Electronic Communications of the EASST
Volume 8 (2008)

Proceedings of the Third International ERCIM Symposium on Software Evolution (Software Evolution 2007)

Evolutionary Success of Open Source Software: an Investigation into Exogenous Drivers

Karl Beecher, Cornelia Boldyreff, Andrea Capiluppi and Stephen Rank

14 pages

Guest Editors: Tom Mens, Ellen Van Paesschen, Kim Mens, Maja D'Hondt
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Evolutionary Success of Open Source Software: an Investigation into Exogenous Drivers

Karl Beecher, Cornelia Boldyreff, Andrea Capiluppi and Stephen Rank[1]

[1] (kbeecher,cboldyreff,acapiluppi,srank)@lincoln.ac.uk
Centre of Research on Open Source Software – CROSS
Department of Computing and Informatics
University of Lincoln, UK

Abstract: The "success" of a Free/Libre/Open Source Software (FLOSS) project has often been evaluated through the number of commits made to its configuration management system, number of developers and number of users. Based on SourceForge, most studies have concluded that the vast majority of projects are failures.

This paper argues that the relative success of a FLOSS project can depend also on the chosen forge and distribution. Given a random sample of 50 projects contained within a popular FLOSS forge (Debian, which is the basis of the successful Debian distribution), we compare these with a similar sample from SourceForge, using product and process metrics, such as size achieved and number of developers involved.

The results show firstly that, depending on the forge of FLOSS projects, researchers can draw different conclusions regarding what constitutes a successful FLOSS project. Secondly, the projects included in the Debian distribution benefit, on average, from more evolutionary activity and more developers than the comparable projects on SourceForge. Finally, the Debian projects start to benefit from more activity and more developers from the point at which they join this distribution.

Keywords: FLOSS, repositories, metrics, success, evolvability

1 Introduction

In terms of Lehman's first law of software evolution, it can be anticipated that a useful and widely used real-world software system, known as an evolutionary (or *E-type*) software system, must undergo continuing change, *i.e.* that it must evolve [LRW+97]. Some well-known Open Source projects, such as the so-called LAMP (Linux, Apache, MySQL, Perl), the Debian family, and *BSDs, have achieved higher evolvability than others [MFH02]; these systems are categorised as E-type. Their evolvability is made possible through these projects attracting a large community of users as well as a strong base of developers. The user community initiates the need for change while the developers make it happen; both are key factors in the evolution process.

The term "success" of FLOSS projects has been often empirically evaluated via endogenous characteristics, such as the amount of development activity, the number of developers, or by using proxies of their pool of users. Moreover, FLOSS literature has traditionally tackled this research topic by sampling well-known FLOSS forges (mostly SourceForge), and concluding

that the vast majority are "unsuccessful", or "dead" projects [ES07]. Perhaps this is because they are too specialised in their functionality, but also it could be that through lack of publicity, they have never achieved the wide spread usage, nor attracted the developers that would drive their evolution. Using the literature terms, those projects never gained a 'bazaar' state, where users join in a self-sustaining cycle and become developers on the project [SM04, RG06, CM07].

Unless an open source evolutionary software project has enough developers to satisfy its users' needs for change, it is likely to fail. An interesting case arises when one open source project becomes incorporated into another, larger one; this is often found in the Open Source operating system projects, such as Debian, *e.g.* in the case of new packages. In such a case, the incorporated package project becomes as widely-distributed as the incorporating project and potentially is able to reach the same user base and benefit from the developer base of the incorporating project.

This paper investigates the exogenous drivers of FLOSS evolvability, and studies whether the inclusion of a specific project in the same forge and distribution of a successful FLOSS project (Debian) has an influence on its evolutionary characteristics. In order to understand the influence of these drivers, we randomly sampled 50 projects from both forges: Debian and SourceForge, and studied their evolution. Our goal is to determine whether the visibility given by the inclusion into Debian increases the number of developers and their activity, compared to the SourceForge sample. Also, we studied the "entry point" of each project into the Debian forge and distribution, and evaluated its activity both prior to and after this event.

This paper is structured as follows: Section 2 reviews related work in the area of FLOSS characterisation and section 3 introduces the traditional Goal-Question-Metric approach [BCR94], as applied to the research topic of this paper. Two major questions will be introduced and later instantiated in several hypotheses in section 4. Section 4.1 empirically evaluates and tests the hypotheses derived from the first question, while section 4.2 presents the results for the second question and section 4.3 discusses threats to validity. Section 5 presents our conclusions.

2 Related work

There are two main types of FLOSS literature, tentatively termed *external* and *internal* to the FLOSS phenomenon: based on the availability of FLOSS data, the former has traditionally used FLOSS artefacts in order to propose models [HG05], test existing or new frameworks [CCP07, LHMI07], or build theories [ACPM01] to provide advances in software engineering.

The latter includes several other studies that have analysed the FLOSS phenomenon *per se* ([SAOB02, Cap03, Ger04] with their results aimed at both building a theory of FLOSS, and characterising the results and their validity specifically as inherent to this type of software and style of development. In this section we review some of the works of the latter category.

The success and failure of FLOSS projects has been extensively studied in the past. Specific forges have been analysed and metrics computed from data extracted from the forges themselves. Examples include the use of the *vitality* and *popularity* indices, computed by the SourceForge maintainers, which have been used to predict other factors of the same forges [SA02], or to compare of the status of the projects between two different observations [FFH+02]. Also data has been collected from SourceForge about community size, time to fix bugs, and the popularity of projects, and this has been used to review some popular measures for success in information sys-

tems to the FLOSS case [CAH03a]. Popularity of FLOSS projects has also been assessed using web-search engines [Wei05]. Other studies have observed projects from SourceForge, and from their release numbers, their activity or success within a sample has been inferred [CAH03b], while other researchers have sampled the whole SourceForge data space, and have concluded that the vast majority of FLOSS projects should be considered as failures [RG05]. Finally, other researchers have created 5 project categories for the overall SourceForge site, based on dynamic growth attributes, using the terms "success" and "tragedy" within the FLOSS development. Again, it has been shown that some 50% of the FLOSS projects should be considered as tragedies [ES07].

There are several tools and data sources which are used to analyse FLOSS projects. FLOSSmole[1] is a single point of access to data gathered from a number of FLOSS forges (*e.g.*, SourceForge, Freshmeat, Rubyforge). While FLOSSmole provides a simple querying tool, its main function is to act as a source of data for others to analyse. CVSAnaly[2] is a tool that can be used to measure and analyse large FLOSS projects [RKG04]. It is used in this research to determine such information as the number of commits and developers associated with a particular project.

3 Goal, Question, Metrics – GQM

The Goal-Question-Metric (GQM) method evaluates whether a goal has been reached by associating that goal with questions that explain it from an operational point of view and providing the basis for applying metrics to answer these questions. The aim of the method is to determine the information and metrics needed to be able to draw conclusions on the achievement of the goal.

In the following, we applied the GQM method to first identify the overall goal of this research; we then formulate a number of questions related to the FLOSS projects and their success relative to their host forge and distribution to which they belong; and finally we collect adequate product and process metrics to determine whether the goal has been achieved.

Goal: The long-term objective of this research is to evaluate metrics to identify successful FLOSS projects, and to provide guidelines to FLOSS developers about practical actions to foster the successful evolution of their applications. Based on two samples from Debian and SourceForge, a comparison of their product and process characteristics will be evaluated to determine which sample should be considered more successful in terms of their evolution. This will also give an indication of the forges and distributions in which developers should include their projects so that they may achieve the best outcomes for their project's future development.

Question: The purpose of this study is to establish differences between samples of FLOSS projects extracted from Debian and SourceForge. Two sets of questions will be evaluated, one comparative and one internal to Debian: the first will deal with a direct comparison of the evolutionary characteristics achieved by the projects in the two samples, and the latter will study the projects in the Debian sample, and evaluate whether their evolution *after* being included in the distribution is different from that *before* this date. The date when a FLOSS project was inserted into the Debian distribution will be termed "entry point". The difference before and after the entry point will be evaluated by comparing the activity and number of developers in each phase.

[1] http://ossmole.sourceforge.net/
[2] http://cvsanaly.tigris.org/

As a summary the two main questions underlying this study can be formulated as follows:

1. Are projects in Debian statistically different from projects in SourceForge?
2. After being inserted into the Debian forge and distribution, do FLOSS projects leverage more activity and developers than before?

In section 4, the first question will be articulated in four research hypotheses, while the second question will lead to two further hypotheses.

Metrics: This study uses three sources of information to assess the above questions: the SourceForge and the Debian forges to select two random samples of projects; each project's own repository (either their CVS or SVN); and, among the projects within the Debian sample, their entry into Debian. Each of these sources has been analysed to obtain the metrics needed to perform the investigation. The metrics for the study will be introduced in each section below.

3.1 Debian and SourceForge samples

The Debian forge (http://www.debian.org/) hosts a large number of FLOSS projects under a common name. At the time of writing, more than 20,000 projects are listed under the "stable" label of the latest version. Using a randomiser, we selected 50 of these stable projects. A summary of the projects retrieved from Debian can be found in the first column of table 1.

The SourceForge site (http://sourceforge.net/) hosts more than 150,000 projects. In order to draw an accurate comparison, the sample from SourceForge has been extracted only from the pool of the "stable" projects, *i.e.* those projects whose core developers labelled the status of the project with the tag "Production/Stable". The number of projects from Debian and SourceForge in this category is comparable (around 22,000). A summary of the projects that have been chosen from the SourceForge site can be found in the first column of table 2.

3.2 Code repositories

The CVS/SVN repository of each project from the Debian or the SourceForge sample has been searched. In the sample of 50 Debian projects, 42 existing repositories were found. In order to provide a similar sample, 42 repositories were also selected from the SourceForge sample.

The following concepts and attributes have been used to build a table of results for each project:

Commit: the atomic action of a developer checking in one or more files (being source code or other) into a central repository.

Modules and subsystems: at a fine level of granularity, both CVS and SVN repositories record activity on files (here termed as "modules") and their containing folder (termed "subsystem").

Date: CVS/SVN repositories record the time when the module and its subsystem has been modified or created from scratch. A date with ISO formatting "YYYY-MM-DD" is recorded.

Developers: we record this information in two ways: firstly by assigning the activity to the actual committer who placed the file into the repository; secondly by using any further developers mentioned in the commit, as being in the coding or patching. This information is used to characterise the input provided to each project.

Touch: Since many modules and subsystems can be committed in the repository within the

project	oldest_date	entry_date	newest_date	days	touches	Dev.	SLOC
acpidump	2003-05-01	2005-09-26	2003-05-01	1	34	1	2349
apmud	2001-12-07	2000-05-23	2001-12-24	18	95	1	2502
clamav	2003-07-29	2002-05-09	2007-06-02	1405	5382	9	116731
dia	1998-10-01	1998-09-02	2007-06-07	3172	12828	126	146550
EtoileWildMenus	2006-03-04	**2006-10-03**	2007-04-16	409	46	3	1711
fte	2000-01-30	1996-12-25	2007-03-15	2602	1937	16	51498
geomview	2000-08-15	1998-08-02	2007-05-21	2471	7777	6	101844
grass6	1999-12-29	**2003-11-10**	2007-06-02	2713	42135	77	107648
gwenview	2006-06-20	2001-09-16	2007-06-06	352	449	5	4580
kdegames	1997-09-11	**1997-09-20**	2007-06-07	3557	19659	243	118479
kdenetwork	1997-11-26	1997-10-19	2007-06-06	3480	43130	818	272576
kmouth	2003-01-17	**2004-01-30**	2007-06-05	1601	647	31	5240
liboil	2004-01-07	**2004-11-04**	2007-05-29	1239	3106	4	52996
mimedecode	2006-06-19	1996-11-29	2006-06-19	1	16	1	631
mod_auth_kerb	2002-05-01	**2004-02-21**	2006-11-22	1667	349	2	119
myphpmoney	2002-11-20	**2003-01-15**	2007-05-27	1650	741	5	19434
octaveforge	2001-10-10	2001-02-25	2007-06-02	2062	16044	48	78150
Pike	1996-09-22	**2002-05-05**	2007-05-30	3903	21449	69	173196
prelude-manager	2001-08-23	**2002-04-11**	2007-05-02	2079	1557	44	10854
ProofGeneral	1996-03-15	**2002-09-03**	2007-05-25	4089	10425	20	48692
ruby	1998-01-16	**2003-08-23**	2007-06-05	3428	23968	143	419942
scid	2002-04-04	2001-02-13	2003-12-12	618	633	2	89402
shorewall	2002-05-01	2001-12-30	2007-06-06	1863	79498	4	25159
skel	2001-05-20	**2003-07-13**	2007-01-24	2076	219	13	120
sylpheed	2005-01-12	2000-09-30	2007-06-04	874	2719	2	106087
tcl	1998-03-26	1997-08-19	2007-05-30	3353	39124	109	165306
tdb	2000-08-14	**2001-05-07**	2005-08-02	1815	295	9	3261
tiobench	2000-03-23	**2000-11-08**	2003-12-22	1370	110	3	1689
txt2html	2007-01-15	2001-03-30	2007-05-10	116	4	1	3623
vlc	1999-08-08	**2000-03-13**	2007-06-06	2860	34736	113	401256
wxWidgets	1998-05-20	**2000-02-13**	2007-06-01	3300	246022	104	2142713
xmakemol	1998-04-03	**2001-10-31**	2006-09-23	3096	1386	4	18724
yaml4r	2002-06-22	**2003-08-23**	2003-04-24	307	498	1	10728
fig2ps	2005-11-16	2003-10-28	2007-02-19	461	105	1	397
syncekde	2003-02-11	**2003-08-15**	2006-11-26	1385	622	5	21684
noteedit	2004-09-15	2001-07-01	2005-07-30	319	590	4	63456
grub	1999-02-28	**1997-11-19**	2007-02-22	2917	5101	76	3536
libsoup	2000-12-06	**2003-03-19**	2007-06-01	2369	1548	42	15012
prcs1	2001-06-25	1997-03-28	2005-02-07	1324	858	5	37360
kphoneSI	2005-10-12	2002-12-20	2007-05-23	589	1630	1	41829
cdparanoia	1999-08-15	1998-05-16	2006-11-15	2650	297	6	9182
rlplot	2002-06-06	**2004-04-16**	2007-05-28	1818	1405	1	69493

Table 1: Summary of attributes of the Debian projects: in bold, the projects where there is a recorded evolution before and after the entry-point

project	oldest_date	newest_date	days	touches	developers	SLOC
audiobookcutter	2006-05-06	2007-05-22	381	958	2	4229
pf	2005-10-03	2006-08-29	330	3207	2	84489
seagull	2006-06-06	2007-05-31	359	707	5	62875
csUnit	2002-12-16	2006-08-14	1337	2147	1	16241
fitnesse	2005-03-26	2007-06-04	800	5172	12	39503
galeon	2000-07-06	2007-04-20	2479	10839	82	93374
expreval	2006-08-29	2007-04-11	225	282	1	3588
cdlite	2005-12-06	2007-04-12	492	46	1	1116
txt2xml	2002-04-27	2006-07-26	1551	155	3	1345
wxactivex	2005-01-26	2005-01-27	1	59	1	3264
ustl	2003-03-21	2007-03-31	1471	11470	1	11416
neocrypt	2003-05-23	2005-06-25	764	108	2	2135
cpia	2000-03-02	2004-10-17	1690	429	15	22954
moses	2002-05-07	2007-02-20	1750	5170	8	105955
critical_care	2002-01-18	2002-09-22	247	1708	5	38994
xmlnuke	2006-03-27	2007-03-07	345	888	2	57944
jtrac	2006-03-18	2007-06-06	445	1577	1	12771
QPolymer	2006-01-10	2007-05-24	499	459	1	86971
kasai	2004-08-31	2007-05-30	1002	673	3	8786
fourever	2005-02-23	2007-05-28	824	1795	2	15163
xqilla	2005-11-01	2007-05-28	573	8867	3	107320
uniportio	2006-05-29	2007-02-03	250	32	1	1096
genromfs	2002-01-18	2005-08-18	1308	94	3	654
Beobachter	2006-08-31	2006-12-10	101	376	1	2715
perpojo	2003-06-10	2003-07-31	51	70	1	1677
oliver	2004-07-22	2006-01-14	541	187	3	1429
hge	2005-11-18	2007-03-18	485	1183	3	45654
fnjavabot	2004-06-18	2007-06-05	1082	660	8	10142
ozone	2001-12-17	2005-12-12	1456	6108	7	63790
juel	2006-05-13	2007-04-25	347	990	1	7284
edict	2002-12-06	2006-12-28	1483	82	1	2556
Aquila	2004-05-04	2004-05-28	24	78	1	893
swtjasperviewer	2004-11-21	2007-05-21	911	188	1	3214
eas3pkg	2006-10-26	2007-05-22	208	274	2	43724
formproc	2001-05-10	2004-12-22	1322	1338	1	3514
toolchest	2002-01-03	2005-07-16	1290	15	1	494
ogce	2006-11-27	2007-06-03	188	26596	3	350997
simplexml	2002-08-23	2002-08-23	0	64	1	1691
intermezzo	2000-11-12	2003-09-30	1052	2276	15	34792
whiteboard	2003-06-15	2003-06-27	12	49	1	4910
modaspdotnet	2004-07-16	2007-03-02	959	688	1	2445
kpictorial	2002-05-09	2002-06-04	26	339	1	18214

Table 2: Summary of attributes of the SourceForge projects

same commit, and the same module could have been modified by more than one developer in the same commit, the term "touch" is used to isolate the atomic information of a unique date, unique union on module and subsystem, and unique developer.

3.3 Entry date

Every project within the Debian distribution has its own page under the Debian website, where the ChangeLog (typically a dated but otherwise unstructured list of amendments to the project) can be used to determine its introduction into Debian. shows the first entry in terms of changes made since its introduction into Debian. Thus, information on the project's lifecycle "before" and "after" its inclusion into Debian can be investigated and recorded. For instance, the Debian Changelog for "clamav" is shown at http://tinyurl.com/2njfon. At the bottom of the page, the first date indicates that this project entered Debian on May 9th, 2002, in its 0.11-1 release. All project history before that date is treated as pre-Debian, after that date it is treated as the post-Debian lifecycle.

4 Hypotheses and Results

Hypotheses have been formed concerning the two questions derived from the GQM approach. Here they are grouped by the question to which they belong along with their results. These hypotheses are concerned with comparing the two samples and establishing specific properties of the Debian sample.

Two statistical test have been used in this evaluation: the t-test and the Wilcoxon test. The t-test is a parametric test: the populations are typically assumed to be normally distributed. On the other hand, the Wilcoxon test is non-parametric [**?**], and the assumption of normality is not needed to run this test. Since it was not possible to clarify whether the two populations could be considered as normally distributed, the t-test was first applied. Nonparametric tests are more powerful in detecting population differences: therefore, since the normality assumptions are not clearly satisfied, the Wilcoxon tests were applied.

4.1 Empirical evaluation of question 1

The first research question has been designed as a direct comparison between the Debian and SourceForge samples, and its objective is to highlight any significant difference on the selected characteristics. Each of these hypotheses is evaluated empirically: given the null hypothesis in the second column of table 3, a statistical test will either reject it or not. A summary of the tests and their results will be provided at the end of this section to wrap up the relevant conclusions.

4.1.1 Hypothesis 1.1 – Period of Activity

This hypothesis posits that the duration of time that projects from each forge have been evolved over differs significantly, measured by the number of days for which activity could be observed on a project's repository. The null hypothesis states that Debian and Sourceforge projects have a similar time-span; this should be rejected if the sample projects display a significant difference.

Hypothesis 1.1: Days of evolution					
H0: Debian and sf.net projects have a similar time span			H1: Debian projects have a longer time span		
	min	Q1	median	Q3	max
debian	1	588	1740	2916	4088
sf.net	1	247	495.5	1290	2479
t-test	**t = -5.279**	**D.F. = 82**	$p \leq 1.142 \times 10^{-6}$		
Wilcoxon	**W = 1320**		$p \leq 3.248 \times 10^{-5}$		

Hypothesis 1.2: Distribution of size					
H0: Debian and sf.net projects have a similar size			H1: Debian projects are larger than sf.net		
	min	Q1	median	Q3	max
debian	119	3,782	48,686	110,945	2,142,554
sf.net	654	2,346	11,416	49,854	106,478
t-test	**t = -1.627**	**D.F. = 82**	$p \leq 0.11$		
Wilcoxon	**W = 1550**		$p \leq 0.036$		

Hypothesis 1.3: Distinct developers					
H0: Debian and sf.net projects have a similar amount of developers			H1: Debian projects have more developers than sf.net		
	min	Q1	median	Q3	max
debian	1	2	5,5	48	818
sf.net	1	1	2	3	82
t-test	**t = -2.294**	**D.F. = 82**	$p \leq 0.02436$		
Wilcoxon	**W = 1343**		$p \leq 7.829 \times 10^{-5}$		

Hypothesis 1.4: Overall touches					
H0: Debian and sf.net projects have a similar amount of touches			H1: Debian projects have more touches than sf.net		
	min	Q1	median	Q3	max
debian	1	2	5,5	48	818
sf.net	1	1	2	3	82
t-test	**t = -2.029**	**D.F. = 82**	$p \leq 0.04577$		
Wilcoxon	**W = 1548.5**		$p \leq 0.03475$		

Table 3: Summary of the hypotheses, tests and results of the tests

Table 3 shows that, apart from the minimum values (just 1 day of activity recorded in the repository), the two samples have different medians, different quartiles Q1 and Q3, and different maximum values. Applying both the t-test and the Wilcoxon test for two independent samples, we can reject the null hypothesis with 99.99% confidence for each test.

4.1.2 Hypothesis 1.2 – Size Achieved

The second hypothesis postulates that the typical size of a project differs significantly for each forge, in terms of SLOC (sources lines of code), with the null hypothesis stating that both forges have similar sizes, to be rejected if project sizes are shown to be significantly different.

The results are based on an evaluation of the extracted data using the R programming language. They show that projects from Debian are larger than those in SourceForge (although we found several outliers in the Debian distribution of sizes). The size of Debian packages also has a greater range, with a greater number of outliers of larger magnitude found in the Debian

distribution, implying the presence of larger communities.

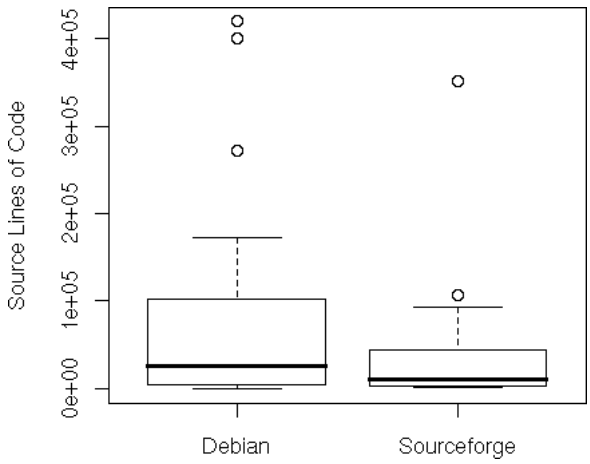

Figure 1: Boxplots of the size distribution (in SLOCs) in the Debian and SourceForge samples

From Table 3 and the boxplot in Figure 1, it can be seen that the two samples show different distributions in terms of size achieved. The null hypothesis is based on the assumption that the two sample come from the same population, and therefore have the same average; based on the tests, we can reject the null hypothesis with 89% and 96% of confidence (for the t-test and the Wilcoxon test, respectively).

4.1.3 Hypothesis 1.3 – Developers

This hypothesis posits that the number of developers that a project attracts is, on average, significantly different for each forge, measured according to the number of unique developers who have contributed source code. The null hypothesis states that Debian and SourceForge projects have approximately equal numbers of contributing developers, to be rejected if this is not the case.

The final column of tables 1 and 2 shows the number of distinct developers (i.e. CVS or SVN committers or external developers acknowledged during a specific commit).

We found several outliers in the Debian sample which rendered the average of the population sample to 51 developers, while the SourceForge sample has an average of some 5 developers only. In table 3 the summaries for the boxplot evaluation are visualised, as well as the results of the t-test (with 82 degrees of freedom) and the Wilcoxon test.

Since the null hypothesis is that the two sample have the same median, the two tests show that, for a confidence of 97.5% and 99.99% (for the t-test and the Wilcoxon test, respectively), we can reject the null hypothesis. That means that there is a statistically significant difference in the distribution of the evolution days in the two samples.

4.1.4 Hypothesis 1.4 – Activity (Touches)

The final hypothesis for question 1 postulates that the amount of activity (or output) observed differs between each forge. Specifically, the null hypothesis states that, on average, individual Debian projects and individual SourceForge projects will have a total number of file touches that does not differ significantly. We may reject this if it is shown that either forge tends to harbour significantly more active projects than the other.

The results are summarised in tables 1 and 2 . As seen in the previous test, some projects (notably wxWidgets and shorewall) clearly skew the distribution of the Debian sample. The two tests show that, for a confidence of 0.04 and 0.03 (for the t-test and the Wilcoxon test, respectively), there is a statistically significant difference in the distribution of the activity (in terms of the amount of overall touches) of the two samples.

4.2 Empirical Evaluation of Question 2

This section examines Debian only, and investigates whether it can be considered an external driver for achieving a better software evolvability. Each project in this sample has been analysed with regards to the two phases of its lifecycle, *i.e.* before and after the date when it was first included into Debian. If the projects experienced two statistically different behaviours before and after the entry date, we could conclude that the "Debian treatment" is responsible for this difference.

As shown in table 1, the bold entries in the entry_date column represent the date of each project's first appearance in Debian. This entry point, e, has been used to separate each project into two phases, so the dependent variable in each hypothesis can be measured between both the earliest available date and e, and between e and the latest available date. For some projects the entry date appears before any data was collected in their repository, hence there is no data to draw a comparison of activities and developers before and after the entry date. For hypotheses 2.1 and 2.2 only projects with data available both before and after the entry point will be considered.

4.2.1 Hypothesis 2.1 – Developers

The first hypothesis for this question postulates that the number of contributing developers a project has before insertion into Debian is significantly different to that figure after insertion. The null hypothesis assumes that no significant difference will be observed between the two durations. The metric used is the number of distinct developers.

The results are shown in the first two columns of Table 4. As can be observed the number of distinct developers in the second part of the lifecycle is always more than or equal to that of the first part. In 18 projects out of 22, the number of distinct developers after the introduction into Debian is strictly larger than before, while 4 projects out of 22 have the same amount of developers both before and after the inclusion. These latter 4 are relatively smaller projects, where at most 1 or 2 developers are currently responsible for the overall development.

Since the majority of the observed projects show a larger number of developers in the second part of the lifecycle, we can reject the null hypothesis.

4.2.2 Hypothesis 2.2 – Activity (Touches)

The second hypothesis posits that the amount of activity a project displays before appearing in Debian is significantly different to that after the event. The null hypothesis states that no significant difference in activity is apparent in the two durations. As in hypothesis 1.4, activity is measured by number of file touches.

The results are summarised in table 4 where each project is given an ID, while the overall number of touches before, T (pre), and after, T (post) are shown in the second and third columns. Considering these unadjusted values, all considered projects show a larger number of touches after joining Debian; these results lead us to reject the null hypothesis.

ID	D (pre)	D (post)		T (pre)	T (post)		T/D (pre)	T/D (post)	
1	9	10	√	10	1417	√	1.11	12.88	√
2	17	41	√	10	88	√	0.59	2.15	√
3	1	2	√	14	18	√	14	6	X
4	5	44	√	15	117	√	3	2.66	X
5	1	1	√	18	18	√	18	18	√
6	2	2	√	22	22	√	11	7.33	X
7	22	42	√	25	86	√	1.14	2.05	√
8	1	1	√	31	37	√	31	37	√
9	2	243	√	32	954	√	16	3.93	X
10	9	31	√	40	41	√	4.44	1.32	X
11	10	13	√	43	50	√	4.3	3.85	X
12	7	9	√	44	55	√	6.29	6.11	X
13	2	2	√	46	49	√	23	24.5	√
14	2	5	√	49	63	√	24.5	12.6	X
15	2	4	√	53	74	√	26.5	18.5	X
16	1	5	√	60	82	√	60	16.4	X
17	1	1	√	67	67	√	67	67	√
18	1	4	√	160	576	√	160	144	X
19	14	20	√	436	779	√	31.14	38.95	√
20	61	69	√	1666	1673	√	27.31	24.25	X
21	50	76	√	3972	6429	√	79.44	84.59	√
22	41	104	√	6923	18595	√	168.85	178.8	√

Table 4: Summary of the number of distinct developers and overall touches in the two samples

Considering the dates shown in table 1, some projects have had a longer time span within the Debian distribution than outside. To consider this point, columns 5 and 6 of table 4 report an adjusted value, given by the touches divided by the relative interval of time spent either outside or inside Debian (T/D (pre) and T/D (post) respectively). As shown by the ticks in the final column, 12 projects out of 22 experienced an adjusted number of touches which was larger before joining Debian than afterwards. This did not allow us to reject the null hypothesis, and hence to consider the observed differences in the two phases as related to the applied treatment.

As reported, the only case where the null hypothesis can be rejected is that concerning the amount of touches done in projects within the Debian forge. In general, projects achieve a larger

amount of activity after the insertion into Debian, but this is not accomplished with the same amount of touches per day (i.e., productivity) than before the entry point.

4.3 Threats to Validity

The limited information (from hypothesis 2) affects the ability to demonstrate a temporal relationship; what exists does not consistently confirm that cause precedes effect. However suggestive the data is of such a relation, more measures applied to these and other forges of similar prestige would help form a stronger position on the relevance of temporal precedence.

The ability to generalize from this study may be threatened by the Debian forge (and others of similar prestige) possessing attributes unique to themselves that may adversely effect their ability to, for example, attract new developers. To provide more confidence of generalizability a future study would need to establish that other forges acting as repositories – yet providing different services, such as Mozilla or KDE – exhibit the same characteristics as those measured here.

5 Conclusions

This research has investigated the presence of exogenous drivers to software evolvability, and proposed that the inclusion in a successful FLOSS forge (Debian) has an influence on the evolutionary characteristics of FLOSS projects (a summary of the hypotheses tested is displayed in 5).

The intended recipients of this paper are both researchers and practitioners. On the researchers' side, it aims to show that by investigating and comparing different FLOSS forges, they are likely to draw different results, and to characterise differently the FLOSS phenomenon. On the practitioners' side, the paper shows that FLOSS developers, if interested in further fostering the development of their project, should consider their project's inclusion in one large distribution-based forges such as Debian.

The paper leveraged the well-known GQM method; two research questions were then formulated, the first based on a direct comparison between the two samples, the second regarding the Debian sample only. The first question postulated that the Debian forge would have significantly different attributes to Sourceforge. Debian projects were shown to have a longer period of evolution, were larger in size, attracted more developers and experienced greater activity than SourceForge projects. All of the designed hypotheses showed a difference in the two random samples, and this positively assessed the first overall research question: Debian projects do indeed show different characteristics than projects from SourceForge.

The second research question was based on the Debian sample only and assessed the presence of two phases of evolution, i.e. before and after the inclusion into the Debian forge. In statistical terms, we studied whether there existed differences before and after applying a treatment to the sample. The first hypothesis proposed that there are more developers after being inserted into Debian, and the majority of projects showed this to be the case. The second hypothesis concerned the activity before and after the entry point. From the results we gathered, we could not conclude that there was a statistically significant difference before and after the treatment. This could be the result of measuring activity in terms of touches.

ID	H0	H1	Metrics	Outcome
1.1	Debian and sf.net projects have a similar time span	Debian projects have a longer time span	Days	H0 rejected
1.2	Debian and sf.net projects have a similar size	Debian projects are larger than sf.net	SLOCs	H0 rejected
1.3	Debian and sf.net projects have a similar amount of developers	Debian projects have more developers than sf.net	Developers	H0 rejected
1.4	Debian and sf.net projects have a similar amount of touches	Debian projects have more touches than sf.net	Touches	H0 rejected
2.1	Same amount of developers before and after the treatment	More developers after the treatment	Debian developers	H0 rejected
2.2	Same amount of touches before and after the treatment	More touches after the treatment	Debian touches	**H0 NOT rejected**

Table 5: Summary of the empirical hypotheses tested in this study (sf.net refers to SourceForge)

Further research is required to substantiate the more general proposition that widespread distribution builds the user base of a FLOSS project thus drives its evolution, while incorporation into a distribution with an existing developer base provides the basis for sustainable evolution.

Bibliography

[ACPM01] G. Antoniol, G. Casazza, M. D. Penta, E. Merlo. Modeling Clones Evolution Through Time Series. In *Proc. IEEE Intl. Conf. on Software Maintenance 2001(ICSM 2001)*. Pp. 273–280. Fiorence, Italy, Nov 2001.

[BCR94] V. R. Basili, G. Caldiera, D. H. Rombach. The Goal Question Metric Approach. In *Encyclopedia of Software Engineering*. Pp. 528–532. John Wiley & Sons, 1994. See also http://sdqweb.ipd.uka.de/wiki/GQM.

[CAH03a] K. Crowston, H. Annabi, J. Howison. Defining Open Source Software Project Success. In *Proceedings of ICIS 2003*. Seattle, Washington, USA, Dec. 2003.

[CAH03b] K. Crowston, H. Annabi, J. Howison. Defining open source software project success. In *ICIS 2003. Proceedings of International Conference on Information Systems*. 2003.

[Cap03] A. Capiluppi. Models for the Evolution of OS Projects. In *Proceedings of ICSM 2003*. Pp. 65–74. Amsterdam, Netherlands, 2003.

[CCP07] G. Canfora, L. Cerulo, M. D. Penta. Identifying Changed Source Code Lines from Version Repositories. *Mining Software Repositories* 0:14, 2007.

[CM07] A. Capiluppi, M. Michlmayr. From the Cathedral to the Bazaar: An Empirical Study of the Lifecycle of Volunteer Community Projects. In Feller et al. (eds.), *Open Source Development, Adoption and Innovation*. Pp. 31–44. Springer, 2007.

[ES07] R. English, C. Schweik. Identifying Success and Tragedy of FLOSS Commons: A Preliminary Classification of Sourceforge.net Projects. In *Proceedings of the 1st International Workshop on Emerging Trends in FLOSS Research and Development*. Minneapolis, MN, 2007.

[FFH+02] J. Feller, B. Fitzgerald, F. Hecker, S. Hissam, K. Lakhani, A. van der Hoek (eds.). *Characterizing the OSS process*. ACM, 2002.

[Ger04] D. M. German. Using software trails to reconstruct the evolution of software. *Journal of Software Maintenance and Evolution: Research and Practice* 16(6):367–384, 2004.

[HG05] A. Hindle, D. M. German. SCQL: a formal model and a query language for source control repositories. *SIGSOFT Softw. Eng. Notes* 30(4):1–5, 2005.

[LHMI07] S. Livieri, Y. Higo, M. Matushita, K. Inoue. Very-Large Scale Code Clone Analysis and Visualization of Open Source Programs Using Distributed CCFinder: D-CCFinder. In *ICSE '07: Proceedings of the 29th International Conference on Software Engineering*. Pp. 106–115. IEEE Computer Society, Washington, DC, USA, 2007.

[LRW+97] M. M. Lehman, J. F. Ramil, P. D. Wernick, D. E. Perry, W. M. Turski. Metrics and Laws of Software Evolution—The Nineties View. In El Eman and Madhavji (eds.), *Elements of Software Process Assessment and Improvement*. Pp. 20–32. IEEE CS Press, Albuquerque, New Mexico, 5–7 Nov. 1997.

[MFH02] A. Mockus, R. T. Fielding, J. Herbsleb. Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology* 11(3):309–346, 2002.

[RG05] A. Rainer, S. Gale. Evaluating the Quality and Quantity of Data on Open Source Software Projects. In Feller et al. (eds.), *First International Conference on Open Source Systems*. 2005.

[RG06] G. Robles, J. M. Gonzlez-Barahona. Contributor Turnover in Libre Software Projects. In Damiani et al. (eds.), *OSS*. IFIP 203, pp. 273–286. Springer, 2006.

[RKG04] G. Robles, S. Koch, J. M. González-Barahona. Remote analysis and measurement of libre software systems by means of the CVSAnalY tool. In *Proceedings of the 2nd ICSE Workshop on Remote Analysis and Measurement of Software Systems (RAMSS '04). 26th International Conference on Software Engineering*. Edinburgh, UK, May 2004.

[SA02] K. J. Stewart, T. Ammeter. An Exploratory Study of Factors Influencing the Level of Vitality and Popularity of Open Source Projects. In *ICIS 2002. Proceedings of International Conference on Information Systems 2002*. 2002.

[SAOB02] I. Stamelos, L. Angelis, A. Oikonomou, G. L. Bleris. Code Quality Analysis in Open-Source Software Development. *Information Systems Journal* 12(1):43–60, 2002.

[SM04] A. Senyard, M. Michlmayr. How to Have a Successful Free Software Project. In *Proceedings of the 11th Asia-Pacific Software Engineering Conference*. Pp. 84–91. Busan, Korea, 2004.

[Wei05] D. Weiss. Measuring Success of Open Source Projects Using Web Search Engines. In Scotto and Succi (eds.), *Proceedings of The First International Conference on Open Source Systems (OSS 2005), Genova, Italy*. Pp. 93–99. 2005.