JOURNAL OF SOFTWARE MAINTENANCE AND EVOLUTION: RESEARCH AND PRACTICE

J. Softw. Maint. Evol.: Res. Pract. 2012; **00**:1–40

Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/smr

Similarities, Challenges and Opportunities of Wikipedia Content and Open Source Projects

Andrea Capiluppi

Brunel University – London, UK

SUMMARY

Several years of research and evidence have demonstrated that Open Source Software (OSS) portals often contain a large amount of software projects that simply do not evolve, developed by relatively small communities, struggling to attract a sustained number of contributors. These portals have started to increasingly act as a storage for abandoned projects, and researchers and practitioners should try and point out how to take advantage of such content. Similarly, other online content portals (like Wikipedia) could be harvested for valuable content.

In this paper we argue that, even with differences in the requested expertise, many projects reliant on content and contributions by users undergo a similar evolution, and follow similar patterns: when a project fails to attract contributors, it appears to be not evolving, or abandoned. Far from a negative finding, even those projects could provide valuable content that should be harvested and identified based on common characteristics: by using the attributes of “usefulness” and “modularity” we isolate valuable content in both Wikipedia pages and OSS projects.

Received . . .

KEY WORDS: Evolutionary patterns; user-generated content; Open Source software; Wikipedia

*Correspondence to: Dr Andrea Capiluppi, Department of Information Systems and Computing (DISC)

Brunel University, Uxbridge UB8 3PH, United Kingdom. Email: andrea.capiluppi@brunel.ac.uk

Prepared using smrauth.cls *[Version: 2010/05/10 v2.00]*

1. INTRODUCTION

During the last few years, the content created and maintained online by users (and termed *User-Generated Content*, or UGC [36, 1]) has become substantial in quantity, relevant in quality [10], and centered around major topics and websites: multimedia (YouTube, Flickr, DeviantArt, etc), expert knowledge on specific topics (Street maps [17], `Slashdot.org`, IMDB, Wikipedia, etc), let alone all the source code released with open licenses through well known OSS repositories (SourceForge, Google Code, etc). The majority of such UGC's faces an "*uncontrolled, self-organized community of volunteer contributors, without the traditional tight conditions and organizational policies imposed in industrial production environments*" [27]. From the research standpoint, some of these UGC's (for instance, the Wikipedia pages and the source code of OSS projects) are based on measurable effort of volunteers and produce measurable output with a determined productivity that can be tracked throughout the content's evolution, by parsing the recorded history of changes.

Some of the similarities between OSS projects and specific UGC projects, such as Wikipedia, have been proposed and discussed in previous works [25, 27, 33]: these UGC environments are based on the *open source* model: people, not paid content-creators, use their time to contribute or revise knowledge and content, in a collaborative effort to create various "types" of publicly available knowledge-based products [12], although a mixed approach to develop OS software by paid developers has been documented, and it's becoming a very effective business model [8]. Furthermore, differently from other UGC, these two environments share interesting commonalities: both count on many distributed contributors; require some expertise before the contribution; are based on incremental changes on top of the same "base" artefacts; and store their evolution logs in an open and measurable way.

This paper analyses the source code produced within OSS projects, and the narrative page produced within Wikipedia: the main objectives of this study are to establish *similarities* in these two types of UGC, and whether the empirical results found in one can be applied to the other; to

Prepared using smrauth.cls
DOI: 10.1002/smr

identify the *challenges* that the two types face in their evolution; and to determine what *opportunities* are available from such content to users, researchers and practitioners.

From the findings of existing literature, there is a good understanding of the various *challenges* faced when evolving OSS systems: challenges in the coordination of efforts, in the continuous evolution and maintenance of content, in the recruitment of new contributors, of the complexity of the task at hand, let aside the *sustainability* of these effort by sparse communities. However, when analysing OSS projects alone, discording results have been reported in the past when analysing their evolutionary trends: on the one hand, some studies have claimed that the OSS model can achieve evolutionary patterns not observed before in proprietary systems [15, 16, 19, 28, 37]. On the other hand, empirical studies on randomly chosen OSS projects have shown that most of the OSS projects are small and inactive, and tend to involve a small number of developers and contributors [5, 13, 21, 34, 35]. One of the reasons for such diverse results is due to the "type" and "size" of the analysed systems: most of the OSS projects are small and developed by one developer alone [11], whereas very few OSS projects achieve a successful status [24, 7]. When randomly sampling, or considering the overall set of projects of large OSS repositories, the findings tend to be uniform and confirmed: the vast majority of OSS projects is small and developed by a small cohort of developers [6]. Conversely, when sampling a specific, representative sample of very large and successful OSS projects, the findings tend to be more sparse and optimistic: this has been found when analysing specific, successful OSS projects, including packages in the Debian/Linux distribution, or some of the flagship OSS projects (Linux, KDE, Apache, Samba, etc. [24]).

This paper is an extension of an earlier work [1], where it was shown that a sample of OSS projects and the pages composing one of the Wikipedia categories conform mostly to two behavioral models, termed "two-phased" and "three-phased" models, respectively. The present work extends the reported findings by increasing the studied samples, in order to obtain more significance in the statistics collected; by proposing a mathematical model to describe the common observed phenomena of UGC evolution; and by designing an approach to identify abandoned content that should be considered for inspection and for further reuse.

Prepared using smrauth.cls
DOI: 10.1002/smr

This paper is articulated as follows: section 2 describes the empirical approach and the definitions used throughout the paper. Section 3 presents a summary of the previous results obtained [1], that are expanded and put in a different context in this paper. Section 4 extends the results with more case studies, and by providing a mathematical reasoning to the aggregated results. Section 5 studies how to identify the value of Wikipedia pages based on the number of views and their modularity, while Section 6 identifies the value of OSS projects and components by analysing the number of downloads and their modularity. Section7 discusses the results from an overall perspective, and the implications for end-users and contributors to online content, Section 8 presents the related work, Section 9 illustrates the threats to validity, and finally Section 10 concludes and provides avenues to further studies.

2. RESEARCH QUESTIONS AND DEFINITIONS

This section details the research questions (RQs) and the formal definitions of the paper. It is divided in 2 subsections: in 2.1, three main research questions, and several subquestions, are derived from two overarching issues at the basis of this research; also, the types of metrics needed are introduced briefly. In 2.2, a more formal list of definitions is presented to clarify what are the key concepts and the terminology used throughout this study.

2.1. Research Questions and Metrics

This paper is built upon two main issues: the first is that some UGC environments face sustainability issues, and a mechanism to understand "how" and "how much" of this content has become unsustainable should be devised; the second issue is related to how to tackle this challenge, and to understand how unsustainable, abandoned UGC projects should be assessed to extract useful value, even if abandoned. The following research questions are grouped around three topics: the similarities between Wikipedia content and OSS projects, the challenges that both environments face in their evolution, and the opportunities that could benefit researchers and practitioners by extracting valuable content from both. They are formulated below to tackle the above issues:

Prepared using smrauth.cls
DOI: 10.1002/smr

$RQ_{similarities}$ – Do the OSS projects evolution and the Wikipedia pages evolution share some similarities? Does the input (by authors, contributors, etc.) measured in UGC projects follow a common pattern? Does the output (measured in wiki pages, source files, etc) produced in UGC projects follow a common pattern?

$RQ_{challenges}$ – Is it possible to differentiate between models of evolution, based on the type of UGC? Do OSS projects and Wikipedia pages face challenges in their evolution?

$RQ_{opportunities}$ – How to provide a classification of the value contained in UGC's? Is it modular? Is it useful? Is it used? Is it reusable?

In order to study the above research questions, the types of metrics that are needed can be categorized in four main groups:

- **Input metrics**: the effort of contributors was evaluated by counting the number of unique (or *distinct*, in a SQL-like terminology) contributors during a specific interval of time. The chosen granularity of time was based on months: different approaches may be used, as on a weekly or on a daily basis, but it is believed that months represent a more convenient way to gather the number of active contributors, as in man-month [4].
- **Output metrics**: the work produced was evaluated by counting the sum of created and modified items during the same interval of time. Each modification or creation affects at least one item, and it is recorded with a plain-text description (in OSS projects, by the relative versioning system; in Wikipedia, by the number of edits gathered during the evolution of a page, and collected by the Mediawiki and the statistics toolservers).
- **Modularity metrics**: in this paper we measure the "value" and the "quality" of the user-generated content by assessing how modular and how much reliant on external components such content is, and, conversely, its degree of modularity. Following common principles of "separation of concerns", and "encapsulation" [30] the value and the quality of any UGC project could be analysed to assess how modular the project itself, and the contained components, are. Although the reusability of a compound has to take into account

Prepared using smrauth.cls
DOI: 10.1002/smr

also external, environmental factors [29], its modularity and independence from others have long been established as necessary (although not sufficient) pre-conditions for their reusability [23]. In this paper we investigate the modularity of the components as an initial proxy for their reusability.

- **Usage metrics**: in order to summarise whether the analysed content provides value to the users, we studied its usage or usefulness: for UGC, these metrics should be based on the external users, by measuring how well such content is perceived. For Wikipedia pages, this was measured by monitoring the number of views that each page benefits along its life-cycle; for OSS projects, this was measured by monitoring the number of downloads by users, per project.

2.2. Definitions

The definitions that we used throughout this study are grouped below into "input and output" definitions, that are used to study how the UGC is created, modified and by who; "modularity" definitions, that are used to assess how modular and indipendent the elements of the UGC are, and to assess if they are used by other elements; and "usage" definitions, that are used to study how useful the UGC and its elements are. They are presented below in the same order.

2.2.1. Input and Output Definitions – The following are the definitions relative to the *input* and the *output* metrics:

- **User-generated content – UGC**: although no formal definition has been given, the Organisation for Economic Co-operation and Development (OECD) pointed out that UGC is any content based on participative effort, that is "published", "based on creative work" and "created outside one's professional routine work" [36].
- **Contributor**: in any UGC, a contributor represents either the original author of the artifact, or anyone else who provides additions or amendments later in the life-cycle of the same artifact. In the specific case of OSS projects, developers and bug-fixers represent the core contributors of a project (let aside users and bug reporters); in the case of Wikipedia, the

Prepared using smrauth.cls
DOI: 10.1002/smr

editors of wiki pages are the named contributors. Since in both OSS "projects" and Wikipedia "pages" the contributors leave traces behind of their actions, we recorded their presence in a similar fashion.

- **UGC item** (or "**item**"): any online content that can be created and modified freely by an online user. In the case of OSS, this could be represented by source files; in the case of Wikipedia, this could be represented by a wiki page. Files, classes, methods or functions, similarly to wikipedia lines and links are added, modified or deleted from the respective UGC. As for the UGC contributors, we recorded these items in a similar way to propose a similitude between the two types of UGC.
- **Created items**: number of new items that have been created during a certain period (daily, weekly, monthly, etc).
- **Changed items**: number of items that have been changed once or more per period.
- **Handled items (or *Total Handlings*)**: sum of the "newly created" and the "changed" items per period.
- **Productivity**: the productivity of the pool of contributors on a UGC project was evaluated through the following formula:

$$Productivity_{(i)} = \frac{Handled\_items_{(i)}}{Contributors_{(i)}} \quad (1)$$

where (i) is the referred period.

2.2.2. Modularity Definitions – The following are instead the definitions relative to the *modularity* metrics:

- **Fan-in**: this is the amount of incoming links to the studied item. For a Wikipedia page, it could be the number of other pages that have a link to it; for an OSS Java class, it could be the number of other classes that import it. An item with high fan-in represents an item providing often-needed services, which is regarded as an acceptable design behavior.
- **Fan-out**: this is the amount of outgoing links of the studied item, and it represents the needed services of an item, or the number of other items "controlled" by the item under analysis.

Prepared using smrauth.cls
DOI: 10.1002/smr

For a Wikipedia page, it could be the number of other Wikipedia pages that are cited in the page; for an OSS source file, it could be the number of other source elements needed by it. Generally a large fan-out is the symptom of poor design choices, since it breaks the principle of encapsulation.

- **Cohesion**: this is the amount of links that connect the item (or part of it) to (other parts of) itself. The cohesion of compound 'A' is stronger if items of A are connected to other elements also from A. If the connections are to different compounds, the cohesion is lower. A high cohesion indicates that the entity represents a single concept, and that it is autonomous to accomplish most of its functionality.
- **External references**: this is the amount of outgoing links that are directed to elements outside the system containing the item. These could be external references for a Wikipedia page, or an external library for an OSS project, package or class. External references show those items that are more "fragile" as their quality depends on external functionality.

2.2.3. Usage Definitions – In addition to these definitions, and in order to evaluate the *usage* of the two types of UGC, we also measure the aggregate number of views that a Wikipedia page receives monthly, and the number of downloads that an OSS project is downloaded by users, again aggregated per month.

3. PREVIOUS RESULTS

This section briefly reports the analysis of two samples of OSS projects and one category of Wikipedia pages [1]: the reported analysis and results lay the foundation for the rest of this paper.

3.1. Wikipedia Category: Data Extraction

A set of pages was studied from the Wikipedia portal, as an example of a massive UGC project. Wikipedia[†] is certainly the most important and diffused online UGC: as of February 2012, the

[†] http://en.wikipedia.org/wiki/Main_Page

Prepared using smrauth.cls
DOI: 10.1002/smr

English Wiki accounts for some 3,9 million articles. The category "Software Engineering" was randomly chosen and studied with its contained pages, sub-categories and sub-pages.

The *Mediawiki Special Export Interface* interface[‡] was used: given a specific Wikipedia category, its pages, sub-categories and sub-pages, the interface allows a researcher to extract an XML file containing all the revisions of these pages and categories, for offline analysis.

The *Mediawiki Dumper* tool finally produced a SQL dump from the previously downloaded XML dump[§]. The "revision" table containing all the revision histories (with timestamps and page references), the "page" table (with the titles and the IDs of each page), and the "text" table (with all the recorded revisions) were used to extract the requested information on effort and produced output. A set of SQL queries was designed in order to be applied in a similar way to the wiki page revisions, and the OSS projects histories.

3.2. OSS Projects: Data Extraction

The Sourceforge repository was chosen as the largest and most representative OSS repository. In order to extract a sample from it, the flossmole.org database [20] was downloaded, containing the basic information of all the Sourceforge projects. The latest available dump of such data, at the time of the extraction (April 2009) contained $126,142$ projects. From this database, three filters were applied: one based on the given status of the projects; one classifying the activity of the project, and imposed by the SourceForge site ("active" and "inactive"); and one relative to the specific topic (or application domain) that each project is developed for.

The most representative Status was found to be the *Production/Stable* one, with $25,877$ projects: other statuses ("Inactive", $3,720$ projects; "Planning", $28,315$; "Pre-Alpha", $22,003$; "Alpha", $23,528$; "Beta", $31,214$ and "Mature", $2,293$) were discarded in order to reduce the bias of inactive projects, or those at the very beginning of their life.

[‡] http://www.mediawiki.org/wiki/Manual:Parameters_to_Special:Export
[§] http://svn.wikimedia.org/svnroot/mediawiki/trunk/mwdumper/

Prepared using smrauth.cls
DOI: 10.1002/smr

The most representative application domains in Sourceforge were found to be “Internet” and “Software Development”, that were hence considered for the initial sampling of the projects.

In order to evaluate the sample sizes, and based on the sample populations ($5,103$ projects in the “Internet” domain, and $6,120$ projects in the “Software Development” domain), we chose the Confidence Level = 95% and the Confidence Interval = 10%. The requested sample size for the “Internet” domain was 94 projects; for the Software Development domain, the sample was 95. In both domains, we rounded the sample size to 100 projects.

An automatic procedure downloaded the CVS and SVN history-logs, that were used to obtain the information about the evolution of systems. Only SVN and CVS projects were considered: projects managed with different version control systems (such as Git or Mercurial) were replaced by other random projects. Projects without an active version control system (around 50%) were also replaced.

3.3. Operationalization – OSS and Wikipedia

Compared to other UGC environments, Wikipedia pages and OSS projects have similar ways of storing the data of past history and revisions. Therefore, from the history logs of both Wikipedia pages and OSS projects, the following fields were extracted for each atomic change recorded in the change log: the objective was to maintain a similar notation of attributes for the two types of UGC.

- month of change
- year of change
- contributor
- change type (creation or change)
- item name (either function or method for OSS projects; pages for Wikipedia categories)
- (only for OSS projects) subsystem name (either containing directory or package)

Each UGC change affects at least one *item*, and is recorded with a plain-text description. The same item could undergo many changes by the same contributor during the same period (month/year): in these cases, such changes were counted only once for the same module in order to avoid inflating excessively the activity of single contributors onto UGC projects.

Prepared using smrauth.cls
DOI: 10.1002/smr

Tables containing these fields were populated for the OSS projects and the Wikipedia pages: SQL queries extracted the *distinct* contributors per month (i.e., "man-month" [4]), and the monthly output. With these metrics, we were able to observe the evolution of both the contributors community working on the project and the output produced, and to evaluate the productivity of each month of the life-cycle. Since several glitches were found in the data of various projects, a cleansing phase was also performed, as described in the Appendix A.

3.4. Results – Wikipedia

The analysis of the pages composing the Software Engineering category of Wikipedia (900 pages, 17 sub-categories) indicated that individual pages have a varied array of evolutionary behaviors (as also observed in the subsections below). On the other hand, the aggregate, category-wise behavior showed three distinct phases: a "lazy-initialization", then a longer period of high activity in the middle, finally the curve tends to lower values. Figure 1 clearly shows these phases, when considering the aggregate behavior of all the pages composing the category.

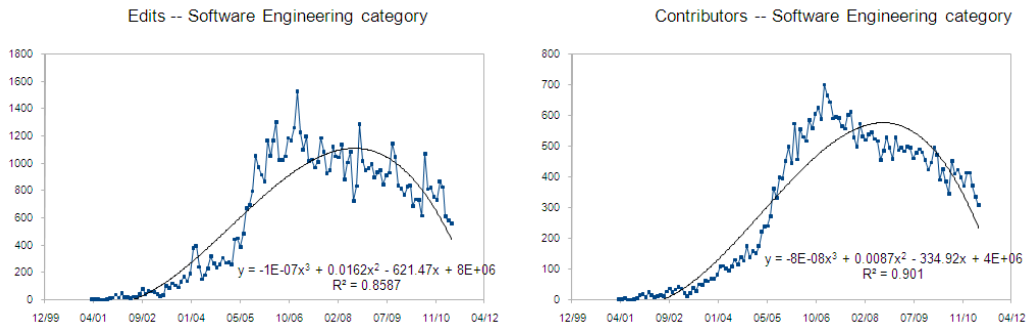

Figure 1. Wikipedia results – Number of edits (left) and number of distinct contributors per month (right)

3.5. Results – OSS projects: Two-phase models

As visible in table I, even when sampling only among the *active* projects, and by focusing on two SourceForge categories, the general results found regarding the bias of SourceForge still apply [13]. Small projects, and with a short-to-medium life-cycle are generally the majority under the OSS largest repository.

Prepared using smrauth.cls
DOI: 10.1002/smr

Category	Internet	Software Development
Sample	100	100
Analyzable (Logs $>$ 2 months)	76	82
Logs $<$ 1 year	28.95%	19,51%
1 year $\geq$ Logs $<$ 2 years	22.37%	14,63%
2 years $\geq$ Logs $<$ 5 years	31.58%	40,24%
Logs $\geq$ 5 years	17.11%	25,61%

Table I. Characteristics of the two studied OSS categories

The majority of the projects in the two samples were found to follow two main models: some 60% of the projects from the "Internet" sample and 52% of the projects from the "Software Development" sample show a recurring 2-phase pattern, described as follows:

1. in the first phase, the activity appears to be sustained with a relevant contribution in terms of created and changed items per month. The amount of contributors can see the addition of new people interested in creating new, or changing existing, items and content;
2. in the second phase, the activity of the contributors shows a considerable slow-down, sometimes dropping to a null activity, in terms of new or modified items. The number of contributors often drops to one, at which time the overall activity becomes even more sparse.

Figure 2 shows such pattern for two of the analysed projects, *epoz* and *xelem*[¶]. Especially for the *epoz* project, it is evident that it started to generate some interest from other contributors (up to 5), who helped in the creation and modification of the underlying contents. However, after this initial phase, the original project owner remained alone to keep the project going, which decreased the overall activity, and its sustainability (and the activity even drops to zero during some months).

The project in Figure 3 (*Emios*) illustrates the typical issue with productivity of two-phased projects: in the first phase of the life-cycle (until June 2005) the contributors activity is very

[¶] http://sourceforge.net/projects/xelem

Prepared using smrauth.cls
DOI: 10.1002/smr

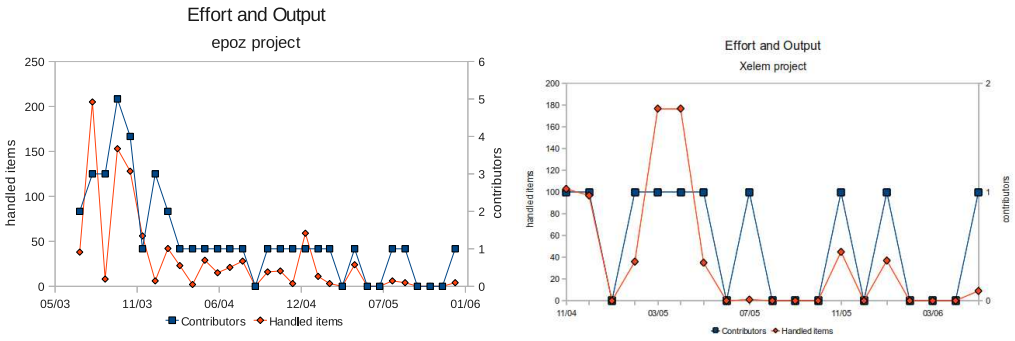

Figure 2. 2-Phases Pattern – *epoz* and *xelem* projects

high, going up to more than 400 Total Handlings for month. After that month, the activity slowly decreases and rarely exceeds 300 Total Handlings for month. More clearly, we can observe the productivity in Figure 3 (middle), where the relationship between the number of contributors and Total Handlings is considered. As we can notice, the productivity in the first half of the life-cycle often exceeds the 200 items/contributor threshold, while in the second half very rarely it exceeds the 100 items/contributor threshold. Considering the cumulative trend of productivity, 80% of the total productivity is expressed in the first half of the life-cycle, the remaining being deployed in the second half. In this case a Logarithmic Regression gives a coefficient of determination R-squared of 0,97% (Fig 3, bottom), that is a very satisfactory value.

3.6. Results – OSS projects: Three-phase models

For other projects, the observed evolution featured a first phase of slow evolution, a larger phase of many contributors and high activity, and a later phase of burnout (see Figure 4 left), while other projects have only two phases (large growth and burnout phase) recorded in their versioning system (see Figure 4 right).

The projects mentioned in section 3.5 above show quite clearly an initial phase, and a burnout phase, but somehow they fail to achieve the explosive phase of “growth and dissemination”. If new contributors are not added to OSS projects, it becomes less likely that such projects will produce an explosive phase of growth [7]; if only one contributor remains in the development, it is more likely that the project will observe a burnout phase.

Prepared using smrauth.cls
DOI: 10.1002/smr

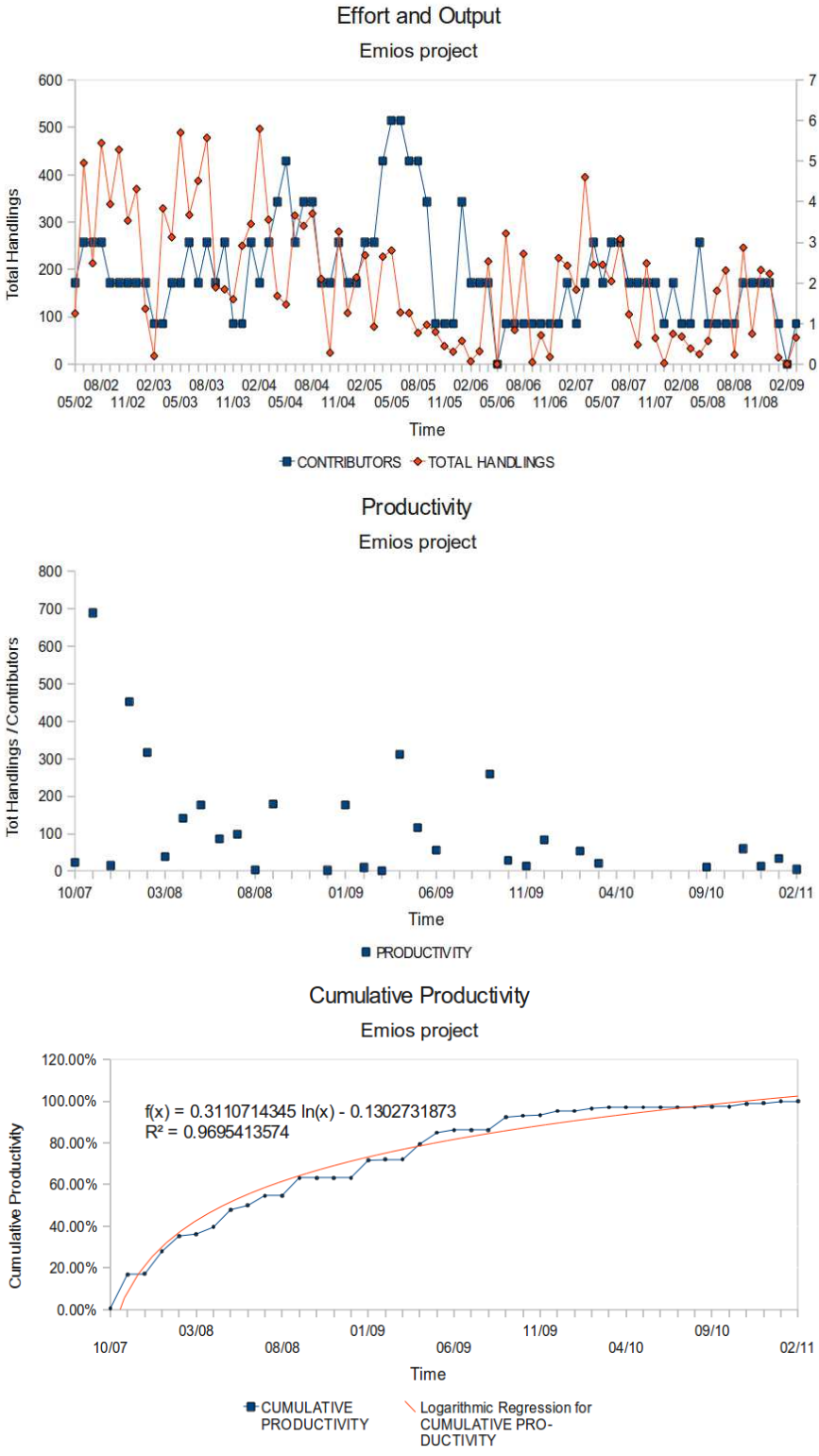

Figure 3. Emios project productivity analysis

Prepared using smrauth.cls
DOI: 10.1002/smr

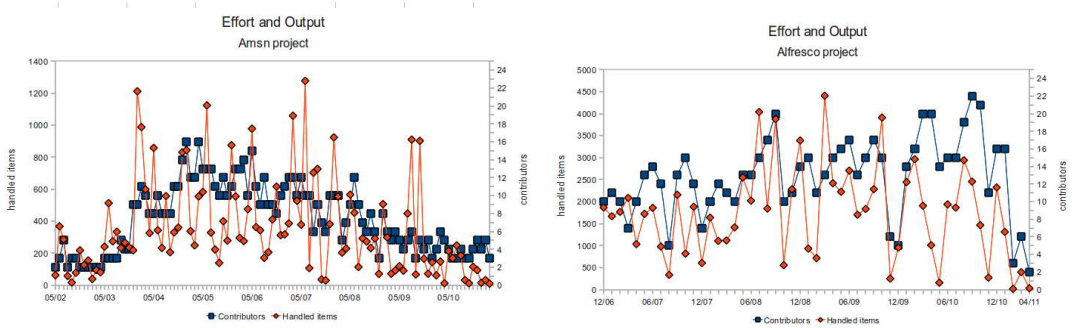

Figure 4. 3 Phases Pattern

4. AGGREGATING THE RESULTS

The results shown above are based on a two OSS samples and a small selection of pages from Wikipedia. The objective of this section is to expand on the above results, and to aggregate the individual facts obtained, especially when analysing the evolution and sustainability of the OSS projects.

4.1. Wikipedia categories

The evolution of the Wikipedia content was replicated by analysing the category "Chemistry"[‖]. As above, the amount of edits were counted, and the number of distinct contributors evaluated for every month of activity of each page, and for the whole category. Table II summarizes the number of pages and categories involved, as compared with the "Software Engineering" category presented earlier.

Category	Software Engineering	Chemistry
Subcategories	17	69
Pages	900	4,900
Edits	74,000	407,000
First recorded change	2001-04-02	2001-03-10

Table II. Characteristics of the two studied Wikipedia categories

[‖] http://en.wikipedia.org/wiki/Category:Chemistry

Prepared using smrauth.cls
DOI: 10.1002/smr

The images in Figure 5 show the amount of edits for the "Chemistry" categories (left), and the number of distinct developers editing the pages composing the category (right)**.

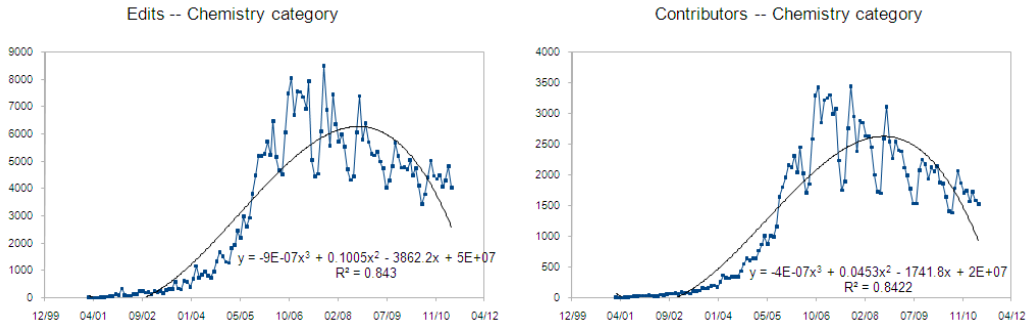

Figure 5. Wikipedia results – Number of edits per month (left) and number of distinct contributors per month (right) for the "Chemistry" category

The regression lines found to be more representative in the four trends of Figures 1 and 5 are polynomial fits of the third order ($y = ax^3 + bx^2 + cx + d$): for the evolution of edits and contributors of the Software Engineering and the Chemistry categories, adequate determination coefficients (R^2) were found in the curve fitting (0.8587, 0.901, 0.843 and 0.8422 respectively).

As mentioned earlier, the growth trends of individual Wikipedia pages are more varied, and do not tend to follow a polynomial pattern: table III summarizes the amount of pages whose cubic regression provided an R^2 larger or smaller than a threshold value of 0.8.

4.2. OSS Projects – Aggregated Results

The evidence shown above shows that, in specific OSS projects, the activity descends to zero, and no additional effort is provided to the project. A further experiment was set up to evaluate the pattern of evolution of the *overall* productivity of the two samples of OSS projects (and not only of single projects). In this case a database was used to store all the projects history in one table and SQL queries were used to evaluate the effort on the "overall" samples and the amount of contributions

**The number of contributors was evaluated by SQL queries, since the values found in the *soxred93* server were not accurate. The values of the monthly edits were also evaluated by SQL statements: when cross-checked with respect to the *soxred93* values, they were also confirmed as inaccurate.

Prepared using smrauth.cls
DOI: 10.1002/smr

Category	Type	$R^2 < 0.8$	$R^2 >= 0.8$
Chemistry	Contributors	4,172	21
	Changes	4,113	59
Software Eng.	Contributors	957	13
	Changes	958	12

Table III. Polynomial interpolation for single pages of Wikipedia

done in all the projects belonging to each of the samples. As we can see in Figure 6, the 2 analyzed samples satisfy the 2 phase pattern if we consider all the 100 projects in the sample as a unique project.

The figure clearly shows that productivity has a decrease in the second half: more specifically, a cubic interpolation is evident, especially for the number of contributors (See table IV). A proper goodness of fit is not found for the output of the SD sample: this mirrors what is found above for the single Wikipedia pages, when considering an individual page or a small sample compared with the overall category.

R^2	TotHandlings	Effort
Internet sample	0.8026	0.9266
SD sample	0.504	0.795
Overall Internet	0.8796	0.9329

Table IV. Goodness of fit of polynomial curves: determination coefficient R^2

In order to obtain a more evident proof, we decided to perform the effort analysis on all the $5,103$ projects belonging to the 'Internet' category. Results are shown in Figure 6 (bottom), and the last row of table IV, and they reflect what is found in the Wikipedia categories studied above: at an aggregate level, the whole category of the "Internet" projects has declining trends of effort and activity. This is reflected by the determination coefficient R^2 of the polynomial interpolations:

Prepared using smrauth.cls
DOI: 10.1002/smr

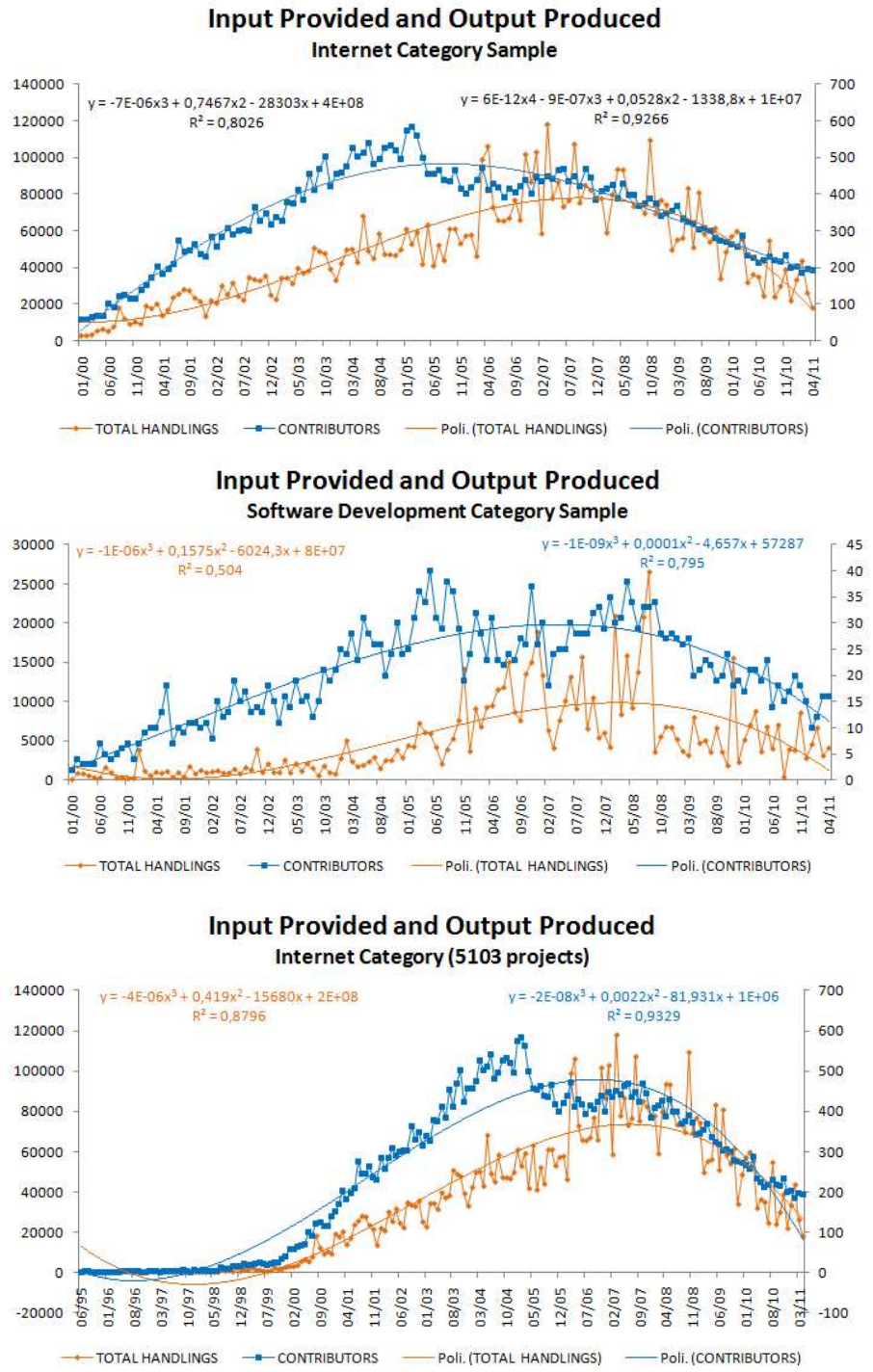

Figure 6. Productivity trend of the 'Internet' and 'Software Development' samples, and the whole "Internet" category

Prepared using smrauth.cls
DOI: 10.1002/smr

a slow start, a fast growth and a rapid descent of both effort and output produced signal an overall "failure" of the category (and not just of the single projects).

5. ASSESSING THE VALUE OF WIKIPEDIA PAGES

The curves shown in Figures 1, 5 and 6 show that, at an aggregate level, not only samples, but also large categories of UGC projects tend to involve less and less contributors over time, and face decreasing amounts of activity which could point to either "abandoned" or "completed" projects. In both meanings, since the core source of such UGC is available to anyone for modifications and additions, the fundamental question for researchers and practitioners is what to gauge from this enormous amount of freely available content. One immediate aspect to be considered is its harvesting, extraction and classification: within large OSS repositories, similarly to "car dumps" or"landfills", systems and components can be searched and their value assessed.

The two main aspects to be considered when harvesting such large collections of items are, at first, the identification of the value to be harvested; and secondly the extraction of such value, that could be even reused into other projects. This and the next sections deal with the issue of such identification, by focusing on the two studied characteristics that should be considered to determine the value of UGC projects, or part of them: usage and modularity.

5.1. Wikipedia Pages – Usage

This section analyses how to determine the value of Wikipedia pages based on their usage by interested readers, measured in number of views per page: when a large number of views is recorded for a page throughout the months, one can argue that the page is considered as a good source of knowledge, that could be promoted to a more established format (books, etc). Provided the list of pages contained in the studied categories, the number of monthly visits were recorded for each

Prepared using smrauth.cls
DOI: 10.1002/smr

page[††], in order to quantify the usage (and usefulness) of the pages in the sample. In general, the temporal data of such views can be grouped in three types:

- Slowly increasing/constant number of views: these pages act as point of reference for few dozens of views per month: after an initial phase, this pattern is quite constant throughout the months (see Figure 7, top). In a sub-set of 50 pages from the two samples studied, we manually checked and observed this pattern 16 times.
- Growing number of views: other pages show that the usage by readers is increasing throughout the months (see Figure 7, middle). In the sub-set of 50 pages, we observed this pattern 23 times.
- Very large number of views: some Wikipedia pages receive a number of views that is one order (or two) of magnitude larger than the rest of the pages. These pages seem to act as a major point of reference for interested users (see Figure 7, bottom). In the sub-set of 50 pages, we observed this pattern 8 times[‡‡].

In both the increasing number and the very large number of views, the impression is that such pages already provide value to the readers. By cross-checking the number of *edits* and the number of *views* on the same page, in some cases one interesting aspect emerges: this metric produces an inverse trend than the number of edits, low at the beginning and then increasing later, remaining quasi-constant in the latest months (see Figure 8, where the growing – then stable – number of views to the "View model" page is depicted). This can reflect the assumption that this page is complete, and it's being visited regularly as a reference by students and researchers. Such status should trigger some sort of recognition, and such pages could be promoted to the status of "stable", or "third-party checked", therefore properly reusable as references.

It is also possible to cluster these pages by averaging the number of visits per month: some 5% of these pages receive less than 10 views per month; some 9% receive between 10 and 100 views per month (on average); the majority of pages (54%) receives between 100 and 1,000 views

[††]By using the *json* service available at `http://stats.grok.se/json/`
[‡‡]In the remaining pages we observed a "descending" pattern of views 3 times

Prepared using smrauth.cls
DOI: 10.1002/smr

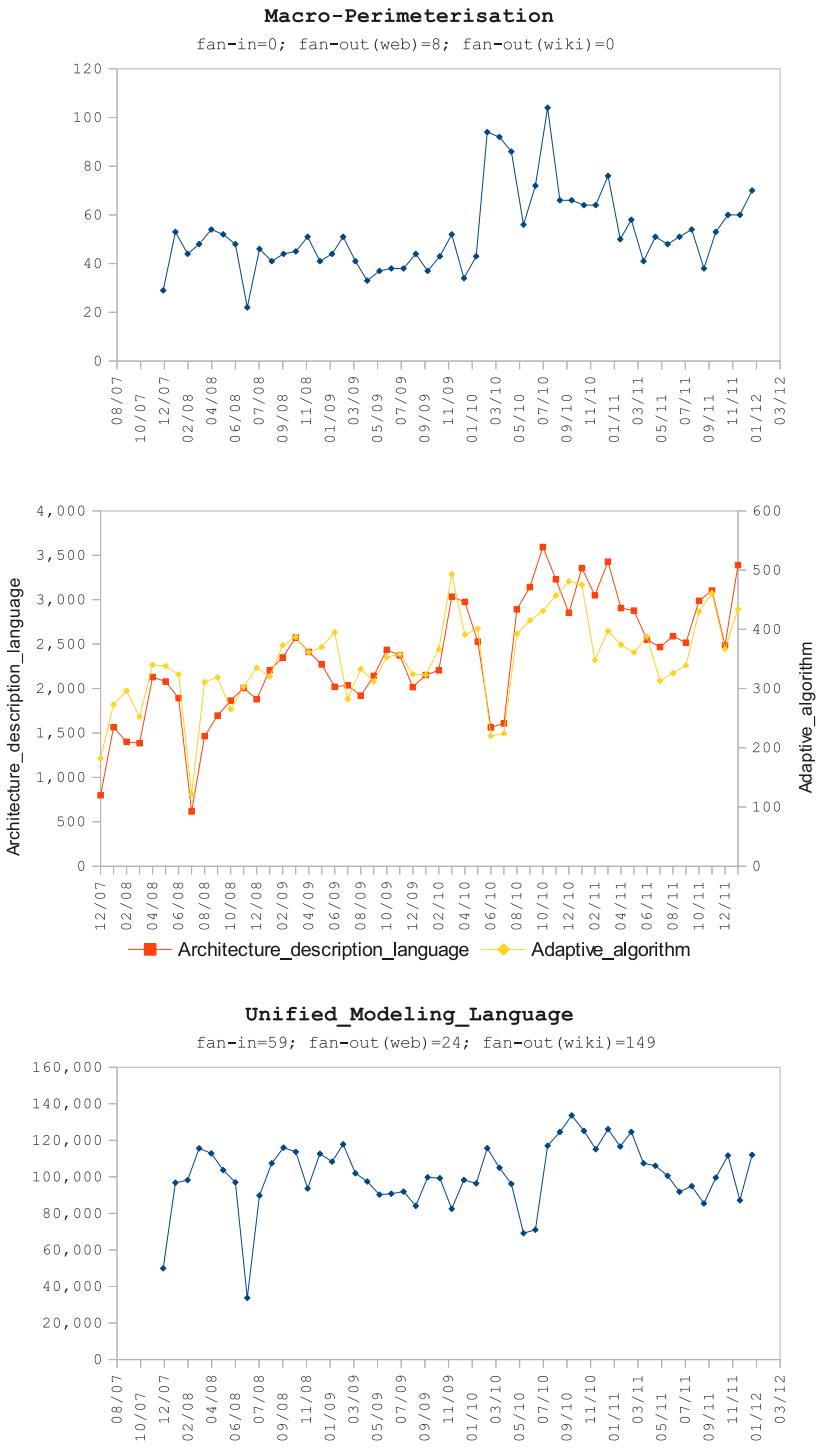

Figure 7. Wikipedia views and patterns – small and constant (top), increasing (middle) and very large and constant (bottom)

Prepared using smrauth.cls
DOI: 10.1002/smr

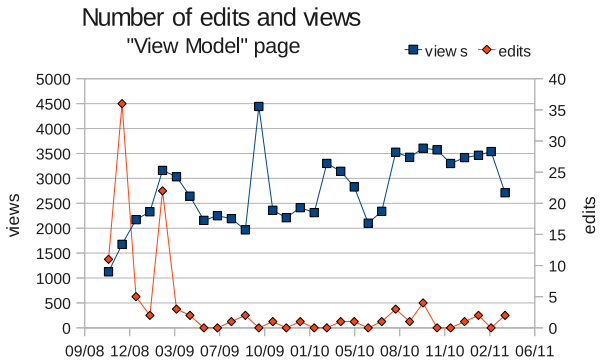

Figure 8. Wikipedia edits and views – “View model” page

per month; more than a quarter (26%) between 1,000 and 10,000; while a smaller portion (7%) receives more than 10,000 views a month. It becomes inevitable to put more value in the pages (the Unified_Modeling_Language page, for example) that receive a number of views which is one or more orders of magnitude larger than other pages.

The next subsection describes whether the value of these pages can be reflected also by their modularity, signaling not only “externally”-valuable resources, but also “internally”-modular ones.

5.2. Wikipedia Pages – Modularity

While it is interesting to observe how a certain page is accessed by users (as an “extrinsic” measurement of its usefulness), the above analysis does not show the “intrinsic” value of that page. As a measurement of the internal value of a page we studied its modularity metrics, adapting the fan-in, fan-out and “external references” concepts to these artefacts:

- Fan-in: for any given page, this is the number of other wiki pages that have a pointer to that page. To properly summarise “who’s calling who” in Wikipedia would require to analyse the whole namespace of Wikipedia: since this would take too long, the following analysis only studies whether a page is being referenced by other pages in the same category;
- Fan-out: for any given page, this is the number of links that are contained in the page: such links could be towards generic wikipedia pages (i.e., *Wiki Fan-out*), or towards page in the same category (i.e., *Cat Fan-out*);

Prepared using smrauth.cls
DOI: 10.1002/smr

- External references (Ext.refs, or *Web Fan-out*): for any given page, this is the number of referenced links that lay the boundaries of Wikipedia.

These measures are analysed in relation to the number of views: it would be important to establish a link between how much a page is viewed by users, and how it provides information also to other Wikipedia pages. Table V shows the Pearson's correlation coefficients between the four modularity metrics and the number of views. None of them is strongly significant (i.e., larger than 80%), but indeed it is found that the relation between the monthly average number of views has a 70% correlation (mild-to-strong) with the amount of Fan-in of the same page[‡]. Other relations are weaker, and show for example that there's little relation between the amount of links to external websites in a Wikipedia page, and the amount of other Wikipedia pages linked by the same page.

The usage and modularity analyses on the Wikipedia pages and categories point to the same result: pages with a large amount of Fan-in and sustained, large number of monthly views form the most valuable set of pages contained in Wikipedia.

	Ext. refs	Fan-out	Fan-in	Cat Fan-out
Fan-out	0.45			
Fan-in	0.14	0.22		
Cat Fan-out	0.23	0.51	0.32	
Avg Nr. of views	0.17	0.33	*0.70*	0.17

Table V. Pearson's coefficients between modularity metrics and monthly average number of views

[‡]The measured Fan-in is a lower-bound of the real one: for the page X, only the pages in the *same* category pointing to X are counted as X's Fan-in, although there could be other pages (outside that category) that have a link to X.

Prepared using smrauth.cls
DOI: 10.1002/smr

6. ASSESSING THE VALUE OF OSS PROJECTS

6.1. OSS Content – Usage

In a similar fashion, we studied the usage of OSS projects by analysing the number of downloads that OSS projects benefit while on their SourceForge websites. A similar set of patterns was also found: few projects benefiting from a semi-constant, large number of downloads; projects observing a seasonal, but overall increasing, amount of downloads; and other projects showing lower and scattered number of downloads.

Apart from those three patterns, we could also observe a fourth one: after a period of relatively large number of downloads, some projects show a sharply descending number of downloads, that essentially degrades to zero. These projects show all the characteristics of unsustained effort, where the contributors do not provide any more code to the project: in order to make a case of extracting value from "abandoned" UGC, in this section we only focus on such projects that could clearly be considered as not evolving. We did so by only selecting the projects showing the "Inactive" special tag: the SourceForge website identifies projects with a low activity, and labels them "inactive"*. In the overall pool of analysed projects, we found 100 projects that show that tag in their SourceForge pages.

Through the SourceForge *json* service† , it is possible to analyse the monthly number of downloads of the inactive projects. In general two patterns can be observed: one, very common, showing that an initially large number of downloads decays until becoming practically null; the second one, less frequent, showing how the available files of a project are still downloaded, even though the project was declared inactive.

It becomes clear that the vast majority of the abandoned projects experiences a sharp decline in the number of downloads, showing that from the users perspective those OSS projects are no longer containing valuable resources. Even if, as pointed out in Israeli et al [22], constant activity

*See for instance the description of the *csstidy* project, http://sourceforge.net/projects/csstidy/
†http://sourceforge.net/p/forge/documentation/Download%20Stats%20API/

Prepared using smrauth.cls
DOI: 10.1002/smr

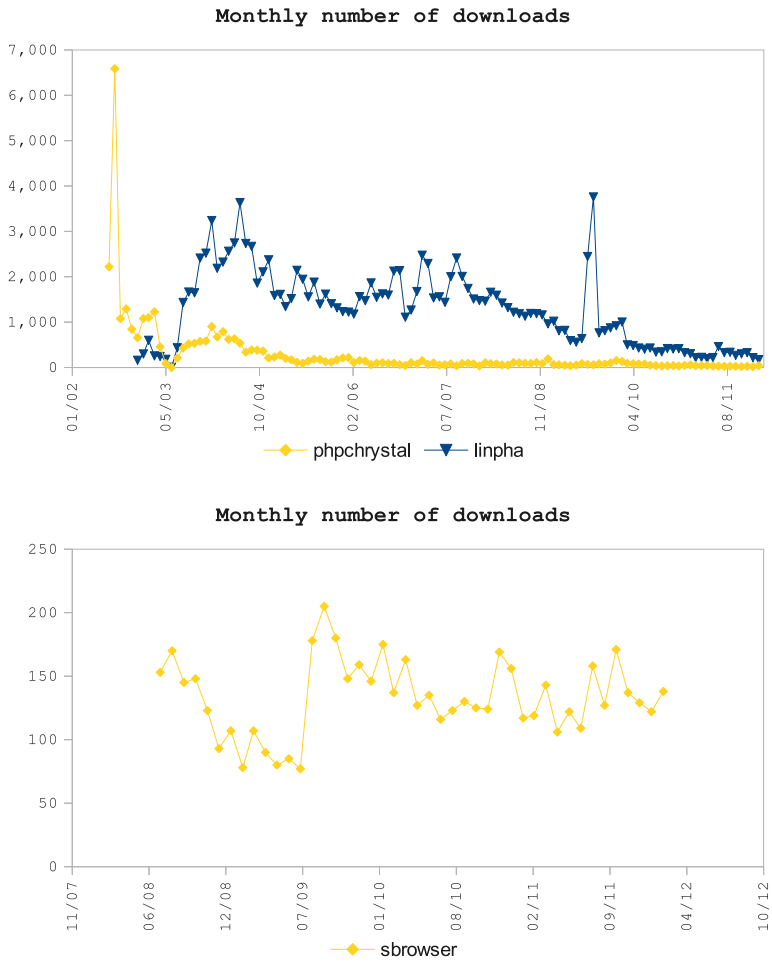

Figure 9. Decreasing (top) and stable (bottom) number of downloads in "inactive" projects

is not always related to users' downloads, and few OSS projects can sustain a constant number of downloads, these projects seem to decline in the productivity by developers, and in the number of end users.

From the developers' perspective, the second question that should be clarified is whether any component of such abandoned projects could be salvaged, based on aspects of internal modularity, and possibly reused in other projects.

6.2. OSS Projects – Modularity

This section deals with the modular characteristics of the OSS projects: it is based on the analysis of one the inactive projects mentioned above (Xelem, see Figure 2): while not conclusive for all

Prepared using smrauth.cls
DOI: 10.1002/smr

the OSS inactive projects, the approach to study the modularity of a system and its packages (or components) is open and reproducible. The same analysis is performed at large in the next subsection.

Differently from the Wikipedia "pages", this section uses the Object Oriented "packages" as the modular characteristics of the Java OSS projects: the value of these "parts-of" UGC is assessed based on the principles of modularity, by making sure that they achieve enough encapsulation, while keeping the dependencies to the external resources to a minimum. The "value" of the OSS content is assessed briefly in the following using well accepted and diffused principles of modularity: in particular, the principles of "High coupling and low cohesion", "information hiding" and the principles of "package design" are put into practice to define if the packages (as components) can be given a higher or lower value mark as long as they respect more or less such principles.

Three releases of Xelem (i.e., r1.0, r2.0 and r3.0) were analysed under the modularity perspective. The results of such extraction show that r1.0, released in 2004-12-10, is composed of 9 packages: the total number of calls performed by all classes in this release is 1,093. It is also visible that 2 of the packages (*nl.fountain.xelem.excel.ss* and *nl.fountain.xelem.excel*) alone produce more than 80% of the total calls: the other packages perform only the 14% of the calls, so one can conclude that most of the logic for this specific project is contained within these two packages.

When analyzing the other two available releases of Xelem project (r2.0 and r3.0), despite the growth of the project in terms of number of packages (14 in r3.0) and number of total calls (1,806 in r3.0), the relevant packages which perform more than 80% of calls are still the same as before, plus another package which however is a sub-package of *nl.fountain.xelem.excel*. The core of the Xelem project has always remained stable during the evolution cycle of the entire project: this could be considered as a first hint to signal valuable resources. Figure 10 shows the two releases, and how the evolution in growth did not affect the overall composition of the architecture and its components: the "Calls IN $\rightarrow$ IN" represents the percentage of calls which start from the core of the specific project and terminate in the core itself (i.e., its cohesion); the "Calls OTHER $\rightarrow$ IN" represents the percentage of calls which start from the others packages of the specific project and terminate in that

Prepared using smrauth.cls
DOI: 10.1002/smr

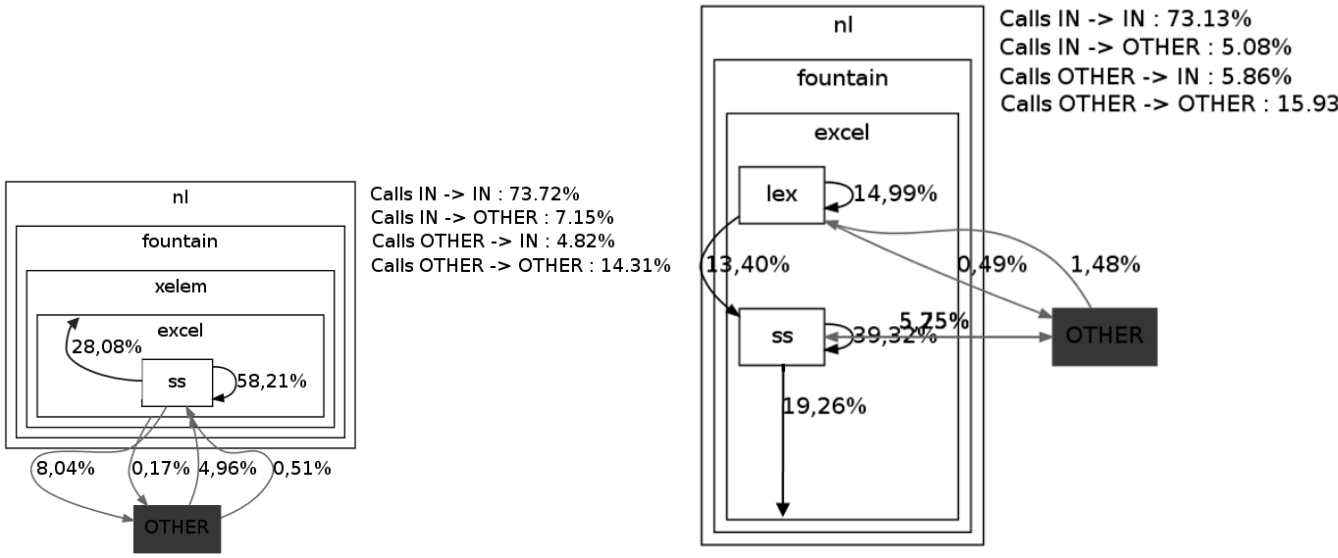

Figure 10. Xelem Architecture Digraph – Releases 1.0 and 3.0

packages that are the core (i.e., the Fan-in); finally the "Calls IN → OTHER" signals the amount of calls from the core to the other packages (i.e., the Fan-out).

The identification of such resources becomes a key issue: without a proper explanation of what these resources achieve, it becomes quite difficult to realize what is inside these packages, even if their modularity is good. A short description of those packages as follows would be needed to provide additional value:

- *nl.fountain.xelem.excel*: provides interfaces and classes that represent elements in SpreadsheetML (the XML schema for Microsoft Office Excel 2003);
- *nl.fountain.xelem.excel.ss*: provides classes that represent the elements of a spreadsheet's namespace;
- *nl.fountain.xelem.excel.ss.lex*: provides classes that enable reading SpreadsheetML.

With these descriptions and packages available, a possible way to reuse this sub-architecture could be by wrapping these packages, which are known to be stable during their evolution, and by making an opportunistic reuse of a large amount of code, and avoiding to "reinvent the wheel".

Prepared using smrauth.cls
DOI: 10.1002/smr

6.3. *Modularity and Abandoned OSS Projects*

A similar analysis to the one performed in 6.2 was replicated for all the Java projects that are flagged out with the “inactive” tag in their SourceForge pages. Out of the 100 inactive projects, 18 are written in Java, and an overall 520 packages were analysed, with regard to their Fan-in, Fan-out, Cohesion and reference to External libraries.

Among all the studied packages, the maximum observed Fan-out is an overall 510 calls to 56 different packages; the maximum observed Fan-in is an overall 2,606 calls from 96 different packages; the maximum observed Cohesion is 548 calls (to itself); while the maximum number of external references is 946 calls to 12 different external libraries (and excluding the standard java.* and javax.* libraries). Apart from these maxima, a large variability is observed in the modularity metrics of the packages studied, which makes it difficult to summarise them graphically. However, by summing up the Fan-in, Fan-out, Cohesion and External References, we obtain the percentages of each metric, whose statistical distribution is summarized in the box-plots of Figure 11: examples of how this was done are reported in table VI below.

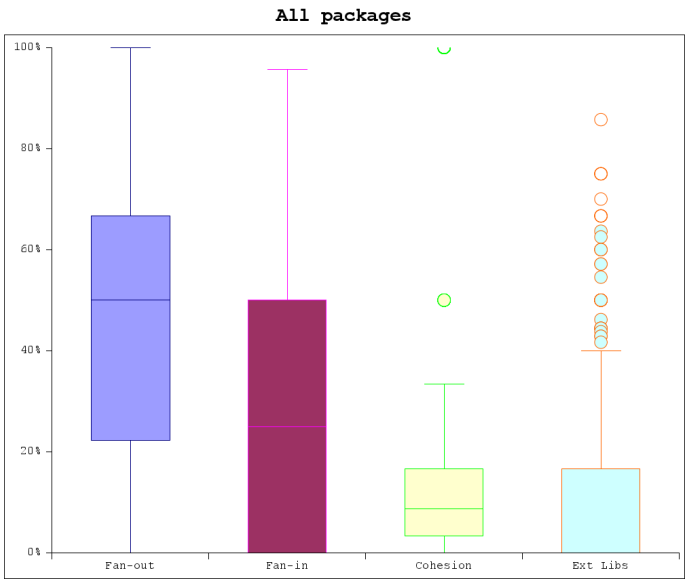

Figure 11. Boxplots: relative distributions of Fan-in, Fan-out, Cohesion and References to External Libraries, in the 540 abandoned packages

Prepared using smrauth.cls
DOI: 10.1002/smr

The Fan-in, Fan-out, Cohesion and External References are evaluated first at a lower level of granularity, by counting the amount of method calls between methods: package p1 has 129 inbound calls (Fan-in), 33 outbound calls (Fan-out), 41 calls to elements contained in itself, and 946 calls to external libraries, totaling 1,149 calls. This produces a percentage Fan-in of 11.23%, a Fan-out of 2.87% and so on. On the other hand, the 5^{th} row of table VI uses a coarser level of granularity, showing that the same package p1 has calls to 12 different packages (Fan-out), receives calls from 5 distinct packages (Fan-in) and issues calls to 10 different external packages.

	F-out	F-in	Cohe	ExtLib	SUM	F-out (%)	F-in (%)	Cohe (%)	ExtLib (%)
METHOD CALLS									
p1	129	33	41	946	1,149	11.23%	2.87%	3.57%	82.33%
p2	0	0	18	143	161	0.00%	0.00%	11.18%	88.82%
p3	0	244	59	5	308	0.00%	79.22%	19.16%	1.62%
p4	1	26	5	0	32	3.13%	81.25%	15.63%	0.00%
PACKAGE CALLS									
p1	12	5	1	10	28	42.86%	17.86%	3.57%	35.71%
p2	0	0	1	6	7	0.00%	0.00%	14.29%	85.71%
p3	0	21	1	2	24	0.00%	87.50%	4.17%	8.33%
p4	1	5	1	0	7	14.29%	71.43%	14.29%	0.00%

Table VI. Modularity characteristics at the method- and package-levels for 4 example packages

It becomes interesting to notice how the packages p1 and p2 differ from p3 and p4: in both p1 (from the inactive *jrdf* project) and p2 (from the inactive *uitags* project), the number of calls to other elements, either internal (Fan-out) or external (ExtLib), is much larger than the Fan-in and Cohesion (both at the method- and package-level). This makes p1 and p2 very brittle, and dependent on other elements, which is not an ideal design characteristic. On the other hand, the packages p3 (from the *tagtraum-jo* inactive project) and p4 (from the *simplessite* inactive project) behave very

Prepared using smrauth.cls
DOI: 10.1002/smr

differently: their Fan-in and Cohesion, at the method- and package-level, present the attributes of a well-encapsulated and independent package, that rely in a very limited way on external resources. Although abandoned, these packages present the characteristics of excellent modularity that is a necessary albeit not sufficient) condition for their reuse into other projects.

These findings have the potential of pointing with sufficient precision the packages that could be evaluated for potential reuse, given their modularity characteristics. Given the overall pool of 520 packages from inactive projects, in over 100 packages the sum (Fan-in + Cohesion) is at least twice as big as the (Fan-out + ExtLib) sum. Although not completely conclusive, it should be possible to enforce these (and other) modularity conditions to scope down a reduced list of potentially reusable resources to analyse further.

7. DISCUSSION AND IMPLICATIONS

From this and other research papers, it is becoming clear that the model proposed by the Open Source advocates provides users with a very large quantity of software at no cost, developed in an open way, trusted and deployed also by large corporations. The other side of the Open Source phenomenon shows that most of these projects do not reach a large developers' community, let alone a large audience of users. What was shown in this paper is therefore part of a larger user-generated content phenomenon, that seems to undergo similar patterns of growth and, more in general, of evolution.

One counter-measure to the observed declining patterns in OSS products is more and more often the inclusion of companies to sponsor the development [8, 9], at various level of involvement, from simple partners, to contributors, and even as project coordinators [3]. In other cases, and when corporate involvement seems less likely, a very large number of OSS projects lay amassed in open portals, without being developed further. The same finding was observed for the Wikipedia content, and similarly to OSS projects, categories loose contributors and activity at an aggregate level.

Prepared using smrauth.cls
DOI: 10.1002/smr

The implications of these findings are two-fold, and they span all sorts of user generated content, from knowledge-sharing sites (e.g., Twitter, blogs, etc) to Questions & Answers sites (e.g., StackOverflow), to Wikipedia and Open Source projects.

7.1. Implications for end-users

From the end-user point of view, corporate businesses and normal users would not want to invest money or time to invest in abandoned content, with no followers and no commitment in the maintenance of its features. On the other hand, providing guidelines to distinguish between "inactive and abandoned" and "inactive but still used" content could serve as valuable information to end-users who could decide to deploy unmaintained, but largely accessed content. Similarly, corporations could even decide to start sponsoring an inactive project, if it contained clearly identified and valuable resources.

7.2. Implications for contributors

From the contributors' point of view, two implications should be considered: (1) reuse and (2) identification of such content. From the *reuse* perspective, the quality of the content itself could be identified through traditional software engineering or modularity metrics, in order to separate single, modular components from the whole. These modules should be extracted, assessed, described and made available in an open way, in order to facilitate their reuse in other projects.

From the *identification* perspective, the original authors of abandoned content should take the responsibility to inform others that they are not interested in supporting or enhancing their systems, and to distinguish their between "inactive" and "abandoned" content. This should be done for two fundamental reasons: first, to be able to clearly distinguish "inactive but alive" projects from the "abandoned" ones, and to give a clearer message to the interested end-users; second, to implement one of the most powerful features of OSS development, namely the ability for other developers

Prepared using smrauth.cls
DOI: 10.1002/smr

to “take over” abandoned projects, in order to sustain a new phase of development, managed and sustained by someone other than the original creators*.

8. RELATED WORK

The concept of UGC is not a new one: it has been described as based on participative effort by a cohort of contributors, it has to be published in some common format and in an open way, it has to be based on creative work, by using and showing technical skills, and it has to be created outside one’s professional routine work, as to say that it should not be part of daily work requirements [36]. The contributors producing such content have been categorized in various ways, either “active”, “passive spectators” and “inactives”, or even by theorizing the various roles that contributors tend to adhere to (“critic”, “active creator”, “collector” or “joiners”) [26].

The paradigm shifts caused by UGC have been multiple: cultural, since recipients have become participants in the creation of content; economic, with the advent of “prosumers”, who are both producers and consumers of online content; and labour-related, since amateurs compete with professionals in the production of content whose quality can be monitored by peers [32].

As a prime example of content generated by hobbyists and non-professionals, the evolution of OSS has attracted attention by researchers in particular when trying to compare “open” and proprietary systems, in order to test, in an open development, the validity of the now widely accepted “laws” of software evolution [15]. As mentioned above, some studies have demonstrated that a number of flagship OSS projects achieve patterns of evolution that have not been observed in proprietary systems: among others, Godfrey et al. [16] demonstrated that the Linux kernel achieved a superlinear evolution; Herraiz et al. [19] examined the growth of 13 OSS systems (mostly from the Debian/Linux distribution), also concluding that the predominant mode of growth was superlinear;

*The process is detailed, for the SourceForge repository, at

`http://sourceforge.net/apps/trac/sourceforge/wiki/Abandoned%20Project%20Takeovers`.

Prepared using smrauth.cls
DOI: 10.1002/smr

similarly other studies, based on a selection of large OSS projects, claimed that some of such projects can practically achieve a linear or a superlinear growth [15, **?**, 28].

On the other hand, further empirical studies on OSS projects have highlighted recurring issues and results that researchers face when mining large repositories of OSS projects, the most important and diffused one being SourceForge: it has been reported that most of the OSS projects are small, tend to involve very few developers, and they are mostly inactive, [5, 13, 35] let alone being actively used by users after the download [14, 21, 34].

One of the reasons for such diverse results is due to the type of analysed systems, and the selection process: when randomly sampling, or considering the overall set of projects of large OSS repositories, the findings tend to be uniform and confirmed. Conversely, when sampling a specific, representative sample of successful OSS projects, the findings tend to be more sparse and optimistic: this has been found when analysing specific, successful OSS projects, including packages in the Debian/Linux distribution, or some of the flagship OSS projects (Linux, KDE, Apache, Samba, etc.). This paper has tackled this second approach to sample OSS projects, in order not to highlight "exceptional" characteristics, but to propose a more general view of what is found in large OSS repositories, and whether "trash" or "treasure" could be identified from the reuse point of view.

The evolution of Wikipedia has also attracted several studies: what is interesting to note is a certain point in time (in 2007) where researchers reported a change of slope in the rapid growth of this online resource. Studies reporting the evolution of Wikipedia before and until 2007 have shown that the number of pages and content grew exponentially, and that this was due to the fast growth of its users and contributors base [2]. A later article proposed instead that the Wikipedia growth can be expressed by a logistic curve, and that "*the population of Wikipedia editors is exhibiting a slowdown in its growth due to limited opportunities to make novel contributions*" [31]. This point was later confirmed either as a real danger for the whole ecosystem within Wikipedia [27], or simply as a normal issue in large system, where growth goes through different stages, modeled by different growth models [25]. Interestingly, the proposal that any UGC (as Wikipedia, or the OSS content) could be uniformed in a "lightweight" or a "heavyweight model for peer production could

Prepared using smrauth.cls
DOI: 10.1002/smr

produce the first differentiation between sustainable and abandoned (or completed) content [18]: an "heavyweight" model in fact produces stronger-ties between participants and better negotiation of purpose.

9. THREATS TO VALIDITY

Like any other empirical study, the validity of ours is subject to several threats. In the following, threats to *internal validity* (whether confounding factors can influence the findings), *external validity* (whether results can be generalized), and *construct validity* (relationship between theory and observation) are illustrated.

9.1. Internal Validity

The following threats to internal validity have been detected:

- As a metric for the input, we used the number of distinct contributors per month. This was done to compare the OSS data with what is available in the Wikipedia edits (also recorded monthly). This means that if a developer only makes a small change in one month, while another does the vast majority of the coding (or editing), we will be counting 2 contributors for that given month.
- As a metric for the output in Wikipedia, we used the total number of edits to the pages composing a given category. We are aware that some of the edits are recorded as "minor" by the *soxred93* toolserver: since we were not completely confident on the accuracy of the "number of edits" metric, let alone the "number of minor edits", we preferred to avoid a cumulation of weakly defined metrics, and just use the "number of edits" alone.
- Apart from their CMSs, OSS projects can be also measured by analysing stable points of development (i.e., releases) while Wikipedia only evolves through small or large changes which are not consistently tagged as 'major' or 'minor' by the contributors. For this reason we use "incremental changes" (i.e., maintenance) as a synonym for "evolution" in both UGC

Prepared using smrauth.cls
DOI: 10.1002/smr

environments. We believe that for sufficiently long periods of analysis the terms "evolution" and "maintenance" can be assimilated.

9.2. External Validity

Recognised threats to external validity are:

- the OSS results apply to a sample of projects from Sourceforge. Surely enough another, higher-profile repository (Debian, KDE, Gnome, etc) will yield different results, in terms of less "one-man bands". The study nonetheless highlights that something precious is hidden in a large repository as Sourceforge, where a large percentage of projects simply do not evolve;
- the Wikipedia sample is also very small compared to the population: instead of randomly extracting Wiki pages, we decided to analyse two full categories. Obviously the results could change considerably by choosing other categories. We are confident though that a random extraction of pages (not categories) from wikipedia could yield similar results to ours.

9.3. Construct Validity

Further threats to construct validity are:

- we assumed that the categorization of "stable" and "active" projects as recorded by Sourceforge is accurate. Since this is not a parameter that we could control, we had to accept it "as is" since all the hosted OSS projects are classified by that metric;
- we assumed that the data of the Wikipedia edits contained in the *soxred93* toolserver are appropriate. Since we saw discrepancies in the measurement data, the data on contributors was re-evaluated by using the dumps by the MediaWiki website

10. CONCLUSION AND FUTURE WORK

This paper is an attempt to define if there are similarities in the evolution of Wikipedia content and OSS projects, as expressions of UGC environments; if both face challenges in their evolution; and whether opportunities can be found in the harvesting of value from these resources.

Prepared using smrauth.cls
DOI: 10.1002/smr

This paper argued that the evolution of Wikipedia pages and the OSS projects share some commonalities in terms of their evolutionary patterns; in particular, it was found that a predefined, cubic model could be used to explain several of the similarities in “abandoned” or “completed” projects and Wikipedia pages. Furthermore, it was also found that at the aggregate level, such behavior is more evident than at a lower level of granularity (e.g., a single page of Wikipedia, or a single OSS project). Large UGC content can be modeled by a cubic function depicting a logistic model, with an initial phase of slow growth, a longer phase of more sustained growth, and finally a phase of burnout. Smaller UGC content is more irregular, and modeled by a two-phase pattern, i.e., the explosive growth phase does not happen.

On the other hand, this paper showed that abandoned (or completed) content in Wikipedia pages or from OSS projects represents a valuable resource for potential reuse: by investigating completed OSS projects, one can find valuable components ready to be reused; by investigating frequently visited Wikipedia pages, one can define third-party checked knowledge that does not need further amendments. By investigating externally-measured value, as the number of views for the Wikipedia pages, and the number of downloads for the OSS projects, it is found that very successful UGC tend to be accessed one or two orders of magnitude more than other content. In some cases, such usefulness is also equipped by structural modularity, in which the underlying content does not require external references to be independent. By leveraging this modularity concept, we finally focused on the Java packages of abandoned OSS projects, arguing that even abandoned but modular components could provide valuable resources to be harvested and reused.

The findings of this paper, if confirmed, have several open avenues of development and research: first, other UGC sites could be added to this analysis, by parsing for instance the results of technorati (`http://technorati.com`), that can browse for a large amount of user-generated media (including weblogs). Second, the effects of the three phases should be clarified: are all UGC’s destined to end up in a burnout phase? What is the repercussion of such finding? Third, the potential of abandoned projects to be reused (fully or in part) as resources in other projects should be studied,

Prepared using smrauth.cls
DOI: 10.1002/smr

by evaluating techniques to automatically index and list resources which comply with the basic requirements of reuse.

11. ACKNOWLEDGEMENTS

The author would like to thank G.M. Alluvatti, G. De Ruvo and Marco Molfetta for the discussion and feedback during the course of their Erasmus placement at University of East London during the summer 2011.

REFERENCES

1. G. M. Alluvatti, A. Capiluppi, G. De Ruvo, and M. Molfetta. User generated (web) content: trash or treasure. In *Proceedings of the 12th International Workshop on Principles of Software Evolution and the 7th annual ERCIM Workshop on Software Evolution*, IWPSE-EVOL '11, pages 81–90, New York, NY, USA, 2011. ACM.
2. R. Almeida, B. Mozafari, and J. Cho. On the evolution of wikipedia. In *International Conference on Weblogs and Social Media*, 2007.
3. K.-J. S. Andrea Capiluppi 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*, 2012.
4. F. P. Brooks, Jr. *The mythical man-month (anniversary ed.)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.
5. A. Capiluppi, P. Lago, and M. Morisio. Evidences in the evolution of OS projects through Changelog Analyses. In *Proceedings of the 3nd Workshop on Open Source Software Engineering, Int. Conf. Software Engineering*, 2003.
6. A. Capiluppi, P. Lago, M. Morisio, and D. e Informatica. Characteristics of open source projects. *Software Maintenance and Reengineering, 2003. Proceedings. Seventh European Conference on*, pages 317–327, 2003.
7. A. Capiluppi and M. Michlmayr. From the cathedral to the bazaar: An empirical study of the lifecycle of volunteer community projects. In J. Feller, B. Fitzgerald, W. Scacchi, and A. Silitti, editors, *Open Source Development, Adoption and Innovation*, pages 31–44. International Federation for Information Processing, Springer, 2007.
8. E. Capra, C. Francalanci, and F. Merlo. An empirical study on the relationship between software design quality, development effort and governance in open source projects. *IEEE Trans. Softw. Eng.*, 34:765–782, November 2008.
9. E. Capra, C. Francalanci, F. Merlo, and C. R. Lamastra. A survey on firms' participation in open source community projects. In C. Boldyreff, K. Crowston, B. Lundell, and A. Wasserman, editors, *Open Source Ecosystems: Diverse Communities Interacting*, pages 225–236. Springer, 2009.

Prepared using smrauth.cls
DOI: 10.1002/smr

10. M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon. I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, IMC '07, pages 1–14, New York, NY, USA, 2007. ACM.
11. A. Deshpande and D. Riehle. The total growth of open source. In B. Russo, E. Damiani, S. A. Hissam, B. Lundell, and G. Succi, editors, *Open Source Development, Communities and Quality, IFIP 20th World Computer Congress, Working Group 2.3 on Open Source Software, OSS 2008, September 7-10, 2008, Milano, Italy*, volume 275 of *IFIP*, pages 197–209. Springer, 2008.
12. C. DiBona, M. Stone, and D. Cooper. *Open sources 2.0*. O'Reilly, first edition, 2005.
13. R. English and C. M. Schweik. Identifying Success and Tragedy of FLOSS Commons: A Preliminary Classification of Sourceforge.net Projects. *Emerging Trends in FLOSS Research and Development, International Workshop on*, 0:11, 2007.
14. D. G. Feitelson, G. Z. Heller, and S. R. Schach. An empirically-based criterion for determining the success of an open-source project. *Software Engineering Conference, Australian*, 0:363–368, 2006.
15. J. Fernandez-Ramil, A. Lozano, M. Wermelinger, and A. Capiluppi. Empirical Studies of Open Source Evolution. In T. Mens and S. Demeyer, editors, *Software Evolution: State-of-the-art and research advances*, chapter 11, pages 263–288. Springer Verlag, 2008.
16. M. W. Godfrey and Q. Tu. Evolution in Open Source Software: A Case Study. In *Proceedings of the International Conference on Software Maintenance (ICSM'00)*, ICSM '00, pages 131–, Washington, DC, USA, 2000. IEEE Computer Society.
17. M. M. Haklay and P. Weber. Openstreetmap: User-generated street maps. *IEEE Pervasive Computing*, 7(4):12–18, Oct. 2008.
18. C. Haythornthwaite. Crowds and Communities: Light and Heavyweight Models of Peer Production. In *System Sciences, 2009. 42nd Hawaii International Conference*, pages 1–10, 2009.
19. I. Herraiz, G. Robles, J. Gonzalez-Barahona, A. Capiluppi, and J. Ramil. Comparison between SLOCs and number of files as size metrics for software evolution analysis. In *Software Maintenance and Reengineering, 2006. CSMR 2006. Proceedings of the 10th European Conference on*, pages 8 pp. –213, march 2006.
20. J. Howison, M. Conklin, and K. Crowston. Flossmole: A collaborative repository for floss research data and analyses. *International Journal of Information Technology and Web Engineering*, 1:17–26, 07/2006 2006.
21. J. Howison and K. Crowston. The perils and pitfalls of mining SourceForge. In *Mining Software Repositories, 26th International Conference on Software Engineering*, Edinburgh, Scotland, 2004.
22. A. Israeli and D. G. Feitelson. Success of open source projects: Patterns of downloads and releases with time. *Software Science, Technology and Engineering, IEEE International Conference on*, 0:87–94, 2007.
23. E.-A. Karlsson, editor. *Software reuse: a holistic approach*. John Wiley & Sons, Inc., New York, NY, USA, 1995.
24. S. Koch. Software evolution in open source projects – a large-scale investigation. *J. Softw. Maint. Evol.*, 19:361–382, November 2007.
25. S. T. K. Lam and J. Riedl. The past, present, and future of wikipedia. *Computer*, 44:87–90, March 2011.

DOI: 10.1002/smr

26. C. Li. Mapping participation in activities forms the foundation of a social strategy. Technical report, Social Technographics Trends Report, Forrester Research Inc, May 2007.
27. F. Ortega. *Wikipedia: A quantitative analysis*. PhD thesis, Universidad Rey Juan Carlos – Escuela Técnica Superior De Ingeniería De Telecomunicación, 2009.
28. J. Paulson, G. Succi, and A. Eberlein. An empirical study of open-source and closed-source software products. *Software Engineering, IEEE Transactions on*, 30(4):246 – 256, april 2004.
29. J. S. Poulin. *Measuring software reusability*, page 126âĂŞ138. Number November. IEEE, 1994.
30. W. Stevens, G. Myers, and L. Constantine. *Structured design*, pages 205–232. Yourdon Press, Upper Saddle River, NJ, USA, 1979.
31. B. Suh, G. Convertino, E. H. Chi, and P. Pirolli. The singularity is not near: slowing growth of wikipedia. In *WikiSym '09: Proceedings of the 5th International Symposium on Wikis and Open Collaboration*, pages 1–10, New York, NY, USA, 2009. ACM.
32. J. Van Dijck. Users like you? theorizing agency in user-generated content. *Media, Culture & Society*, 31(1):41–58, 2009.
33. J. Voss. Measuring wikipedia. In *Proceedings of the 10th International Conference of the International Society for Scientometrics and Informetrics*, Stockholm (Sweden), July 2005.
34. D. Weiss and B. E. X. Entry. A large crawl and quantitative analysis of open source projects hosted on sourceforge. In *University of Technology*, 2005.
35. A. Wiggins, J. Howison, and K. Crowston. Heartbeat: Measuring Active User Base and Potential User Interest in FLOSS Projects. In C. Boldyreff, K. Crowston, B. Lundell, and A. I. Wasserman, editors, *Open Source Ecosystems: Diverse Communities Interacting, 5th IFIP WG 2.13 International Conference on Open Source Systems, OSS 2009, Skövde, Sweden, June 3-6, 2009. Proceedings*, volume 299 of *IFIP*, pages 94–104. Springer, 2009.
36. S. Wunsch-Vincent and G. Vickery. Participative web: User-created content. *Working Party on the Information Economy*, page 74, 2007.
37. G. Xie, J. Chen, and I. Neamtiu. Towards a better understanding of software evolution: An empirical study on open source software. In *Software Maintenance, 2009. ICSM 2009. IEEE International Conference on*, pages 51 –60, sept. 2009.

Prepared using smrauth.cls
DOI: 10.1002/smr

A. APPENDIX – DATA CLEANING

Before the evaluation of the effort metrics, a cleaning phase was performed, because anomalies were identified. Figure 12 details an example of an anomaly detected in the trend of total handlings and how it is transformed after the cleaning. The reported example is based on the *Exuserfolder* project: during a low productivity period of the contributors, month 11/2004 shows a unexpected peak of Total Handlings. This high value could represent a breakpoint separating a previous phase of poor activity from a new one with more activity but, as we can notice, this peak is an isolated occurrence. Probably the peak is due to a refactoring activity (not intended as code restructuring, but as folders moving, file renaming...) which doesn't represent an effort related to creation or modifications of code lines. The cleaning process consists of the following steps :

1. the first month of activity is not considered to calculate the Total Handlings metric. Usually in the first month a large number of new files are found because they are related to the initial import of the project in the repository, so the Modules Created metric doesn't represent a trustworthy description of the effort. In this case the value of the metric is set to 0 only in the first month.
2. An anomalous peak is detected if the current value is larger than 200% of the moving average (months with zero activity are not considered in the average computation). If a peak is detected 2 cases were computed:
    - if at least two of the following values are larger than 200% of the moving average, the peak is not removed because it represents a new phase of productivity for the project;
    - otherwise the peak is removed and replaced by the average value of the data series till that point.

Prepared using smrauth.cls
DOI: 10.1002/smr

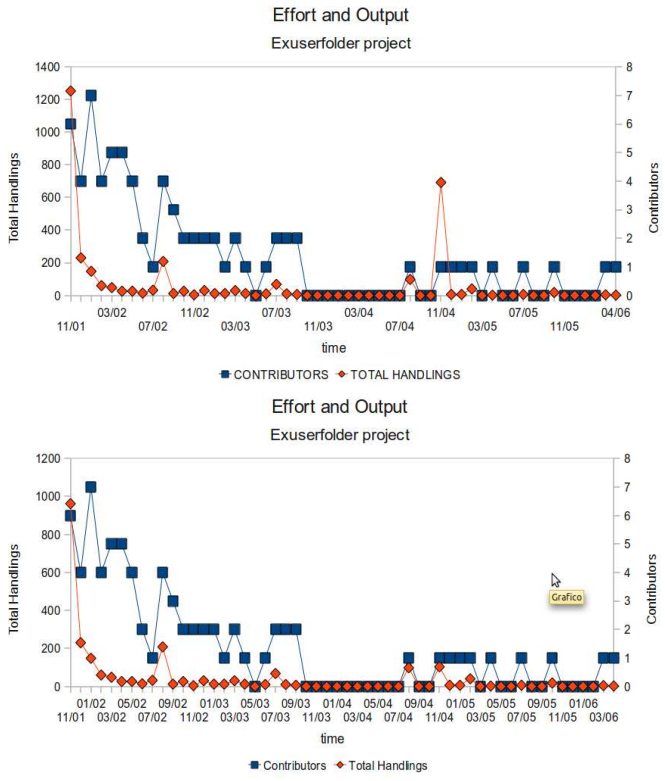

Figure 12. Data Cleaning Phase

Prepared using smrauth.cls
DOI: 10.1002/smr