Is It All Lost?
A Study of Inactive Open Source Projects

Jymit Khondhu[1], Andrea Capiluppi[1], and Klaas-Jan Stol[2]

[1] Department of Information Systems and Computing
Brunel University, United Kingdom
[2] Lero—The Irish Software Engineering Research Centre
University of Limerick, Ireland
j_khondhu@hotmail.co.uk, andrea.capiluppi@brunel.ac.uk, klaas-jan.stol@lero.ie

Abstract. Open Source Software (OSS) proponents suggest that when developers lose interest in their project, their last duty is to "hand it off to a competent successor." However, the mechanisms of such a hand-off are not clear, or widely known among OSS developers. As a result, many OSS projects, after a certain long period of evolution, stop evolving, in fact becoming "inactive" or "abandoned" projects. This paper presents an analysis of the population of projects contained within one of the largest OSS repositories available (SourceForge.net), in order to describe how projects abandoned by their developers can be identified, and to discuss the attributes and characteristics of these inactive projects. In particular, the paper attempts to differentiate projects that experienced maintainability issues from those that are inactive for other reasons, in order to be able to correlate common characteristics to the "failure" of these projects.

Keywords: Open Source, Inactive Projects, Maintainability Index

1 Introduction

The vast diffusion of Open Source Software (OSS), and the availability of several (hundreds of) thousands of OSS projects, has started to show that this online phenomenon is not always successful [21, 19]. A lack of activity by developers, and a low interest by users, show that many OSS projects, after a certain long period of evolution, stop evolving, and in fact become "inactive" projects. This phenomenon has been observed before, and metrics have been developed to automatically detect inactivity, or to correlate which characteristics are more relevant to such inactive projects [18]: how many days, months or years in "*lack of activity*" are necessary to tag a project as inactive? How much should two subsequent releases be apart to declare that a project's development is inevitably facing a downturn? On the other hand, is a specific programming language more likely to cause inactivity in projects? A specific application domain (games, office productivity, etc.)? A specific OSS license?

Although the semi-automatic detection of inactivity can produce useful insights on the general trends of large sets of OSS projects, any set of assumptions

Please cite as:
J Khondhu, A Capiluppi and K Stol (2013) Is It All Lost? A Study of Inactive Open Source Projects,
In: Proceedings of the 9th International Conference on Open Source Systems.

used to produce figures of inactive projects could be criticized for its inclusion criteria, or for not being validated with external factors. One could inadvertently misclassify dormant projects as "inactive"; projects could be renamed in the meantime, or even just moved to dedicated repositories.[3]

The information on such inferred inactivity has relevant external effects: from an end-user's point of view, organizations and private users would not want to invest money or time to deploy or evaluate inactive projects, or those with few or no contributors to perform future maintenance. From a developers' point of view, the original authors should take the responsibility to inform others that they are no longer interested (or able) to support or enhance their project. Or, as Raymond [15, p.26] suggested:

> "*When you lose interest in a program, your last duty to it is to hand it off to a competent successor.*"

This would introduce one of the most powerful mechanisms in OSS development, namely the possibility for another developer to take over the project from one or more previous developers who lost interest in it [15, 3]. Thus, we identified two key motivations for accurately labelling inactive or abandoned projects:

1. The ability to clearly distinguish alive projects from the clutter, so as to give confidence to potential users and adopters;
2. To implement one of the most powerful mechanisms of OSS development, namely the ability for other developers to "take over" inactive or abandoned projects, in order to sustain a new phase of development, managed and sustained by someone other than the original creators.

This paper studies the abandonment of OSS projects without inferring their lack of activity with measures. It studies the abandoned projects by spidering a specific HTML tag (i.e., $<$ *inactive* $>$) that the authors are encouraged to use in specific cases for their projects on SourceForge.[4] When developers do not intend to continue working on their projects any longer, they have the ability of placing such a tag onto the SourceForge "main" project page,[5] thus sending a clear indication to potential users about the lack of maintenance and support; but also to other developers, who can follow a specific process to become the new maintainers and owners of the project, if they want to do so.[6]

A fundamental reason for studying abandoned projects is to analyze what type of resources have been left behind when developers "moved on", their quality, and whether one of the reasons for such abandonment is because those resources have become ultimately unmaintainable. Both as a user and as a prospective new contributing developer, it is important to understand the quality of an

[3] As an example, the Moodle CMS was moved from SourceForge to its own dedicated repository, after they have "outgrown it."

[4] `http://sourceforge.net`

[5] See for instance the main page of the *busiprocess* project, `http://sourceforge.net/projects/busiprocess/`.

[6] The process is detailed, for the SourceForge repository, at `http://sourceforge.net/apps/trac/sourceforge/wiki/Abandoned\%20Project\%20Takeovers`.

Please cite as:
J Khondhu, A Capiluppi and K Stol (2013) Is It All Lost? A Study of Inactive Open Source Projects,
In: Proceedings of the 9th International Conference on Open Source Systems.

abandoned project and its resources, its general maintainability, and how it progressed over time, also in relation to taking over "control" of the overall project, or potentially the reuse of some of its carefully designed internal components [5]. Therefore, this paper has three objectives:

1. To *quantify* the phenomenon of abandonment of OSS projects on the SourceForge repository;
2. To *investigate* whether the abandonment of these projects should be linked with issues of maintainability of the source code itself;
3. To *discuss* the quality of the resources that were left behind for others to reuse or "upcycle."

The remainder of this paper proceeds as follows. Section 2 presents the research design. Section 3 presents the results of our study. Section 4 positions this study with respect to previous work in this area. We conclude in Section 5.

2 Research Design

2.1 Research Strategy

We adopted the *Goal-Question-Metric* approach as the overall research strategy [2].

Goal. The overall goal of this study is to develop an understanding of the scale, nature and characteristics of abandonment of OSS projects. We defined two sub-goals: (1) to quantify abandonment of OSS projects, and (2) to evaluate the quality of the code that was left abandoned by developers.

Questions. This paper addresses the following research questions:

1. How many "active," "dormant" and "inactive" projects exist in SourceForge?
2. Do these categories achieve similar growth patterns?
3. Do these categories result in comparable quality and maintainability attributes?

Metrics and Definitions. We used the following metrics and definitions in this study:

– **Category of projects**: in this study we aim to produce three clusters of projects: the "active" ones, whose activity is updated and recent; the "dormant" ones, whose activity is visible in the past evolution, but it has stopped (for any reason) for a defined period (e.g., one year, two years, etc.); and the "inactive" ones, that have been explicitly tagged as such by the previous developers.

Please cite as:
J Khondhu, A Capiluppi and K Stol (2013) Is It All Lost? A Study of Inactive Open Source Projects,
In: Proceedings of the 9th International Conference on Open Source Systems.

- **Latest update**: in this study we will extract the latest recorded activity for any Sourceforge project by parsing the specific `dateUpdated` tag on each project's summary page.
- **Inactive project**: in principle, the inactivity of an OSS project should be evaluated by pre-defining an interval of time when no development effort is registered by its contributors, in terms of commits, messages on the mailing lists or public releases. The SourceForge repository places the specific $< inactive >$ HTML tag[7] on the project pages that show no sign of activity. This could be either be directly communicated by a project's administrators (i.e., an "abandoned" project), or inferred from activity logs (i.e., an "inactive" project). The possibility of isolated inactive projects, and the availability of the source code, is one of the fundamental characteristics of OSS development, which allows new developers to take over and manage an abandoned project [3].
- **Number of lines of code (LOC)** measures the physical size of the program code, excluding blank lines and comments.
- **Percentage of lines of comments** with respect to the number of lines of code (PerCM) describes the documentation and self-descriptiveness of the code.
- **Halstead Volume (HV)**. Halstead [10] defined four metrics that can be measured from a program's source code: n1 (the number of distinct operators), n2 (the number of distinct operands), N1 (the total number of operators) and N2 (the total number of operands). Based on them, Halstead defined program vocabulary n (given by n = n1 + n2) and program length N (given by N = N1 + N2). Finally, he defined Volume, a composite metric given by the formula $V = N * (LOG2n)$.
- **Cyclomatic Complexity (CC)**. Proposed by McCabe [13], this metric counts the number of independent paths in the control flow graph of a program component. Its value depends on the number of branches caused by conditional statements (`if-then-else`). It measures the structural complexity of the component.
- **Maintainability Index (MI)**: based on the definitions above, the Software Engineering Institute (SEI) has developed a maintainability index defined as follows:

$$171 - 5.2ln(HV_{avg}) - 0.23CC_{avg} - 16.2ln(LOC_{avg}) + 50.0sin\sqrt{2.46COM} \quad (1)$$

where HV_{avg} measures the average Halstead metric per module (function, method, class, file or the whole system), CC_{avg} the average McCabe index per module and LOG_{avg} the average lines of code per module.

2.2 Data Collection and Analysis

In this subsection we detail the steps undertaken to perform this empirical study.

[7] `<span id="dev-status" class="inactive">inactive</span>`

Please cite as:
J Khondhu, A Capiluppi and K Stol (2013) Is It All Lost? A Study of Inactive Open Source Projects,
In: Proceedings of the 9th International Conference on Open Source Systems.

1. **SRDA database query**: in order to have an initial list of all the projects contained in the SourceForge archive, we queried the SRDA database [9].[8] In the data-dump of September 2012, the total number of SourceForge projects retrieved in the SRDA database is 174,845, and their unique names were recorded.
2. **Status extraction**: each of the project IDs was used to compose a URL relative to the SourceForge online structure.[9] The retrieved page contains a generic summary of the status of the project, along with information on its programming languages, the intended audience, the date of the latest update to the project and so on. The tag "inactive" was searched for all the projects, and a subpool of **inactive projects** was identified. This additional attribute is not available from the SRDA database, and it had to be specifically extracted.
3. **Latest activity extraction**: from the same summary page, we extracted, per project, the latest recorded date of activity, which is automatically copied by SourceForge from the list of activities of the project.[10] The information on the latest activity date served to discriminate between an "active" and a "dormant" project. If no activity was registered throughout the last year since the analysis (i.e., since November 2011), a project was categorized as "dormant" (instead of abandoned as proposed in [19]). As a result of this step, we obtained a subpool of **active projects**, and another subpool of **dormant projects**. The inactive projects, when their latest activity flagged them also as dormant, were discarded from the subpool of dormant ones, in order to have mutually exclusive subsets.
4. **Sampling of projects**: three equally sized samples of 25 projects each were randomly extracted from the three subsets of projects ("active," "inactive" and "dormant"), respectively. An SQL "random" statement was used to extract the three samples. Each of the projects in all the samples IDs was used to compose a URL relative to retrieve the relative code repository in the SourceForge online structure.[11]
5. **Three-point selection**: considering the number of revisions of a project, three data points were considered:
   - *Initial Point – IP*: this point refers to the first revision that was committed into the code repository of the project;
   - *Final Point – FP*: the last available point in the evolution log, corresponding to the latest available revision N contained in the repository of the project;

[8] The table that lists the projects is named *trove_agg*.

[9] The homepage of any Sourceforge project 'p' hosted appears in the form `http://sourceforge.net/project/p`.

[10] This is coded within the summary page of each project and marked with the `<dateUpdated>` tag.

[11] The repository of any Sourceforge project 'p' can be found under their "develop" page under `http://sourceforge.net/project/p/develop`, and marked with the `<code>` tag.

Please cite as:
J Khondhu, A Capiluppi and K Stol (2013) Is It All Lost? A Study of Inactive Open Source Projects,
In: Proceedings of the 9th International Conference on Open Source Systems.

- *Middle Point – MP*: considering the final point, the middle point is evaluated dividing by two the overall number of revisions N contained in the repository. In this way, we observed a system in three points not temporally, but *logically* equidistant.

6. **Evaluation of size**: the size of each project in the samples of the three categories ("active," "dormant" and "inactive") was evaluated using the `sloccount`[12] tool, that measures the physical lines of code (SLOCs) rather than the overall number of lines in a class, file or a whole system. The size was evaluated in the three points mentioned above (IP, MP and FP), and boxplots were produced to compare the growth in size of the projects in the different samples.
7. **Calculation of the Maintainability Index (MI)**: the maintainability index (MI) can be used to evaluate single functions, classes, files, compounds (as packages, namespaces, etc.) and even a system as a whole. The general understanding of such index is that with an $MI \geq 85$, the function or the compound can be considered to have a good maintainability; with an index $85 > MI \geq 65$ the compound has a moderate maintainability; while for a $MI < 65$ the compound should be considered as a difficult to maintain compound, signaling low quality pieces of code. In this work, the index was used to compare and contrast the projects within the abandoned sub-pool, and to evaluate the changes in such index when considering the evolution of the projects, before the abandonment, rather than as an absolute measure. The MI was evaluated for all the systems in the three samples, and along the three points forming their life-cycles: an automated Perl script[13] from the Understand suite[14] was used to evaluate the MI of each compound, as well as the system-wide MI.

3 Results

As outlined in Section 2, we defined two sub-goals. The first, that of quantifying the abandonment of OSS projects, is addressed by research questions one and two listed in Section 2.1. Subsections 3.1 and 3.2 address these research questions, respectively. The second goal, that of evaluating the quality of the code left behind by developers is operationalized by research question three, and is addressed in Subsection 3.3.

3.1 A Classification of Project Activity Status

To address the first research question, *How many 'active,' 'dormant' and 'inactive' projects exist in SourceForge?*, we conducted a characterization of the pool

[12] `http://www.dwheeler.com/sloccount/`

[13] Available at `http://scitools.com/plugins/perl_scripts/acjf_maint_index_halstead.pl`

[14] Understand by Scitools, `http://scitools.com/index.php`

Please cite as:
J Khondhu, A Capiluppi and K Stol (2013) Is It All Lost? A Study of Inactive Open Source Projects,
In: Proceedings of the 9th International Conference on Open Source Systems.

of over 174,000 projects in SourceForge (as of November 2012), into three major clusters: "active," "inactive" and "dormant."

The inactive cluster was the easiest to evaluate: as reported in the sections above, the specific "inactive" tag was searched in each of the projects' summary pages. As depicted in Figure 1, the pool of SourceForge projects contains over 10,000 projects tagged as "inactive" by their own developers (as of November 2012). The inactive projects represents around 6% of the overall population of the hosted projects.

The "active" and "dormant" clusters of projects were evaluated more subjectively: given the date of sampling (DS) at 11/2012, and knowing the date of the latest activity (LA) for each project, at first it was decided to evaluate as "active" those projects whose latest update was within a year (i.e., 365 days) of the date of sampling, "dormant" otherwise. We used the formulas below:

$$LA - DS < 365 \Longrightarrow active \tag{2}$$

$$LA - DS > 365 \Longrightarrow dormant \tag{3}$$

Using this approach, over 86% of Sourceforge projects were classified as "dormant," i.e., there is no recorded activity in their latest year on SourceForge. Only some 7% of projects have experienced an update within the last year. These initial three clusters are visible in the left part of Figure 1.

Using two years (i.e., 730 days) as a threshold for the latest activity, some 65% of projects were classified as "dormant," while some 28% were classified as "active" (that is, development occurred in the last two years). This second set of clusters are visible in the right-hand side of Figure 1.

These results form an updated picture of what is well known of OSS portals by researchers and practitioners: on the one hand, OSS portals tend to host an increasingly large number of projects. On the other hand, the large number of inactive and dormant projects has started to become an issue of space (i.e., should abandoned projects be erased from repositories?), as well as of credibility of the overall OSS community (i.e., are OSS projects doomed to failure?).

A further investigation of the inactive projects resulted in a clustering of three subsets:

1. **Inactive "moved" projects**: some of those projects were moved from SourceForge to their own servers, and were tagged as inactive (but "moved") to indicate to interested developers where to find the up-to-date releases or the repository that hosts the source code. The number of moved projects represents some 14% of the overall inactive projects.
2. **Inactive "stale" projects**: a larger portion of the inactive projects are additionally tagged as "stale" projects, therefore indicating to other interested

Please cite as:
J Khondhu, A Capiluppi and K Stol (2013) Is It All Lost? A Study of Inactive Open Source Projects,
In: Proceedings of the 9th International Conference on Open Source Systems.

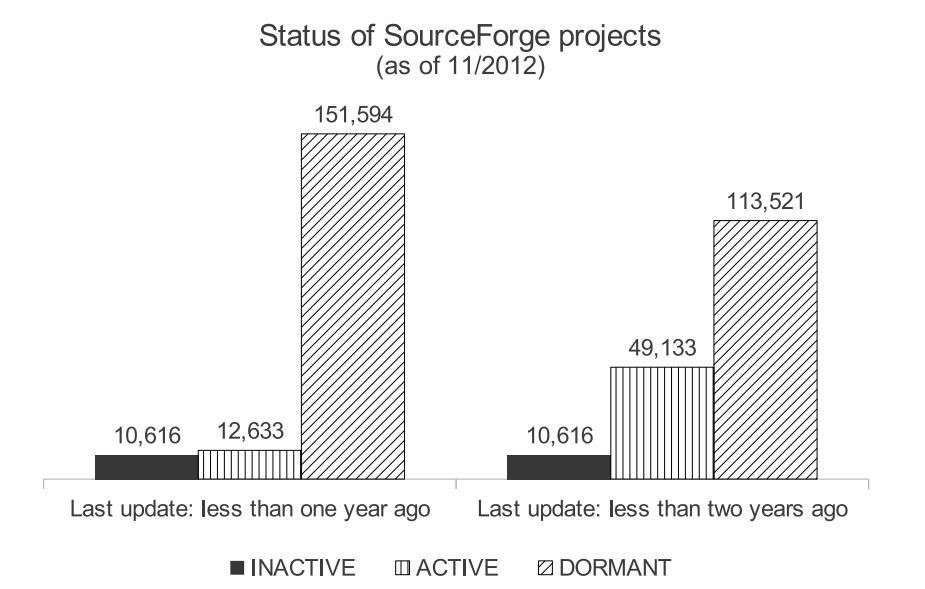

Fig. 1. Clusters of "active," "inactive" and "dormant" projects in Sourceforge, using different thresholds (as of 11/2012)

developers that the project is properly "dead" and it could benefit from further attention and development.[15] SourceForge makes sure also to record the date since when those projects were declared as "stale," which becomes an additional piece of information for the evaluation of such systems. Over 60% of the inactive projects are also categorized as stale, as shown in Figure 2.

3. **Other inactive projects**: the remaining 23% of the inactive projects provide no further information on specific details regarding when the project was declared inactive.

The remainder of the analysis in this paper focuses only on the projects which were appropriately tagged as "stale" by their developers, to produce a more thorough understanding of what sort of resources were left behind by the original developers for others to take over.

3.2 Growth Patterns Across Categories

The second research question was defined as: *do these categories achieve similar growth patterns?* As described above, three points were considered: the size of a project in its "first revision" under the versioning system (i.e., its initial point –

[15] For instance, by being developed in a parallel branch onto another OSS repository as GitHub, as anecdotally reported in `http://www.quora.com/What-is-the-appeal-of-GitHub-com`

Please cite as:
J Khondhu, A Capiluppi and K Stol (2013) Is It All Lost? A Study of Inactive Open Source Projects,
In: Proceedings of the 9th International Conference on Open Source Systems.

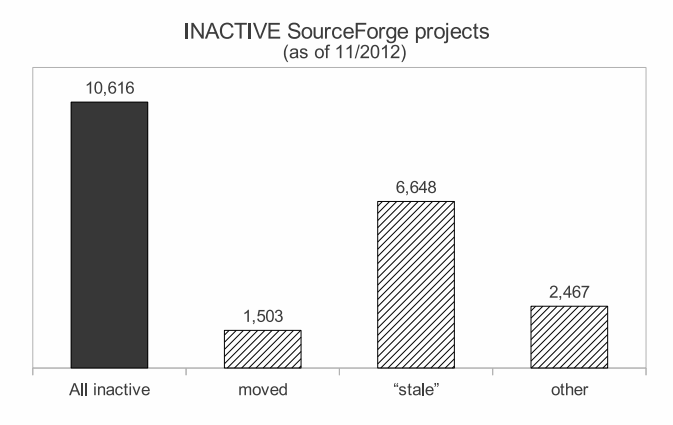

Fig. 2. Distribution of inactive projects in SourceForge (as of 11/2012)

IP); in its latest revision (i.e., its final point – FP); and in the revision midway between the first and the latest (i.e., its midway point – MP). The size of the systems was evaluated with the *sloccount* tool for the IP, MP and FP points. The size in SLOCs of the systems are shown, per point, in the Appendix section (Tables 2, 3 and 4).

The distribution of size per project and point, and across samples was visualized using boxplot diagrams to take into consideration the distribution of very diverse projects in the same sample. The distribution per sample, and relative to the initial point (IP) is shown in Figure 3. The other boxplot distributions, albeit similar to the one reported below, are not displayed due to space limitations.

The boxplots in Figure 3 show that the distribution of size (in SLOCs) among the "active" projects is generally larger than the other samples ("dormant" and "inactive"). Such disparity is clearly visible (albeit not reported as a graph) also in the other analyzed points (MP and FP): the outliers of the dormant and inactive projects (shown in Figure 3) exacerbate the difference between few outstanding projects and the more general projects in the MP and FP points.

Despite the large variances (indicated by the whiskers and the outliers of Figure 3), we report the averages and medians of these distributions among samples and in the three points IP, MP and FP (Table 1). In both the average and median measurements, and across samples, the average system increases its size along the three points, but the average active system achieves a growth of 7 times (or 6 times for the medians) between the initial and final points of its growth. "Dormant" and "inactive" projects appear less distinguishable in what they achieve in their growth.

The results of the growth analysis suggest that there is a difference between the analyzed samples, hence between the described clusters: "active" projects begin as larger (in size) than both "dormant" and "inactive" projects, and they grow considerably larger than the other samples along their lifecycle. Whether

Please cite as:
J Khondhu, A Capiluppi and K Stol (2013) Is It All Lost? A Study of Inactive Open Source Projects,
In: Proceedings of the 9th International Conference on Open Source Systems.

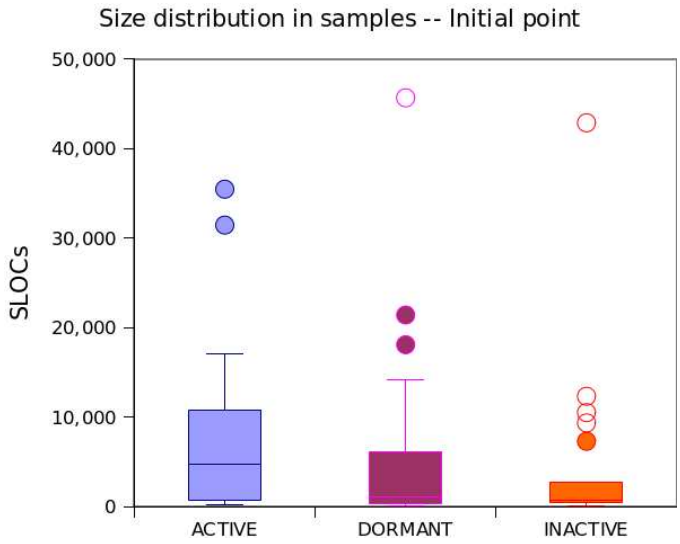

Fig. 3. Boxplots of size (in SLOCs) across samples, initial point (ip)

Table 1. Size growth in the sample of "active," "dormant" and "inactive" projects – Average and median values (in SLOCs)

Average	IP	MP	FP
Active	7,680	24,486	50,189
Dormant	5,585	12,361	16,862
Inactive	4,056	7,775	15,014
Median	IP	MP	FP
Active	4,789	12,361	20,217
Dormant	1,114	3,047	4,520
Inactive	720	2,163	3,267

or not a dormant project returns in an active state, the difference with active projects still persist.

These differences in size will be evaluated in the next section by monitoring their maintainability indexes along the three selected points.

3.3 Quality and Maintainability Across Categories

The third research question defined in Section 2 was: *do the categories result in comparable quality and maintainability attributes?* The overall objective of this is to assess whether there is a difference in the quality of projects across the different categories (active, inactive and dormant).

Please cite as:
J Khondhu, A Capiluppi and K Stol (2013) Is It All Lost? A Study of Inactive Open Source Projects,
In: Proceedings of the 9th International Conference on Open Source Systems.

In this section, the issue of quality is discussed by measuring and monitoring the Maintainability Index [14] of the projects composing the abandoned subpool. The index has been used several times in the past [8], especially to evaluate the effect of the index on the resulting maintenance needed, but the index has also been exposed to various criticism [11].

Code quality is a multi-dimensional attribute that cannot be evaluated only by a static analysis, as the one that we performed and reported below. Nonetheless, the maintainability index (MI) offers an advantage over other low-level metrics and measurements (such as the Kemerer and Chidamber suite of object-oriented metrics, the cyclomatic complexity or others): such index can be used to evaluate single functions, classes, files, compounds (as packages, namespaces, etc.) but also whole systems.

In this study, MIs were used more to compare and contrast the projects within the samples, and to evaluate the changes in such index when considering the evolution of the projects, before the abandonment, rather than as an absolute measure.

The maintainability indexes for the projects composing the samples are reported in the right-hand side of Tables 2, 3 and 4 included in the appendix) for the IP, MP and FP points. In particular, we were interested in two aspects: (1) whether the maintainability indexes are comparable across samples and categories; and (2) whether the projects in the "inactive" sample have clearly issues with their quality, reflected in low maintainability indexes.

Figure 4 shows the distribution of MIs within the samples in the initial point (IP, top of the figure) and of the latest available revision (FP, bottom of the figure). Two things can be observed: first, that the "active" sample shows a boxplot which achieves less variability between IP and FP (reflected in a more compact boxplot). Second, the difference between samples is not evident. The average scores in the maintainability indexes show that "active" projects are slightly higher than the other samples, but in general the samples seem to decrease their quality with their growth. This is more evident in those projects that achieve large changes of size (reflected in Tables 2, 3 and 4).

The second aspect that was investigated was related to the relative change in overall maintainability between the initial and the final point of each project, and it was measured with a Delta, defined as:

$$Delta(MI)_p = \frac{MI(fp)_p - MI(ip)_p}{MI(ip)_p} \tag{4}$$

where $MI(fp)_p$ represents the maintainability index at the final point for project 'p' and $MI(ip)_p$ the MI in the initial point for the same project 'p.' No relative changes were recorded in proximity of smaller changes, i.e. when the Delta was less than 1%. If the Delta was positive, nil or the change was minimal, we put a ✓in the last column of Tables 2, 3 and 4. If there was a substantial negative change, we recorded it with a X in the same column.

As shown in the tables, 14 projects in the "active" sample scored a positive change (or an equal score) between IP and FP in their MIs. On the other hand,

Please cite as:
J Khondhu, A Capiluppi and K Stol (2013) Is It All Lost? A Study of Inactive Open Source Projects,
In: Proceedings of the 9th International Conference on Open Source Systems.

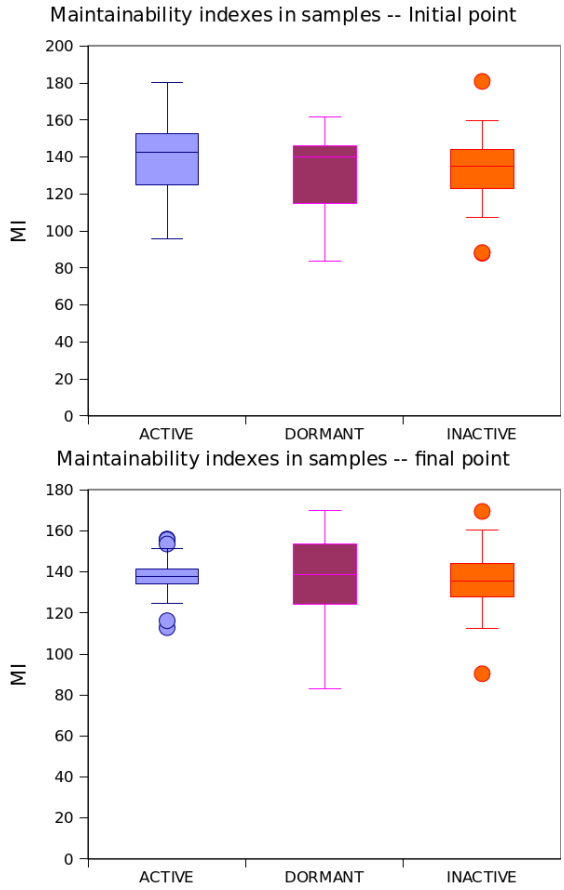

Fig. 4. Boxplots for the maintainability indexes across samples, initial point (IP, top) and final point (FP, below)

11 projects experience a substantial negative change in their MIs. The proportion of "dormant" projects with an increasing or stable MI to those with a decreasing MI is 16 to 9; the proportion for "inactive" projects is 17 to 8. This shows that the majority of "dormant" and "inactive" systems are left (or abandoned) with a similar or increased maintainability, in comparison to their initial status.

If the results were confirmed in a larger sample, or the whole population of inactive projects, the implications would be straightforward:

- From a *developer's* perspective, one would want to take over projects which are inactive, and that showed some degree of growth (otherwise they could be perceived as "toy systems" that were developed over a short time and abandoned soon after storing in an OSS repository);
- From a *commercial stakeholder's* perspective aiming to participate in OSS development [6], the project to take over (or get involved in) should have

Please cite as:
J Khondhu, A Capiluppi and K Stol (2013) Is It All Lost? A Study of Inactive Open Source Projects,
In: Proceedings of the 9th International Conference on Open Source Systems.

Table 2. Size growth and maintainability indexes in the sample of "active" projects

	SLOCs			Maintainability Index			
Project ID	IP	MP	FP	IP	MP	FP	$\Delta \geq 0$?
adirectoryimp	10,818	11,918	12,202	152.43	156.34	156.06	✓
Cw-evolver	2,449	4,147	5,994	120.73	123.63	131.53	✓
eid2ldap	444	5,697	7,606	122.75	139.79	134.35	✓
Enrich-ege	780	139,857	88,878	170.17	141.71	141.6	X
eps2pgf	2,372	8,014	19,315	153.13	153.11	155.68	✓
fenixweb	8,555	8,777	9,817	138.95	140.35	140.31	✓
flugbuchz	12,313	5,941	47,587	137.33	134.87	135.85	X
fluidcubed	749	774	887	132.35	144.72	144.55	✓
foxtrot	450	2,161	5,156	143.28	140.03	137.59	X
ieplugin4rm	14,071	18,619	25,355	144.58	144.72	140.68	X
javajson	318	2,620	3,345	95.83	138.65	137.84	✓
loconetovertap	557	5,107	9,622	145.57	140.97	129.66	X
micepfl	6,812	39,458	47,716	122.5	137.23	135.26	✓
midiagentsystem	4,789	9,007	13,329	145.81	141.66	140.92	X
mythtvplayer	9,545	11,700	20,217	133.07	132.23	133.27	✓
oswing	31,476	64,856	128,879	145.53	143.62	144.64	✓
quartica	17,086	12,205	101,063	154.78	149.78	137.09	X
rcpforms	1,861	22,039	74,036	161.3	139.66	137.62	X
resteasy	2,220	79,237	151,681	180.08	155.32	151.46	X
spl	35,488	55,560	257,447	125.08	151.88	135.24	✓
vizkit	6,444	9,542	40,130	142.37	139.2	141.01	✓
xith3d	5,658	51,062	122,811	169.66	153.38	153.62	X
xmms2-qtcli	1,157	2,074	2,486	138.65	126.97	124.68	X
yaco	15,397	41,445	58,550	96.88	116.34	112.82	✓
zmkk	201	325	605	104.68	115.7	116.22	✓

a certain minimum degree of quality and maintainability. The MI could be a useful metric to show this quality of compounds, such as files, classes, modules and even the whole system, at various points of its evolution, and ensuring that the quality or maintainability was not compromised too significantly in its evolution. The MI may be a simple alternative to the plethora of more extensive evaluation methods and frameworks that have been proposed so far to evaluate open source products [20].

However, further research is needed to investigate this in more detail and to confirm these findings.

Please cite as:
J Khondhu, A Capiluppi and K Stol (2013) Is It All Lost? A Study of Inactive Open Source Projects,
In: Proceedings of the 9th International Conference on Open Source Systems.

Table 3. Size growth and maintainability indexes in the sample of "dormant" projects

	SLOCs			MaintainabilityIndex			
Project ID	IP	MP	FP	IP	MP	FP	$\Delta \geq 0$?
build-status	1,075	1,075	1,572	154.73	154.73	153.59	✓
dbtoolbox	7,146	8,541	9,214	161.47	160.14	159.95	✓
dekware	45,693	46,181	46,644	113.36	113.36	111.79	X
dynolab	190	553	841	146.22	178.44	170.14	✓
emfincpp	260	1,229	3,226	154.24	154.53	154.53	✓
fit7h-projects	66	192	201	143.08	124.21	123.98	X
gcts	1,114	1,611	642	148.75	147.92	155.48	✓
gerber2eps	1,201	9,616	9,652	135.62	126.74	127.1	X
hemus	14,229	88,665	139,147	111.36	104.78	99.24	X
jrdesktop	6,206	11,327	12,104	127.46	139.36	138.63	✓
libnary	9,159	9,792	10,059	86.01	90.92	89.82	✓
llads	381	46,843	56,182	114.81	124.15	124.22	✓
msgtext	253	4,431	4,520	111.97	135.17	135.02	✓
nexplorer	18,111	19,574	19,523	142.06	149.51	150.54	✓
numenor	348	1,981	2,734	135.55	138.92	142.37	✓
osgmaxexp	4,229	6,632	8,448	139.94	140.55	137.48	X
overflowandroid	308	5,140	2,055	111.18	112.16	86.18	X
psimpl	4,051	3,047	10,283	133.69	133.7	127.28	X
siprop	21,429	68,828	108,816	153.39	158.3	158.41	✓
svnauthzctl	905	941	1,075	83.65	82.65	82.95	✓
tbltools	1,804	2,949	5,688	145.66	140.11	146.52	✓
vmatrixlib	1,048	1,108	1,377	139.98	148.04	146.3	✓
yahoofinanceapi	1,288	1,834	2,339	138.09	142.41	148.17	✓
ybwopenpilot	820	1,729	1,949	161.9	156.33	153.98	X
zcommons-mojos	454	800	1,212	140.05	130.82	130.82	X

4 Discussion

4.1 Comparison to Related Work

This study positions itself in the area of the categorization of OSS projects: the abandonment of OSS projects has been extensively studied in the past in [19, 12] and in [7, 4], firmly establishing that the vast majority of OSS projects suffers from the "abandonment tragedy." What the present study contributes is an understanding of how developers are currently handing over projects to others, by means of signaling the inactivity of their projects through a specific tag. Also, different from other studies, we did not try to predict the inactivity or projects based on specific coding characteristics (such as languages, application domain, and so forth), but we used readily available identifiers to infer a project's inactivity.

The maintainability of OSS packages has been studied, in a slightly different formula by Samoladas et al. [16], and was also reproduced with OSS projects

Please cite as:
J Khondhu, A Capiluppi and K Stol (2013) Is It All Lost? A Study of Inactive Open Source Projects,
In: Proceedings of the 9th International Conference on Open Source Systems.

Table 4. Size growth and maintainability indexes in the sample of "inactive" projects

	SLOCs			Maintainability Index			
Project ID	IP	MP	FP	IP	MP	FP	$\Delta \geq 0$?
aesop-server	489	489	489	128.12	128.12	128.12	✓
arduinoforum	508	576	1,426	124.42	123.03	112.33	X
artgarden	1,990	2,349	2,649	132.97	133.39	130.96	X
carjam	2,484	2,495	3,267	153.72	153.54	147.18	X
cgiscaler	42,901	29,488	58,958	135.67	135.9	137.47	✓
cherrygis	9,374	9,510	10,436	135.19	135.46	138.24	✓
csamegame	2,753	2,753	2,753	129.06	129.06	128.69	✓
cuatroenraya	189	342	1,190	128.99	127.88	115.77	X
doubles	1,028	3,857	6,130	107.31	132.27	124.25	✓
foobar1914	665	1,051	1,904	139.89	167.5	169.52	✓
gimp-vs	1,536	6,073	151,164	107.48	118.58	125.27	✓
icbor	576	7,907	3,488	116.74	140.23	139.76	✓
icepidgen	921	919	13,061	180.88	181.01	144.27	X
icerp	577	2,163	4,763	146.57	159.44	148.71	✓
javimojamucho	154	704	1,457	144.43	145.74	160.45	✓
jnetclip	285	285	285	135.39	135.39	135.39	✓
kino	7,328	93,385	81,883	123.13	122.63	127.73	✓
minigames	428	428	404	87.96	87.96	90.35	✓
pycdk	104	152	152	88.32	131.74	131.74	✓
tankz	634	705	848	139.15	135.03	139.05	✓
timeedition	10,532	10,532	10,532	156.66	156.66	156.66	✓
tuxnotebook	12,353	12,375	9,903	117.96	117.9	114.31	X
unlesbar	720	1,974	3,713	159.78	155.53	152.72	X
utopicengine	2,710	2,947	3,370	136.7	141.58	141.19	✓
webappservicefi	151	913	1,136	144.11	129.09	129.09	X

by Heitlager et al. [11]. Rather than analyzing single compounds, we have used the MI as a system-level metric, but without inferring quality or maintainability in an absolute way, rather pointing at a relative change between points of the evolution.

Existing studies on the maintainability of OSS projects have shown that evolving software tends to decrease its quality and maintainability [1, 17]. We found that this is not necessarily true, given a pool of projects and analyzing their growth of lines of code and maintainability indexes.

Maintainability studies have been performed also to predict the quality of OSS projects using well-known design metrics [23]: using the MI metric as a response, and object-oriented metrics (such as the depth of inheritance tree, average response per class, etc.) as input factors, it was found that OSS projects written in Java show a dependency between the MI and the control flow of the program itself, hence validating the MI as a metric for maintainability. We used this result to acknowledge the value of MI as a metric.

Please cite as:
J Khondhu, A Capiluppi and K Stol (2013) Is It All Lost? A Study of Inactive Open Source Projects,
In: Proceedings of the 9th International Conference on Open Source Systems.

Finally, it has been noted that studies of maintainability in the OSS domain are flawed by the quality of data that is available for investigation [22]. Given the number of resources that OSS projects make available (change logs, mailing lists, source code, etc.), and the diversity in quality in each source, it becomes inevitable to avoid a strict comparison between projects to assess which one has more quality. What we produced in this study was rather an assessment, per project and not inter-project, of how these quality measurements change, and how to interpret them.

4.2 Limitations of this Study

We are aware of a few limitations of this study, which we discuss next. Since we have not studied any causal relationships, we do not discuss threats to internal validity.

Construct validity refers to the extent to which a concept is correctly operationalized, i.e., whether the researcher is measuring what he or she meant to measure. In particular, a key construct that we operationalized is abandonment of OSS projects, for which we defined three categories: active, dormant, and inactive. We relied on (ex-)maintainers of OSS projects who have indicated their projects to be inactive, using the `inactive` tag; active and dormant projects were distinguished by measuring the latest activity for each project (see Section 3). As shown in Fig. 1, results depend on how the terms "active" and "dormant" are operationalized.

Growth of OSS projects was another construct that we measured, which we operationalized using source lines of code (SLOC) as a metric. We believe that SLOC is the most useful metric to measure size, as it is source code that developers are dealing with. (For that reason, measuring size using memory footprint or number of files or classes, to mention two alternatives, is less useful). Important here is that we only measured the projects at three points: the initial point (size of first revision in the source repository), the final point (last revision in the source repository), and the middle point which is logically positioned between IP and FP.

External validity refers to the extent to which a study's findings can be generalized, or in which other settings the findings are valid. While the results of the classification of SourceForge projects into clusters has been performed on *all* in the SourceForge repository, we extracted three samples (size 25) that were randomly chosen from the three sets of projects (active, dormant and inactive). While the samples were chosen at random (using SQL's random statement), a different and larger set of samples may result in a different set of findings. Replication and extension of this study should be performed so as to overcome this limitation.

Please cite as:
J Khondhu, A Capiluppi and K Stol (2013) Is It All Lost? A Study of Inactive Open Source Projects,
In: Proceedings of the 9th International Conference on Open Source Systems.

Reliability of a study refers to the degree to which a study can be repeated while attaining the same results. In this paper, we have provided an extensive description of the research method and operationalization, as well as pointers to relevant sources and tools. This description can help other researchers to replicate or extend this study using the same methodology. While results of one study can be "revelatory," research results need to be replicated and confirmed with other studies and research methods, so as to triangulate findings.

5 Conclusion and Future Work

This paper positioned itself in the practice (and duty) of OSS developers to hand off their project to other developers when they lose interest in its further development. The study started off by clustering the population of SourceForge projects into three clusters, using both objective (i.e. the "inactive" tag that can describe an OSS project), and subjective indicators (i.e., the amount of days that a project remains without updates) to draw an updated picture of the status of the projects on one of the largest OSS portals. We isolated projects that are inactive because they have been moved to another portal, and other projects that have been classified as "stale," and therefore properly abandoned by the original developers. We further identified as "dormant" those projects whose latest date of activity was over a year from the date of the analysis (November 2012). Projects with updated activity were characterized as "active."

Using a sample of "active," "dormant" and "inactive" projects, we investigated whether differences could be detected in terms of the projects' overall growth: we found that "active" projects start generally as larger than "dormant" and "inactive," and that they grow consistently larger than the other two clusters.

Finally, we investigated whether projects in the different categories show differences between them by using the maintainability index (MI): we selected this metric since it produces a useful system-wide measurement, ensuring not to use it as an *absolute* measure, but rather observing its *relative change*. The study revealed that the majority of "inactive" projects' MIs increase or remain stable. This result is important for other developers and potential commercial stakeholders to evaluate whether a project is worth taking over, or whether its quality has degraded excessively.

We identified a number of potential research directions for future work. Firstly, this study should be replicated on a larger sample of projects, so as to confirm these results, which will help in improving the generalizability of these findings. Secondly, whether or not the maintainability index is a useful metric to reflect the quality of an open source project as perceived by commercial stakeholders remains an open issue. While this paper presents quantitative results, we believe that triangulation of these results through qualitative studies would be the next step so as to increase the confidence in the findings presented in this paper.

Acknowledgements. The authors would like to thank Dr Angela Lozano for

Please cite as:
J Khondhu, A Capiluppi and K Stol (2013) Is It All Lost? A Study of Inactive Open Source Projects, In: Proceedings of the 9th International Conference on Open Source Systems.

the feedback on an early draft of the paper. This work was supported, in part, by Science Foundation Ireland grant 10/CE/I1855 to Lero—the Irish Software Engineering Research Centre (www.lero.ie).

References

1. T. Bakota, P. Hegedus, G. Ladanyi, P. Kortvelyesi, R. Ferenc, and T. Gyimothy. A cost model based on software maintainability. In *28th IEEE International Conference on Software Maintenance (ICSM)*, pages 316–325, 2012.
2. V. R. Basili, G. Caldiera, and D. H. Rombach. *The Goal Question Metric Approach*. John Wiley & Sons, 1994.
3. A. Capiluppi, J. M. González-Barahona, I. Herraiz, and G. Robles. Adapting the 'staged model for software evolution' to free/libre/open source software. In *Ninth international workshop on Principles of software evolution*, pages 79–82, 2007.
4. A. Capiluppi, P. Lago, and M. Morisio. Characteristics of open source projects. *European Conference on Software Maintenance and Reengineering*, 2003.
5. A. Capiluppi, K. Stol, and C. Boldyreff. Software reuse in open source: A case study. *International Journal on Open Source Software and Processes*, 3(3), 2011.
6. A. Capiluppi, K. Stol, and C. Boldyreff. Exploring the role of commercial stakeholders in open source software evolution. In *Proceedings of the 8th International Conference on Open Source Systems*, pages 178–200, 2012.
7. R. English and C. M. Schweik. Identifying success and tragedy of floss commons: A preliminary classification of sourceforge.net projects. In *Proceedings of the 29th International Conference on Software Engineering Workshops*, ICSEW '07, 2007.
8. F. Fioravanti and P. Nesi. Estimation and prediction metrics for adaptive maintenance effort of object-oriented systems. *IEEE Transactions on Software Engineering*, 27(12):1062–1084, 2001.
9. Y. Gao, M. Van Antwerp, S. Christley, and G. Madey. A research collaboratory for the open source software research. In *First International Workshop on Emerging Trends in FLOSS Research and Development (FLOSS'07)*, 2007.
10. M. H. Halstead. *Elements of Software Science (Operating and programming systems series)*. Elsevier Science Inc., New York, NY, USA, 1977.
11. I. Heitlager, T. Kuipers, and J. Visser. A practical model for measuring maintainability. In *Proceedings of the 6th International Conference on Quality of Information and Communications Technology*, QUATIC '07, pages 30–39, 2007.
12. J. Howison and K. Crowston. The Perils and Pitfalls of Mining Sourceforge. In *Proc. International Workshop on Mining Software Repositories (MSR)*, 2004.
13. T. J. McCabe. A complexity measure. *IEEE Transactions on Software Engineering*, 2(4):308–320, 1976.
14. P. Oman and J. Hagemeister. Construction and testing of polynomials predicting software maintainability. *The Journal of Systems and Software*, 24(3):251–266, 1994.
15. E. S. Raymond. *The Cathedral and the Bazaar*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, revised edition, 2001.
16. I. Samoladas, I. Stamelos, L. Angelis, and A. Oikonomou. Open source software development should strive for even greater code maintainability. *Commun. ACM*, 47(10):83–87, 2004.
17. S. Schach, B. Jin, D. Wright, G. Heller, and A. Offutt. Maintainability of the Linux Kernel. *IEE Proceedings – Software Engineering*, 149(1), 2002.

Please cite as:
J Khondhu, A Capiluppi and K Stol (2013) Is It All Lost? A Study of Inactive Open Source Projects,
In: Proceedings of the 9th International Conference on Open Source Systems.

18. C. M. Schweik and R. English. *Internet Success: A Study of Open Source Software Commons*. MIT Press, Cambridge, MA, USA, 2012.
19. C. M. Schweik, R. English, Q. Paienjton, and S. Haire. Success and abandonment in open source commons: Selected findings from an empirical study of sourceforge.net projects. In *2nd workshop on Building Sustainable Open Source Communities (OSCOMM 2010)*, pages 91–101, 2010.
20. K. Stol and M. Ali Babar. A comparison framework for open source software evaluation methods. In P. Ågerfalk, C. Boldyreff, J. M. González-Barahona, G. Madey, and J. Noll, editors, *Open Source Software: New Horizons*, volume 319 of *IFIP Advances in Information and Communication Technology*, pages 389–394. Springer, 2010.
21. A. Wiggins and K. Crowston. Reclassifying success and tragedy in floss projects. In P. J. Ågerfalk, C. Boldyreff, J. M. González-Barahona, G. R. Madey, and J. Noll, editors, *Open Source Software: New Horizons*, volume 319 of *IFIP Advances in Information and Communication Technology*, pages 294–307. Springer, 2010.
22. L. Yu, S. Schach, and K. Chen. Measuring the maintainability of open-source software. In *International Symposium on Empirical Software Engineering (ISESE)*, 2005.
23. Y. Zhou and B. Xu. Predicting the maintainability of open source software using design metrics. *Wuhan University Journal of Natural Sciences*, 13:14–20, 2008.

Please cite as:
J Khondhu, A Capiluppi and K Stol (2013) Is It All Lost? A Study of Inactive Open Source Projects, In: Proceedings of the 9th International Conference on Open Source Systems.