The (quantitative) history of LibreOffice

Jesus M. Gonzalez-Barahona

jgb@bitergia.com
`http://identi.ca/jgbarah` `http://twitter.com/jgbarah`
Bitergia
GSyC/LibreSoft (Universidad Rey Juan Carlos)

LibreOffice Conference, Berlin, October 17th, 2012

Presentation of a preview

Analysis still being completed

...still unvalidated

...could have errors

It will be published when complete

`http://blog.bitergia.com`

Main characteristics of the analysis

Quantitative analysis

Focus on activities related to development and maintenace

View of the evolution of the project

Specific questions:

- Activity in changing the code base
- Developers involved
- Profile of the activity of the developers
- Activity in reporting and closing tickets
- Ticket openers, ticket closers
- Time to close, time to attend (tickets)
- How state of tickets change
- Some comparison with OOo, AOO

Data on git, Bugzilla

Data source: git (commits, changes)

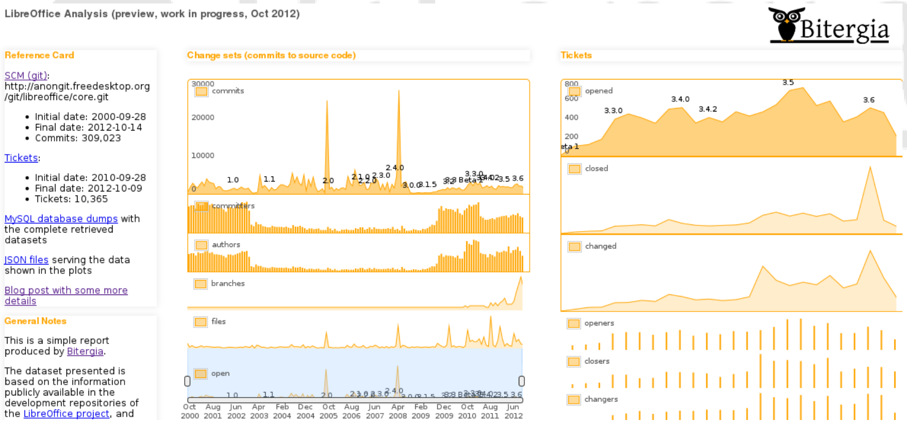
- `http://anongit.freedesktop.org/git/libreoffice/core.git`
- 2000-09-28 to 2012-10-14
- 309,023 commits

Data source: Bugzilla (tickets)

- `https://libreoffice.org/bugzilla/`
- 2010-09-28 to 2012-10-09
- 10,365 tickets

Data source: released source code of OpenOffice.org, LibreOffice, Apache OpenOffice

General overview (git, Bugzilla)



LibreOffice Analysis (preview, work in progress, Oct 2012)

Reference Card

SCM (git):
http://anongit.freedesktop.org/git/libreoffice/core.git

- Initial date: 2000-09-28
- Final date: 2012-10-14
- Commits: 309,023

Tickets:

- Initial date: 2010-09-28
- Final date: 2012-10-09
- Tickets: 10,365

MySQL database dumps with the complete retrieved datasets

JSON files serving the data shown in the plots

Blog post with some more details

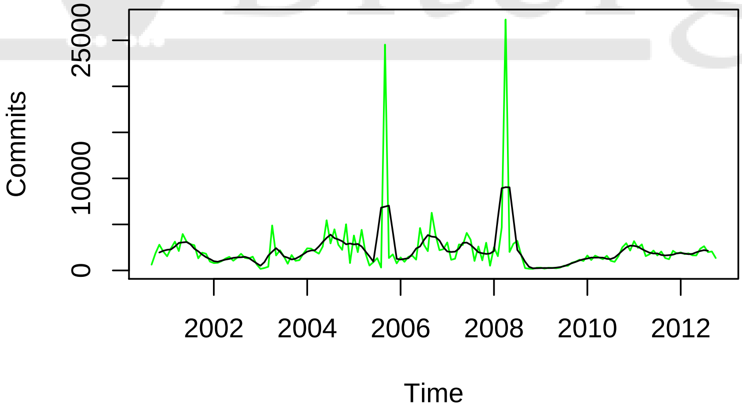
General Notes

This is a simple report produced by Bitergia.

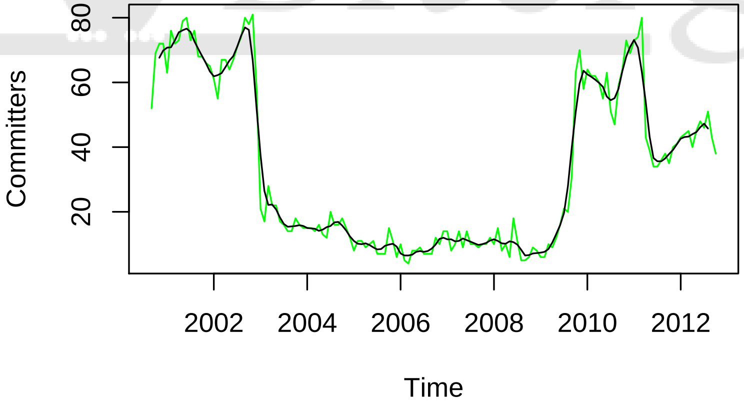
The dataset presented is based on the information publicly available in the development repositories of the LibreOffice project, and

http://bitergia.com/public/previews/2012_10_libreoffice/

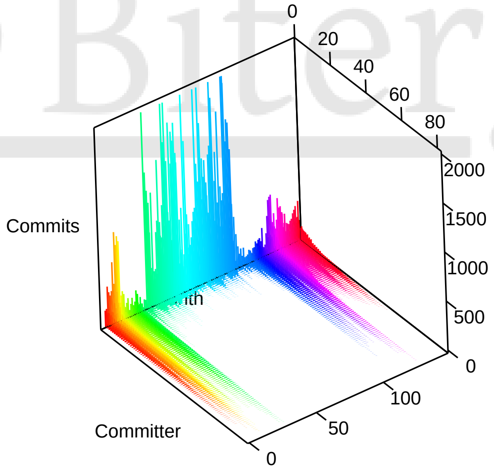
Commits per month



Committers per month

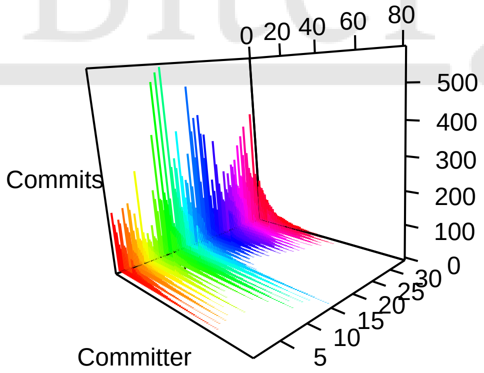


Commits for each committer per month



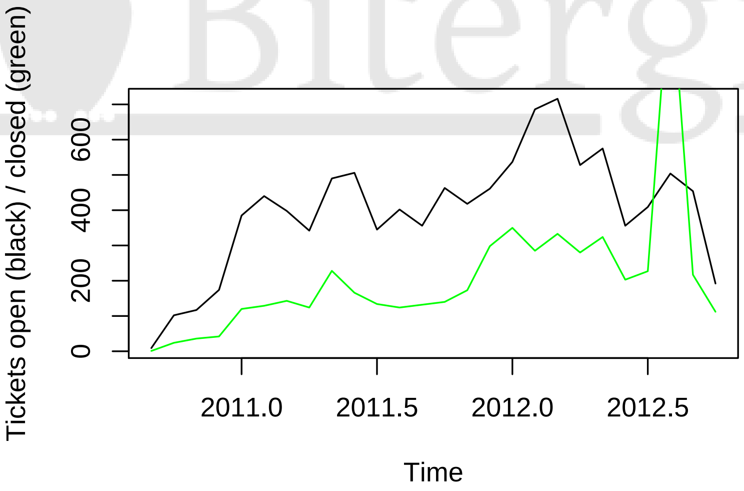
[Contributions of more than 2,000 commits trimmed]

Commits for each committer per month



[Since 2010-01-01]

Tickets open / closed per month



Bugzilla: how tickets were closed

Resolution	Number of tickets
NOTCLOSED	5400
FIXED	1458
DUPLICATE	1217
INVALID	947
WORKSFORME	844
NOTABUG	307
WONTFIX	98
NOTOURBUG	91
MOVED	3

Field "resolution" of Bugzilla

Bugzilla: how tickets were not closed

Of 5,400 “not resolved”:

- 2,009 didn’t change in status
- 3,392 tickets did (5,882 changes):

Status changed to	Number of changes
NEW	2959
NEEDINFO	1465
RESOLVED	503
REOPENED	398
UNCONFIRMED	285
ASSIGNED	258
CLOSED	12
VERIFIED	2

Bugzilla: changes of status

Status	Total	2010	2011	2012
ASSIGNED	702	24	359	319
CLOSED	42		21	21
NEEDINFO	2,998		2,076	922
NEW	3716	2	731	2,983
REOPENED	649	10	198	441
RESOLVED	5,731	105	2,018	3,608
UNCONFIRMED	368		38	330
VERIFIED	19		3	16
OPEN	10,365	402	5,006	4,957
FIXED	5,773	105	1,039	3,629

FIXED: CLOSED + RESOLVED

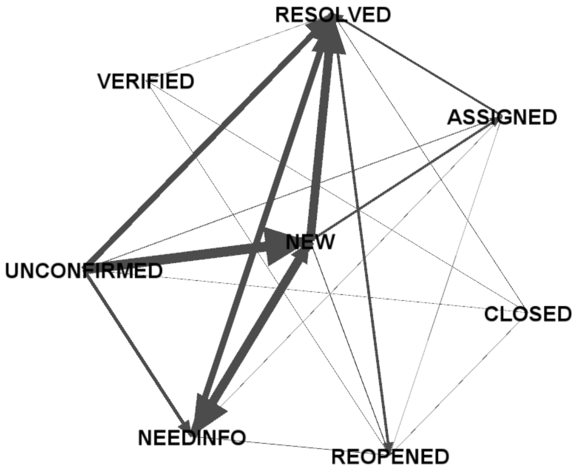
Bugzilla: how tickets change their status

	ASSIG	NEED	NEW	REOP	RESOL	UNCF
ASSIG			541			
NEED			2,171			757
NEW		1,092				2,428
REOP					578	
RESOL	437	1,532	2,121	212		1,424
UNC		220				

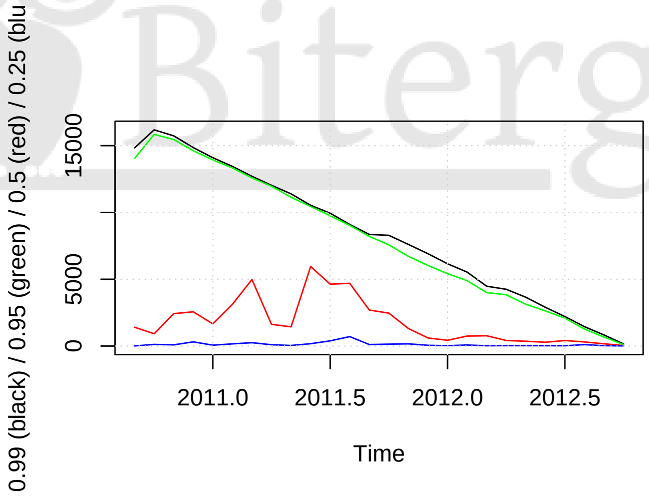
(X,Y): Change from X to Y

(changes with $>$ 200 occurrences)

Bugzilla: how tickets change their status (graph)



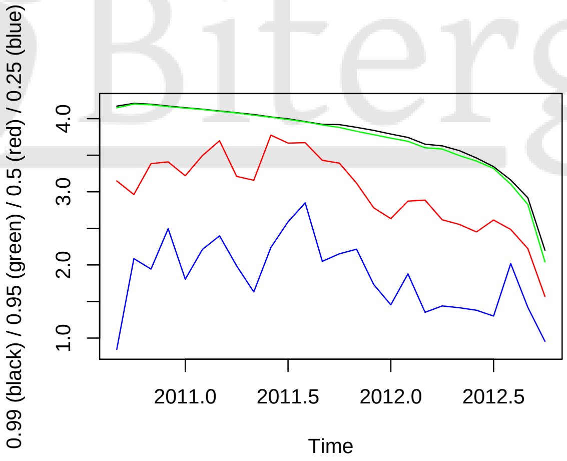
How long does it take to close tickets (hours)



Time to close tickets opened during the month and getting closed

5,000 hours: 7 months

How long does it take to close tickets (log10 hours)



10^2 hours: 4 days, 10^3 hours: 1.3 months

Comparing the many * Office *

	Release	Date	Files
OOo	OpenOffice.org 3.3.0	Jan 2011	42,731
LOa	LibreOffice 3.5.1	March 2012	42,160
LOb	LibreOffice 3.6.2	October 2012	39,637
AOO	Apache OpenOffice 3.4.1	August 2012	50,463

Comparing: size

	Cloc	SLOCCount
AOO	6,004,901	5,570,062
OOo	5,309,587	4,753,965
LOa	5,437,769	4,852,832
LOb	5,309,587	4,720,906

http://cloc.sourceforge.net/

http://www.dwheeler.com/sloccount/

Comparing: languages (SLOCCount)

	C++	Java	XML
AOO	4,696,598 (84.32 %)	406,520 (7.30 %)	188,105 (3.38 %)
OOo	4,004,178 (84.23 %)	382,284 (8.04 %)	145,300 (3.06 %)
LOa	4,066,780 (83.80 %)	394,926 (8.14 %)	168,222 (3.47 %)
LOb	3,958,585 (83.85 %)	387,448 (8.21 %)	167,411 (3.55 %)

Comparing: similarity-tester

- Find percentage of a file included in some other
- Not symetric (imagine a small file being 100 % in a much larger file)
- Run for all files in two releases, pair to pair
- (ignoring binary files)
- Find all files included above a certain threshold (eg 95 %)
- Do it in both directions

similarity-tester Debian package

Comparing: similarity-tester (ii)

	AOO	OOo	LOa	LOb
AOO	50,463	4,348	-	4,381
OOo	2,672	42,731	12,581	7,260
LOa	-	15,363	42,160	27,610
LOb	3,357	7,253	27,259	39,637

(X, Y) means similarity X $\rightarrow Y(95\,\%)$

(number of files in X for which at least 95 % of their content is found in some file in Y)

Let's talk about methodology

Data lives in repositories not always designed to release all their data easily:

tools are needed to retrieve and extract it

Data includes many complexities and details

tools are needed to assist in its mining, analysis

The Metrics Grimoire approach

Set of tools specialized in retrieving information from different kinds of repositories. Among them:

- CVSAnalY: source code management (CVS, Subversion, git, etc.)
- Bicho: issue tracking systems (Bugzilla, Jira, SourceForge, Allura, Launchpad, Google Code, etc.)
- MLStats: mailing lists (mbox files, Mailman archives, etc.)

Store all the information in SQL databases with similar structure

`http://metricsgrimoire.github.com`
`https://github.com/MetricsGrimoire`

MetricsGrimoire: CVSAnalY

- Browses an SCM repository producing a database with:
  - All metainformation (commit records, etc.)
  - Metrics for each release of each file
- Also produces some tables suitable for specific analysis
- Multiple SCMs: CVS, svn, git (Bazaar partially)
- Whole history in the database, it's possible to rebuild the files tree for any revision
- Tags and branches support
- Option to save the log to a file while parsing
- Extensions system, incremental capabilities
- Multiple database system support (MySQL and SQLite)

MetricsGrimoire: CVSAnalY extensions

- Extension: a “plugin” for CVSAnalY
- Add information to the database, based in the information in the database and maybe the repository
- Usually: new tables for specific studies
- Simple example: commits per month per commiter
- Extensions add one or more tables to the database but they never modify the existing ones

MetricsGrimoire: CVSAnalY extensions

Some examples:

- FileTypes: adds a table containing information about the type of every file in the database (code, documentation, i18n, etc.)
- Metrics: analyzes every revision of every file calculating metrics like sloc and complexity metrics (mccabe, halstead). It currently supports metrics for C/C++, Python, Java and ADA.
- CommitsLOC: adds a new table with information about the total lines added/removed for every commit

MetricsGrimoire: Bicho

Parsing issue tracking systems

Results stored in a MySQL database

Information about each issue (ticket), and its modifications

Currently it supports:

- SourceForge (HTML parsing)
- BugZilla: GNOME, KDE, others
- Jira, Google Code, Allura, Launchpad (API)

It can work incrementally

MetricsGrimoire: MailingListStats

Parses mbox information (RFC 822)
Deals with Mailman archives
Stores results (headers, body) in a MySQL database:

- Sender, CCs, etc.
- Time / Date
- Subject
- ...

It can work incrementally
It can store multiple projects in a single database

Milking the databases

Once information is retrieved, and in suitable format for querying:

- it can be queried directly in the database
- it can be analyzed from R
- it can be filtered, manually inspected, improved
- it can be combined, cross-analyzed
- it can be visualized

We're building tools to simplify all of this: vizGrimoire

`https://github.com/VizGrimoire`

Why this approach?

Quantitative, objective data: facts, not opinions
Powerful: many specific questions can be answered
Transparent: you can reproduce the analysis easily
Even simple analysis may help stakeholders:

- Developers:
  Understanding, improving development processes
- Users, integrators:
  Long-term sustainability, evolution, reaction to issues
- Investors:
  Attraction of external resources, growth rate

In summary

- FLOSS development repositories have a wealth of information
- Their analysis is potentially interesting to any stakeholder
- Getting the data out of the repository is not that difficult...
- ...but the analysis may be difficult
- We're interested in deep analysis
- We're interested in working with developers, managers, users

What would you like to know about your pet project?

Bitergia: a start-up on free software metrics

Started operations in July 2012
Builds on the experience of LibreSoft R&D group
Offering professional products and services
Focused on:

- Metrics about software developent (including community metrics)
- Specialized support for development forges (including metrics for projects)

http://bitergia.com
http://blog.bitergia.com
http://libresoft.es

This is the end

Have you learned something useful?

[I would love to know what interested you the most]
[...and the least]

`http://blog.bitergia.com/2012/10/17/presentation-at-the-libreoffice-conference/`

`http://wp.me/p2cQGW-4d`