Evidences in the evolution of OS projects through Changelog Analyses

Andrea Capiluppi
andrea.capiluppi@polito.it

Patricia Lago
patricia.lago@polito.it

Maurizio Morisio
maurizio.morisio@polito.it

Dip. Automatica e Informatica
Politecnico di Torino
Italy

ABSTRACT

Most empirical studies about Open Source (OS) projects or products are vertical and usually deal with the flagship, successful projects. There is a substantial lack of horizontal studies to shed light on the whole population of projects, including failures. This paper presents a horizontal study aimed at characterizing OS projects.

We analyze a sample of around 400 projects from a popular OS project repository. Each project is characterized by a number of attributes. We analyze these attributes statically and over time.

The main results show that few projects are capable of attracting a meaningful community of developers. The majority of projects is made by few (in many cases one) person with a very slow pace of evolution.

We then try to observe how many projects count on a substantial number of developers, and analyze those projects more deeply. The goal is to achieve a better insight in the dynamics of open source development.

The initial results of this analysis, especially growth in code size and tendency to stability in modularity, seem to be in line with traditional close source development.

1. INTRODUCTION

The Open Source (OS) model of software development has gained the attention of both the business, the practitioners and the research communities. The OS process has been described by the seminal paper by Eric Raymond 4 and . However, sound empirical studies are still very limited in number and mostly vertical, i.e. they deal with a single, high impact project [3], [6], [9] and [10]

On the other hand, few are the preliminary horizontal studies that have been performed on major OS repositories, like[9] and [14],but still they remain on the surface of the data calculated by the administrators of the site themselves. In these cases, those data are parsed from the HTML pages and used to perform descriptive statistics. Our study uses a similar horizontal approach, but goes deeper into data collection: besides readily available metrics that can be computed automatically, other project attributes are extrapolated or computed by hand. Further, we consider evolution of the projects. The measures are computed at a point in time, then repeated some time later. Our long term objective is to gain further understanding about OS project dynamics, and also to draw useful lessons for software development in general. It should be noted that the OS process provides *open* process and product data, and therefore is a rare opportunity for empirical research.

As an example of an open process-oriented issue, the literature studies the evolution of traditional (non OS) projects. As a result, evolution is organized in a significant number of releases in a short time, and this is usually considered an instability factor [16], and [17]. On the contrary, in the OS community this type of evolution is an evidence of vitality showing the commitment of the authors, and the level of appreciation from users [18]

Koch and Schneider [9] study the GNOME project, especially at the level of size and programmers. Size and number of programmers grow steadily, the study confirms that an 'inner circle' of programmers gives most of the contributions.

Among horizontal studies we are aware of the FLOSS project, which is analyzing this topic from an Economic as well as technological point of view [2].

Stamelos et al. [11] analyze the quality of source code of five open source projects and conclude that the quality level is not different from closed source projects.

In this study we concentrate on a very large sample (406 projects) selected randomly from an OS portal [20], and give some descriptive statistics and an initial analysis of evolution. The evolution aspect is considered since we observe the attributes of the projects twice, with an interval time of 6 months.

The vast majority of projects are 'solo' works (small size, one developer and no users) and belong to horizontal application domains (software used to produce other software, such as operating system components, data bases etc.). Many do not evolve (no change in version number, no change in size) for months, suggesting that they are 'dead'. Few projects are 'alive' (several developers, growth in size and developers). This suggests that, also in the OS

community, the competition for attracting developers is harsh, successful projects are a minority and mortality is high. Examples from flagship projects like Linux and Apache should not be taken as the rule for all OS projects.

Next, we observe 12 'alive' projects from the initial set of 406. We reformulate some attributes defined for the first phase (modularity is defined, and modularity level is discarded), while some others are no more considered. We analyze the evolution of these projects in more detail, through a smaller set of project attributes (modularity, size, developers) computed at each release. The goal is to understand if there are common trends, and if there are key differences from closed source development.

We define three clusters of projects: 'large' projects as long as they are based on more than 1000 KB (40 KLOC), 'medium' projects (between 100 and 1000 KB), and 'small' projects (up to 100 KB).

First results demonstrate that there's substantially small differences between close and open software development.

2. DEFINITIONS

We formulate several project attributes. Here we report their definitions only: in we show the values of the attributes for the 12 chosen projects

2.1 Age

Age of the project (evaluated in days). As a proxy we use the date of first posting of the project on the portal, defined by portal owners. We calculated the time buckets based on that.

2.2 Application domain

Main domain covered by the application. As examples, Scientific, Security, Database, etc.Their definition is up to the portal owners.

2.3 Programming language

Programming language used to develop the application. As examples, C, C++, Java, etc. They are selected by the project and reported by the portal owners.

2.4 Code Size

Size of the source code of an instance of the product developed by the project. We analyzed the projects in order to purge the code from auxiliary files (html, documentation, images, etc.) and legacy code (developed by somebody else and included as library in the code). For the sake of automation, we formulate sizes in Kbytes.

2.5 Number of developers, stable developers, transient developers

A developer is a person who contributes isolated code patches, as well as a continued contribution of code. Bug reporting or contribution of ideas are not considered as development. Developers are further divided into transient and stable. We therefore define as stable those developers submitting more than one isolated patch: this definition holds as long as patches are submitted in different versions, or through different modules.

2.6 Number of users

A user uses the application developed by the project. The number of subscribers to a project is used as a proxy of number of users. This metric is calculated by the portal owner and it's publicly available.

2.7 Modularity level

Degree of modularity of the source code. As a proxy of modularity we use the number of directories the source code is divided into. The possible values of this attribute are one directory (dirLev 1), two (dirLev 2), more than two (dirLev 3).

2.8 Modularity

When analyzing the evolution of the 12 chosen projects, we refine the above definition as follows: modularity is the number of modules (as a proxy we use the number of directories the source code is split into). We further define the average size of module (= code size/number of modules).

2.9 Documentation level

Level of documentation of a project (source code, APIs). We define three documentation levels: comments in the source code (docLev 1), a README file or a Unix-like MAN page (docLev 2), availability of a user manual or API specification (docLev 3).

2.10 Popularity

Defined by portal owners as follows:

$$P = \sqrt[3]{\frac{U * R * (S+1)}{3}}$$

- U stands for the count of visits to the project home page
- R is the number of visits to the project on FreshMeat pages
- S is the subscribers number

2.11 Status

Stage of development of a project. This attribute is coded in six different values (Planning, Pre-alpha, Alpha, Beta, Stable, Mature), and it's value is selected by the project and reported by the portal owners.

2.12 Vitality

Defined by portal owners as follows:

$$V = \frac{R * A}{L}$$

- R is the number of releases in a certain period (t)
- A is the age of project (in days)
- L stands for the number of releases in t

2.13 Version Number

Identifier of an instance of the product developed by the project, in the form 1.1, 1.2 and similar. They're selected

by the project owner. Rather than by the absolute number, we're interested in analyzing the relative changes in version numbers and the rate of their changing.

2.14 Date of version

Date a version was released. This measure is used to analyze the rate of activity per period (month, age, and so forth).

3. HORIZONTAL ANALYSIS

Size of projects is typically small (Figure 1), GPL the most popular license. Horizontal application domains (applications used to build other software, the end user is required to program and is, likely, a software professional) prevail (66%).

Most used languages are C, C++, PERL. Surprisingly, Java comes after those.

Number of developers is typically low: 57% have one or two developers. Only 15% of them have more than 10 developers. We believe for the latter category only the issue of chaotic vs. organized development becomes meaningful. For this category, a core team of coordinators exists, and the ratio of coordinators to developers is, on average, 1 to 4 (Figure 3 and Figure 4).

Number of subscribers (a proxy of users of the application developed by a project) is also low. 80% of projects have less than 11 subscribers, 1% has more than 100. Surprisingly, the number of subscribers does not appear to grow with the age of a project, nor with its size. 72% of projects with more than 10 subscribers belong to a horizontal domain. To us, this confirms that successful OS software is developed by experts for experts.

As for evolution of projects, another striking fact is that, over six months, 97% of projects did not change size or changed less than 1% (Figure 2).

These data suggest that, despite the huge number of OS projects listed on OS portals, the overall effort put in OS projects (and the pool of developers) is a scarce resource that concentrates on very few projects. Very successful OS projects such as Linux and Apache are probably not the 'average' OS project.

While some validity threats must be considered to interpret these results correctly (especially the use of portals as an advertising means that inflates the number of single developer projects with no chance of success), we believe that this analysis can bring useful insight in the debate about the Open Source movement.

4. VERTICAL ANALYSIS

An initial observation of the projects in our sample suggests that they cluster around both the number of authors and size (Figure 5): two projects have more than 15 authors, six have between 2 and 4, and four have one author.

In all projects we have analyzed there are some common behaviors: size grows, authors and contributors grow over time (Figure 6, Figure 7 and Figure 8, where we report the evolution of two projects only, ARLA and MUTT, which we consider as large projects). Both behaviors indicate that a project is alive, with a community of developers that work on it and let it evolve. This finding is not surprising, since we selected the projects with the aim of isolating especially alive projects. However, the result was not warranted, since we selected projects with a static indicator like total number of developers > 10. In this sense we can say that a large number of developers (authors + contributors) may be a good predictor of alive projects.

A constant growing size may be an implication of Lehman's and Belady laws on software evolution [6], [7], and [12]. In this sense we can hypothesize that Lehman's laws apply to alive OS projects, or, in other words, OS projects could behave in the same way as closed source ones under this point of view.

On the other hand a growing number of contributors, and (for large and medium projects) of authors is a typical OS characteristics, usually unmatched by closed source endeavors. Contributors grow with a linear-wise trend. (Figure 6 and 8) While there is no warranty that contributors always grow, a growing pattern indicates a healthy project. Knowing why certain projects attract contributors (alive projects) and others don't would be extremely beneficial. Unfortunately the data we have does not allow us to answer to this question. Similarly we are in no position to tell if and when the evolution of a project stops, and in case it stops if this is due to a status of maturity achieved or to end of interest from the community (Figure 7).

In large and medium projects authors grow too, but the growth rate is much more limited. New authors were always contributors, contributing to a project seems to be the preferred way to access the core group. The number of authors is always limited, in medium and large projects the limit is set by an overall reduced number of contributors, in large projects with a very large number of contributors (such as Mutt) the ceiling could be set by organizational issues. In Mutt the ceiling is at around 20 authors (Figure 6).

In large and medium projects the number of modules grows, probably due to the parallel code growth and internal reorganizations. An intriguing observation is that, while the size of modules changes considerably from project to project, in all of them it tends to evolve to a stable value (Figure 7).

In large and medium projects parallel distributions (i.e. parallel versions of the product, with enhanced or limited functionality or with variations in internal design) are sometimes used, and eventually merge in a single version. We are not aware if a similar behavior happens in closed source too. In small projects the number of modules tends to be one, the issue of reorganizations and redesign is probably premature for the state of the project.

5.REFERENCES

[1] Capiluppi A., Lago P., Morisio M., 2002, “Characteristics of Open Source Projects”, on the Proceedings of the 7th European Conference on Software Maintenance and Reengineering, March 2003

[2] FLOSS (Free/Libre and Open Source Software), http://www.infonomics.nl/FLOSS/outline.htm

[3] Mockus, R.T. Fielding, J.D. Herbsleb (2002): “Two Case Studies of Open Source Development: Apache and Mozilla”. ACM Transactions on Software Engineering and Methodology Vol.11, No. 3, 2002, 309-346.

[4] B. Behlendorf, 1998, “Open Source as a Business Strategy” contained in DiBona C., Ockman S., Stone M., “Open Source: Voices From the Open Source Revolution”. Sebastopol, California. O’Reilly & Associates, and available on-line at http://www.oreilly.com/catalog/opensources/book/brian.html.

[5] Raymond, E., “The Cathedral and the Bazaar”, FirstMonday, Vol.3 N.3 - March 2nd. 1998, on line at http://www.tuxedo.org/~esr/writings/cathedral-bazaar/

[6] Lehman MM, Perry DE, Ramil JF, Turski WM and Wernick PD, Metrics and Laws of Software Evolution – The Nineties View, Proc. Metrics '97, IEEE - CS Albuquerque, NM, 5 - 7 November 1997, pp. 20 – 32

[7] Belady LA and Lehman MM, "An Introduction to Program Growth Dynamics", in *Statistical Computer Performance Evaluation*, W. Freiburger (ed.), Academic Press, NY, 1972, pp. 503 - 511

[8] Raymond, E., “Homesteading the Noosphere”, FirstMonday, Vol. 3 N. 10 - October 5th. 1998.

[9] Stefan Koch, Georg Schneider, Results from Software Engineering Research into Open Source Development Projects Using Public Data, in Diskussionspapiere zum Tätigkeitsfeld Informationsverarbeitung und Informationswirtschaft, Hans R. Hansen und Wolfgang H. Janko (Hrsg.), Nr. 22, Wirtschaftsuniversität Wien, 2000.

[10] Mockus A., Fielding R., Herbsleb J., Two Case Studies of Open Source Software Development: Apache and Mozilla, ACM Trans. On Software Engineering and Methodology, 11, 3, (2002), 309-346.

[11] Stamelos, I., Angelis, L., Oikonomou, A., Bleris, G.L., ‘Code Quality Analysis in Open-Source Software Development’, Information Systems Journal, 2nd Special Issue on OS Software, 12(1), January 2002, pp. 43-60.

[12] Lehman M.M., Ramil J.F.,, 2002, "Software Evolution and Software Evolution Processes", Annals Of Software Engineering, Vol. 14, pp. 275-309

[13] Fenton A., (1994), “Software Measurement: A Nec-essary Scientific Basis”. IEEE Transactions on Software Engineering, Vol.20, No.3, Pagg. 199-206

[14] Kienzle, R., 2001, “Sourceforge Preliminary Project Analysis”, available on-line at http://www.osstrategy.com/sfreport/

[15] Fuggetta A., (2002), OS software: an evaluation, web.cefriel.it/~alfonso/documents/ Papers/opensource.pdf

[16] Bezroukov, N., “OS Software Development as a Spe-cial Type of Academic Research”, FirstMonday, Vol. 4 N. 10 - October 4th. 1999.

[17] Bezroukov, N., “A Second Look at the Cathedral and the Bazaar”, FirstMonday, Vol. 4 N. 12 - December 6th 1999.

[18] Tirole J., Lerner J., “Some Simple Economics of OS”, Journal of Industrial Economics, N. 52, July 2001.

[19] Kienzle, R., 2001, “Sourceforge Preliminary Project Analysis”, available on-line at http://www.osstrategy.com/sfreport/

[20] FreshMeat portal, http://freshmeat.net

Table 1 – Attribute values for the 12 chosen projects

Name	Function	Size [Kbytes]	Modules	Average module size [KBytes]	Authors	Contributors
Arla	cache manager for the AFS file system	4290	71	105	16	29
Gnuparted	Manipulates logical disks partition	927	23	40.3	4	10
Mutt	e-mail client	2134	4	533.5	23	101
Weasel	Reads electronic contents on a palm pilot	482	2	241	3	10
Bubblemon	displays system's load with a graphical interface	66.1	2	33	2	9
Calamaris	gets statistical data out of parsed documents	111	1	111	1	16
Dailystrips	searches cartoon strips over the web	42	1	42	1	10
disc-cover	Searches disc covers over the web	83	2	41.5	2	16
Edna	Mp3 file server	52.4	4	13.1	1	12
Motion	detects motion in a video device	160	1	160	3	24
Rblcheck	Email monitoring and protection	20	6	3.33	1	9
Xautolock	Monitors console activity	70	2	35	2	14

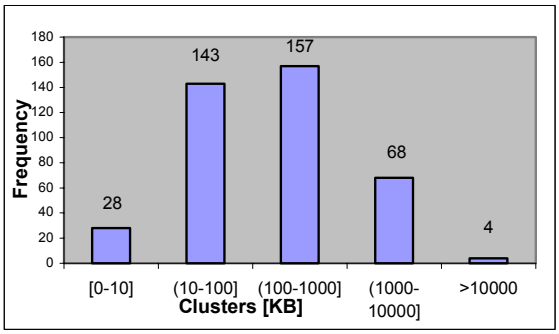

Figure 1 - Distribution of Code Sizes

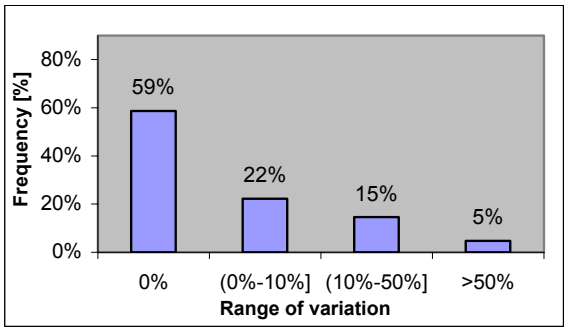

Figure 2 - Dynamic Variation of Code Sizes

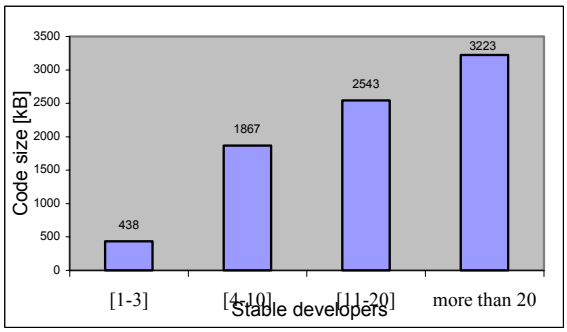

Figure 3 - Distribution of stable developers over projects

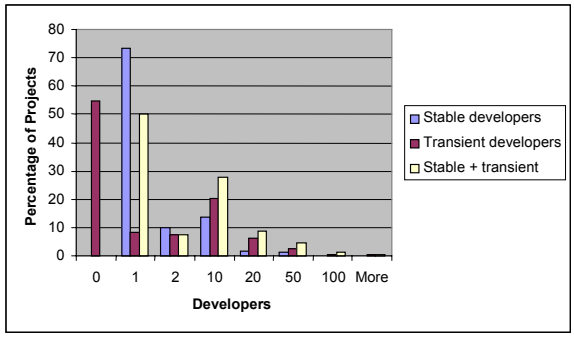

Figure 4 - Distribution of developers over projects

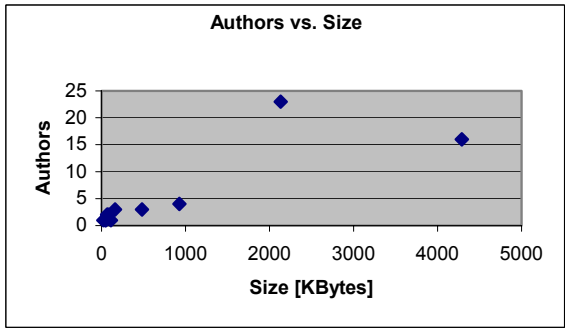

Figure 5 - Scatterplot of authors vs. code size (KB)

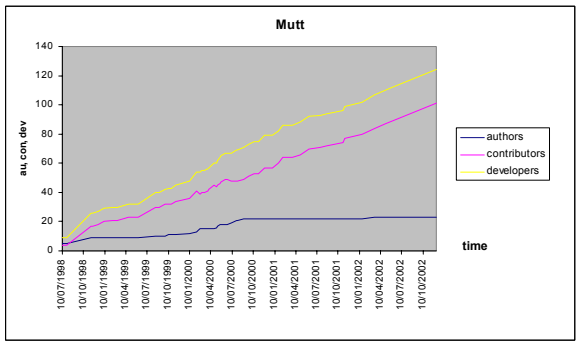

Figure 6 - Authors, Contributors and Developers in the MUTT project

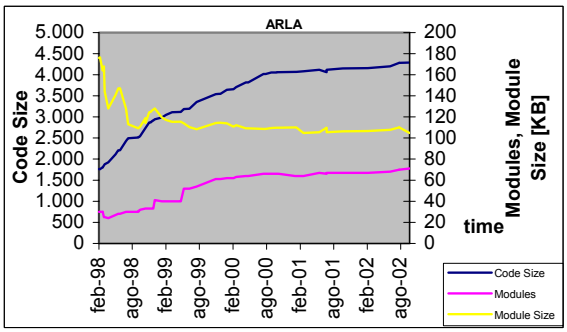

Figure 7 - Code Size, Modules and Module Size in the Arla project

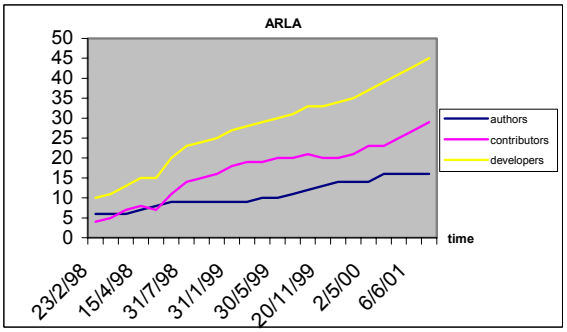

Figure 8 - Authors, Contributors and Developers in the ARLA project