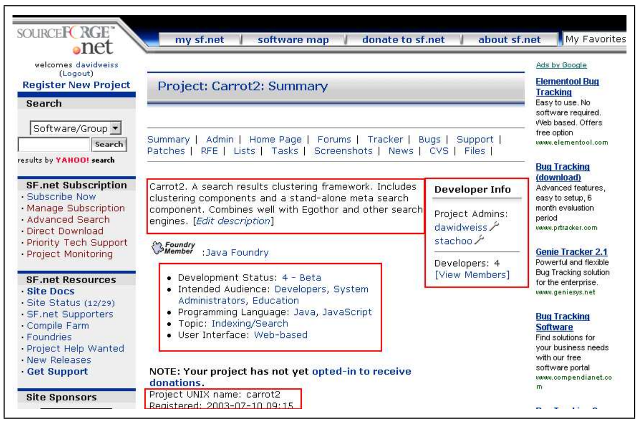
Quantitative Analysis of Open Source Projects on SourceForge

Dawid Weiss
Poznań University of Technology
Poznań, Poland
dawid.weiss@cs.put.poznan.pl

***Abstract*— Relatively easy accessibility of high volumes of information about open source software makes it an interesting target for quantitative analysis meant to discover some hidden properties and trends of this software development model. In this work we demonstrate how such information can be acquired from the largest open source hosting facility — SourceForge — with nearly minimal effort. We compare our data with an identical data set collected a few months earlier by the OssMole [2] project, of which we were not aware at the time of performing the experiment, but which allowed us to make some interesting cross-comparisons and derive conclusions about temporal changes going on at SourceForge.**



Carrot2. A search results clustering framework. Includes clustering components and a stand-alone meta search component. Combines well with Egothor and other search engines. [*Edit description*]

:Java Foundry

- Development Status: 4 - Beta
- Intended Audience: Developers, System Administrators, Education
- Programming Language: Java, JavaScript
- Topic: Indexing/Search
- User Interface: Web-based

NOTE: Your project has not yet opted-in to receive donations.

Project UNIX name: carrot2
Registered: 2003-07-10 09:15

Developer Info

Project Admins:
dawidweiss
stachoo

Developers: 4
[View Members]

Elementool Bug Tracking
Easy to use. No software required. Web based. Offers free option
www.elementool.com

Bug Tracking (download)
Advanced features, easy to setup, 6 month evaluation period
www.prtracker.com

Genie Tracker 2.1
Powerful and flexible Bug Tracking solution for the enterprise.
www.geniesys.net

Bug Tracking Software
Find solutions for your business needs with our free software portal
www.compendianet.com

Fig. 1. Home pages of projects hosted at SourceForge demonstrate a very predictable layout of content structure (here depicted with red rectangles).

I. Introduction and motivation for the work

It is seldom the case that so little is known about the sources of success of so many. The open source movement has gained an incredible momentum and still many questions concerning its internal workings remain unclear. Why is open source development attractive for programmers? What kind of factors make an open source project successful? How can a programming model be so far from modern commercial software engineering standards and yet spawn stable, commercial-quality software?

Researchers start to investigate the above questions and more and more is known about the mechanics and processes ruling the open source world [1]. In this work we would like to contribute certain quantitative analyses of open source projects that we performed on data acquired (*crawled*) from the largest open-source hosting facility — SourceForge.[1] Under the term ‘quantitative’ we understand applying certain statistics and counting properties of hundreds or thousands of open source projects.

A single statistical snapshot of so many open source projects is interesting in itself, but as an additional bonus we also provide elements of temporal analysis of changes going on at SourceForge, which we believe has not been presented before. We do this by comparing results made for our data set with identical statistics published a few months earlier by the OssMole project [2]. It should be stressed that we were not aware of OssMole until our data was halfway crawled; approaches to crawling exhibited in the two vary significantly.

II. Procedure of crawling SourceForge data

The primary observation that led us to the concept of crawling SourceForge data automatically was that all home pages of projects hosted there have a very similar and predictable structure (see Figure 1), suggesting an underlying database of some sort. The structure and availability of data in that database was for obvious reasons unknown, but by manual analysis of several dozen pages, we ended up with the following features that could be automatically retrieved from tha raw HTML source of home pages:

- **Registration date.** For all projects, there is a registration date on SourceForge services.
- **Activity percentile.** A measure of a project’s ‘activity’.[2]
- **Total number of developers in a project.** Note that this is the total number of so-called *committers* (people with write-access to the repository). The number of committers heavily depends on the internal project policy of assigning such rights to people and is hard to compare between projects.
- **Unix name of the project.** Name of the folder on SourceForge hosts (also indicates project names found in the URLs).
- **Summary of the project.** More verbose information about the project (if defined by the admin).
- **Project tracker information.** Trackers for: bugs, support requests, feature requests and patches. Total and open counts are available for projects that declared to use such facilities.
- **Trove classification data.** Trove[3] classification system is based on self-declaration of project maintainers about features and properties of the project. These include: database environment, development status,

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

[2]Calculated by SourceForge every week using the formula described at `http://sourceforge.net/docman/display_doc.php?docid=14040\&group_id=1#calculate`

[3]`http://sourceforge.net/softwaremap`

intended audience, used license(s), programming language, deployment operating system, topic, translations of the project, its documentation and type of user interface.

Project maintainers may locate the project in Trove's categories by assigning it to any of the values predefined by SourceForge. A project may opt not to use this classification system at all. Usually more than one value from each category may be also assigned to a project (i.e. more than one language of translation or a target operating system).

The next thing we observed was that the URLs of home pages follow a predictable pattern of:

$$\underbrace{\texttt{http://sourceforge.net/projects/}}_{\text{Base address}}\underbrace{\texttt{carrot2}}_{\text{Unix project name}} \tag{1}$$

We then fetched home pages of all projects hosted at SourceForge in order to extract the interesting information to a relational database for further analysis. The remaining part of this section presents this process in an outline. For a detailed step-by-step description and additional resources, refer to [3].

1) Acquire a list of all Unix names of projects at SourceForge. This can be done easily if you have a shell account at SourceForge (which every project administrator does).
2) Process the list of project names and fetch all home pages of known projects using the URL pattern mentioned above. Play it nice with SourceForge servers and make long delays between fetches. There were over 89 thousand projects at the time we did the crawl. It takes a good couple of days to collect all the data.
3) Process locally stored home pages and extract all available information about the projects. We used a set of quite trivial regular expressions and a simple Python script.
4) Insert all the extracted information into a relational database to facilitate data mining. We used a database's schema shown in Figure 2 together with MySQL database engine. There is one table to hold all of project-related information (`projects`) and one table to store predefined values of categories found in the Trove system (`trove_categories`). Since there is a many-to-many relationship between `projects` and `trove_categories`, this relationship is split and forms an additional table `trove`. An average query with multiple joins and where clauses is usually completed under a few seconds on average commodity hardware.

Using this procedure we crawled and extracted information from 89557 projects present in SourceForge at the end of December, 2004.

III. Cross-comparison of results with OssMole

Our crawl took place roughly two months after OssMole's. We believed certain trends and differences should

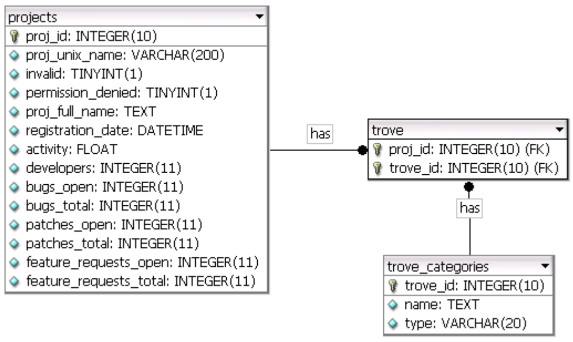

projects
proj_id: INTEGER(10)
proj_unix_name: VARCHAR(200)
invalid: TINYINT(1)
permission_denied: TINYINT(1)
proj_full_name: TEXT
registration_date: DATETIME
activity: FLOAT
developers: INTEGER(11)
bugs_open: INTEGER(11)
bugs_total: INTEGER(11)
patches_open: INTEGER(11)
patches_total: INTEGER(11)
feature_requests_open: INTEGER(11)
feature_requests_total: INTEGER(11)
has
trove
proj_id: INTEGER(10) (FK)
trove_id: INTEGER(10) (FK)
has
trove_categories
trove_id: INTEGER(10)
name: TEXT
type: VARCHAR(20)

Fig. 2. Database schema for storing information extracted from SourceForge. `trove` table is a normalization of many-to-many relationship.

be visible by comparing results of queries[4] published by OssMole to identical queries on our data set.

A. Number of developers for each project

(OSSMole: `sfSummaryDeveloperData12-Nov-2004.txt`)

We have found out the following:

- Projects with no access (invalid or with permission denied) are omitted in OSSMole's results. We include all projects, even if information about them is unavailable.
- Our list has 4744 new projects compared to OssMole's. This is also confirmed by charts of projects registration per month (see Section III-G).
- 1187 projects gained developers, on average by 1.51 developer per project.
- 397 projects *lost* developers, on average by 1.80 developer per project (which is more than the developer gain mentioned above).
- Interestingly, 15 projects went completely inactive (or should we say have been abandoned by the captain) dropping to zero developers.
- Even more interesting, however, was the fact that several projects were *not present at all* in our results. Since our database of projects contained all entries in the filesystem, it either implies that those projects had been physically removed in the meantime (against SourceForge's policy?), or a network file system was disconnected at the time we crawled project names (not likely because missing project names range throughout the alphabet). A total of 93 projects had been removed in the period of 2 months.

B. Intended audience for projects

(OSSMole: `sfSummaryIntAudData21-Oct-2004.txt`)

Comparison of intended audience charts yields an interesting thing: apparently new categories have been added to Trove (see Table I). Apart from that, a steady increase of project counts among all categories can be observed.

C. Usage of licenses

(OSSMole: `sfSummaryLicenseData21-Oct-2004.txt`)

As it is shown in Table II, the use of open source licenses is

[4]OssMole only published the scripts and several results of their crawl. The database they collected is not available for the public.

Intended audience	Experiment	OssMole	Increase	Ranking change
Developers	32248	31104	1144	0
End Users/Desktop	28381	27406	975	0
System Administrators	13111	12710	401	0
Other Audience	6329	6206	123	0
Information Technology	5155	4895	260	0
Education	4254	4051	203	0
Science/Research	3978	3749	229	0
Customer Service	1292	1232	60	0
Telecommunications Industry	1205	1121	84	0
Financial and Insurance Industry	612	572	40	0
Manufacturing	495	460	35	0
Healthcare Industry	437	424	13	0
Advanced End Users	329	#N/A	#N/A	#N/A
Religion	239	227	12	1
Legal Industry	219	210	9	1
Quality Engineers	48	#N/A	#N/A	#N/A

TABLE I

INTENDED AUDIENCE FOR PROJECTS IN OSSMOLE AND OUR EXPERIMENT. NOTE THE MISSING CATEGORIES IN OSSMOLE

License	Experiment	OssMole	Increase	Ranking change
GNU General Public License (GPL)	40275	37947	2328	0
GNU Library or Lesser General Public License (LGPL)	6455	5070	1385	0
BSD License	4172	3962	210	0
Public Domain	1579	1271	308	0
Artistic License	1160	1112	48	0
MIT License	1041	904	137	-1
Other/Proprietary License	1000	555	445	-1
Apache Software License	930	918	12	2
Mozilla Public License 1.1 (MPL 1.1)	716	537	179	0
OSI-Approved Open Source	473	471	2	0
Common Public License	416	360	56	0
zlib/libpng License	288	245	43	0
Open Software License	270	208	62	0
Apache License V2.0	260	198	62	0
Mozilla Public License 1.0 (MPL)	227	187	40	0
Qt Public License (QPL)	214	152	62	-1
Academic Free License (AFL)	182	160	22	1
Python License (CNRI Python License)	139	117	22	0
Python Software Foundation License	123	102	21	0
PHP License	112	58	54	-1
IBM Public License	74	63	11	1
Apple Public Source License	51	51	0	0
wxWindows Library Licence	48	33	15	0
Sun Industry Standards Source License (SISSL)	47	29	18	-2
Sun Public License	45	26	19	-2

TABLE II

SUMMARY OF USED LICENSES IN OSSMOLE AND OUR EXPERIMENT. NOTE THE STABILITY IN THE RANKING AMONG THE TOPMOST LICENSES.

quite stable, especially among the topmost licenses. Why is GNU-family licensing so popular is an interesting question. Some people say it is GNUs idealistic concept of forever-free software people are attracted to. Others claim it is a *consequence* of these idealistic concepts — a set of 'contagious' legal terms and obligations people have little choice but to obey.

D. Target operating systems

(OSSMole: `sfSummaryOpSysData21-Oct-2004.txt`)

Target operating system summaries differ very much between OssMole and our experiment. It is mostly due to major changes in names of categories. This is somewhat of a mistery to us — we suppose either SourceForge changed Trove classification system at some point, or OssMole's data has been modified from Trove's classification to reduce the set of classes (compare listings of topmost classes in Table III). Another factor that obscures the view of results is that Trove categories are hierarchical, but projects get assigned to internal hierarchy nodes and leaves at the same time.

E. Language of implementation

(OSSMole: `sfSummaryProgLangData21-Oct-2004.txt`)

A ranking of implementation languages remained virtually unchanged from OssMole's version (see Table IV); a few new languages apparently have been added to Trove and there is a steady increase in the project counts. Note strong position of Java in comparison with C#. Assembly language, even though strong in the ranking, gained only 7 projects in two months.

OssMole		Our experiment	
Operating system	Projects	Operating system	Projects
OS Independent	20802	All POSIX (Linux/BSD/UNIX-like OSes)	27660
Linux	20576	OS Independent (interpreted language)	22135
Windows	7720	Linux	21642
Windows 95/98/2000	6429	All 32-bit MS Windows (95/98/NT/2000/XP)	18766
Windows NT/2000	4983	32-bit MS Windows (95/98)	6628
POSIX	4786	32-bit MS Windows (NT/2000/XP)	5358
MacOS X	2694	WinXP	5269
SunOS/Solaris	1972	Win2K	5212
FreeBSD	1397	All BSD Platforms (FreeBSD/NetBSD/OpenBSD/Apple OS X)	2978
BSD	1353	OS X	2888
Other OS	1072	Solaris	2057
Microsoft	1007	FreeBSD	1451
MacOS	821	Other Operating Systems	1081
Other	705	Other	738
MS-DOS	540	MS-DOS	581
PalmOS	462	PalmOS	499
BeOS	421	BeOS	430
MacOS 9	386	Apple Mac OS Classic	409
PDA Systems	297	OS Portable (Source code to work with many OS platforms)	372
OpenBSD	291	Handheld/Embedded Operating Systems	310

TABLE III

TOPMOST CLASSES FOR TARGET OPERATING SYSTEMS IN OSSMOLE AND OUR EXPERIMENT. NOTE MAJOR DIFFERENCES IN NAMES OF CLASSES.

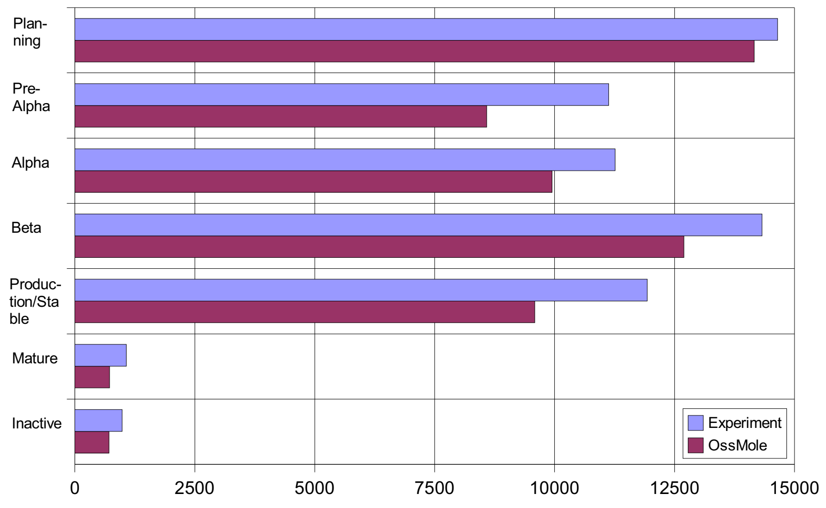


Fig. 3. Declared development statuses in OssMole and our experiment.

Implementation language	Our experiment	OssMole	Increase	Ranking change
C++	14326	13793	533	0
C	13962	13547	415	0
Java	13552	12872	680	0
PHP	9976	9480	496	0
Perl	5554	5407	147	0
Python	3608	3432	176	0
JavaScript	2054	1946	108	0
C#	2052	1847	205	-1
Visual Basic	1989	1939	50	1
(contd.)				

TABLE IV

IMPLEMENTATION LANGUAGES IN OSSMOLE AND OUR EXPERIMENT.

F. *Declared development status*

(OSSMole: `sfSummaryStatusData21-Oct-2004.txt`)

Comparison of development statuses between OssMole and our experiment shows major increases in products marked as *Pre-Alpha* and *Production/Stable* (by over 2000 projects). Pre-alpha growth could be explained by projects that were added to SourceForge in between the two experiments. There seems to be no clear source for such increase in stable projects number (all statuses noted an increase in counts and the number of recently added projects does not entirely balance this increase). We were unable to rationally explain this phenomenon.

G. *Registration history*

(OSSMole: `sfSummaryRegistrationData21-Oct-2004.txt`)

As mentioned before in Section III-A, we could expect certain discrepancy in the number of registrations over time between OssMole's and our data. Theoretically, the past numbers should be identical, but as we showed, several projects had been physically removed from SourceForge. Large increase in 2004 is of course caused by the two months of difference between crawl times of the two experiments (however, the difference accounts only for 4221 new registrations and as we showed in Section III-A, the actual difference between data sets is 4744 projects).

Development status	Experiment	OssMole	Increase
1999	433	433	0
2000	5660	5662	-2
2001	16354	16412	-58
2002	21618	21676	-58
2003	22377	22436	-59
2004	20864	16643	4221

TABLE V
NUMBER OF REGISTRATIONS PER YEAR IN OSSMOLE AND OUR EXPERIMENT.

H. *Declared topic*

(OSSMole: `sfSummaryTopicData21-Oct-2004.txt`)

As for projects' declared topics, no major changes could be observed between OssMole and our experiment. When looking at the assignments, however, one cannot escape the conclusion that the work on categorization of software on SourceForge is still in progress, or is badly designed. A highlight example: in the *DocBook* category of Trove[5] there was only one project at the time of our crawl[6] and it was *not* the main DocBook stylesheets project, which is also hosted on SourceForge... In fact, DocBook stylesheets project is not even categorized in Trove.

I. *Summary of the comparison*

Comparison of our data with OssMole's yields some interesting observations, especially about changes undergoing the SourceForge software classification system Trove. Analysis of open source projects data over time additionally shows certain aspects of its dynamics, which is obscure with static snapshots. It may be a good idea for the future to focus on this temporal activity going on at SourceForge.

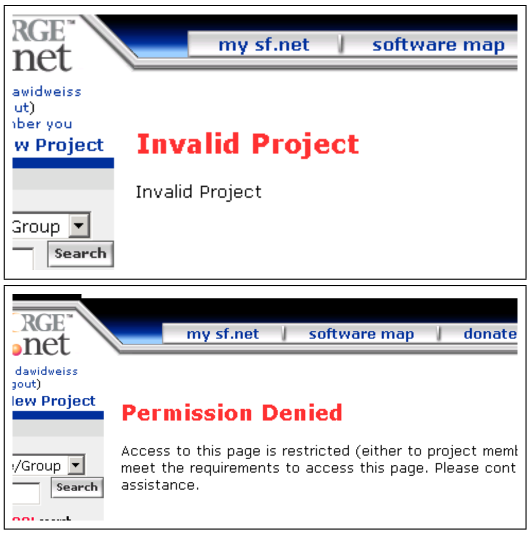
IV. SELECTED ANALYSES PERFORMED EXCLUSIVELY ON OUR DATA

A. *Project activity*

1) Invalid and inaccessible projects: While crawling SourceForge we encountered many projects with home pages either inaccessible — 74 projects, or invalid — 2251 projects, a bit over 2% of the total number of projects (see Figure 4). Inaccessible pages are blocked by project administrators. We could not come up with any sensible explanation for invalid pages other then that these pages might have been scheduled for removal and inaccessibility is a way of detecting whether the project is still maintained.

2) Development status declared by projects: Figure 5 presents a chart of counts of projects with a given development status. Note the disproportion between software declared as stable and software in alpha, or planning phase. Table VI shows the number of 'discontinued' projects — those that registered over a year ago and never changed their status from alpha. Again, it might be surprising, but over 43% of projects fall into this category.

[5] `http://sourceforge.net/softwaremap/trove_list.php?form_cat=555`
[6] `http://sourceforge.net/projects/xml2texml/`



Invalid Project

Invalid Project

Permission Denied

Access to this page is restricted (either to project meml meet the requirements to access this page. Please cont assistance.

Fig. 4. SourceForge returned invalid or inaccessible pages for a number of projects.

3) Number of projects with at least one bug: Number of projects with at least one bug is 14475, a mere 16% of the total. Yet again a proof that most SourceForge projects never really attracted a serious user community.

At an average, each project has approximately one open bug out of four ever submitted. unfortunately, the averages are not good representatives of the distribution of bugs, which is exponential (see Figure 6 for a chart). This again would suggest that only minority of SourceForge projects are truly active.

Figure 7 also presents an interesting chart — projects with *most* submitted bugs (and hence most active user communities). On a more relaxing note, the project with most submitted feature requests is... SourceForge itself!

B. *Average number of developers per project*

An average number of developers per project is 2, but again, the distribution of this statistic is rather exponential (see Fig 8) with maximums reaching hundreds of developers in one project (TINYOS, JEDIT, JBOSS).

C. *Number of language translations per project*

Majority of projects is unilingual (40649 projects), with English being the most popular language (48482 projects) and German with the second position (4809 projects). Relatively high number of bilingual projects (9486 projects) may be a result of foreign projects being later (or even at the time of development) translated to English. Higher number of translations are seldom.

	Registration date					Total
Status	1999	2000	2001	2002	2003	
1 - Planning	24	524	3308	3991	3462	11309
2 - Pre-Alpha	43	686	2271	2794	2758	8552
3 - Alpha	61	1027	2163	2651	2769	8671
Total	128	2237	7742	9436	8989	**28532**

TABLE VI
NUMBER OF DISCONTINUED PROJECTS

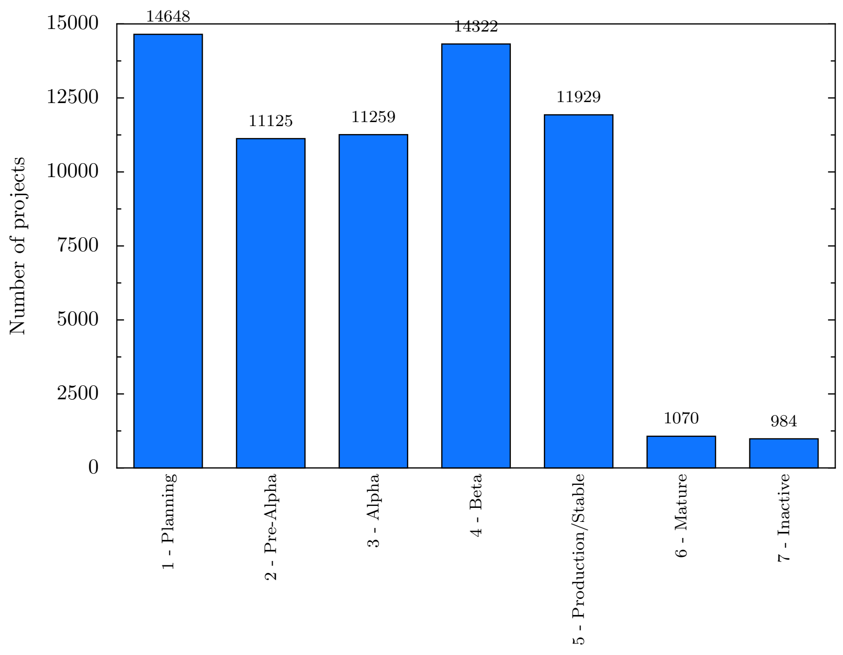


Fig. 5. Count of projects with a given development status

V. Summary and conclusions

We have presented a simple and low-effort method of crawling SourceForge data that provides very interesting information about numerous open source projects. We have also shown how our data compares to a similar project OssMole.

The following conclusions and observations can be drawn from the experiments:

- OssMole's data is not complete — it lacks information about inactive and inaccessible projects. We are able to indicate at least names of those projects.
- Open source projects on SourceForge are constantly initiated and go inactive; some of the inactive projects vanish completely from the server.
- Trove categorization seems to be under constant refinement by SourceForge. Categories change names, new leaves and hierarchy nodes are added.
- Developers themselves are the main target audience for open source projects. This class of target audience also exhibits the most rapid growth in the number of new projects (see Table I).
- With outstanding lead over any other license, GPL and LGPL licensing types are the most common in the open source world (see Table II). GNU's 'contagious' licensing terms might have resulted in its leading role; it is difficult to say whether this is really the case. From the point of view of commercial partners, this strong leading position of GPL might be seen as a drawback because GPL is commonly considered hostile to the commercial world.
- The 'war' between Java and C#, at least in the world of open source, does not exist — Java reigns together with C and C++ (see Table IV).
- Surprisingly, vast majority of open source projects declares their development status at the 'unstable' level or even in the planning phase (see Section IV-A.2). Together with the information concerning community activity, such as the number of submitted bugs, patches and feature requests, this seems to indicate that only minority of open source projects ever makes it to the level of being popular (and thus successful). open source looks much like software incubator — out of many attempts only a few make it, but are good enough to catch on with a larger group of end users.
- There is something essentially odd about the way 'activity' is measured by SourceForge. Linear characteristic of values of this activity suggests that it is a form of a ranking, rather than a product of some independent factors. In any way, the activity factor is on an ordinal scale rather then ratio scale — any comparisons of activity ratios of two projects do not make sense.

Future work could focus on detecting temporal activity of the open source projects and community: what type of projects are becoming more successful and which lose interest? Which languages are becoming more important and which not? What operating systems dominate the market? Seeking answers to such questions is justified for purely research reasons, but might turn to be useful to commercial companies investing in open source software as well. Elements of such analysis have been started after this paper was finalized: the OSSMole project integrated the outcome of the experiment described in this paper and initiated the process of collecting temporal data.

Acknowledgement. The author of this work would like to thank Maciej Hapke, Andrzej Jaszkiewicz and Krzysztof Kowalczykiewicz for fruitful discussions. This research has been funded by the European Commission via FP6 Co-ordinated Action Project 004337 in priority IST-2002-2.3.2.3 **CALIBRE** (`http://www.calibre.ie`).

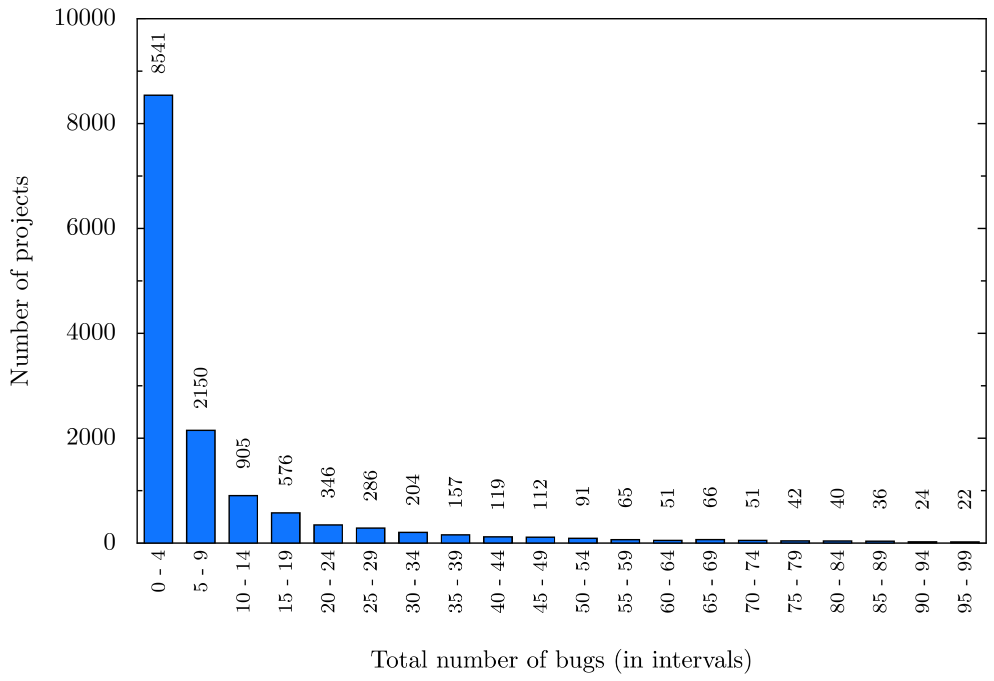


Fig. 6. Histogram of distribution of the total number of bugs for projects with less than a 100 bugs.

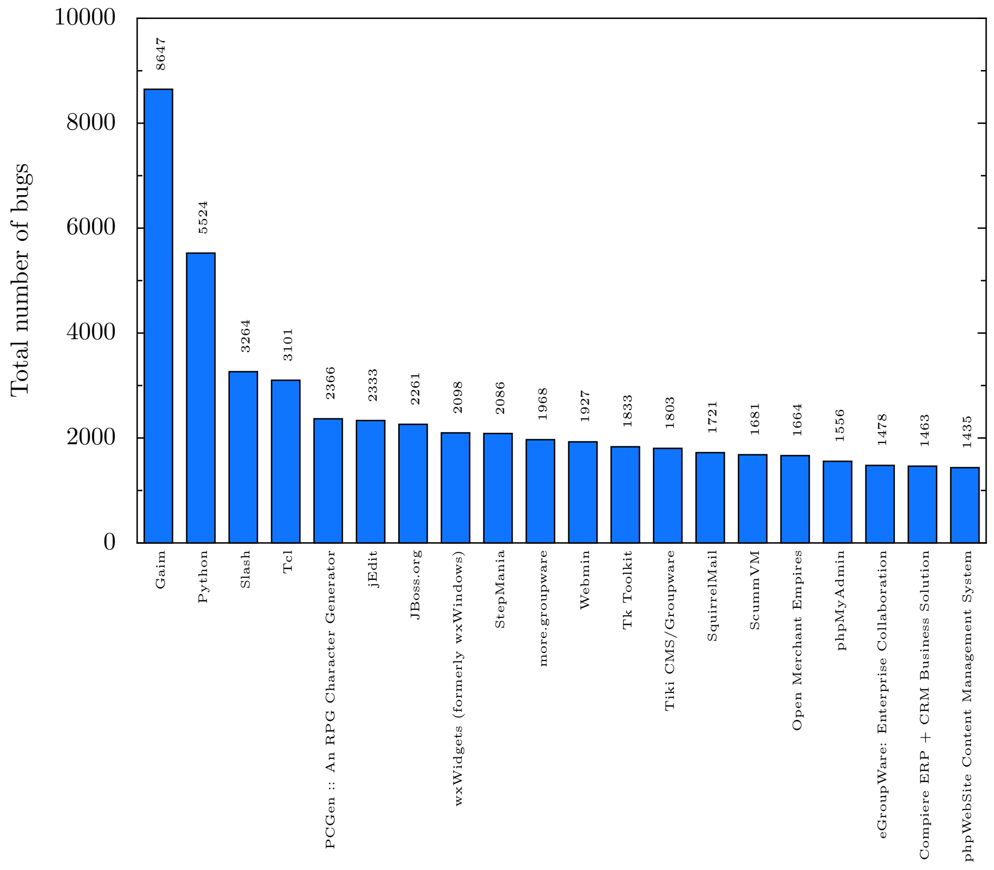


Fig. 7. Projects with most total submitted bugs.

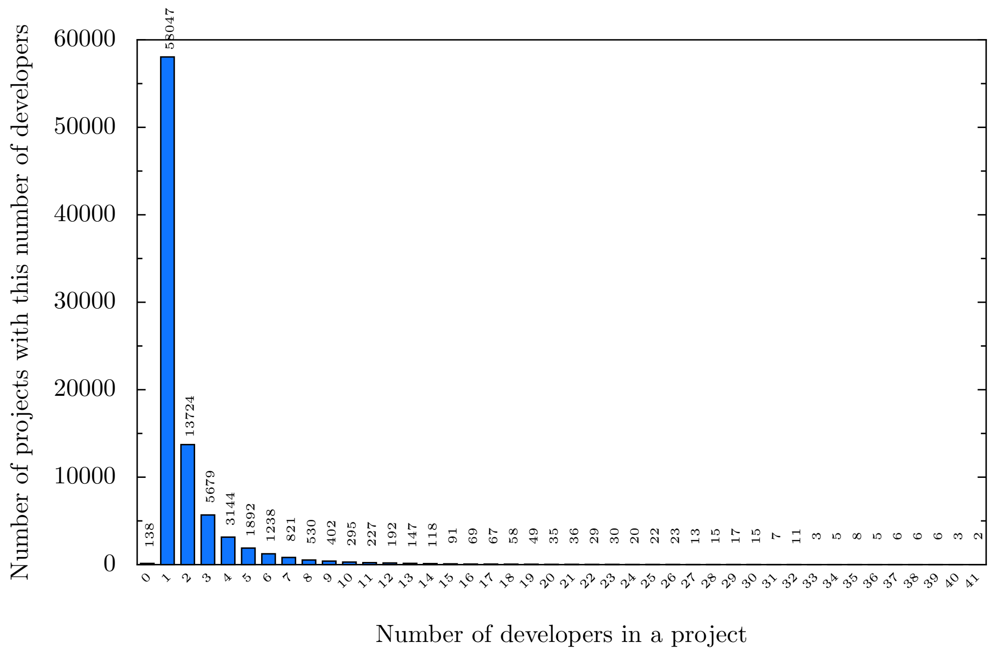


Fig. 8. Distribution of the number of developers in projects.

References

[1] M. Hahsler and S. Koch, “Discussion of a large-scale open source data collection methodology,” Proceedings of the Hawaii International Conference on System Sciences (HICSS-38), to appear., 2004.

[2] Ossmole, “Ossmole: a project to provide academic access to data and analyses of open source projects,” 2004. [Online]. Available: http://ossmole.sourceforge.net

[3] D. Weiss, “A large crawl and quantitative analysis of open source projects hosted on sourceforge,” Institute of Computing Science, Poznań University of Technology, Poland,” Research Report RA-001/05, 2005.