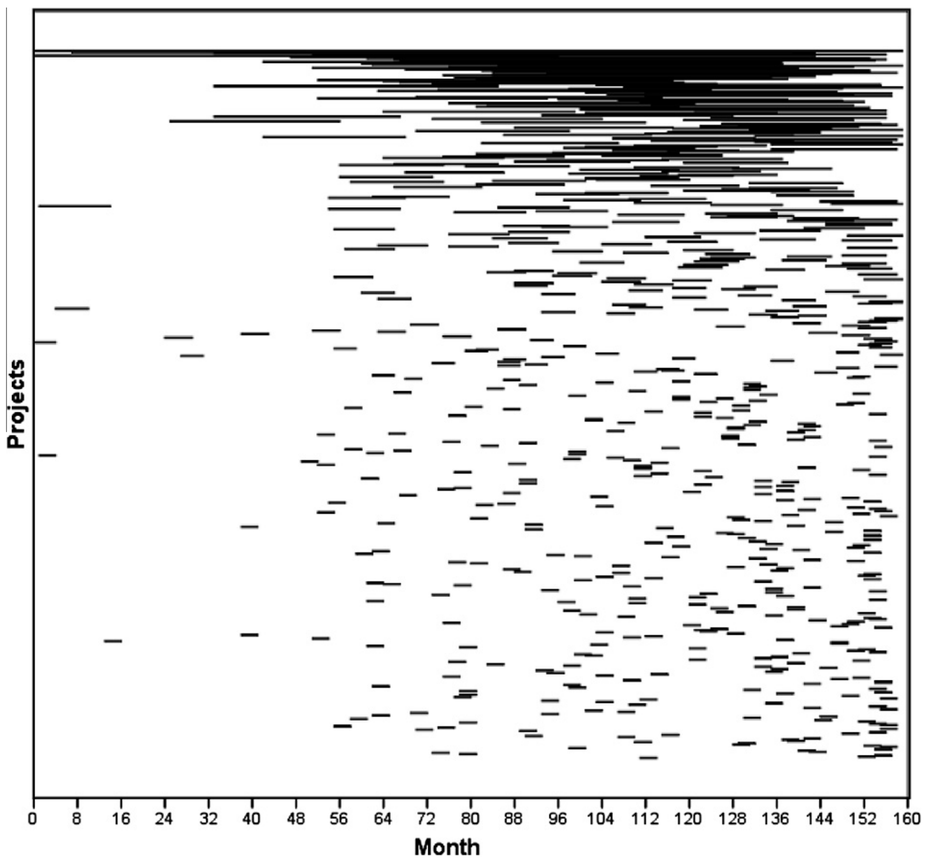

Fig. 5. Duration of projects characterized as "Lost to follow up".

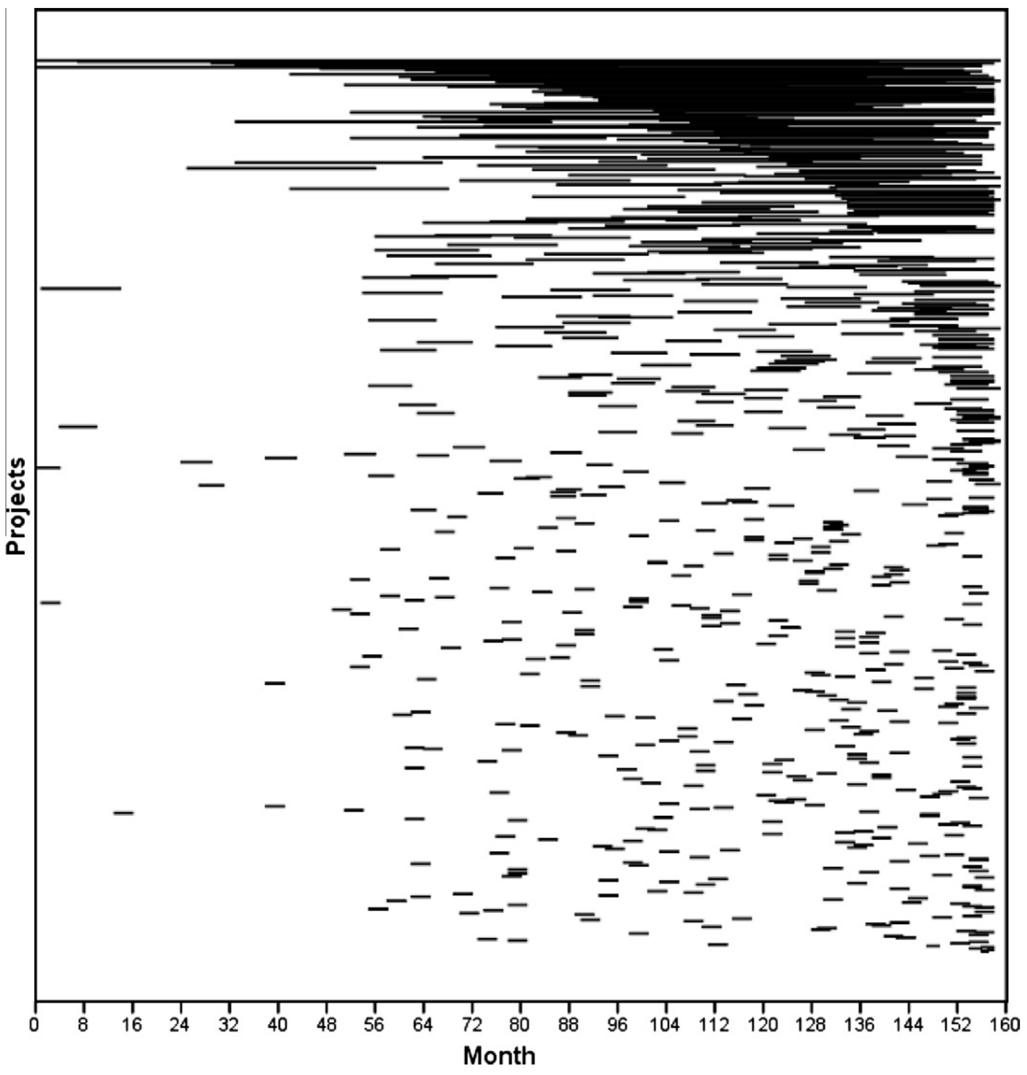

Fig. 6. Duration of all projects except from active ones.

Fig. 1, all projects.
Fig. 2, active projects.
Fig. 3, inactive for 1 month projects.
Fig. 4, inactive for 2 months projects.
Fig. 5, lost-to-follow-up projects.
Fig. 6, all projects which are not active.

Each project's duration is represented by a straight line segment starting from the birth month and terminating at the death month. Each line represents a single project. The line for active projects stops at month 159 since this is the cut-off month of the study. Table 1 shows the number of projects of each class depicted in Figs. 2–5.

We can see clearly that the active projects are much more than the others and include a large number of projects that were active for the whole period of research (0–159 month).

Table 1
Number of instances of projects in each class.

Project class	Number of projects
Active projects	494
Inactive for 1 month projects	50
Inactive for 2 months projects	20
Lost-to-follow-up projects	583

3.2.1. Basics of survival analysis

The distribution of duration can be described by three functions:

1. The probability density function.
2. The survival function.
3. The hazard function.

These three functions are mathematically equivalent but their interpretation is different. Also, these functions are estimated from the data by approximation methods. In what follows, we denote by T the positive continuous random variable representing the duration time of software projects while by t we denote its values.

The *Probability Density Function* (*PDF*) is a nonnegative function denoted by $f(t)$ and is the most known probability function for any random variable. It is defined as the limit of the probability that a project is completed in the time interval $(t, t + \Delta t)$ per unit width Δt. The mathematical formula defining PDF is:

$$f(t) = \lim_{\Delta t \to 0} \frac{P(t < T < t + \Delta t)}{\Delta t}$$

Another function defining the distribution of the duration times is the *Cumulative Distribution Function* (*CDF*) denoted by $F(t)$:

$$F(t) = P(T \leqslant t) = \int_0^t f(u)du$$

The *Survival Function* (*SF*) is denoted by $S(t)$ and is defined as the probability that the duration of a project is longer than t:

$$S(t) = P(T > t) = 1 - F(t)$$

$S(t)$ is a non-increasing function of time t with the properties:

$$S(t) = \begin{cases} 1 & \text{for } t = 0 \\ 0 & \text{for } t = \infty \end{cases}$$

The graph of $S(t)$ is called the *survival curve* and its shape can be interpreted in terms of short or long duration. Specifically, a steep curve shows short duration while a gradual or flat curve is an indication of longer duration.

The *Hazard Function* (*HF*) is denoted by $h(t)$ and is defined as the probability of completion during a very small time interval, assuming that the project is still active by the beginning of the interval:

$$h(t) = \lim_{\Delta t \to 0} \frac{P(t < T < t + \Delta t | T \geqslant t)}{\Delta t}$$

An alternative definition is given by:

$$h(t) = \frac{f(t)}{1 - F(t)} = \frac{f(t)}{S(t)}$$

The hazard function is a measure of the tendency for project completion as a function of the duration of the project.

The curve of $h(t)$ can have any shape. It may be increasing, decreasing, constant, or indicate a more complicated process. It should be noted that $h(t)$ has inverse meaning of $S(t)$ in the sense that $S(t)$ is focused on the extension of the project duration while $h(t)$ is focused on the termination of the project. That is why large values of $S(t)$ correspond to small values of $h(t)$ and inversely. HF is very important for the Cox regression models.

The C*umulative Hazard Function* (CHF) is defined as

$$H(t) = \int_0^t h(u)du$$

3.2.2. The Kaplan–Meier method

In the case where the data contain active projects we need to use a nonparametric estimation technique to estimate the survival and hazard functions. The most widely used method for estimating the SF in the presence of censored values is known as *product-limit* (P-L) or *Kaplan–Meier* (K–M) method [13].

First, the duration times of all the projects participating in the study are arranged in ascending order $t_1 < t_2 < t_3 \cdots$ then, at each time point t_i we calculate:

a. The number of terminations, denoted by d_i.
b. The number of censored durations, denoted by c_i.

The number of projects *waiting for termination* is denoted by n_i and is computed by:

$$n_i = n_{i-1} - d_{i-1} - c_{i-1}$$

Based on the above numbers, the *conditional probability of survival* is estimated by:

$$P(T > t_i | T > t_{i-1}) = \frac{n_i - d_i}{n_i} = 1 - \frac{d_i}{n_i}$$

The corresponding *unconditional probability* coincides with the notion of SF and is estimated by the recurrent relation:

$$\hat{S}(t_i) = P(T > t_i) = P(T > t_i | T > t_{i-1}) \cdot P(T > t_{i-1}) = \left(1 - \frac{d_i}{n_i}\right)\hat{S}(t_{i-1})$$

In order to derive an explicit expression of SF we denote by t_0 the starting point of the study where we can safely assume that $P(T > t_0) = 1$. Under this assumption, the above recurrent relation results in the explicit relation of the K–M estimate of the survival curve:

$$\hat{S}(t_i) = \prod_{j=1}^{i} \left(1 - \frac{d_j}{n_j}\right)$$

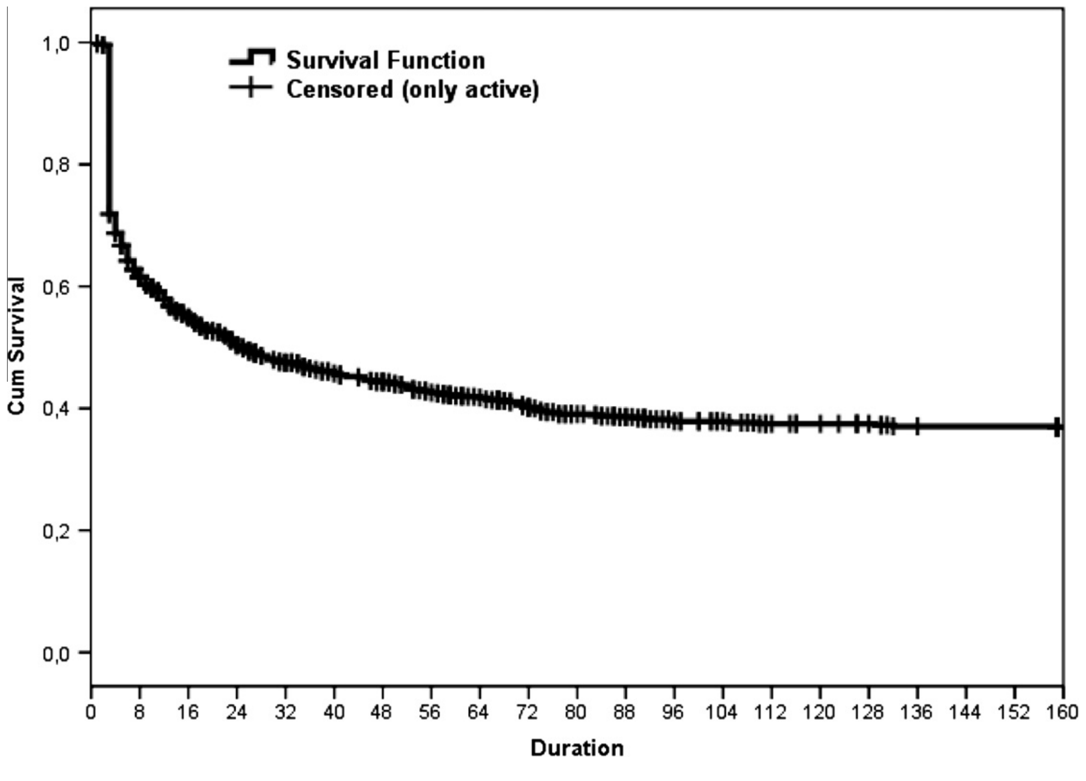

Fig. 7. Kaplan–Meier estimation of the survival function of all projects. Only the active projects are considered as censored cases.

The corresponding values of the CHF, $\hat{H}(t_i)$, are estimated from the K–M estimations $\hat{S}(t_i)$. The estimations $\hat{S}(t)$ and $\hat{H}(t)$ can then be graphically represented by curves useful for inferences and comparisons. For more details see [19,16].

Note that the values of $H(t)$ are not probabilities and the interpretation of the function is not easy. However, it is a function that plays a very important role in the survival analysis theory since it is used for estimation of distribution parameters and comparisons

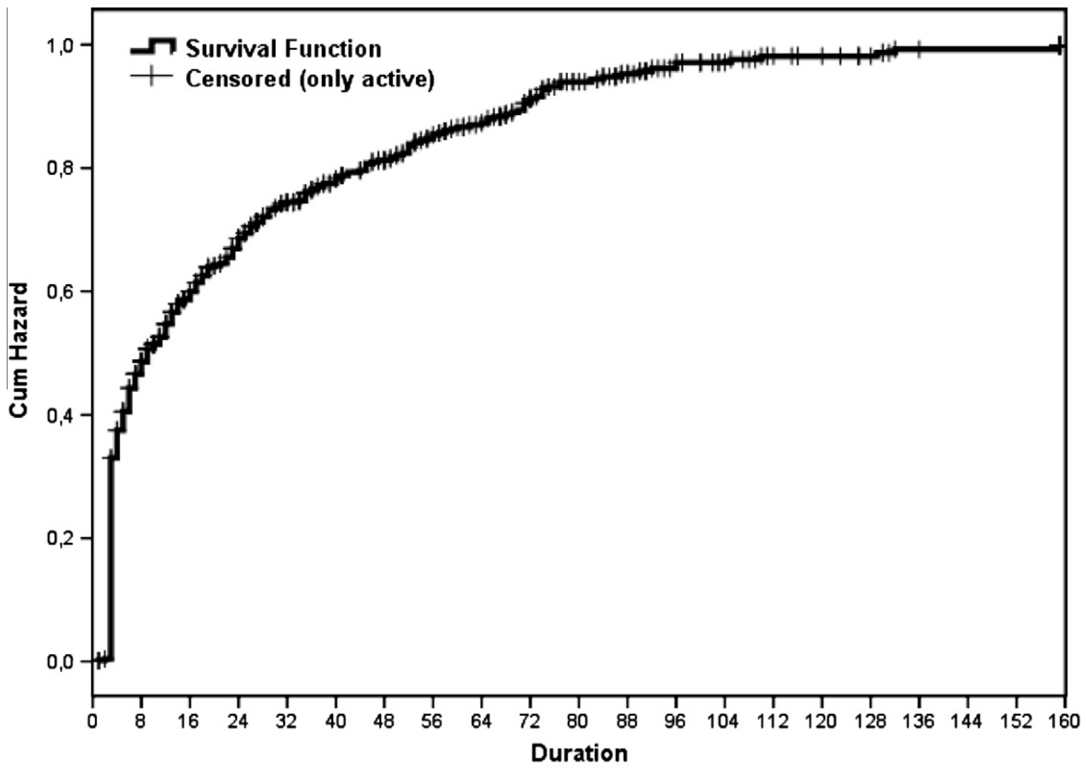

Fig. 8. Kaplan–Meier estimation of the cumulative hazard function of all projects. Only the active projects are considered as censored cases.

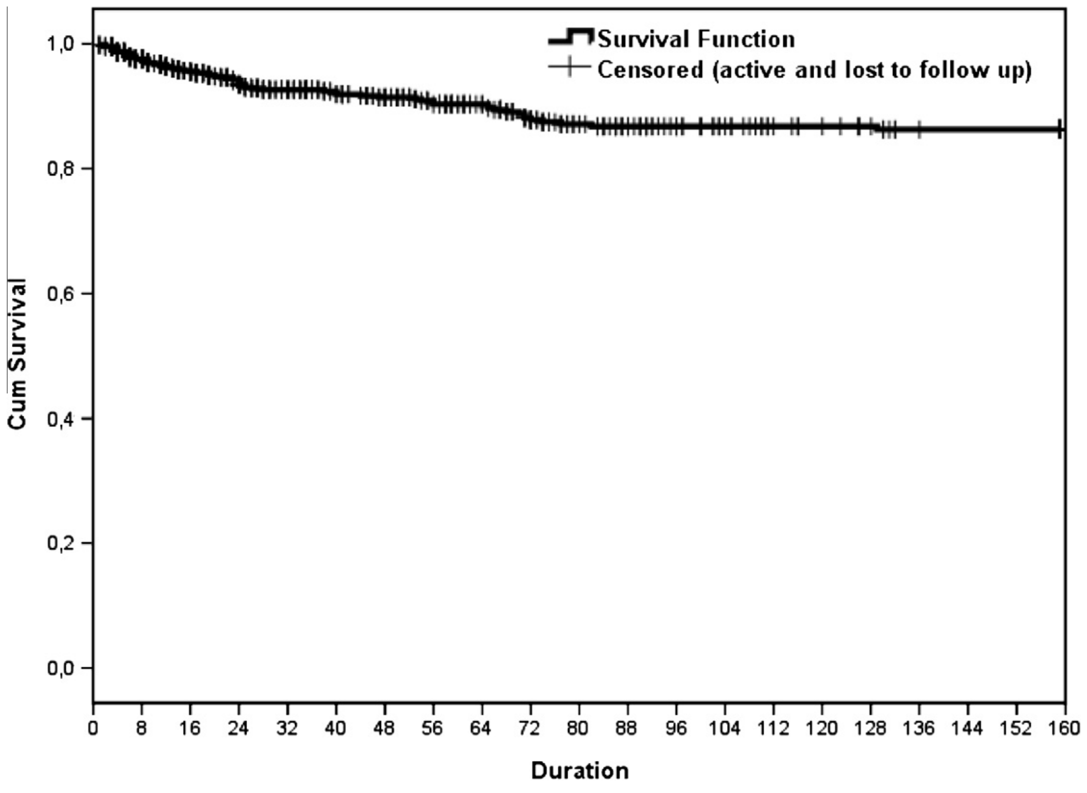

Fig. 9. Kaplan–Meier estimation of the survival function of all projects. Lost to follow up projects are also considered as censored.

between non parametric and parametric distributions through models connecting the duration time t and the values of $H(t)$.

The K–M curve enables us to describe graphically all the durations in the available dataset, even the censored ones.

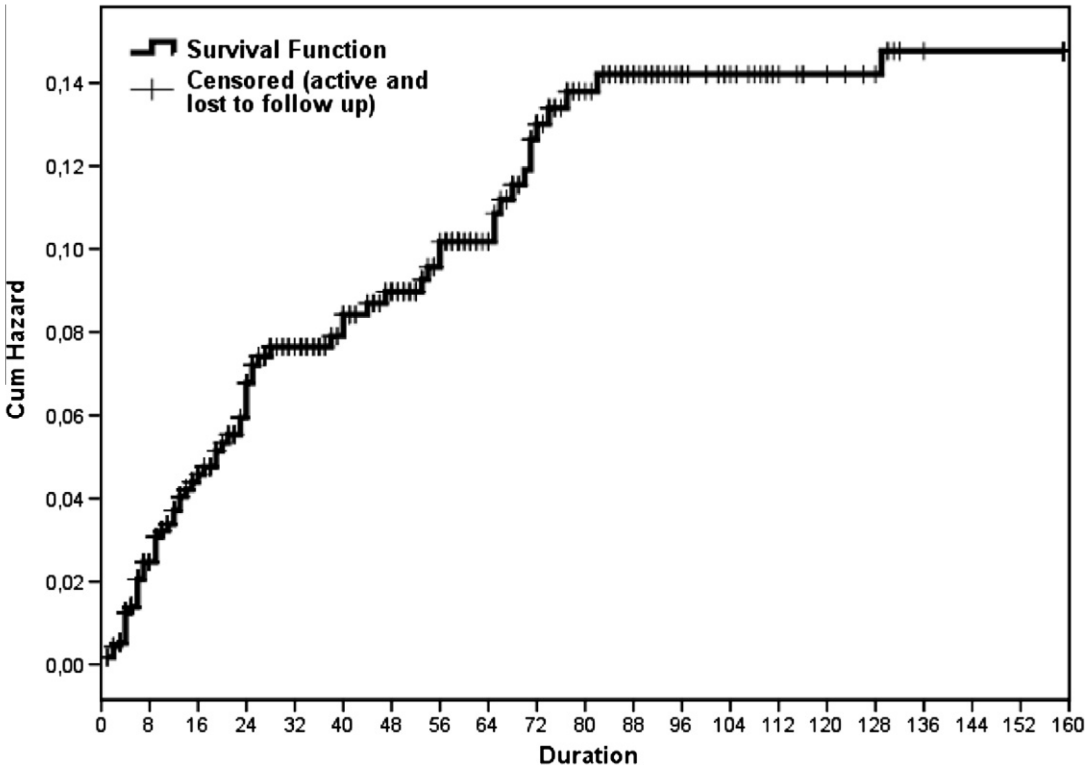

Fig. 10. Kaplan–Meier estimation of the cumulative hazard function of all projects. Lost to follow up projects are also considered as censored.

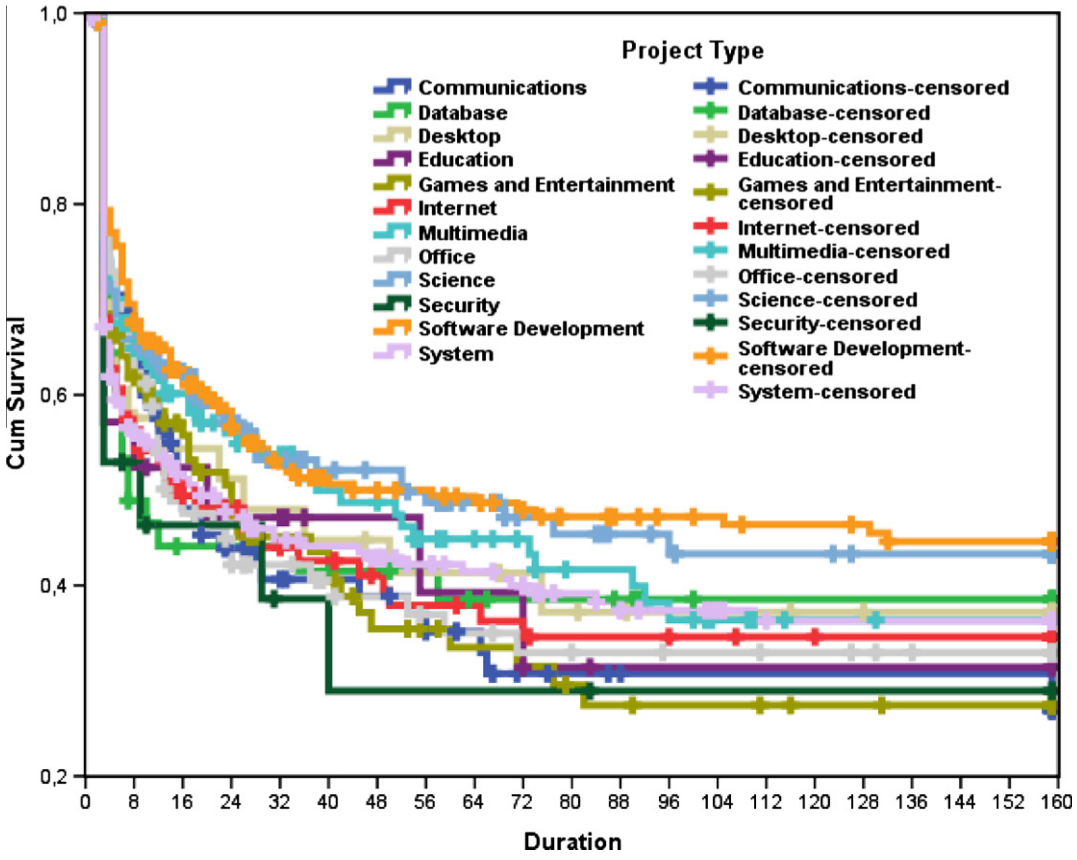

Fig. 11. Kaplan–Meier estimations of the survival functions, separately for each project type. Only active projects are considered as censored.

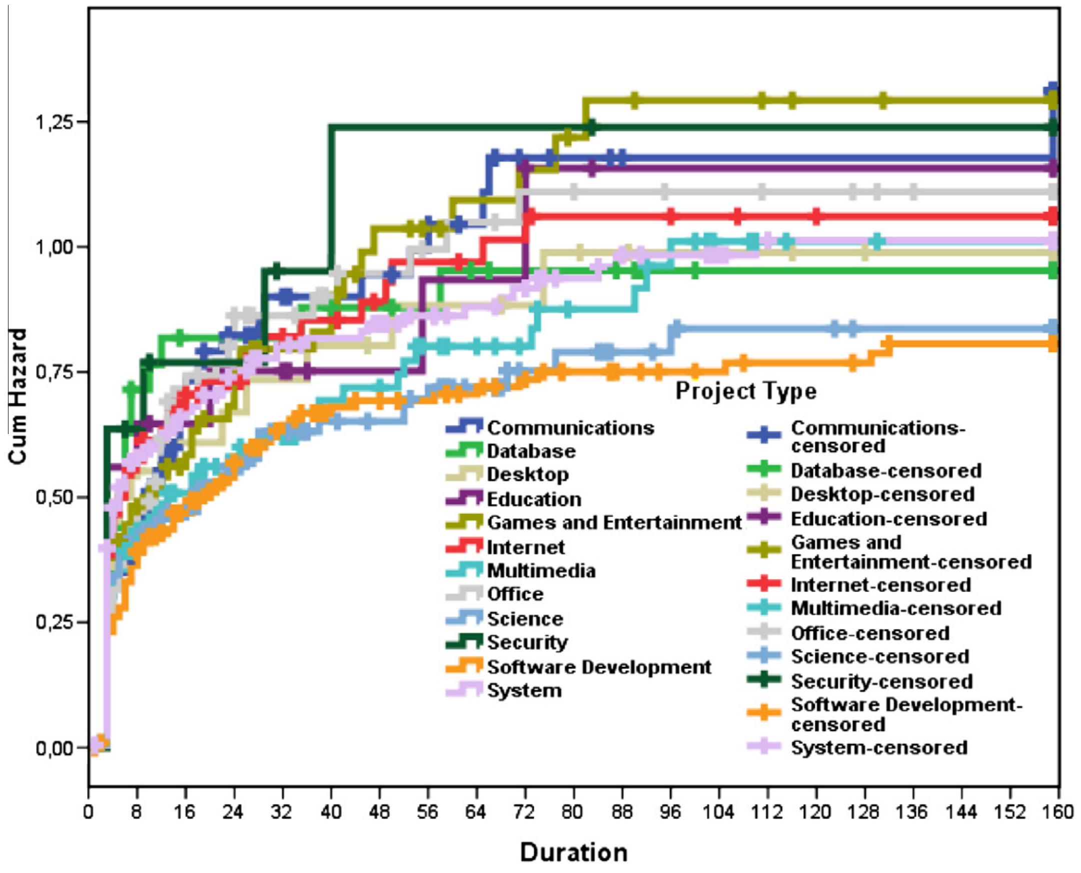

Fig. 12. Kaplan–Meier estimations of the cumulative hazard functions, separately for each project type. Only active projects are considered as censored.

In the following figures we can see the SF and the CHF estimated by K–M for all projects (Figs. 7 and 8). The censored cases are the projects characterized as active. We can see in general long durations even in this pessimistic estimation where all projects that were lost to follow up were considered as terminated (Fig. 7). However, the probability of durations more than 160 months falls below 40%. Another interesting remark is that under that assumption, the duration of a large percentage of projects (actually 28.7%), from all categories, is less than or equal to 3 months. This is represented by the early steep decent of the curve which shows that the probability of surviving more than 3 months is 71.3%.

In order to see the SF under the most optimistic estimation, we considered all the lost-to-follow-up projects as still active. The SF and the CHF are given below (Figs. 9 and 10). We can see clearly that the probability of survival of a project for more than 159 months does not fall below 80%.

The SF of all the project types simultaneously is depicted in Fig. 11, while the cumulative hazard functions are depicted in Fig. 12. SF figures of all project types separately are given in Appendix (Figs. 16–27). Some curves seem to have different behavior (for example see the "software development" which shows higher probabilities of long durations and the "security" and "games and education" with lower durations). This is not strange since software development tools, such as compilers, are often very prolific. However, the tests (such as the log-rank test) did not show significant difference. First we give the distribution of project types (Table 2) and the K–M estimation of their mean and median (Table 3). In the case where censored times are present, the K–M curve is first estimated and the median is found as the value of M satisfying the

Table 2
Distribution of project types.

Category	Frequency	Percent
Communications	88	7.7
Database	45	3.9
Desktop	33	2.9
Education	21	1.8
Games and entertainment	92	8.0
Internet	94	8.2
Multimedia	122	10.6
Office	85	7.4
Science	121	10.5
Security	17	1.5
Software development	215	18.7
System	214	18.7
Total	1147	100.0

Table 3
K–M estimation of means and medians.

Project type	Mean estimate	Median estimate
Communications	59,859	16,000
Database	66,516	7000
Desktop	63,561	26,000
Education	62,664	20,000
Games and entertainment	57,924	24,000
Internet	63,792	16,000
Multimedia	72,253	38,000
Office	61,896	14,000
Science	79,176	52,000
Security	54,141	9000
Software development	81,265	57,000
System	67,349	18,000
Overall	69,520	25,000

equation $\hat{D}(M) = 0.5$. The interpretation is quite simple: it is the time in which 50% of the projects will be terminated while the other 50% will be active beyond this time. Median is more reliable measure than mean, due to high skewness. Skewness is inherent in our dataset because of the intense, often abrupt project changes, in the open source world and the multitude of diverse FLOSS projects. For example, many developers start a small project that is abandoned after a couple of months.

3.2.3. Cox regression models for identification of prognostic factors

The prediction of the future of a project is called *prognosis*. When data contain a number of project attributes, it is often difficult to identify which ones are most closely related to prognosis.

These variables usually interact in complicated ways and their individual effects are confounded in unpredictable manners. For this reason it is preferable to study the combined effect of all the variables on the duration by modeling this relation.

There are various parametric models for modeling the relationship of duration with other variables but they require the knowledge of an underlying theoretical distribution for the error components of the model. This knowledge is often not available, so the common practice is to use a class of comparative, semi-parametric models for duration data. The most known methodology is the *Cox regression analysis* which will be outlined here.

First of all we have to point out the importance of the hazard function $h(t)$. The HF by definition is an expression of the accumulated knowledge over time, generally known as *aging*. Since HF is applied only to projects that have stayed active up to a particular time, it accounts for the aging that has taken place in the dataset [11]. The point here is that when projects are observed over time, we essentially witness an aging process. It is therefore reasonable to use the HF for modeling the aging process in a dataset.

Suppose that the HF is a function depending on a set of predictors $\mathbf{x} = (x_1, \ldots, x_p)$, denoted by $h(t;\mathbf{x})$. The basic assumption for the formation of the model is that any pair of different projects with corresponding values of the predictors (say) $\mathbf{x_1} = (x_{11}, \ldots, x_{1p})$ and $\mathbf{x_2} = (x_{21}, \ldots, x_{2p})$ has proportional HFs, i.e. the *hazard ratio* (*HR*)

$$HR(t;\mathbf{x}_1,\mathbf{x}_2) = \frac{h(t;\mathbf{x_1})}{h(t;\mathbf{x_2})}$$

is a constant, in the sense that it does not vary with time. The meaning of such an assumption is that the ratio of the conditional completion rate of two projects is the same no matter how long their duration is.

The constant HR property leads to a model where the HF given the set of predictors for a project is expressed as a product of two functions:

$$h(t;\mathbf{x}) = h_0(t)g(\mathbf{x})$$

The function $h_0(t)$ characterizes how the HF changes as function of only the duration. The other function, $g(\mathbf{x})$ characterizes how the CCR changes as the function of project predictors and represents their effect. Note that $h_0(t)$ is called *baseline HF* and represents HF when $g(\mathbf{x}) = 1$. In order to interpret the $h_0(t)$ as the HF when we remove the effect of all the predictors, a usual choice of $g(\mathbf{x})$ requires that $g(\mathbf{0}) = 1$.

The *Cox model* assumes that $g(\mathbf{x})$ is an exponential function of the predictors $\mathbf{x} = (x_1, \ldots, x_p)$, i.e. an expression of the form:

$$h(t;\mathbf{x}) = h_0(t)\exp\left(\sum_{i=1}^{p} b_i x_i\right)$$

whereby x_i, $i = 1, \ldots, p$ we denote the ith predictor (continuous or categorical) and by b_i the corresponding unknown regression coefficient. Here, the baseline HF corresponds to a situation where all predictor variables are equal to zero.

An interesting aspect of the Cox regression model is that it can be represented in terms of either the HF or the SF. It can be proved that the equivalent duration model can be written as:

$$S(t;\mathbf{x}) = [S_0(t)]^{g(\mathbf{x})}$$

where $S_0(t)$ is the baseline duration function and $g(\mathbf{x}) = \exp(\sum_{i=1}^{p} b_i x_i)$. The regression coefficients and the baseline function are estimated from the data by a version of the maximum likelihood method using the partial likelihood function [16,11].

Special attention should be paid on the basic assumption of the Cox model, that the ratio $h(t;\mathbf{x})/h_0(t)$ is not a function of time. The validity of such a hypothesis is tested by statistical tests. However, if the assumption is violated, an extended version of the Cox regression model can be used so as to include time-dependent predictors [12]. Available statistical packages (like SPSS and S-plus) provide procedures with a wide variety of options in order to build and validate an efficient model. One of the most important features is the execution of algorithms (e.g. stepwise regression) which are able to produce models containing the most important variables. This is very useful for the cases where the predictors are correlated.

In our case we have information about two predictors, i.e. the numerical variable *committers* and the categorical variable *project type*. In order to include in the equation the categorical variables, there is a need to use binary or "dummy" variables taking only values 0 and 1. When the levels of a factor are only two, these can be represented by 0 and 1. In general, when we have p levels in a factor, we need $p - 1$ dummy variables to represent them in the form $(1,0,\ldots,0),(0,1,\ldots,0),\ldots(0,0,\ldots,0)$. These are useful for interpreting the regression coefficients.

For building the model we used a stepwise Cox regression, as implemented in the statistical program SPSS, which is capable to identify only the most significant predictors for the Cox model. The algorithm resulted in a model with only one significant variable, namely the committers ($p < 0.001$). The categorical variable was not considered significant for the model.

The final model is:

$$h(t;\mathbf{x}) = h_0(t)\exp(-0.172 \times committers)$$

The value $e^{-0.172} = 0.842$ means that the HF is reduced by $100\% - (100\% \times 0.842) = 15.8\%$ each time a unit is added to committers. The 95% confidence interval (CI) for e^b is [0.819, 0.866].

It is interesting to see the equations for each type of projects separately. Table 4 gives the coefficients of the models, the confi-

Table 4
The Cox regression models for each type separately.

Type of project	b	exp(b)	% of reduced HF when one committer is added	95% CI for exp(b)	
Communications	−0.152	0.859	14.1	0.774	0.954
Database	−0.145	0.865	13.5	0.767	0.975
Desktop	−0.086	0.917	8.3	0.803	1.048
Education	−0.068	0.934	6.6	0.843	1.035
Games and entertainment	−0.161	0.851	14.9	0.786	0.921
Internet	−0.165	0.848	15.2	0.765	0.941
Multimedia	−0.250	0.779	22.1	0.699	0.868
Office	−0.219	0.803	19.7	0.702	0.920
Science	−0.226	0.798	20.2	0.726	0.877
Security	−0.476	0.621	37.9	0.371	1.039
Software development	−0.145	0.865	13.5	0.815	0.919
System	−0.192	0.825	17.5	0.766	0.889

dence interval for e^b and the percentage of the HF reduction each time a committer is added.

It should be emphasized that the Cox regression does not produce point estimations (single values) of the duration of a project. Its prognosis is a whole distribution of duration values with assigned probabilities. So for any value of the predictors we can get graphical representations of SF. For example in the following graph (Fig. 13) we can see the SF curve corresponding to the *average*

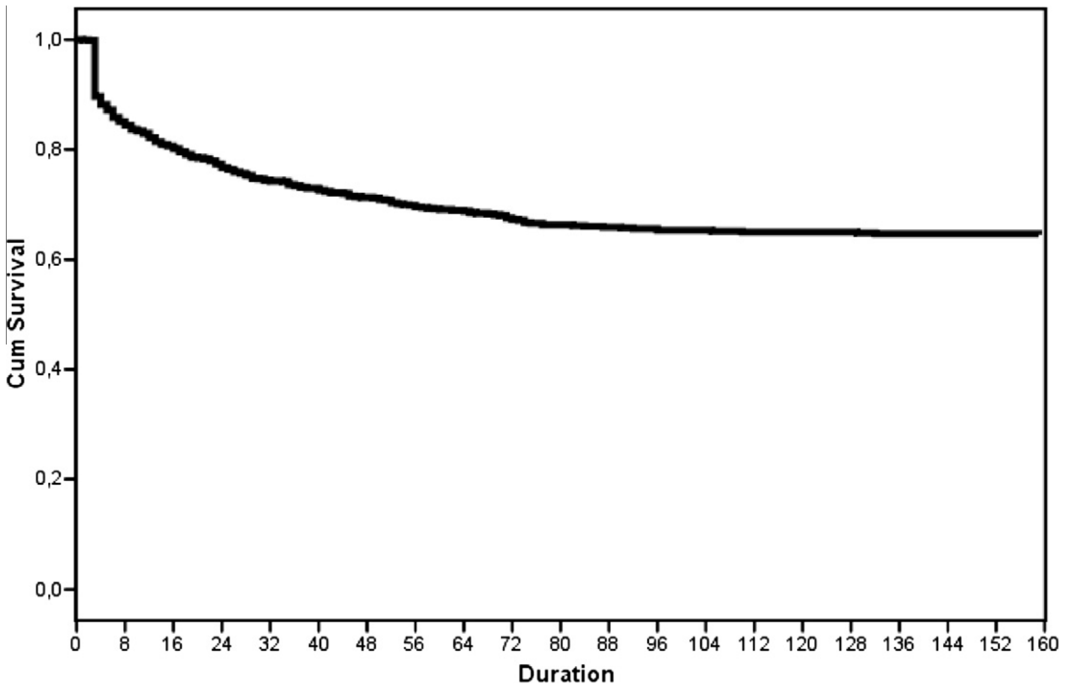

Fig. 13. Estimation of the survival curve by Cox regression for committers = 10.5.

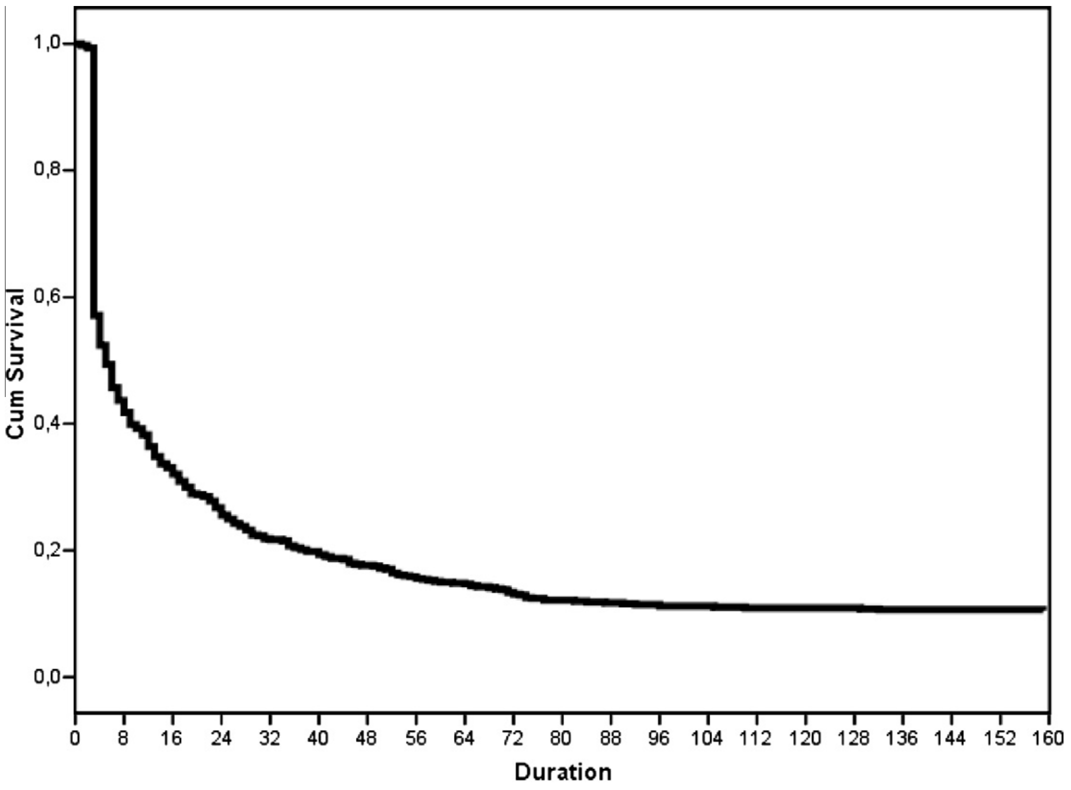

Fig. 14. Estimation of the survival curve by Cox regression for committers = 1.

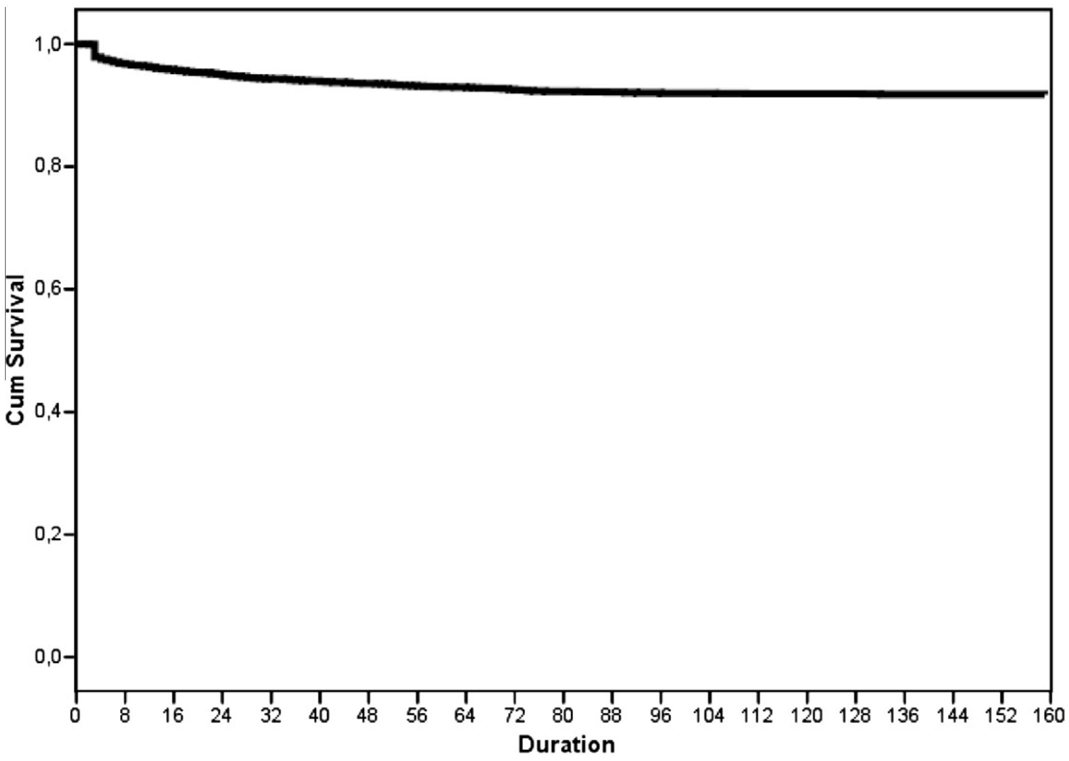

Fig. 15. Estimation of the survival curve by Cox regression for committers = 20.

project in the sense that in Cox regression equation we substitute the variable *committers* by its mean value. In this way we have the curve for a hypothetical project having committers = 10.5.

We can see that for about 11 committers the probability of long duration does not fall below 60%. On the other hand, the following Fig. 14 shows the SF when committers = 1. The probability of long duration falls below 20%.

For committers = 20 the curve does not fall below 80% (Fig. 15).

4. Discussion

Looking at the tables and the analysis of the previous sections, a first observation that can be clearly drawn is that projects which have already a prolonged activity and presence in open source landscape are difficult to be abandoned. Fig. 1 shows that projects that existed more than 10 years ago continue to evolve. This could mean that there are open source projects which have proved their value and established a sustainable community.

Regarding the probability of survival of open source projects, it can be argued that with our pessimistic calculations, this is about 40% for more than 5 years. The optimistic calculation, that with the *lost to follow up* projects included, raises this probability up to 80%. We consider these numbers encouraging.

Although statistical tests do not show any significant difference, Tables 1 and 2 show that there are certain types of software which seem to have higher probability of survival. For example projects in the "software development" and "science" category have more chances to continue to evolve, while projects in "database" or "security" less. Maybe this is so because both these categories are crucial and important and users usually choose the most popular projects and avoid trying new ones. However, it may be also attributed to the fact that software engineers are predisposed to software development projects. In any case this is a finding that deserves better examination.

One other interesting result of our study is the outcome of the Cox regression analysis that showed that for every new programmer (committer) we add in a project, its survivability is increased by 15.8%. Scweik et al. [21] showed that for each developer added to an open source project, the chances for this project to become successful increases 1.24 times. The idea that in open source larger team means successful projects, dubbed as "Linus' Law" is a hot research issue nowadays. Recently Capiluppi and Adams [3] conducted a study questioning Brook's law and found that, regarding communication channels, it is valid for the core developer team of an open source project. Our finding provides additional insight to these claims. However, we have to point that we do not distinguish between core developers and sporadic ones. Future analysis on open source manpower will give us more insight and would prove a useful tool for open source coordinators.

The proposed framework of Survival Analysis is able to take into account prognostic or influential factors as we clearly show in our analysis. These influential factors can be recognized by separate survival curves and Cox regression. So, a coordinator can have an early prognosis of the survivability of a project. Also, the impact of certain actions (like attracting more developers, adding interesting functionality or improve marketing of their project) could be studied within the framework we propose, although this problem is beyond the scope of the present paper. The interested reader is referred to papers such as [5] for quality indicators of open source projects that should be pursued and monitored.

5. Threats to validity

The following concerns have been identified as a threat to the validity of our study. We have categorized our concerns into two sections, namely internal and external validity:

5.1. Construct – internal validity

We have considered as the initiative critical event, the *start month*, the first activity recorded in the FLOSSMetrics database. This date for some projects, which account for the 12% of the projects analyzed, does not match the actual beginning of the project. Our study started at 00:00:00 01/01/1996 and many projects like the GNU C Compiler existed well before that. For these projects we assume that they have been started at 00:00:00 01/01/1996. The choice to include these projects in our analysis was based on the following fact: the majority of these projects had a very long duration within the period of the study and therefore their exclusion from the analysis would cause significant loss of valuable information, particularly regarding the maximum duration. So, although there is an underestimation of the actual duration, since the duration before the study starting date is unknown, the removal would result to a less realistic estimation of the maximum observable duration.

Another issue is with the projects characterized as *lost to follow up*. For some reasons, open source project coordinators decide either to move their code repositories to a new URL or to a new source code management system. In addition it is possible for a project to fork and continue as a new name, but with the same codebase. These are cases that can happen in open source software, although the FLOSSMetrics database team makes efforts to assure continuity in a projects history. We also have to stress here that this is a problem of all empirical open source studies that manipulate public available data, particularly with Sourceforge.net [12]. It is not rare for a project to start in a forge and then move to its own infrastructure. We tried to heal this issue by conducting two analyses: one with the *lost to follow up* projects considered being "inactive" and another with these projects considered as "active". We believe that with these two approaches we have minimized the effect of this phenomenon.

Another assumption that affects the results of our study is that of the definition of the "death" of a project. We took a rather optimistic view by assuming that a project is considered abandoned if it has less than two commits per month (zero or just one). Healthy projects often have hundreds of commits per month even per week, so it is fair to assume that a project with one commit per month has ended. For our analysis we did not take into account other activities of open source development like mailing and bug reporting. So even if a project has an active mailing list and bugs being reported, but no code activity, we consider that this project has ended.

Finally, an additional validity threat might originate from the way projects entered our sample as described in Methodology. Such a threat would not exist if an entirely random sample were available.

5.2. External validity

The size of the population is an issue for our study. Given the fact that the open source landscape comprises of thousands of projects, the number of the projects we studied is relatively small. However, the requirements for the selection process of the projects in the FLOSSMetrics database dictated that projects of all kinds (successful and not) and sizes should be considered. In addition the FLOSSMetrics database is constantly being fed with new data and new projects and our methodology can be easily applied to any sample size.

6. Conclusions – future work

This study demonstrated how survival analysis can be applied to a full extent to large samples of FLOSS projects in order to provide useful insight for the chances such projects have to be continued in the close future and understand how the number of committers and application domain affect their duration. We showed how probability of termination or continuation may be calculated and how a prediction model may be built to upraise project future. In addition, we quantified the benefit of adding more committers to FLOSS projects. Information of this kind may be useful as a decision making tool for FLOSS coordinators, sponsors, users and participants, since in most cases, project critical attributes and dynamics are hard to be understood and assessed. For example, project coordinators may be alerted whenever project survival becomes critical in order to manipulate project popularity

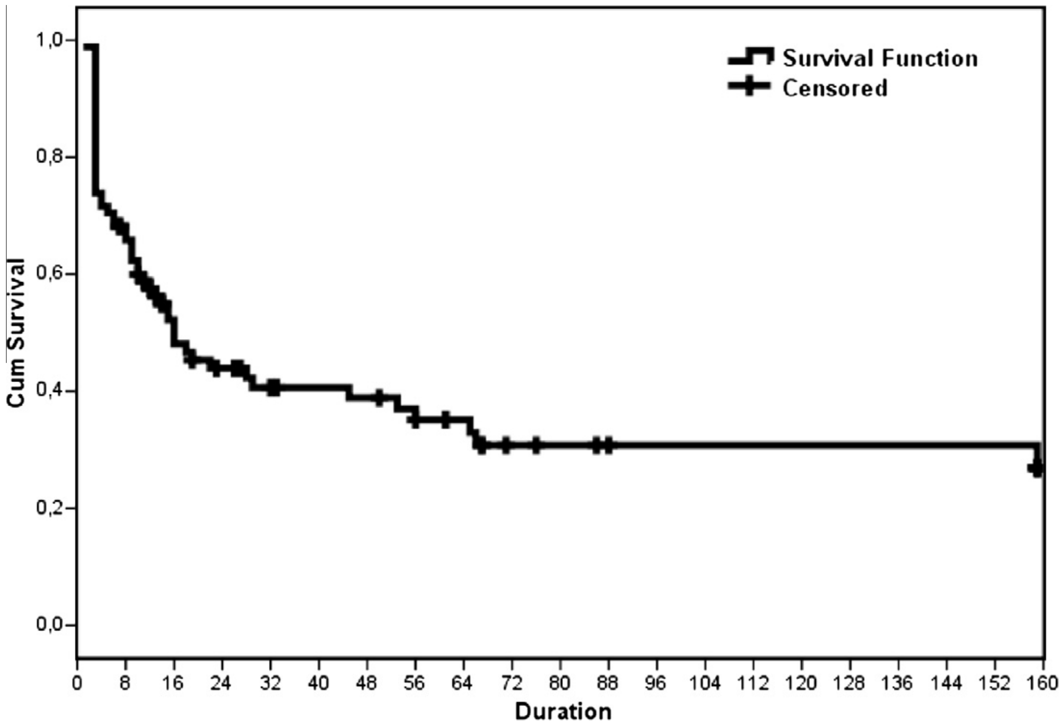

Fig. 16. Kaplan–Meier estimation of the survival function of Communications projects. Only active projects are considered as censored cases.

growth mechanisms and attract more developers. In addition, survival analysis may provide a holistic view of the open source phenomenon, providing findings w.r.t. FLOSS project duration and success. Studying such figures over long times and for large number of projects may reveal such characteristics as intensity, vividness and longevity of the average FLOSS project.

The analysis we presented here has two main research issues to be addressed in the future. The first one is to expand the analysis to

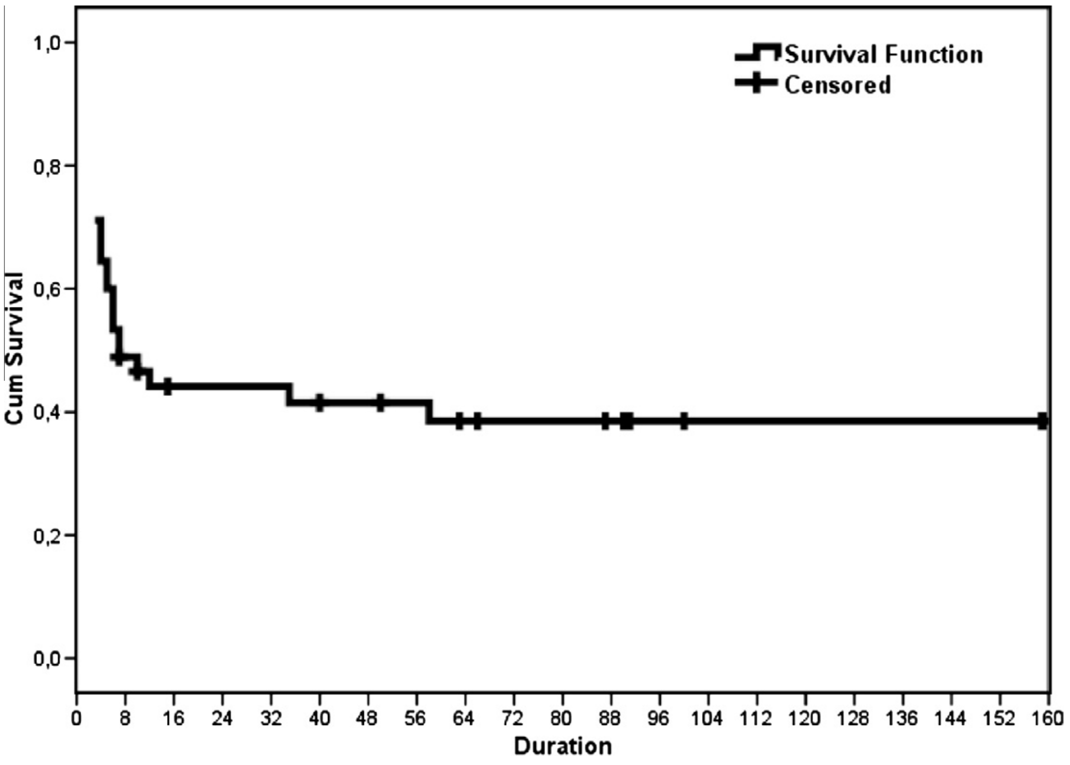

Fig. 17. Kaplan–Meier estimation of the survival function of database projects. Only active projects are considered as censored cases.

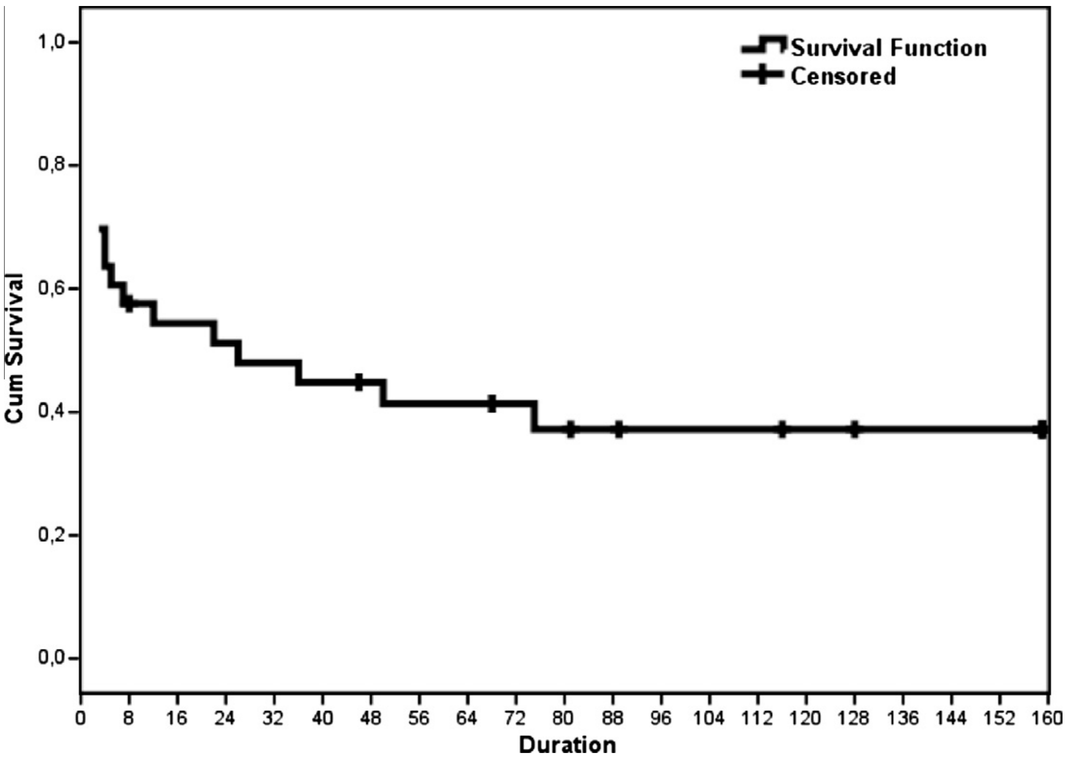

Fig. 18. Kaplan–Meier estimation of the survival function of desktop projects. Only active projects are considered as censored cases.

more projects, pursuing a bigger sample of FLOSS projects. Different categorization of projects may produce more interesting results across diverse application domains. In addition, the effects of more project parameters, such as programming language should be examined. This is not a trivial exercise since typically more than on language is used in each project, and altering the relative weight of different languages in the code over the life span of a project may add further validity threats to our study. Other parameters that worth examination are mailing lists activity and bug reporting, data already available in the FLOSSMETRICS database. Apart from adding more parameters to our model, a further examination of the conditions under a project will be considered a failure, is also

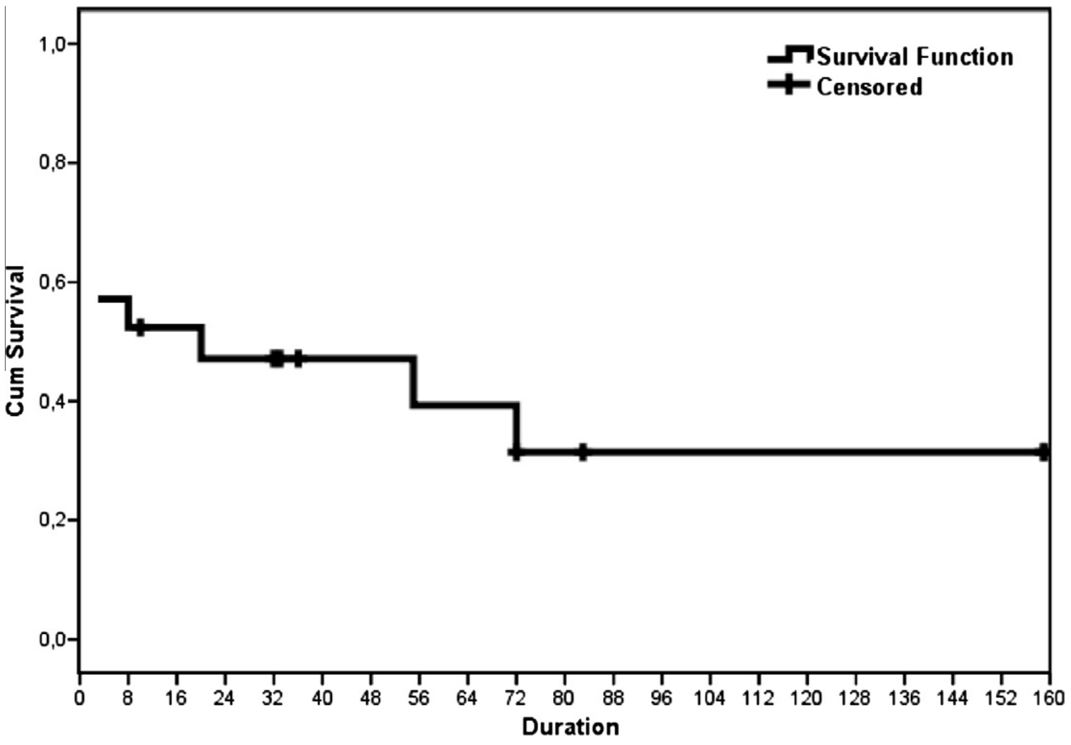

Fig. 19. Kaplan–Meier estimation of the survival function of education projects. Only active projects are considered as censored cases.

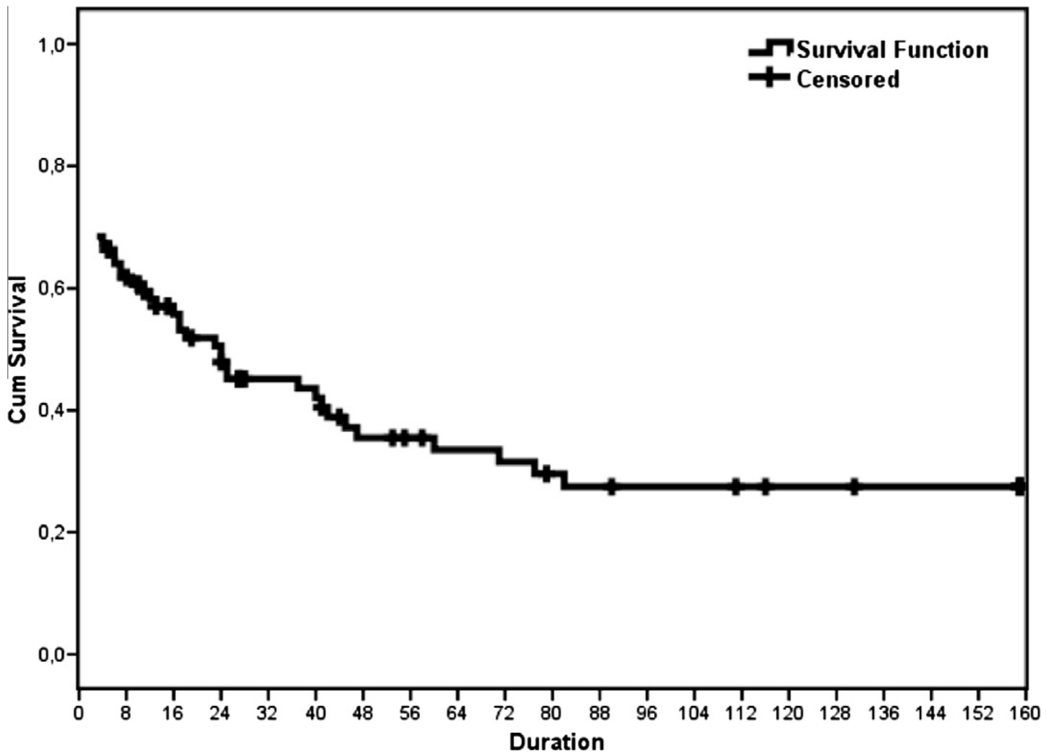

Fig. 20. Kaplan–Meier estimation of the survival function of games and entertainment projects. Only active projects are considered as censored cases.

included in our research agenda. This evolves conducting a sensitivity analysis on the number of commits that denote a failure and the inclusion of other parameters like mailing list archives and bug reports. Expanding the time slot of our research is in our future research agenda, too. Another interesting issue that deserves investigation is the existence of "sporadic" projects, i.e. projects stopping and then starting after some time. Their inclusion in the proposed models is a challenging problem.

Finally, we have assumed that a standard pattern of survivability exists in relation to the number of committers. The Cox

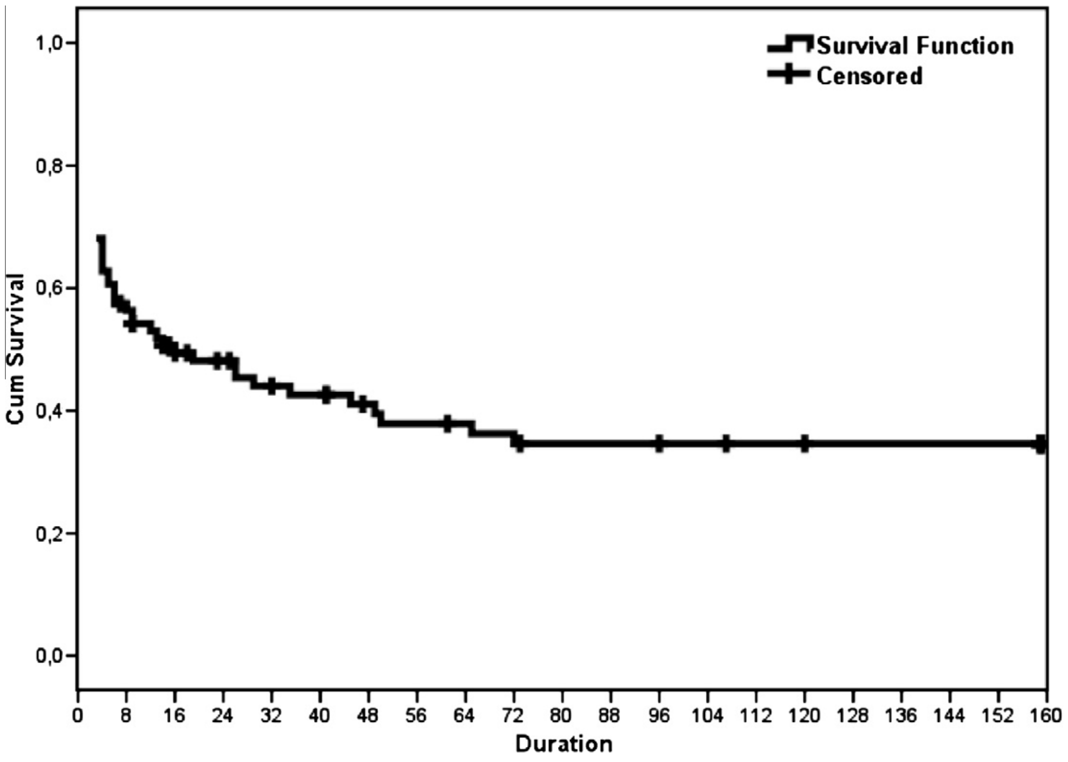

Fig. 21. Kaplan–Meier estimation of the survival function of internet projects. Only active projects are considered as censored cases.

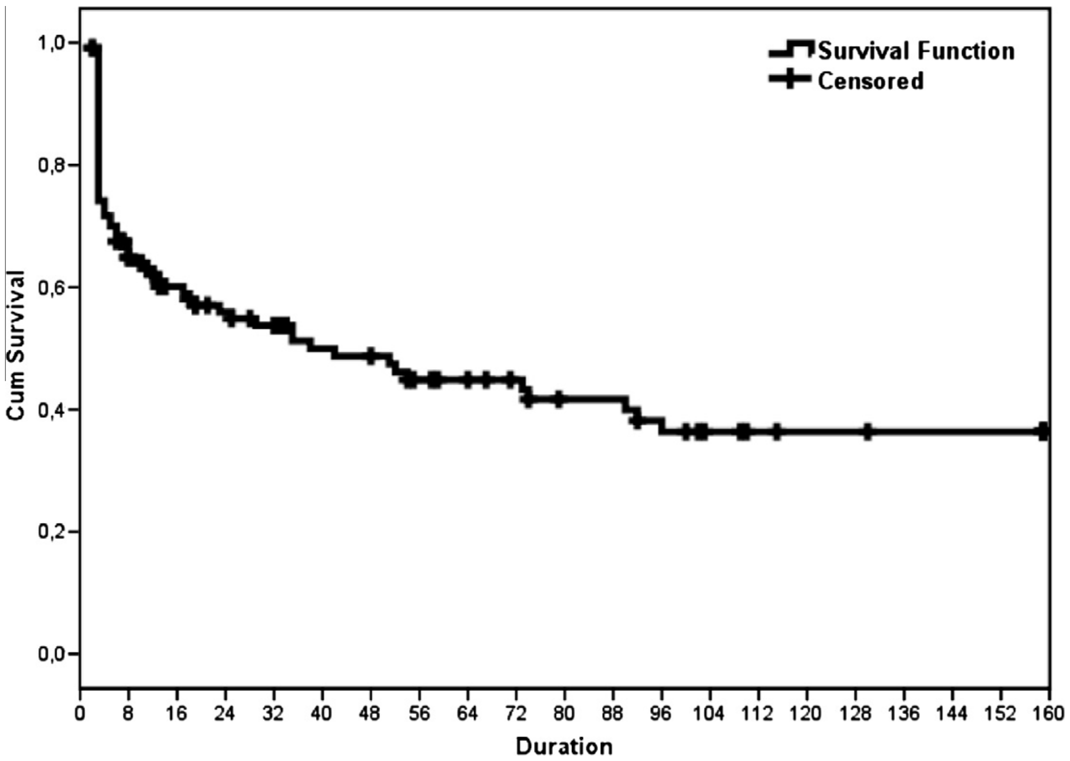

Fig. 22. Kaplan–Meier estimation of the survival function of multimedia projects. Only active projects are considered as censored cases.

regression is capable to incorporate any available information in the form of variables, either categorical or numerical. It would be interesting to study pattern variations across projects of different scales and types.

The second issue to be addressed in the future is the progress, if any, of the projects characterized as "lost to follow up". This cross repository analysis will give us better insight about the survival probability of open source in general. Apart from the purposes of

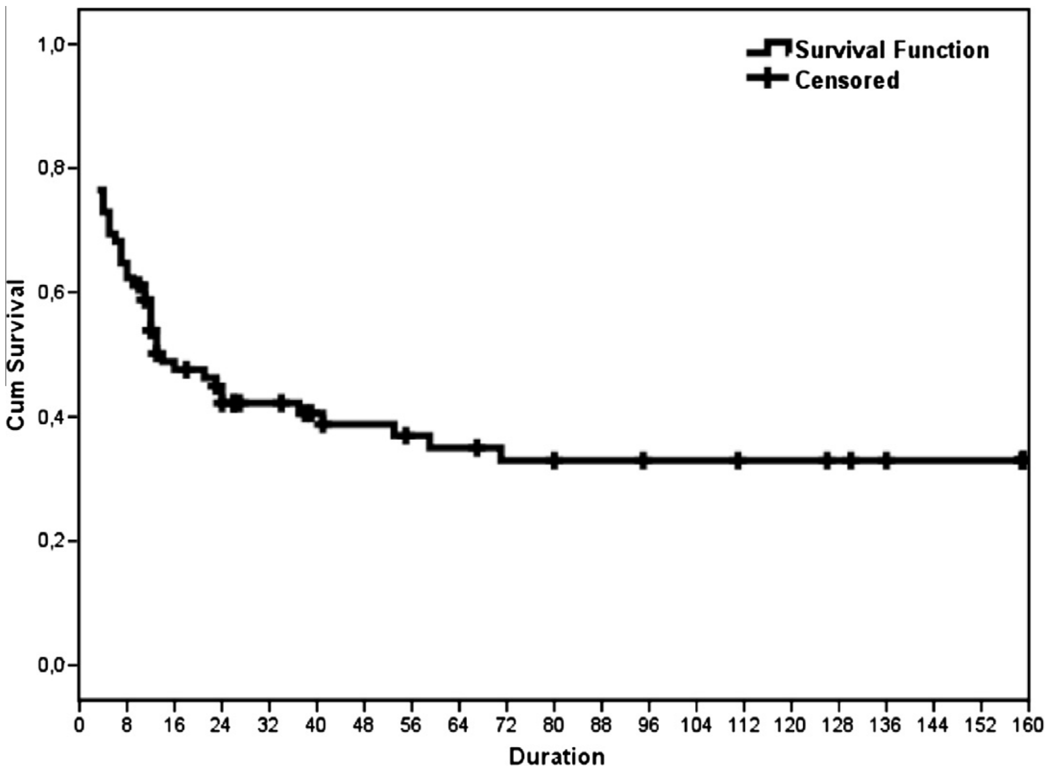

Fig. 23. Kaplan–Meier estimation of the survival function of office projects. Only active projects are considered as censored cases.

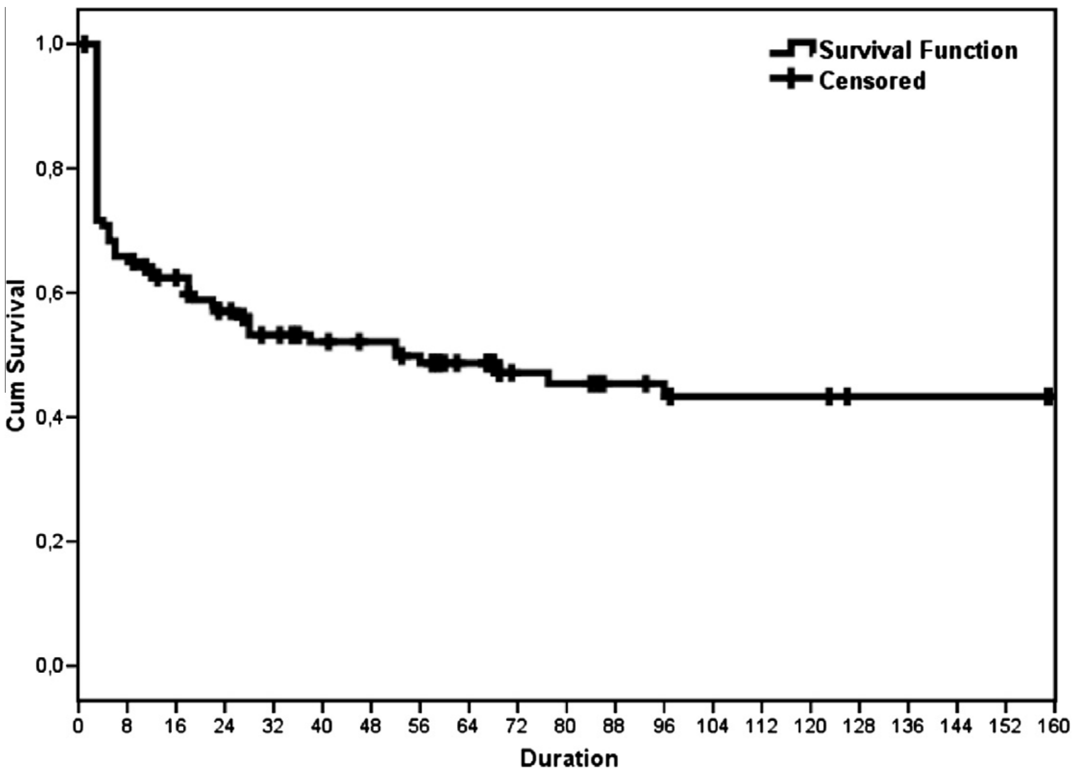

Fig. 24. Kaplan–Meier estimation of the survival function of science projects. Only active projects are considered as censored cases.

our study, cross repository analysis will enable us to understand better the evolutionary aspect of open source software, why projects change their infrastructure by moving to another host and if there are certain patterns on this.

Investigation of all these aspects of open source development would eventually lead us to model and predict the evolution of a project, allowing thus its coordinators and participants to act accordingly.

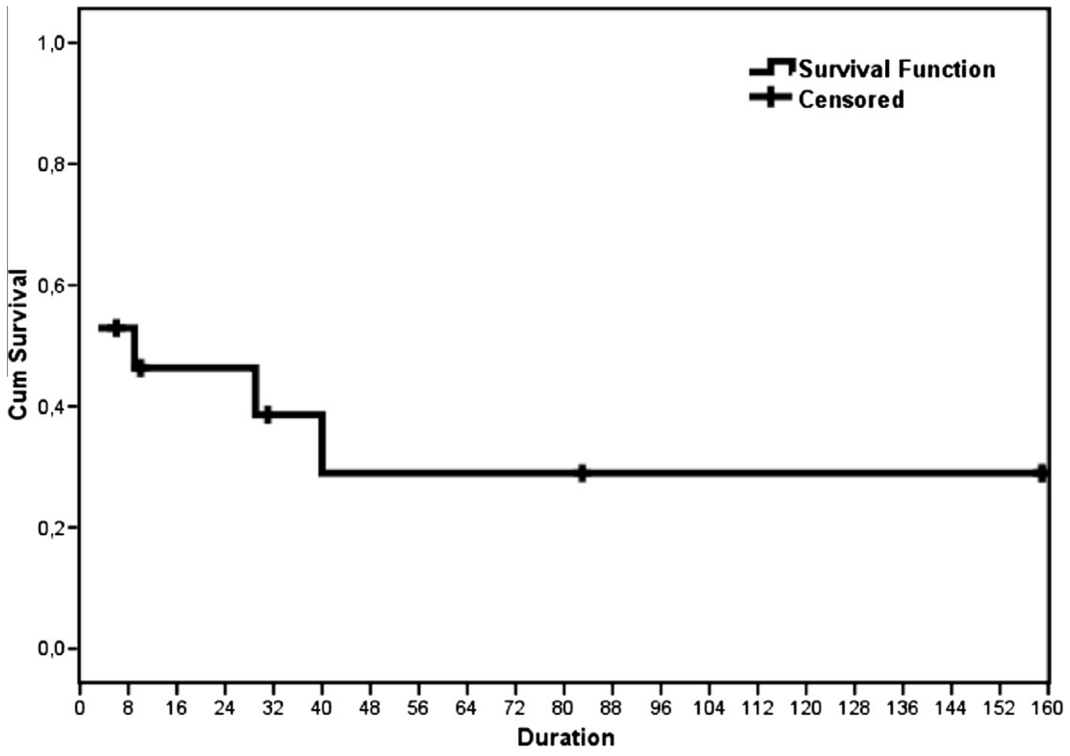

Fig. 25. Kaplan–Meier estimation of the survival function of security projects. Only active projects are considered as censored cases.

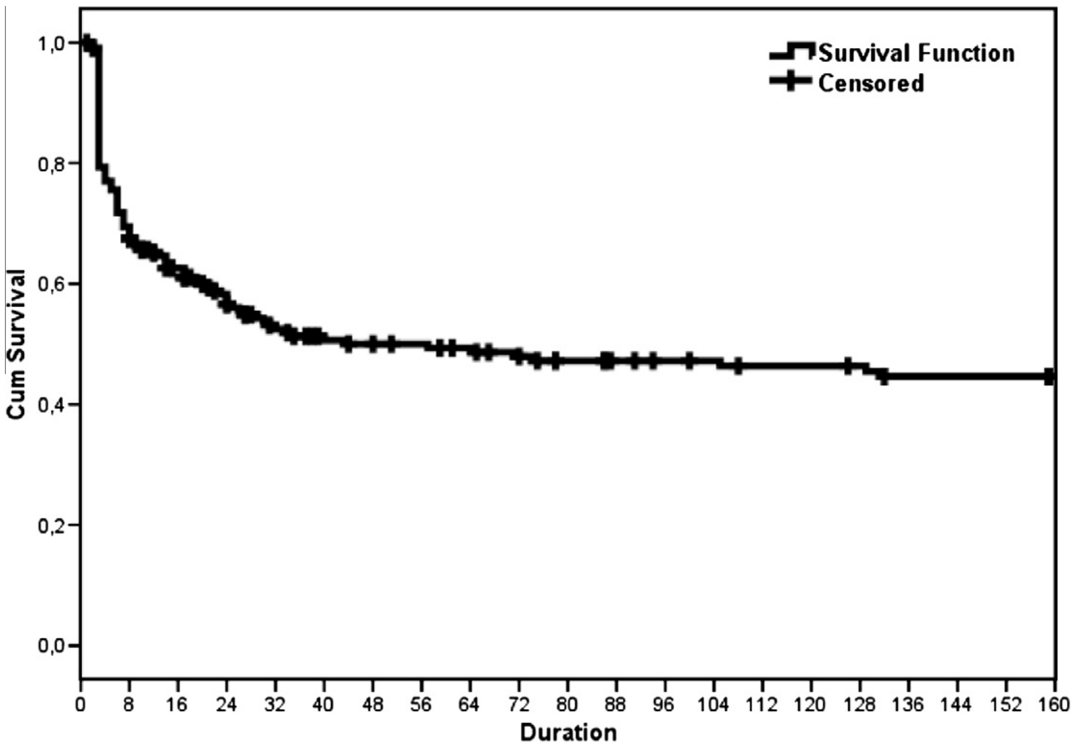

Fig. 26. Kaplan–Meier estimation of the survival function of software development projects. Only active projects are considered as censored cases.

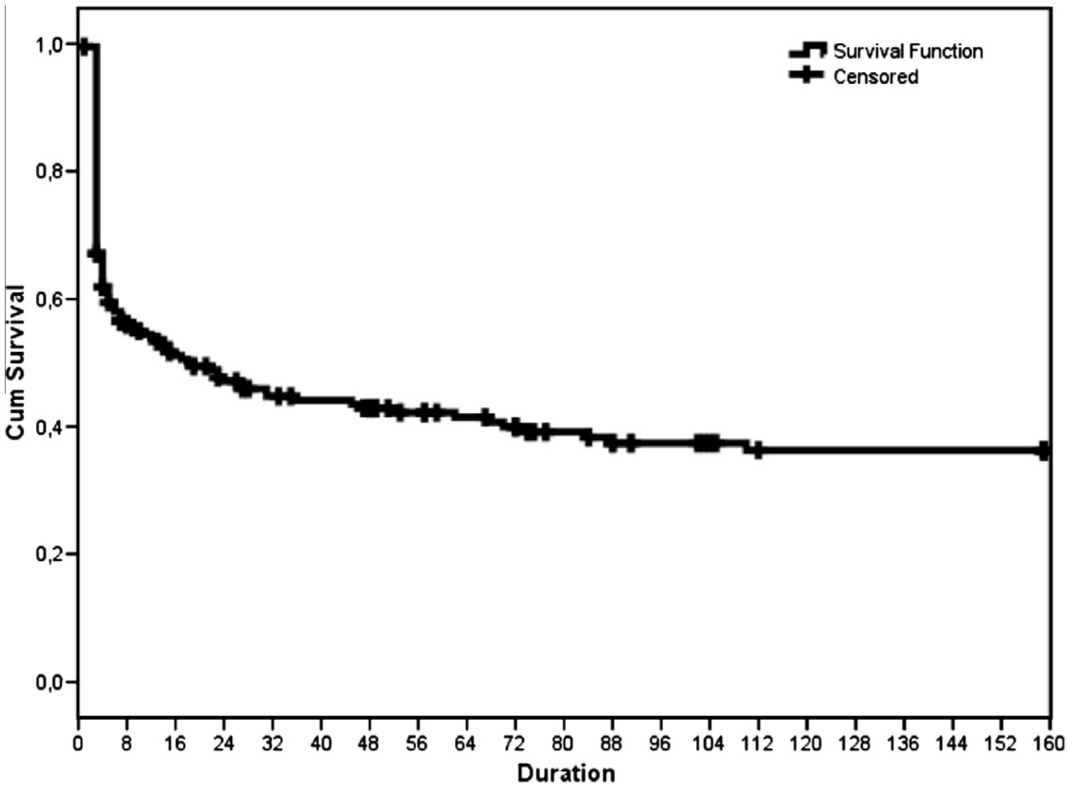

Fig. 27. Kaplan–Meier estimation of the survival function of system projects. Only active projects are considered as censored cases.

Acknowledgements

This work has been funded by the FLOSSMetrics project, funded by the European Commission via Project FP6-033982. We would like to thank the Libresoft group at Universidad Rey Juan Carlos, Madrid, Spain and our colleagues Eleni Konstantinou, Eleni – Maria Stea, George Kakarontzas, and Anastasia Deliga for their valuable help. Finally, we thank the editor and the reviewers for their fruitful comments which led us to improve the paper.

Appendix A

Figs. 16–27.

References

[1] L. Angelis, P. Sentas, Duration analysis of software projects, in: Proceedings of the 10th Panhellenic Conference on Informatics, 2005, pp. 258–269.
[2] K. Beecher, A. Capiluppi, C. Boldyreff, Identifying exogenous drivers and evolutionary stages in FLOSS projects, The Journal of Systems and Software 82 (2009) 739–750.
[3] A. Capiluppi, P. Adams, Reassessing Brooks' law for the free software community, in: Proc. IFIP 5th International Conference on Open Source Software Systems (OSS2009).
[4] K. Crowston, H. Annabi, J. Howison, Defining open source software project success, in: Proceedings of ICIS 2003, Seattle, WA, 14–17 December.
[5] K. Crowston, J. Howison, H. Annabi, Information systems success in free and open source software development: theory and measures, Software Process – Improvement and Practice 11 (2) (2006) 123–148.
[6] R. English, C.M. Schweik, Identifying success and abandonment of FLOSS commons: a classification of Sourceforge.net projects, Upgrade: The European Journal for the Informatics Professional VII (6) (2007).
[7] N. Evangelopoulos, A. Sidorova, S. Fotopoulos, I. Chengalur-Smith, Determining process death based on censored activity data, Communications in Statistics – Simulation and Computation 37 (2009) 1647–1662.
[8] D. German, A. Mockus, Automating the measurement of open source projects, in: ICSE '03 Workshop on Open Source Software Engineering, Portland, Oregon, May 3–10, 2003.
[9] I. Hann, J. Robert, A. Slaughter, Why developers participate in open source software projects: an empirical investigation, in: Proceedings of 25th International Conference on Information Systems, 2004, pp. 821–830.
[10] I. Herraiz, J. Gonzalez-Barahona, G. Robles, Forecasting the number of changes in Eclipse using time series analysis, in: 29th International Conference on Software Engineering Workshops (ICSEW'07).
[11] D.W. Hosmer, S. Lemeshow, Regression Modelling of Time to Event Data, Wiley, 1999.
[12] J. Howison, K. Crowston, The perils and pitfalls of mining SourceForge, in: Proc. of Workshop on Mining Software Repositories at the International Conference on Software Engineering, ICSE, 2004.
[13] E.L. Kaplan, P. Meir, Nonparametric estimation for incomplete observations, Journal of the American Statistical Association 53 (1958) 457–481.
[14] G. Koru, D. Zhang, H. Liu, Modelling the effect of size on defect proneness for open source software, in: 2nd Workshop on Predictor Models in Software Engineering (PROMISE 2007).
[15] G. Von Krong, S. Spaeth, K. Lakhani, Community, joining, and specialization in open source software innovation: a case study, Research Policy 32 (2003) 1217–1241.
[16] E.T. Lee, J.W. Wang, Statistical Methods for Survival Data Analysis, third ed., Wiley, 2003.
[17] J. Lerner, J. Tirole, Some simple economics of open source, Journal of Industrial Economics 52 (2002).
[18] F. Ortega, D. Izquerdo-Cortazar, Survival analysis in open development projects, in: Proceedings of the 2009 ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development.
[19] M.K.B. Parmar, D. Machin, Survival Analysis. A Practical Approach, Wiley, 1995.
[20] A. Rainer, S. Gale, Evaluating the quality and quantity of data on open source software projects, in: The First International Conference on Open Source Systems, Genova, Italy, July 11–15, 2005.
[21] C.M. Schweik, R. English, M. Kitsing, H. Sandra, Brooks' versus Linus' law: an empirical test of open source projects, in: The Proceedings of 9th International Digital Government Research Conference, 2008.
[22] P. Sentas, L. Angelis, Survival analysis for the duration of software projects, in: Proceedings of the 11th IEEE International Software Metrics Symposium, Como, Italy, 2005.
[23] P. Sentas, L. Angelis, I. Stamelos, A statistical framework for analyzing the duration of software projects, Empirical Software Engineering 13 (2008) 147–187.
[24] I. Stamelos, Teaching software engineering with free/libre open source projects, International Journal of Open Source Software and Processes 1 (2009) 72–90.
[25] C. Subramaniam, R. Sen, M. Nelson, Determinants of open source software project success: a longitudinal study, Decision Support Systems 46 (2009) 576–585.
[26] D. Weiss, Measuring success of open source projects using web search engines, in: M. Scotto, G. Succi (Eds.), Proceedings of the First International Conference on Open Source Systems (OSS2005), Genova, Italy, pp 93–99.