Software Development Analytics with GrimoireLab

Jesus M. Gonzalez-Barahona

Universidad Rey Juan Carlos
@jgbarah http://github.com/jgbarah/presentations

Intl. Summer School on Visual Soft. Analytics
Leipzig (Germany), September 23rd 2019

It is difficult to improve
if you cannot measure
and track your improvement

Our plan today

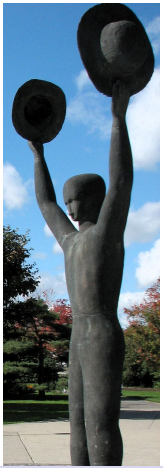
1. A bit of context
2. Dealing with dynamic complexity
3. Data sources
4. GrimoireLab
5. Case studies
   - Activity
   - Remaining code
   - Performance
   - Demographics

Me and my two hats

Uni Rey Juan Carlos:

- Understanding free, open source software
- Data analytics approach
- Data visualization in XR

http://gsyc.es/jgb

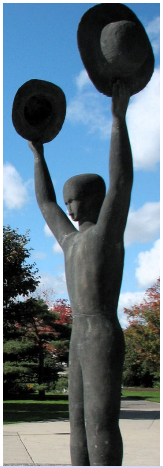


Me and my two hats

Bitergia:

- From research to the real world
- Understanding software development
- Data analytics approach

`http://bitergia.com`

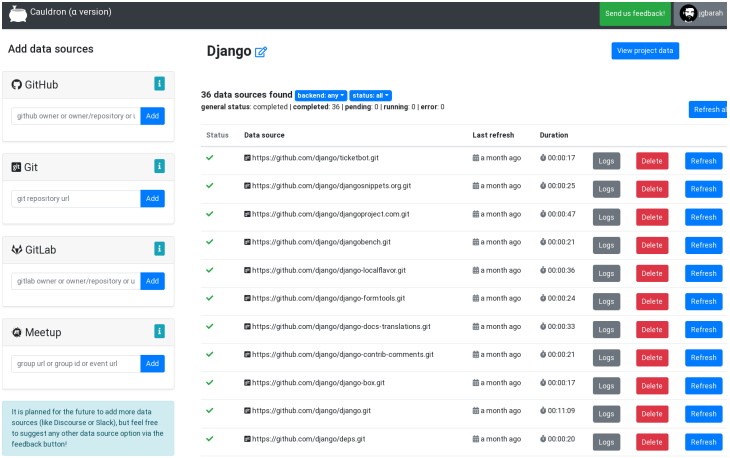


Recommendations

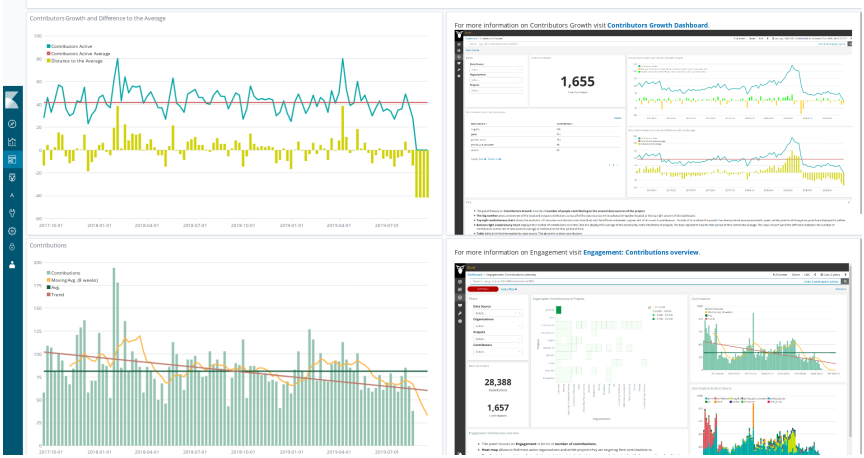
- Open your laptop
- Download the slides (they have links)
- Visit Alpha.Cauldron.io and produce your own dashboard
- Play with the dashboards
- Understand the interpretations behind the numbers

`https:/alpha.cauldron.io`

Cauldron Alpha



Cauldron Alpha



Dealing with dynamic complexity

Development projects may be large and complex

Projects may be large and complex...
and dynamic

It's difficult to...

- ...track what's happening
- ...understand why it's happening
- ...react quickly
- ...evaluate results of reaction

If data is available
analytics may come to the rescue

A continuous process

Figure out your interest

Find out available data

Define key parameters

Monitor, understand, detect deviations

Act to correct, improve

Track results

$$Measure \rightarrow Monitor \rightarrow Act$$

A continuous process

Case example: Overall development activity

Interest: activity

Data: changes to code, tickets

Parameters: commits, tickets closed

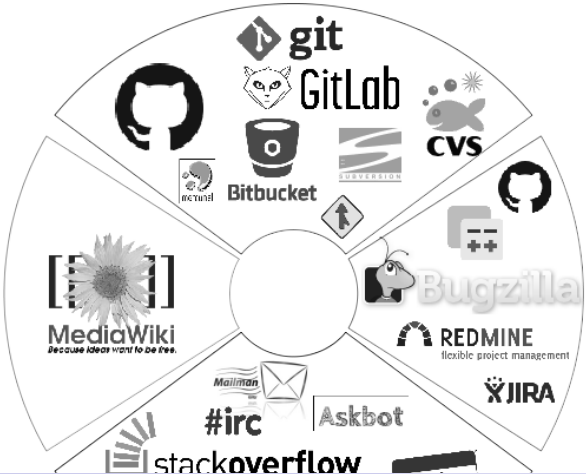
Monitoring: charts, numbers

Observation: numbers declining

Action: allocate more developer effort

Data sources

Repositories, repositories...



Source code management

- Client/server: CVS, Subversion
- Decentralized: git, Mercurial, Bazaar, etc.
- Most of them accessible through git... (with some problems)
- Can be integrated with other tools: Gerrit, GitHub, GitLab, etc.

Issue tracking

Many different systems:

- Bugzilla
- Jira
- GitHub issues
- GitLab Issues
- Phabricator
- RedMine...

Each with a different model, data, operations...

Code review

Usually: peer review pre-merge review

Different methods:

- Mailing lists (eg: Linux)
- Gerrit (eg: OpenStack)
- GitHub pull requests (eg: ElasticSearch)
- GitLab merge requests (eg: GNOME)
- or even Jira, Bugzilla...

Much of the control on the software lies here

Async communication

Mailing lists:

- Mailing lists systems (Mailman)
- Google Groups
- Mailing list archivers

Forums: too many to mention

Question/Answer sites: StackOverflow, Askbot

Information is always archived

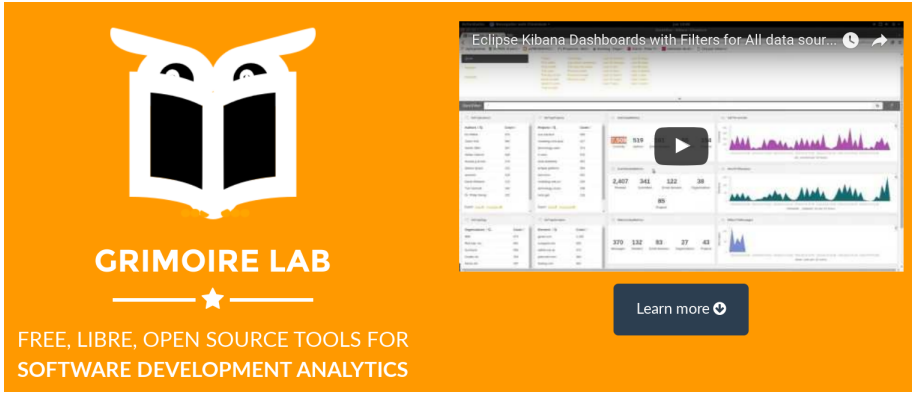
Sync communication

Systems:

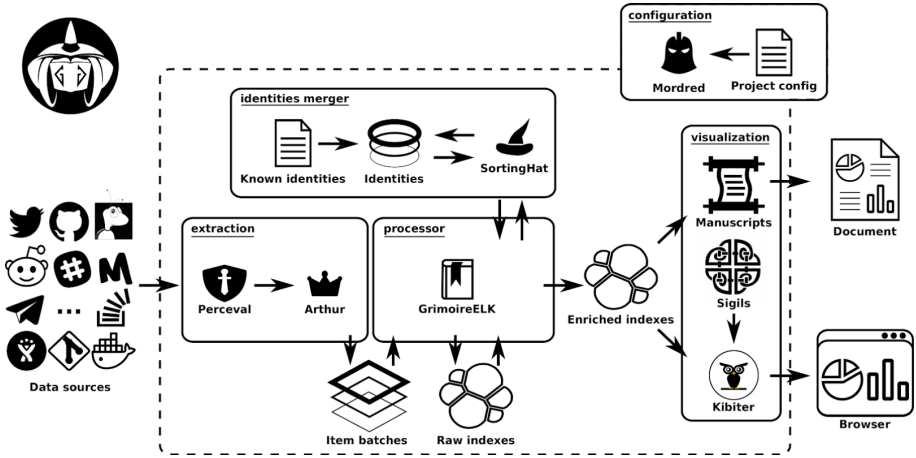
- Traditionally: IRC
- Nowadays: Slack & many others
- Not always text/based (eg: videoconferences)

Notes:

- In many cases, lack of archives
- Privacy concerns: considered informal



https://chaoss.github.io/grimoirelab



https://chaoss.github.io/grimoirelab

Main components

- Perceval: data retrieval
- Arthur: retrieval orchestration
- GelK: enrichment
- SortingHat: identity management
- ElasticSearch (*): database
- Kibiter: dashboard (light fork of Kibana)
- Sigils: visualizations for Kibana/Kibiter

(*) Not a part of GrimoireLab

Perceval

```
$ python3 -m venv gl
$ source gl/bin/activate
(gl) $ pip install grimoirelab
(gl) $ perceval git \
  https://github.com/chaoss/grimoirelab-perceval
(gl) $ perceval github \
  chaoss grimoirelab-perceval
```

https://chaoss.github.io/grimoirelab-tutorial/perceval

```
{"backend_name": "Git",
"backend_version": "0.11.1",
"category": "commit",
"classified_fields_filtered": null,
"data": {
  "Author": "Santiago Due\u00f1as <sduenas@bitergia.com>",
  "AuthorDate": "Tue Aug 18 18:08:27 2015 +0200",
  "Commit": "Santiago Due\u00f1as <sduenas@bitergia.com>",
  "CommitDate": "Tue Aug 18 18:08:27 2015 +0200",
  "commit": "dc78c254e464ff334892e0448a23e4cfbfc637a3",
  "files": [{
     "action": "A",
     "added": "10",
     "file": ".gitignore",
```

```
{"backend_name": "GitHub",
"backend_version": "0.22.1",
"category": "issue",
"classified_fields_filtered": null,
"assignee_data": {},
"assignees": [],
"assignees_data": [],
"author_association": "CONTRIBUTOR",
"body": "Based on Sphynx, prepared...",
"closed_at": "2016-01-04T13:51:56Z",
"comments": 0,
"comments_data": [],
"comments_url": "https://api.github.com/...",
"created_at": "2016-01-03T23:46:04Z",
```

Perceval as a module

```
#! /usr/bin/env python3
from perceval.backends.core.git import Git

repo_url = 'http://github.com/chaos/grimoirelab-perceval'
repo_dir = '/tmp/perceval.git'

repo = Git(uri=repo_url, gitpath=repo_dir)
for commit in repo.fetch():
    print(commit['data']['commit'])
```

```
import argparse
from perceval.backends.core.git import Git

parser = argparse.ArgumentParser(description = "Count comm
parser.add_argument("repo", help = "Repository url")
parser.add_argument("--print", action='store_true', help =
args = parser.parse_args()

repo = Git(uri=args.repo, gitpath='/tmp/perceval.git')
count = 0
for commit in repo.fetch():
    if args.print:
        print(commit['data']['commit'])
    count += 1
print("Number of commmits: %d." % count)
```

SirMordred

Producing a dashboard:

- Elasticsearch installed
- Kibana / Kibiter installed
- MariaDB installed
- Config: mordred.cfg, projects.json, identities.yaml, menu.yaml

```
(gl) $ mordred -c mordred.cfg
```

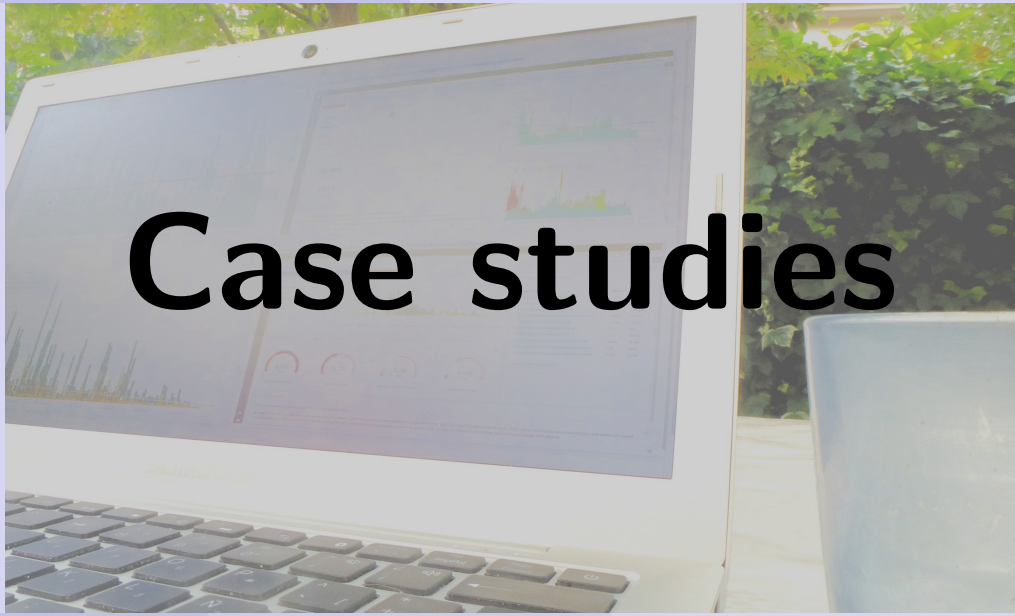
https://chaoss.github.io/grimoirelab-tutorial/sirmordred

GrimoireLab

Software development analytics with free, open source software

(a CHAOSS project)

chaoss.github.io/grimoirelab
chaoss.github.io/grimoirelab-tutorial



Case studies

Tracking involved parties

Development is much more than developers (this is explicit in FOSS & inner sourcing)

- Developers: all repositories
- Contributors: issue tracking, async communication
- Users: async communication, ...
- Ecosystem: difficult to track

Activity / size

- committing patches:
  source code management system
- reporting, commenting or fixing bugs:
  issue tracking system
- submitting patches or reviewing them:
  code review system
- sending messages:
  async or sync communication systems

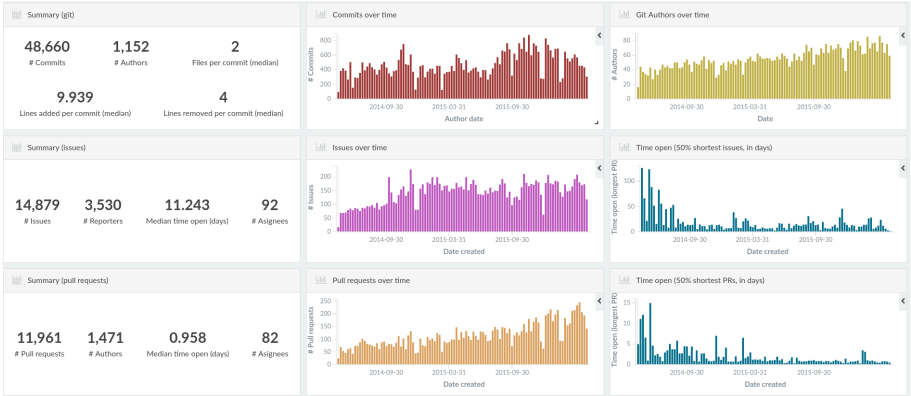
Most common cases

- Parameters reflecting activity for a period.
- People active for a certain period.
- Evolution of any of them.
- Trends for any of them.

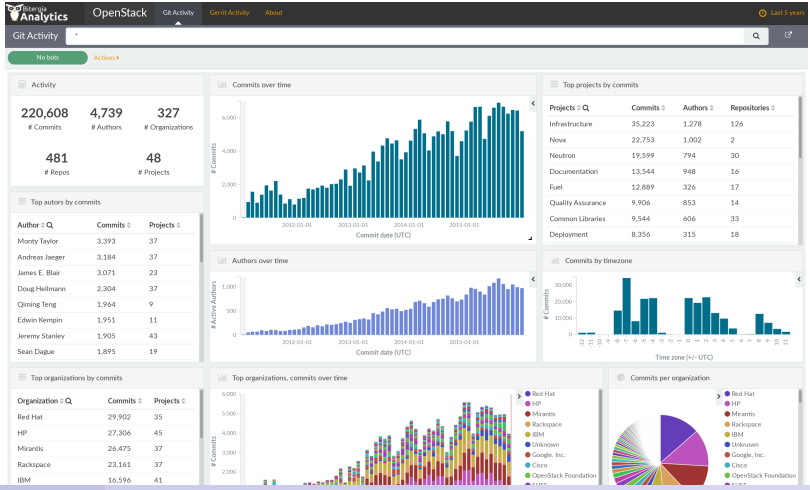
Difficult to compare between projects

Interesting to compare in-project

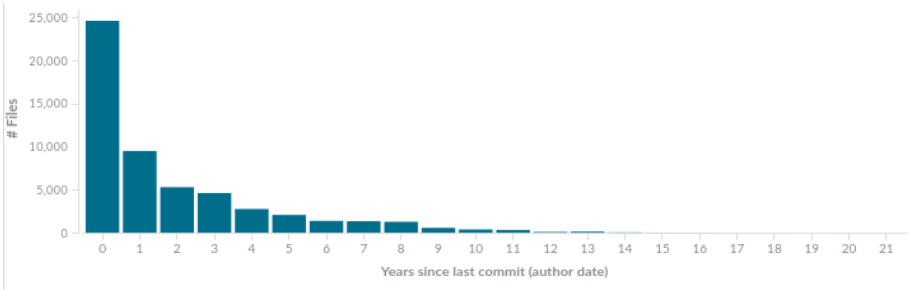
Many facets



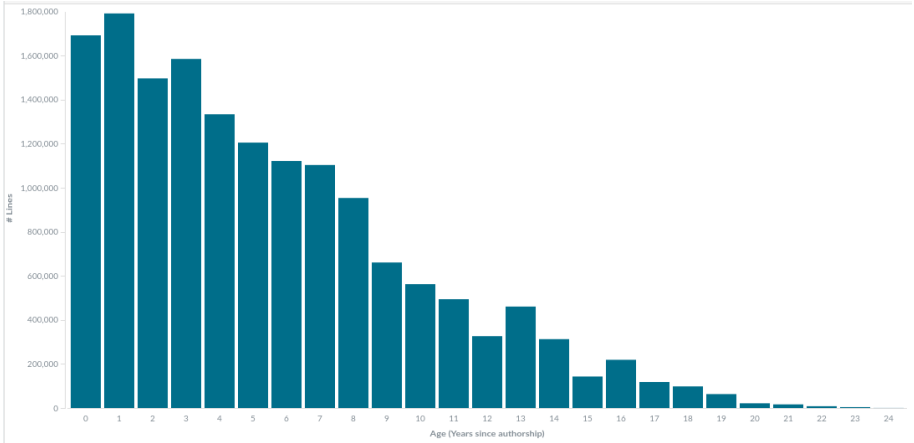
Many facets



How old is code?

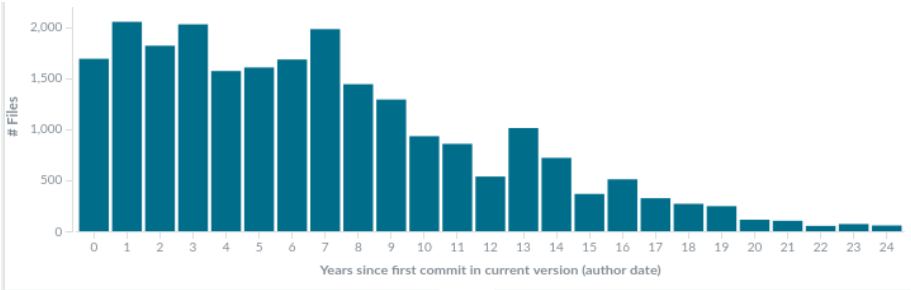


[Linux kernel, July 2016, C files by last commit]

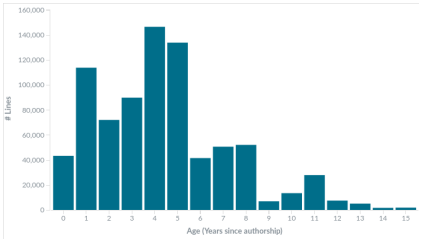


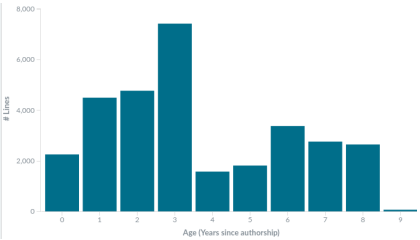
[Linux kernel, July 2016, lines in C files by age]

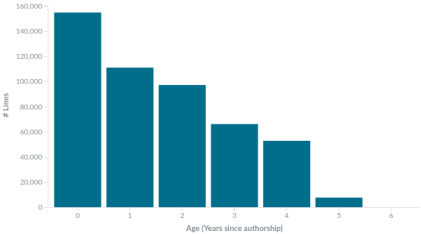
How old is code (3)

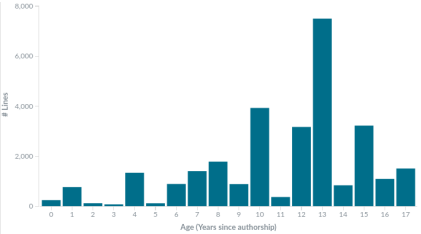


[Linux kernel, July 2016, C files by first remaining commit]





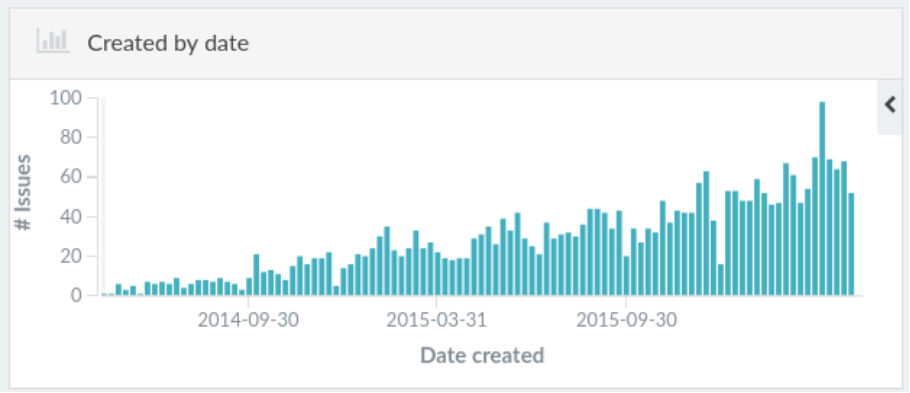


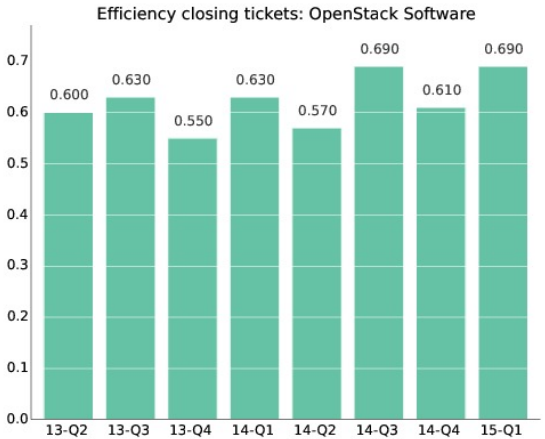


Age of lines (data of authorship, ".c" files in Linux)

From top left, clockwise: Wireless, USB, IRDA Ethernet

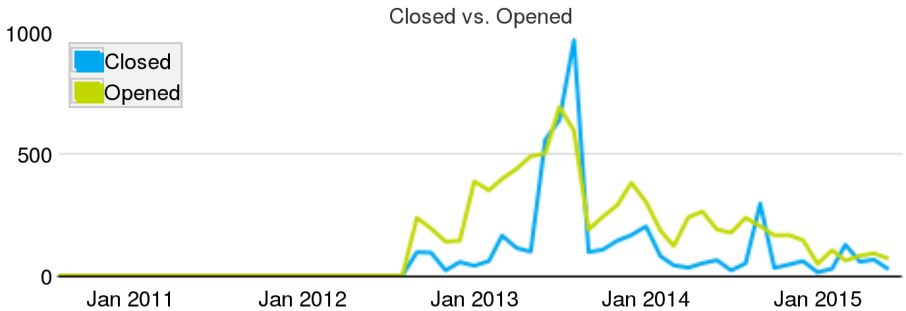
Performance (backlog)



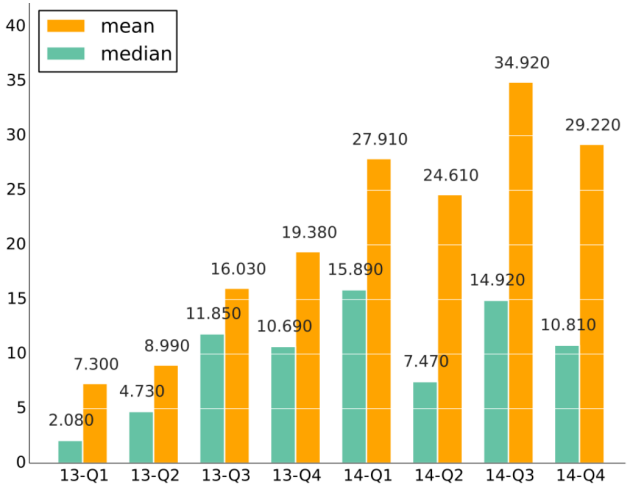


Efficiency. Example: closed/opened tickets per quarter

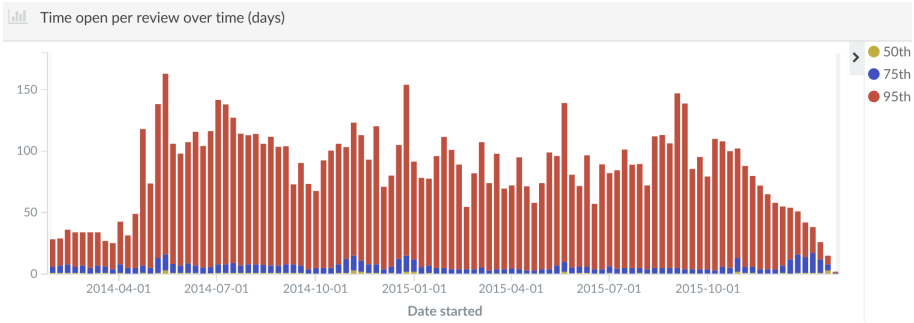
Tickets



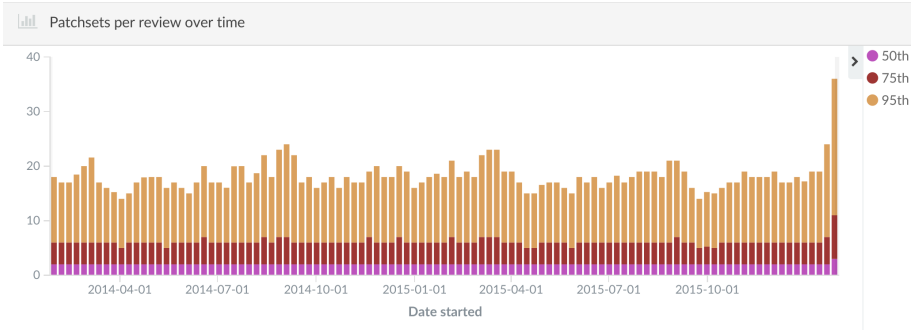
Review: time to merge



Review: time to merge



Versions per review



The coding process

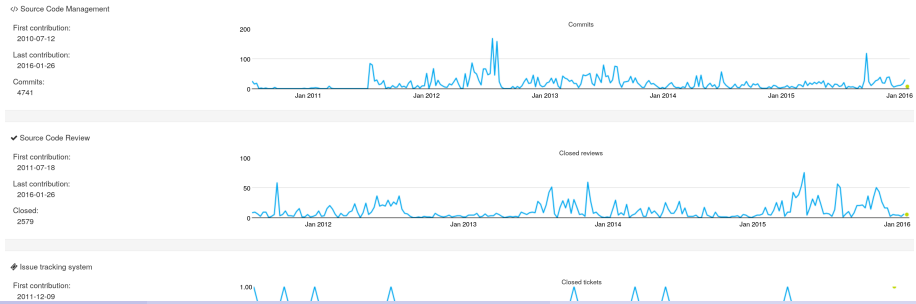
From idea to implementation

- Story, design
- Ticket(s)
- Code review
- Automated testing
- Commit in code base

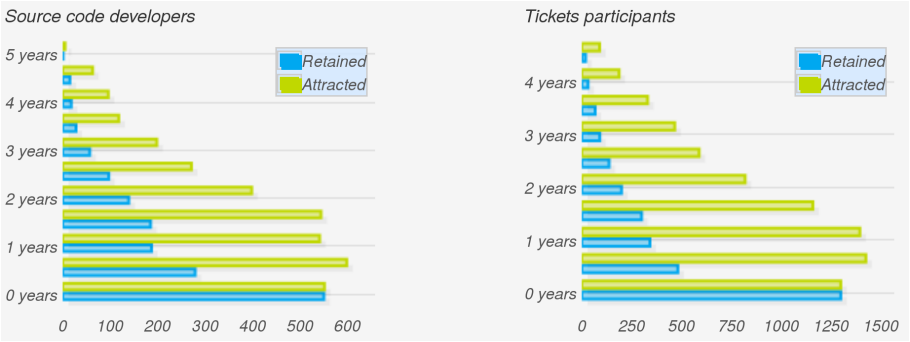
The OpenStack case

- Blueprint (if feature), Launchpad
- Ticket (bug, feature), Launchpad
- Code review, Gerrit
- Automated testing, Jenkins
- Commit in code base, Gerrit, Git

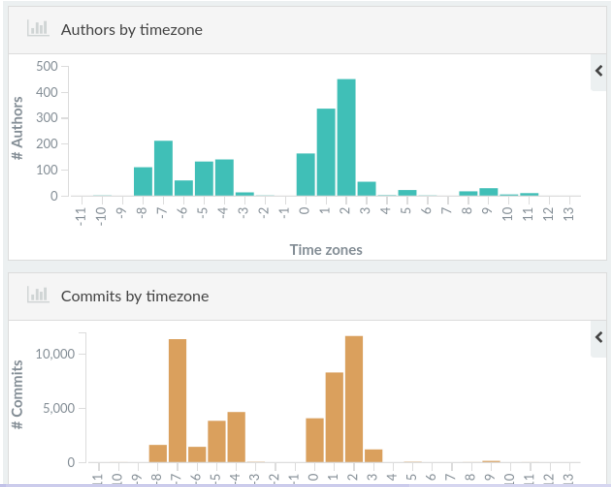
- The repository level.
- The class of repository level.
- The project level.
- The global level.



The aging chart

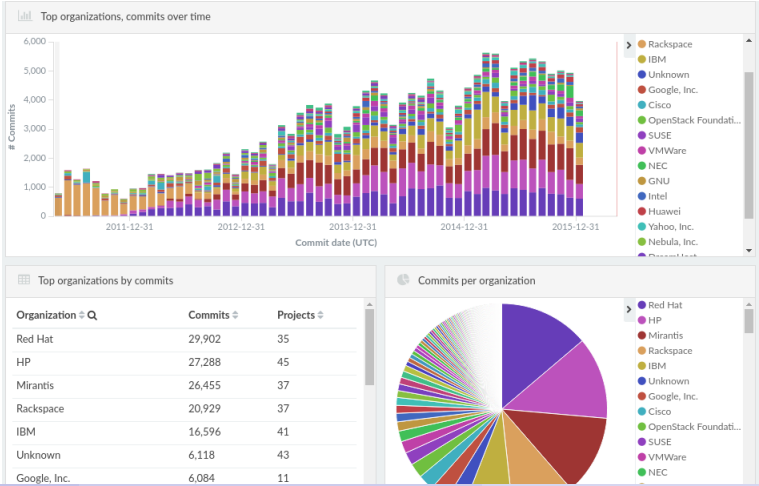


Time zones



GitHub profiles

Affiliation



Apache Pony Factor

Pony Factor (PF) shows the diversity of a project in terms of the division of labor among committers in a project.

Pony Factor is determined as: **"The lowest number of committers whose total contribution constitutes the majority of the codebase"**

ke4qqq.wordpress.com/2015/02/08/pony-factor-math/

Bitergia Elephant Factor

Bitergia Elephant Factor

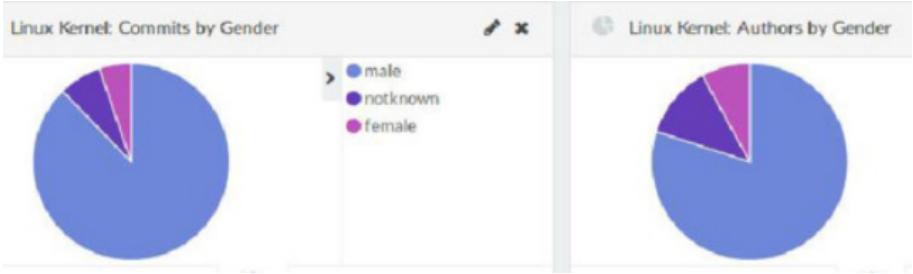
The elephant factor shows the diversity of a project in terms of the division of labor among companies (by mean of developers affiliated with them).

Elephant factor is determined as:
"The lowest number of companies whose total contribution (in commits by their employees) constitutes the majority of the commits"

Some projects (2016)

	Pony Factor	Elephant Factor	Commits (excl bots)
OpenNebula	4	1	12K
Eucalyptus	5	1	25K
CloudStack	14	1	42K
OpenStack	>100	6	126K
CloudFoundry	41	1	60K
OpenShift	10	1	15K
Docker	15	1	18K
Kubernetes	12	1	7K

Diversity: Gender gap



Commits by women: 6.8 % (4 Kcommits)

Women: 9.9 % (330 developers)

Linux kernel, Nov 2015 – Oct 2016

Final remarks

Room for improvement

- Many other aspects... explore your own
- Refine what is important
- Explore new ways of making data useful
- Tell interesting stories based on data
- Visualization is very important
- Higher-order metrics
- Simplify results, make them meaningful

Summary

You cannot improve
what you cannot measure

Fortunately, you can measure a lot of things...

`http://chaoss.github.io/grimoirelab`

Credits (1)

- "Man With Two Hats"
  Statue by Henk Visch, located in Otawa, Canada
  Picture by Lezumbalaberenjena in Wikimedia Commons
  License: Public domain
  `https://commons.wikimedia.org/wiki/File:Man_With_Two_Hats_Ottawa_Statue_by_lezumbalaberenjena.jpg`
- "Crowd at FOSDEM 2008"
  by Jesús Corrius
  License: CC Attribution 2.0
  `http://www.flickr.com/photos/jcorrius/2302302707/`

This document (including source) is available from https://github.com/jgbarah/presentations