UMONS
Université de Mons

Understanding the evolution of software project communities

Tom Mens & Mathieu Goeminne
Service de Génie Logiciel, Institut d'Informatique
Faculté des Sciences, Université de Mons

Faculté
des Sciences

Research topic

- Study of open source software evolution
- Taking into account the community (social network) of persons surrounding a software project (developers, users, ...)
- By analysing and combining data from different types of repositories

Goals

- Understand (through repository mining and analysis)
  - How software quality is impacted by how the community is structured
  - How do software product, project and community co-evolve
  - How developers and users interact and influence one another
- Improve (through guidelines and tool support)
  - The way in which communities should be structured
  - The quality of the software process and product
  - The interaction between developers and users

Questions

- Is there a core group (of developers and/or users) being significantly more active than the others?
- How does a person contribute to different types of activities?
- Are the recurrent patterns of activity in the community?
- How do particular “events” impact the project?
- How does developer intake and turnover affect the project?
- Do we find evidence of Pareto principle (inequality of activity)
- Is there a “bus factor” effect?
- How does the activity distribution evolve over time?
- How does the activity distribution vary across different projects?

Why open source?

- Free access to source code, defect data, developer and user communication
- Observable communities
- Observable activities
- Increasing popularity for personal and commercial use
- A huge range of community and software sizes

Methodology

- Exploit available data from different repositories
  - code repositories
  - mail repositories (mailing lists)
  - bug repositories (bug trackers)
- Select open source projects
- Use Herdsman framework
  - Based on FLOSSMetrics data extraction
  - Use identity merging tool
  - Use of econometrics
  - Use of statistical analysis and visualisation

Selected Projects

- Criteria for selecting projects
  - Availability of data from repositories
  - Data processable by FLOSSMetrics tools
    - CVSAnaly2, MLStats, Bicho
  - Size of considered projects: persons involved, code size, activity in each repository
  - Age of considered projects

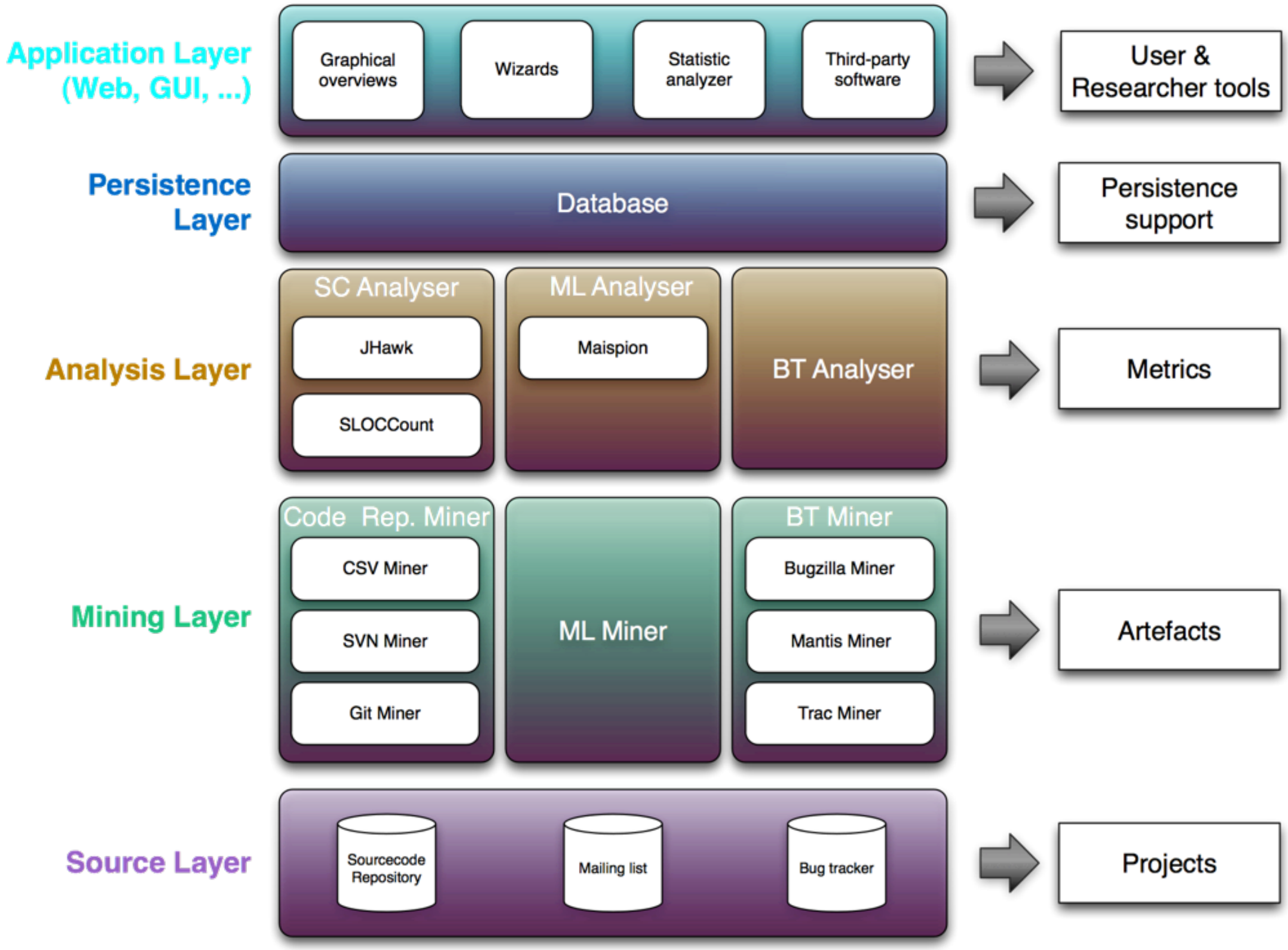
Selected Projects

	Brasero	Evince	Subversion	Wine
versioning system	git	svn	svn	git
age (years)	8	11	11	11
size (KLOC)	107	580	422	2001
#commits	4100	4000	51529	74500
#mails	460	1800	24673	14000
#bugs	250	950		3300

Herdsman framework

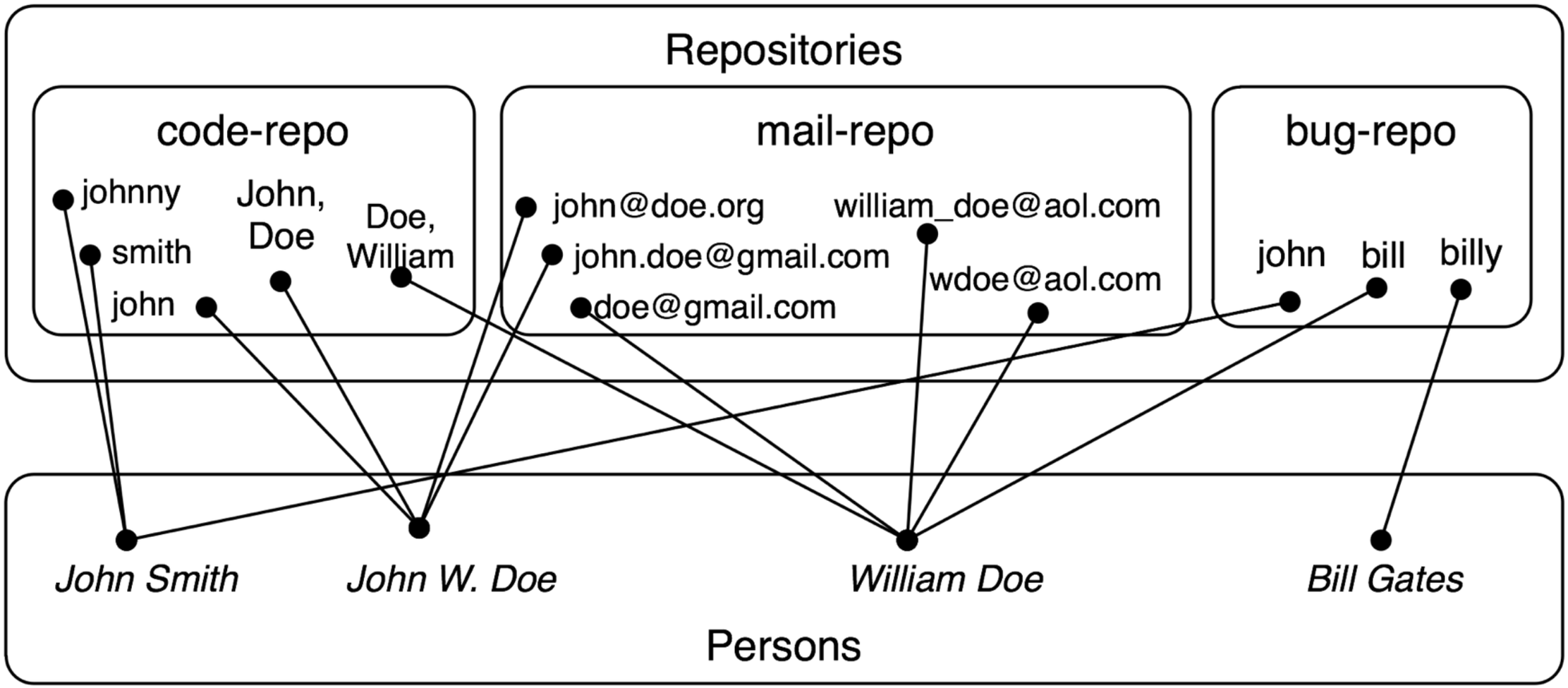
- Exploiting available data from different repositories
  - code repositories: to detect committer activity
  - mail repositories (mailing lists): to detect mailer activity
  - bug repositories (bug trackers): to detect bug-related activities

Herdsman Framework



UMONS
Université de Mons

Identity Merging



Faculté
des Sciences

UMONS
Université de Mons

Comparing Merge Algorithms

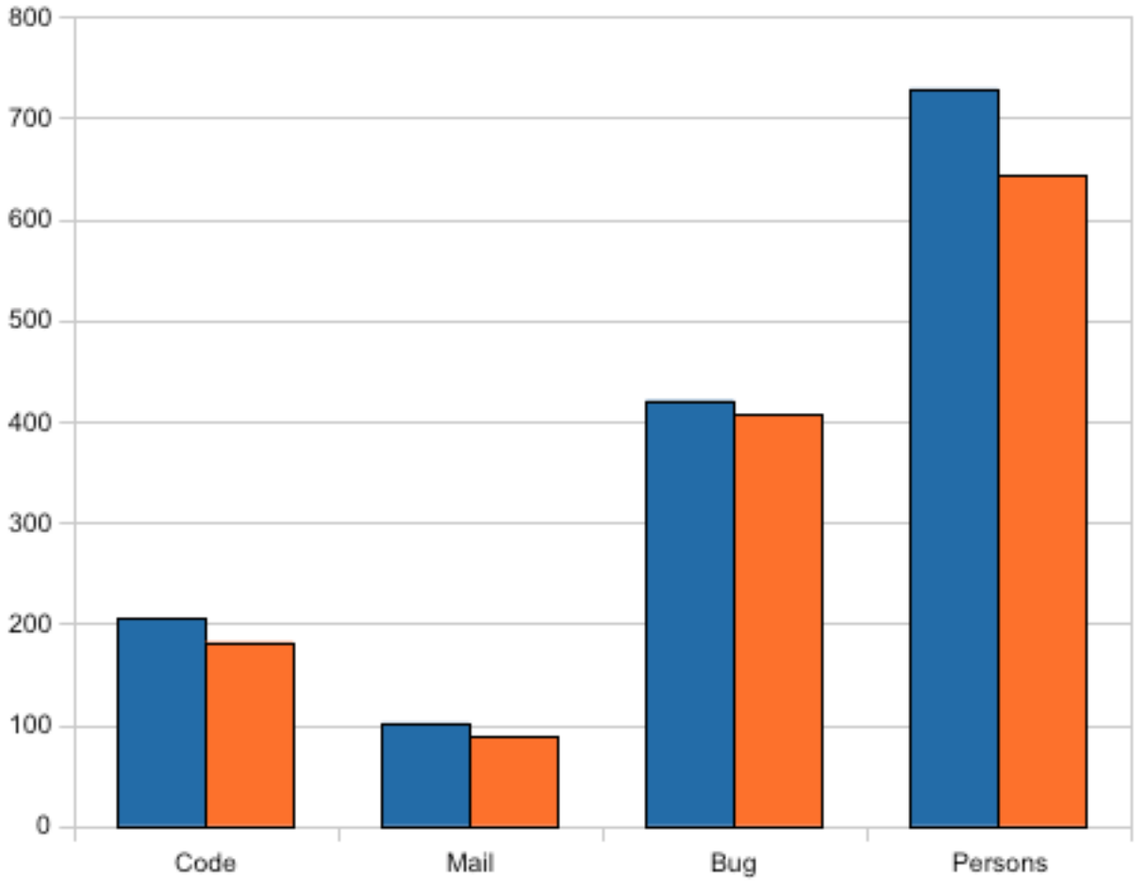
- Based on a *reference merge model*
  - manually created
  - iterative approach
    - relying on information contained in different files (COMMITTERS, MAINTAINERS, AUTHORS, NEWS, README)
- Compute, for each algorithm, *precision* and *recall* w.r.t. reference model

Faculté
des Sciences

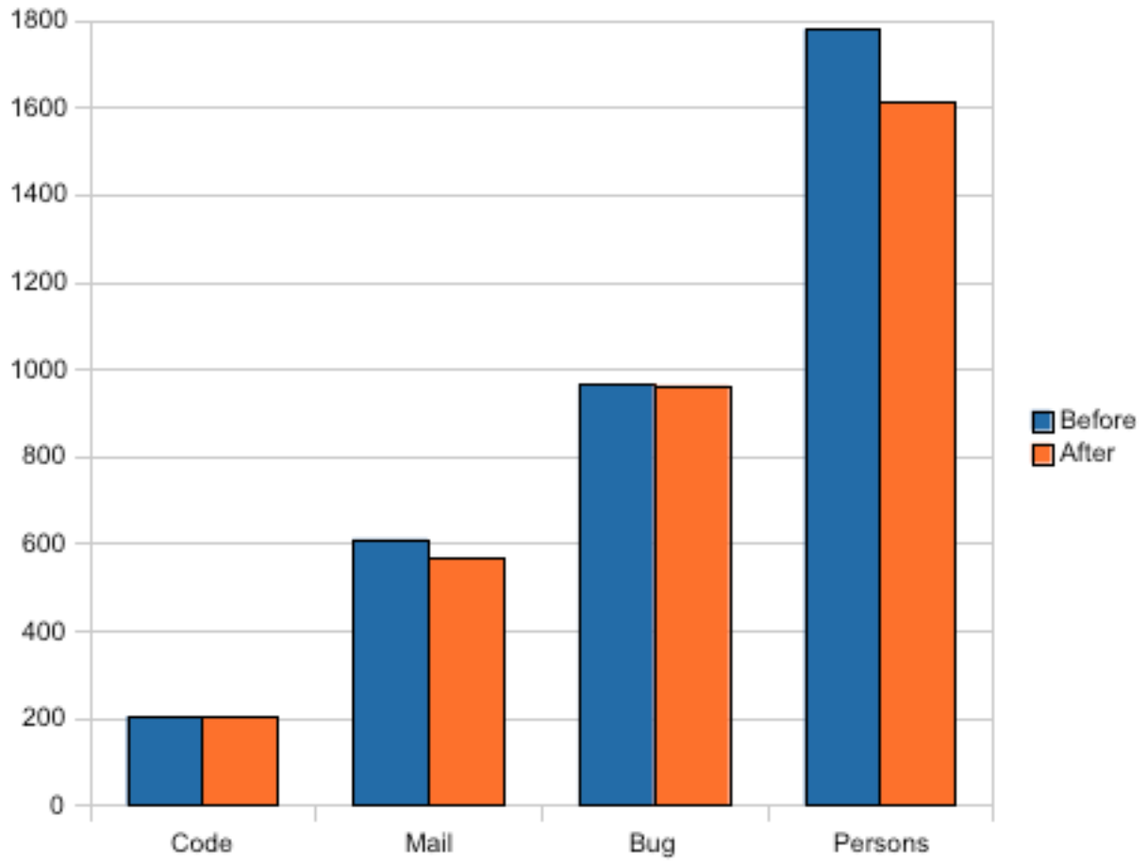
UMONS
Université de Mons

Reference merge model

Brasero



Evince



Before: one repository account = one person
After: one person may have multiple accounts

Faculté
des Sciences

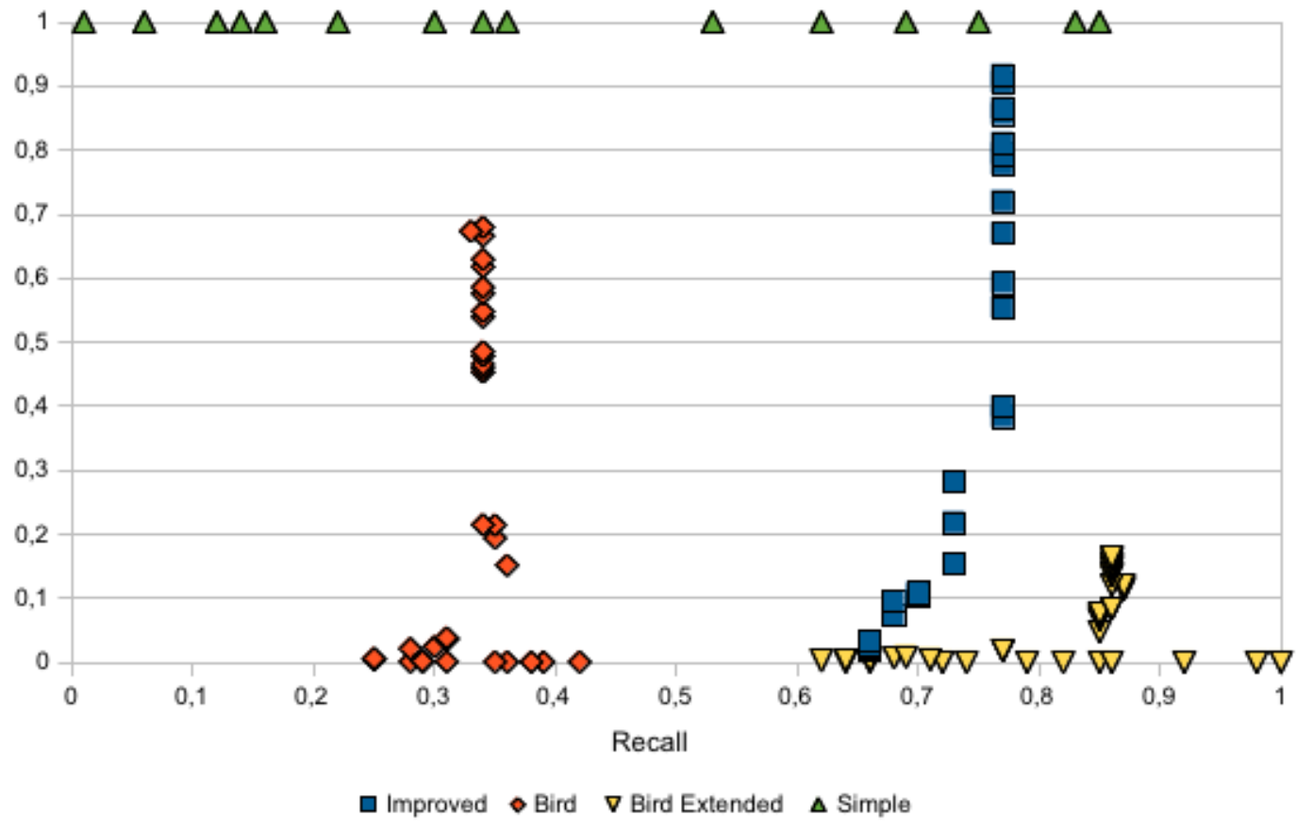
Comparing Merge Algorithms

- Simple
- Bird (code and mail repositories only)
  - based on Levenshtein distance
- Bird extended for bug repositories
- Improved
  - Combining ideas from Bird and Robles

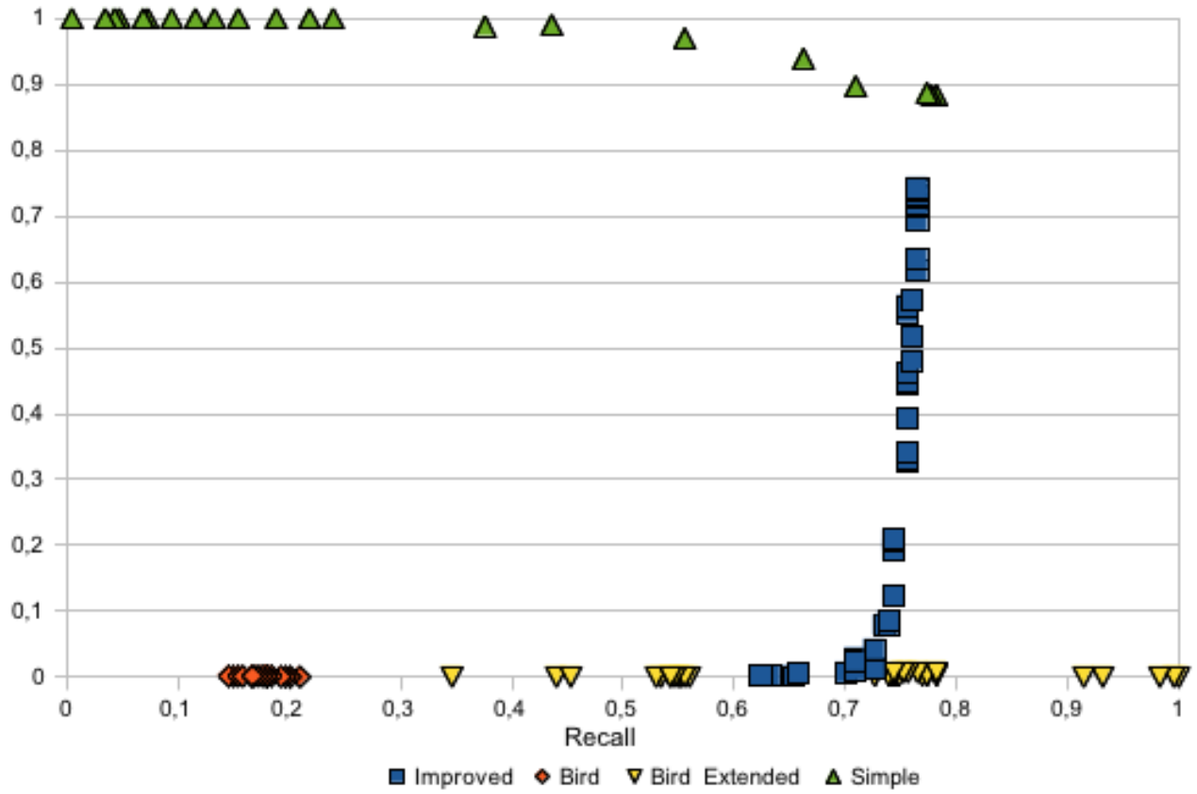
UMONS
Université de Mons

Comparing Merge Algorithms

Brasero



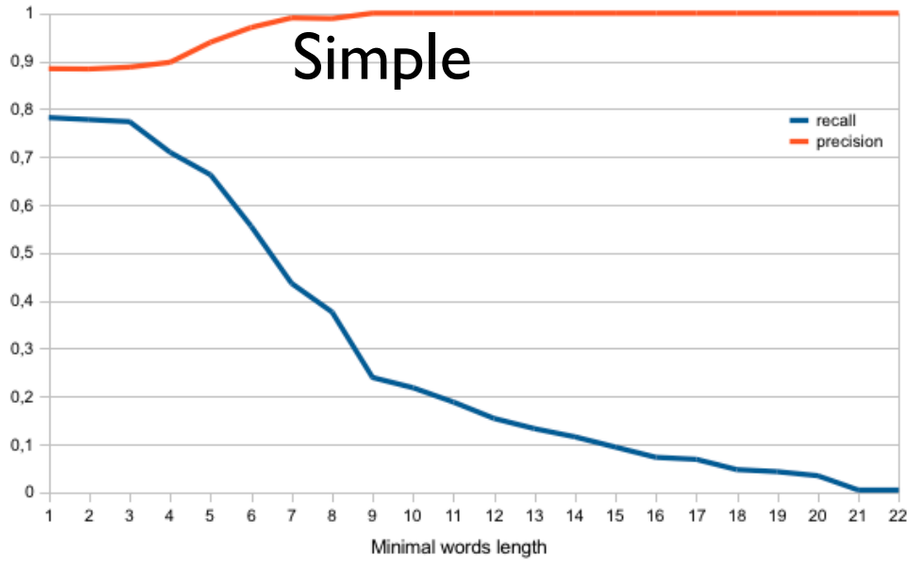
Evince

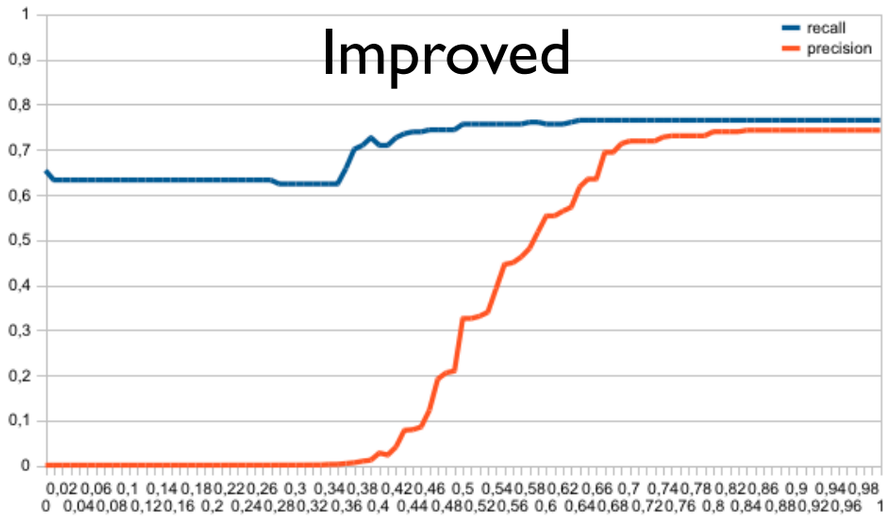


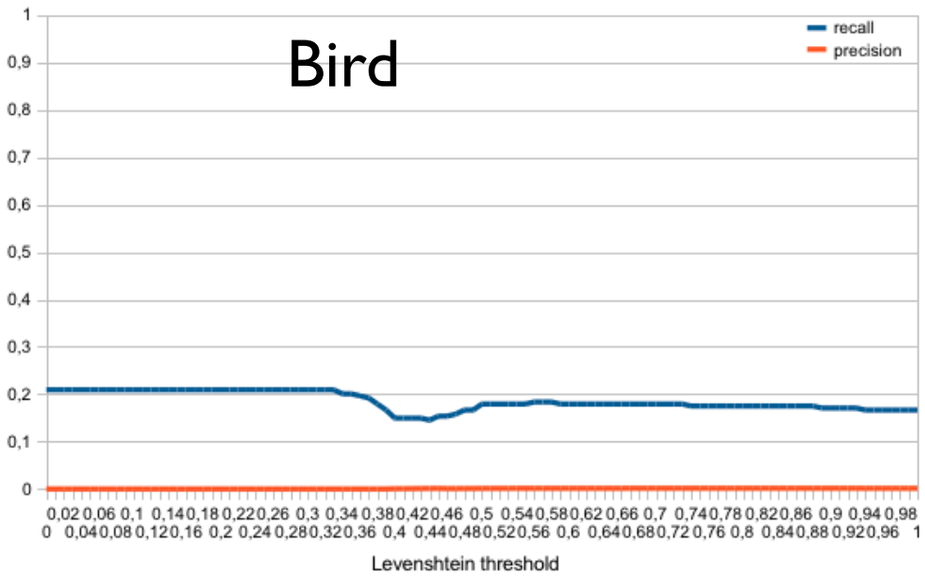
Faculté
des Sciences

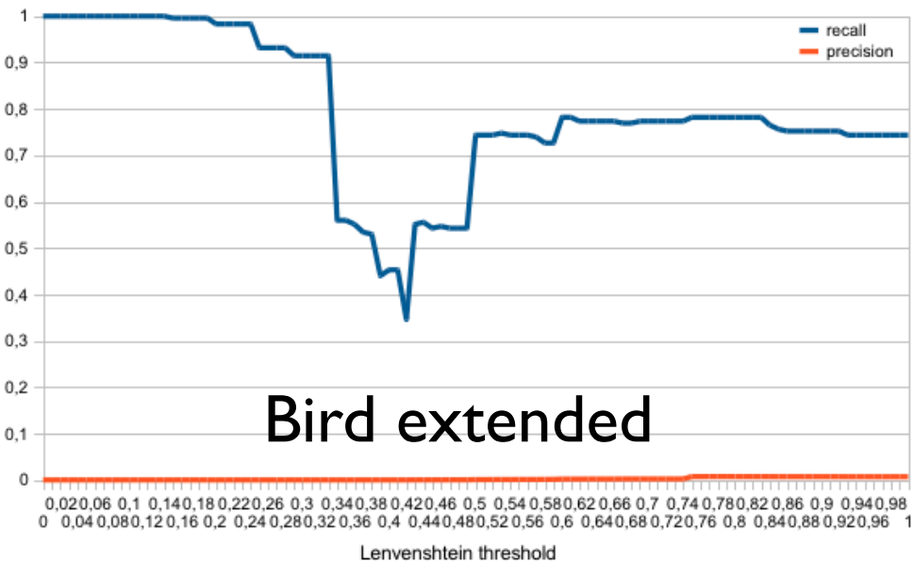
UMONS
Université de Mons

Comparing Merge Algorithms (varying parameter values) - Evince









Faculté
des Sciences

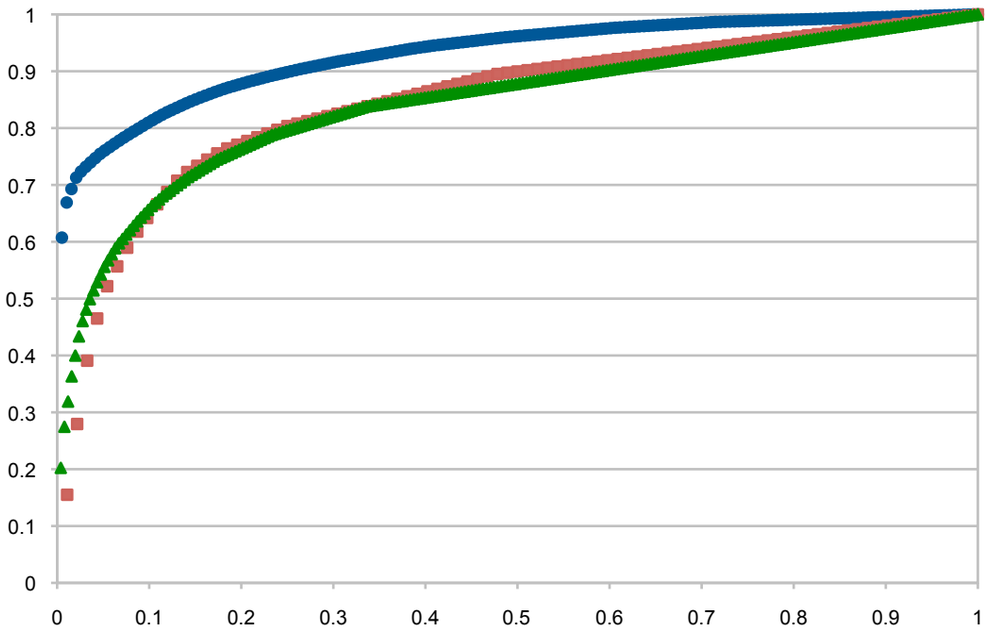
Activity distribution

- How are developer activities (commits, mails, bug fixes) distributed?
  - For a single release: do we observe an unequal distribution ?
  - Over time: do we observe a change in this distribution ?

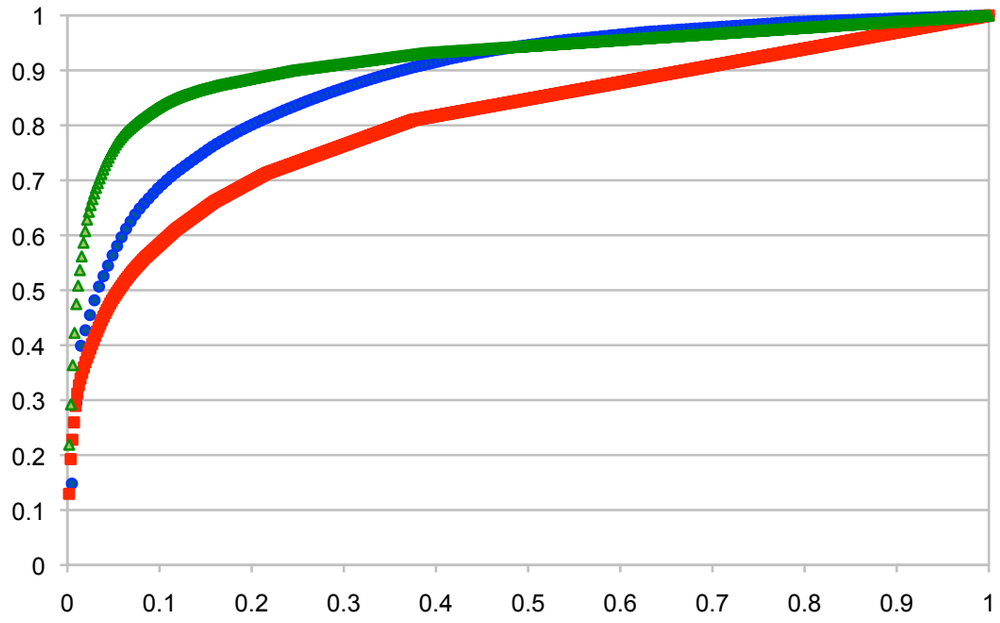
Activity distribution For a single release

- Evidence of Pareto principle (20/80 rule)?
  - Most activity is carried out by a small group of persons.
    - Typically : 20% do 80% of the job.
- Doesn't necessarily imply that the activity distribution follows a Pareto law

Activity distribution

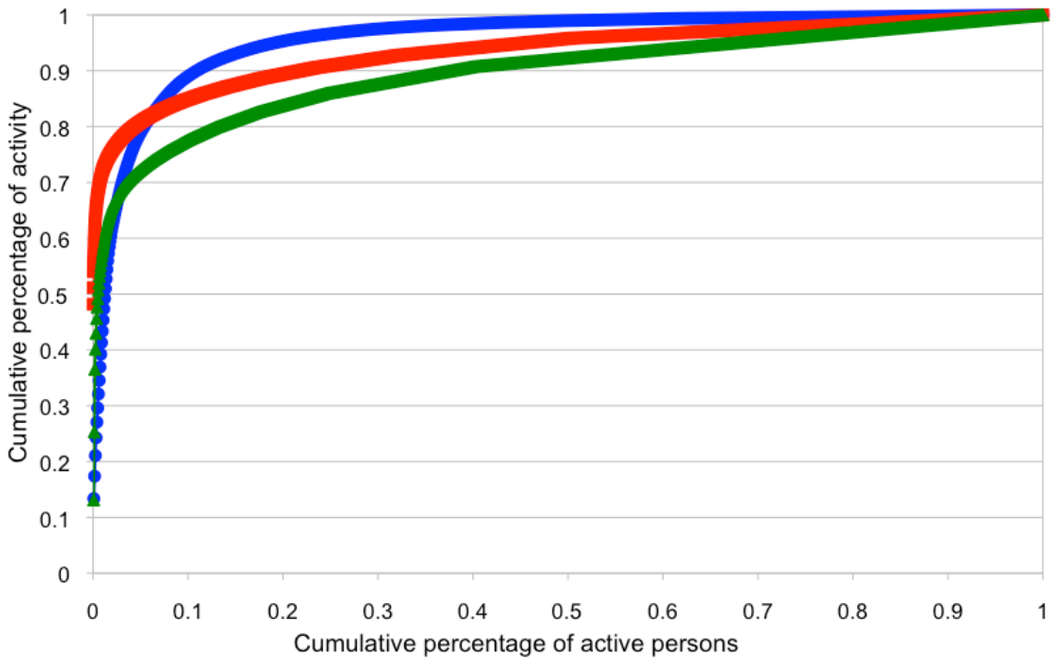


Brasero



Evince

Wine



commits

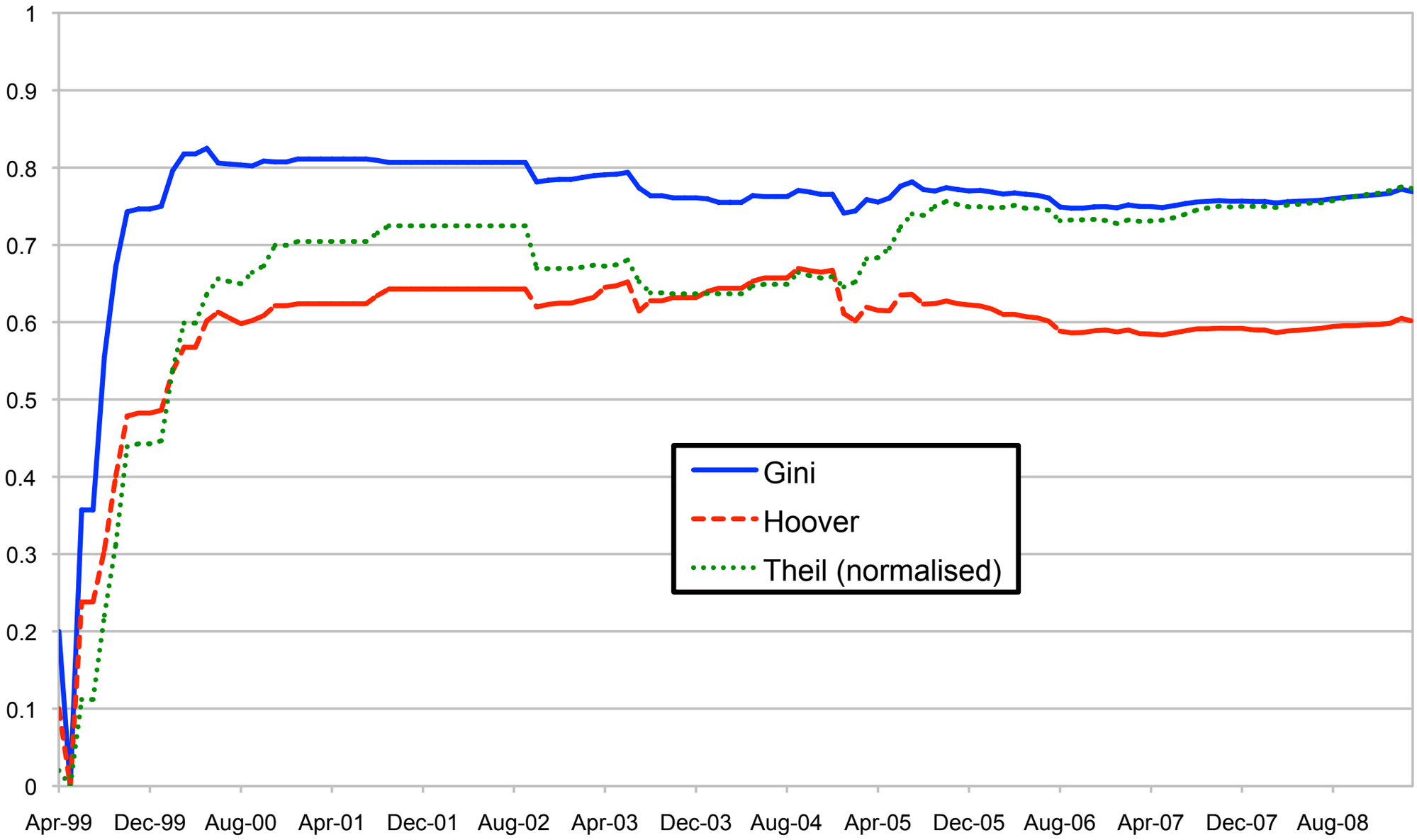
mails

br changes

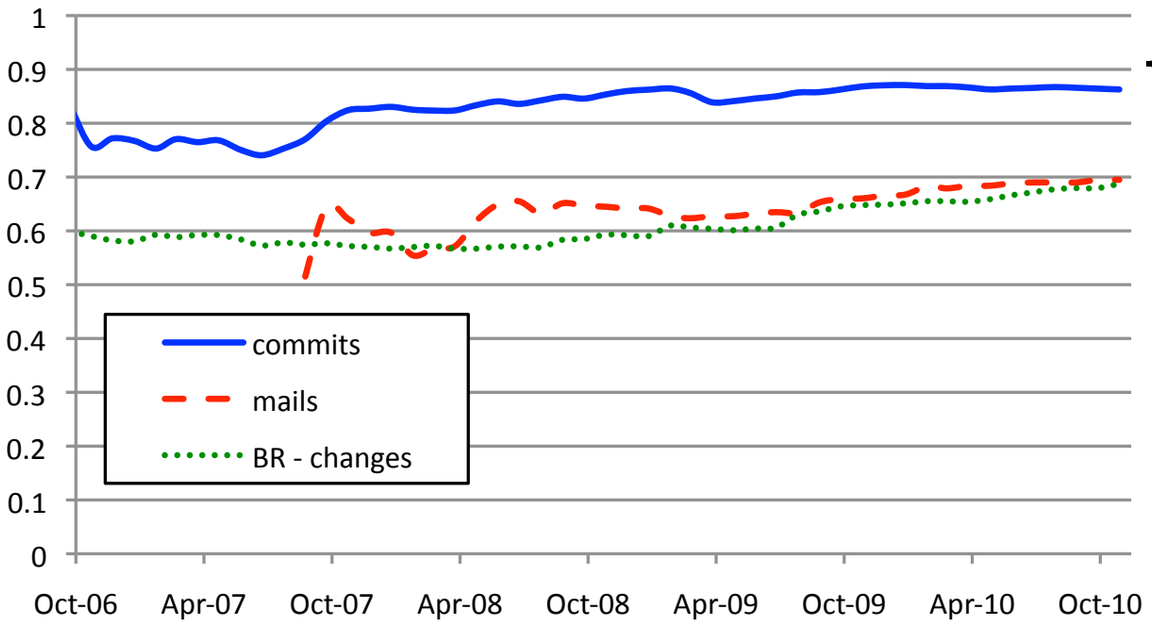
Activity distribution Over time

- Econometrics
  - express inequality in a distribution
  - aggregation metrics:
    Gini, Hoover, Theil (normalised)
  - Values between 0 and 1
    - 0 = perfect equality; 1 = perfect inequality

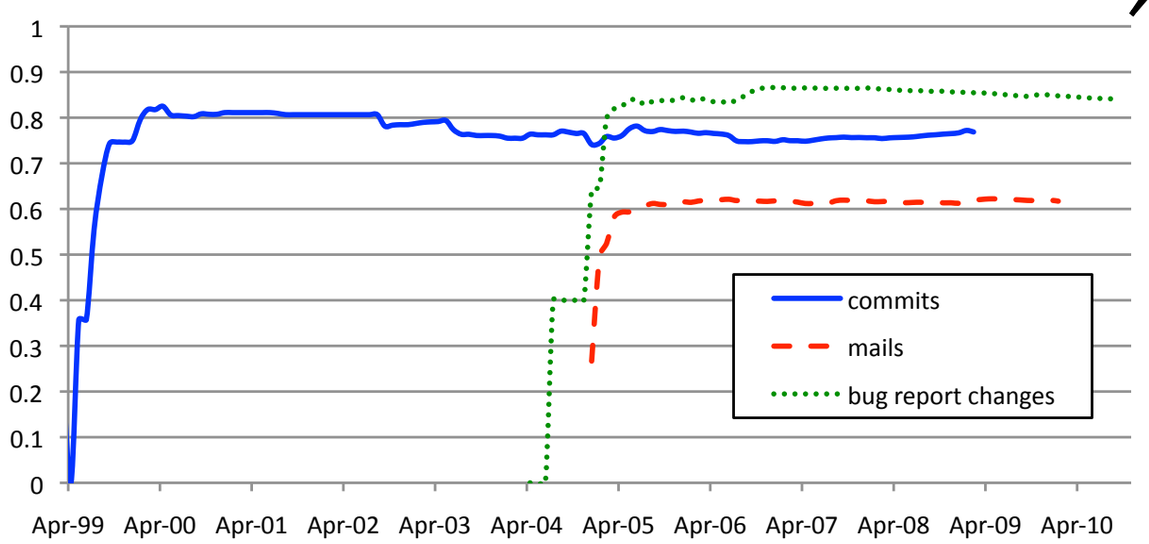
Activity distribution (aggregation indices for Evince)



Activity distribution (Gini index)

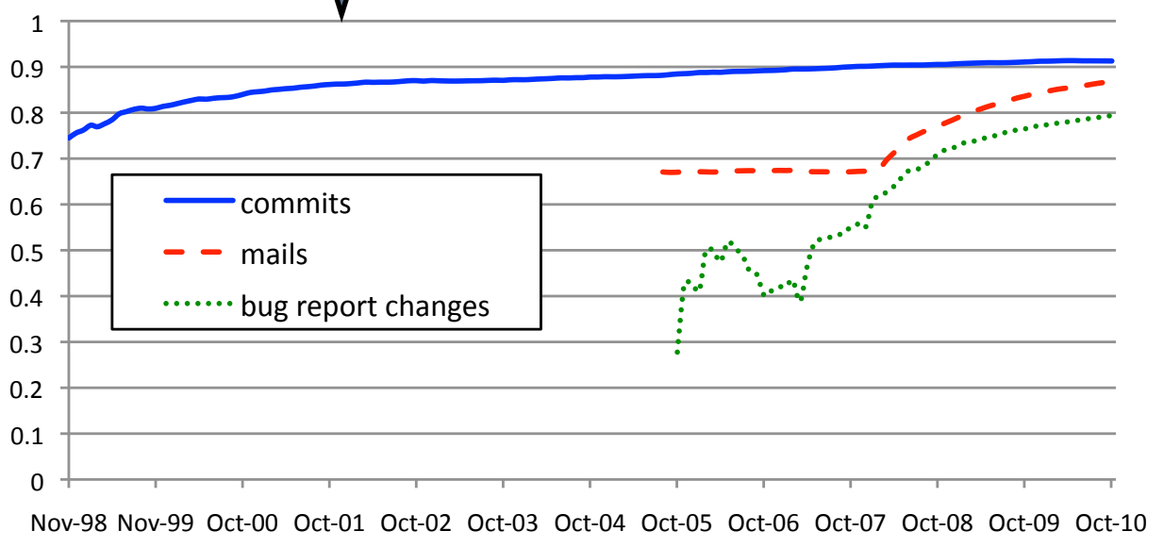


Brasero



Evince

Wine



Activity Distribution Conclusion

- Activity distributions seem to become more and more unequally distributed
- The Pareto principle is clearly present in studied projects

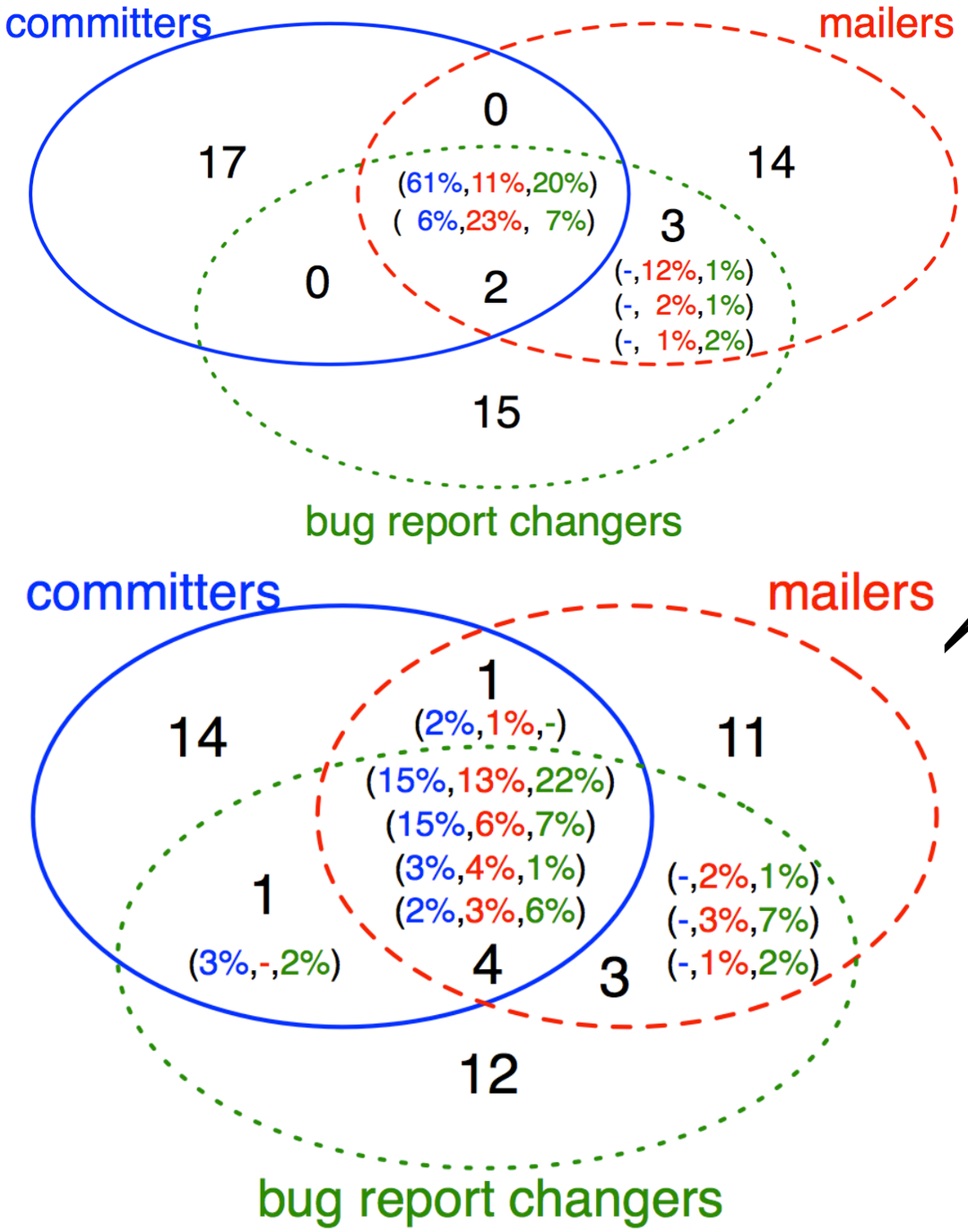
Activity distribution
Future work

- Identify the type of statistical distribution
- Use sliding windows for studying activity distribution over time
  - useful to detect impact of personnel turnover
  - ignore persons that have become inactive, and discover new active persons.

Identifying core groups

- Display Venn diagrams of most active (top 20) persons, according to each definition of activity (committing, mailing, bug report changing)
- For each person, show the percentage of activity attributable to this person
- Take into account identity merges

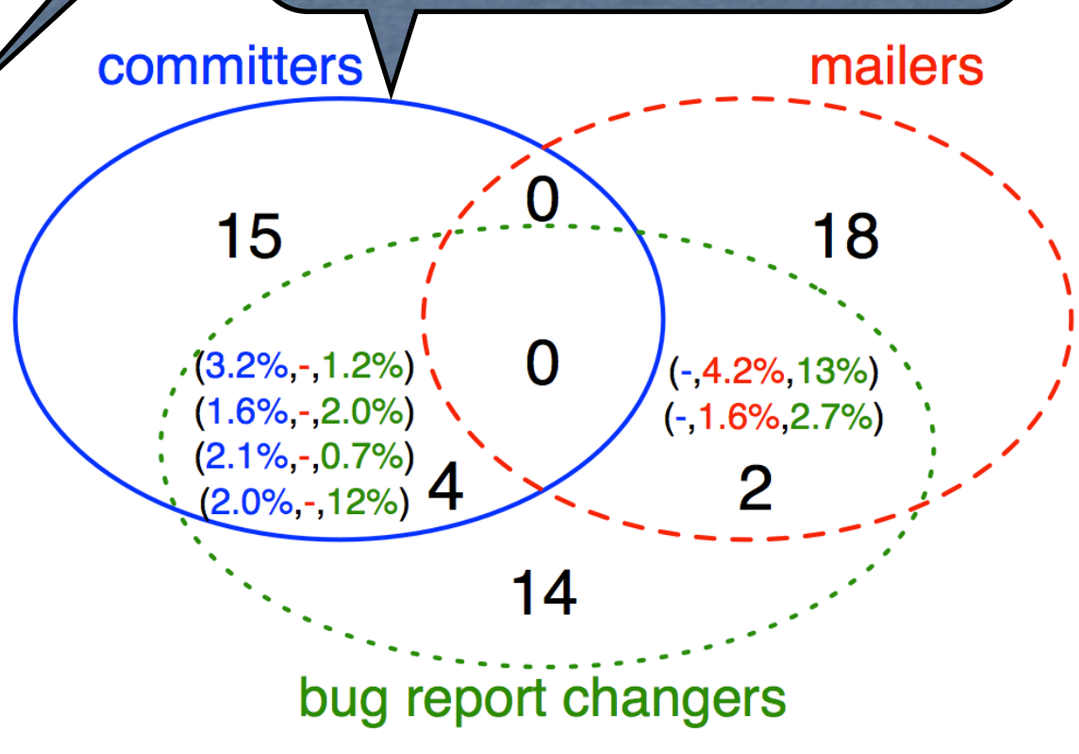
Identifying core groups



Brasero

Evince

Wine



Identifying core groups Conclusion

- For Brasero and Evince, the activity is led by a limited number of persons involved in 2 or 3 of the defined activities.
- For Wine, it seems not to be the case.

Identifying core groups
Future work

- Automate this process for the entire project community
- Study the evolution of core groups over time
  - Does a core group remain stable or does it change often? Why?
- Can we find evidence for a "bus factor"?

More future work

- Study correlation between community structure (social network) and source code quality (as computed using software metrics).
- Extend and refine types of activity:
  - different types of commit activity (doc, source code, test, etc.); of mail activity (information, asking, answering, etc.); of bug repository activity (bug creation, modification and commenting)

Thank you