UMONS
Université de Mons

On the analysis of OSS ecosystem evolution

Mathieu Goeminne & Tom Mens
UMONS

Faculté
des Sciences

Subject

- Goal : study the evolution of Open Source Software (OSS) and its quality
- Analysing only the source code is not sufficient :
  - Not all development activity is reflected in the source code evolution.
  - No information about the user and developer implication.

Why OSS?

- Open access to the source code
- Observable communities
- Observable activities
- Increasing popularity for personal and professional use
- A huge range of communities and software sizes (from 1 person /100 LOC to 1.000 persons / 6.000.000 LOC projects)

So what?

- Need to take into account user and developer activities.
- Merge data from
  - Revision control system
  - Bug tracker
  - Mailing list
  - ...

Ecosystem

What can other data sources provide?

- **Bug tracker** : Provides information about
  - how software issues are handled by the developers community.
  - user interests and software quality.
  - Number of submitted issues, time needed to react to change requests, who resolves each kind of issue.

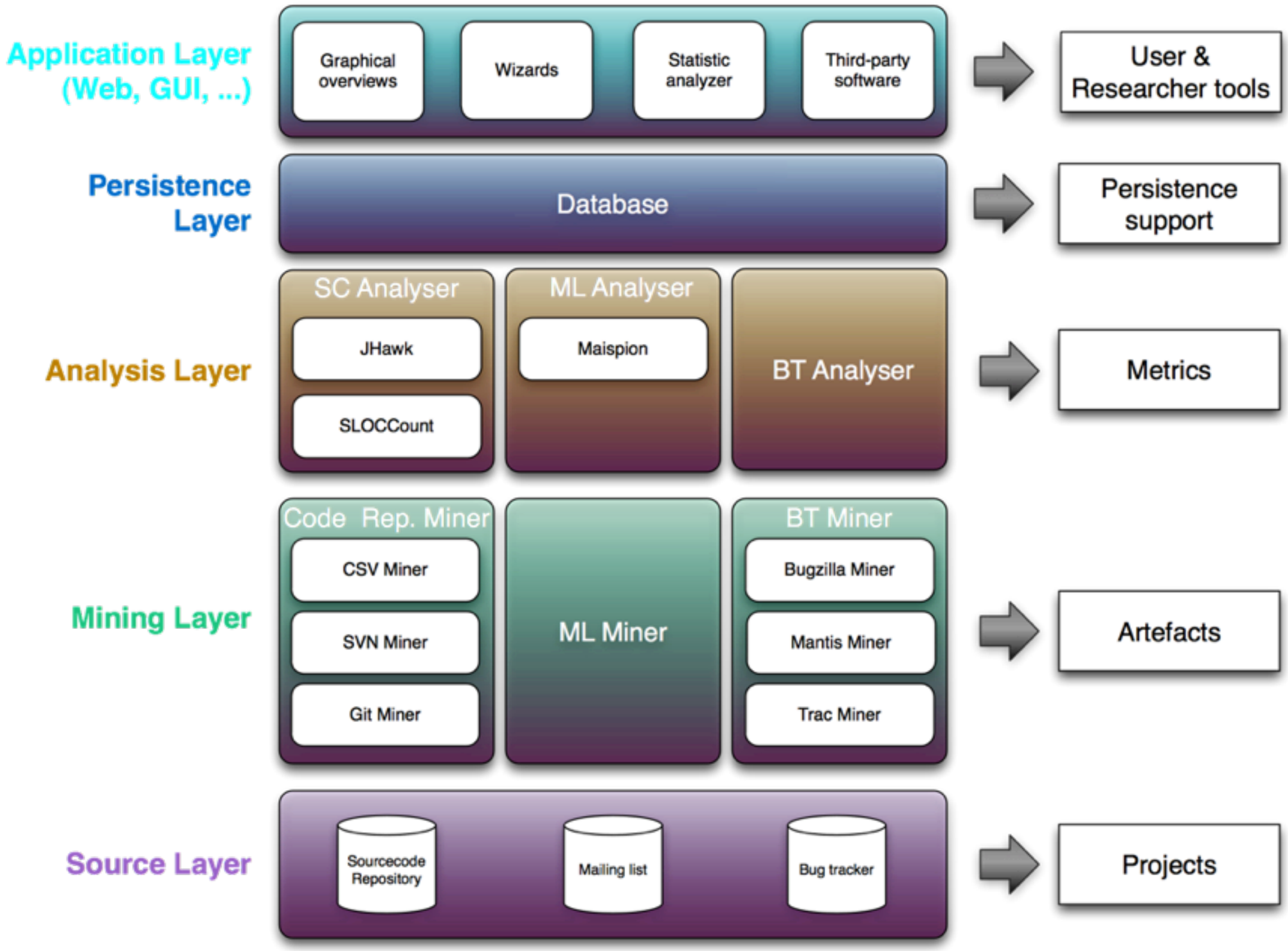
What can other data sources provide?

- **Mailing list** :
  - The developer activity (often the main communication channel among devs and between users and devs)
  - Interactions among persons
  - Number of subject threads, number of messages in a thread, number of users and devs involved in a thread, time needed to answer a message, mails answered for each dev, etc.

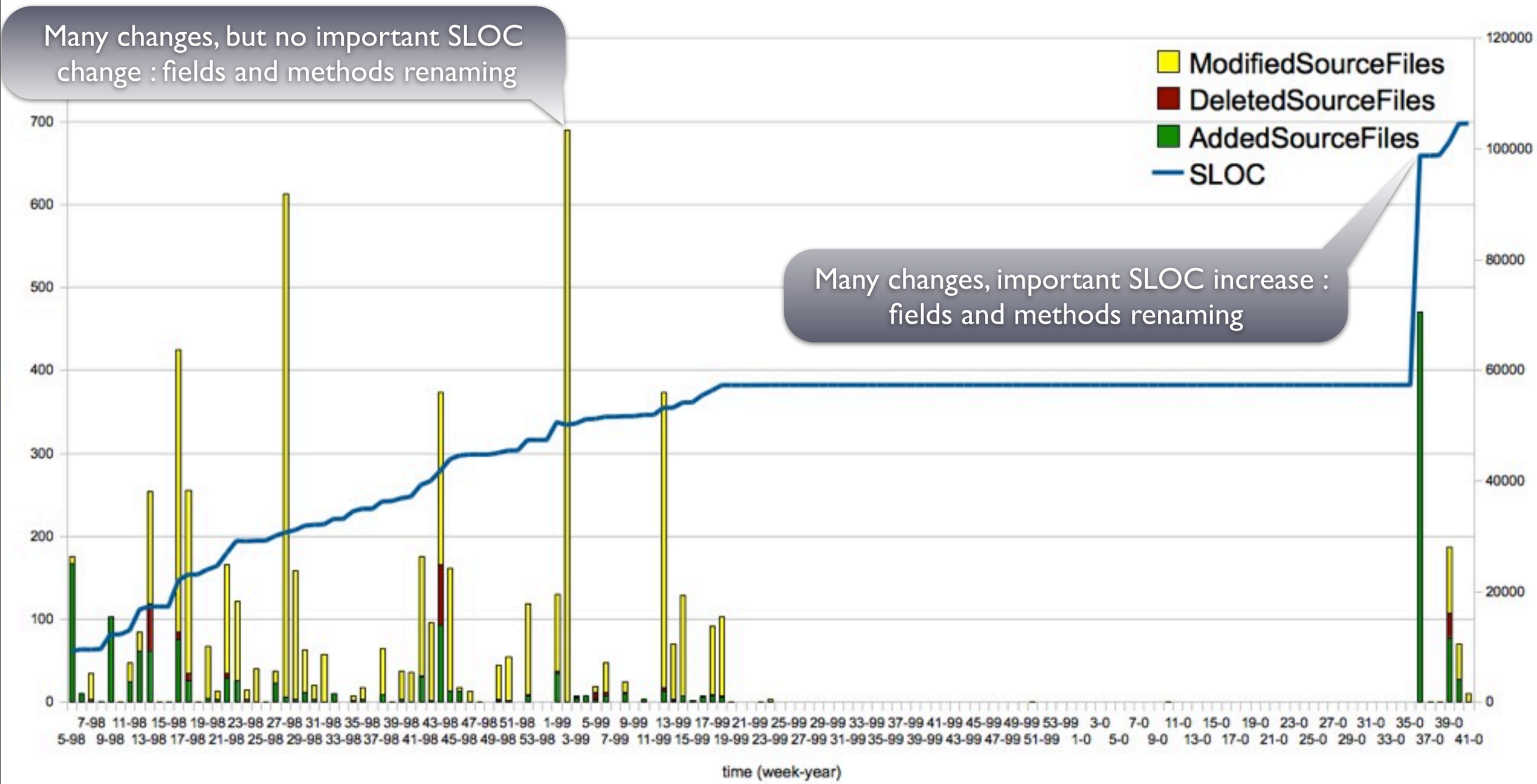
Work in progress

- Generic framework to extract metrics about the OSS ecosystem
- Use of FLOSSMetrics data repository

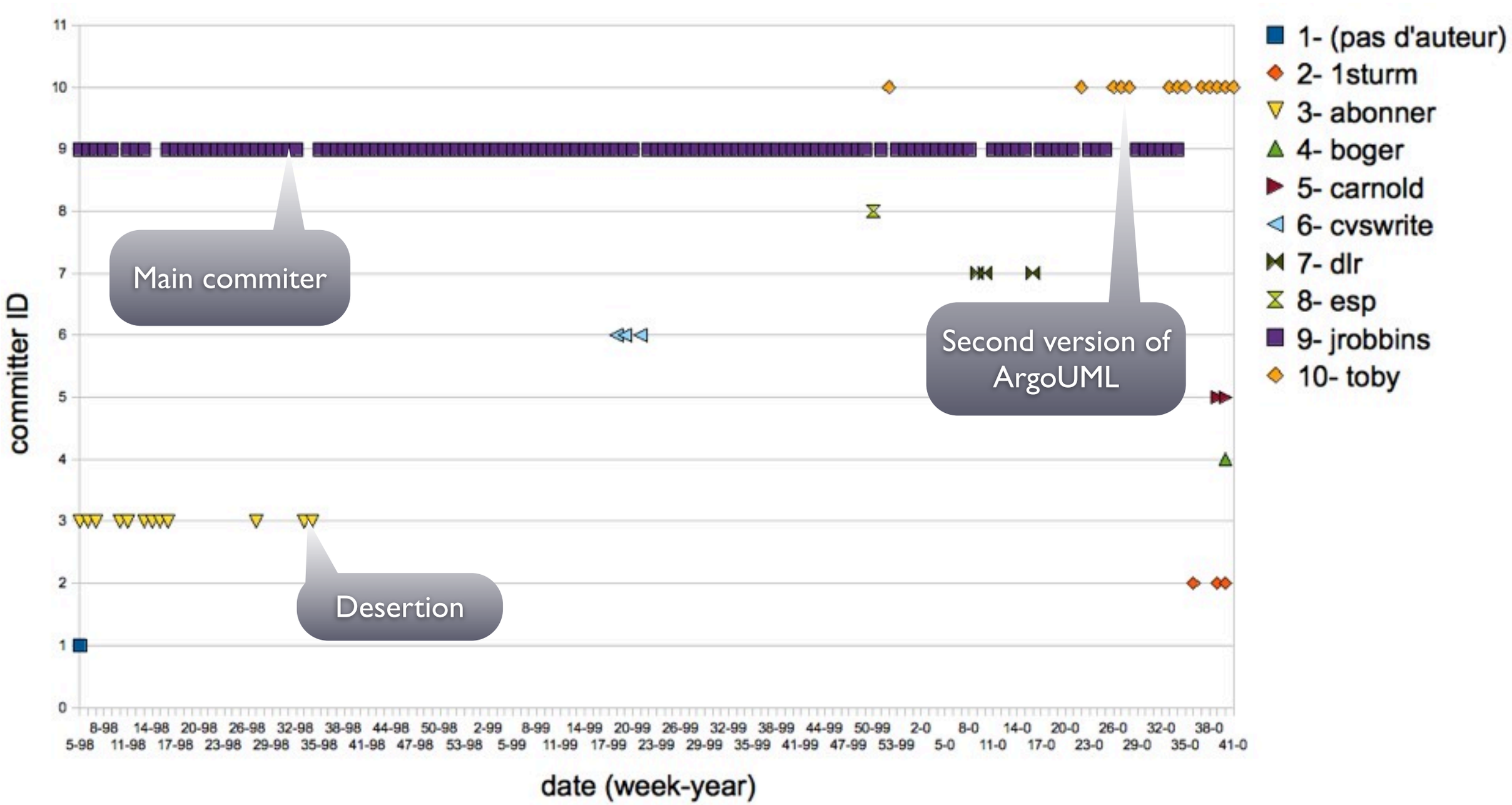
Framework


Application Layer
(Web, GUI, ...)
Graphical overviews
Wizards
Statistic analyzer
Third-party software
User & Researcher tools
Persistence Layer
Database
Persistence support
Analysis Layer
SC Analyser
JHawk
SLOCCount
ML Analyser
Maispion
BT Analyser
Metrics
Mining Layer
Code Rep. Miner
CSV Miner
SVN Miner
Git Miner
ML Miner
BT Miner
Bugzilla Miner
Mantis Miner
Trac Miner
Artefacts
Source Layer
Sourcecode Repository
Mailing list
Bug tracker
Projects

Framework usage : source files of ArgoUML



Framework usage : committers activity of ArgoUML



FLOSSMetrics usage

- Available tools
  - CVSAnaly2 for source code repository.
  - Bicho for the bug tracker.
  - MailingList for... the mailing list.
  - Other tools can be added.
- All data generated by these tools is put in a database.

Comparison

	Pros	Cons
My framework	Many quality metrics, tailored behavior	Immature, needs a lot of effort to become really interesting
FLOSSMetrics	Mature, many projects are treated	Harder to add/ extend features

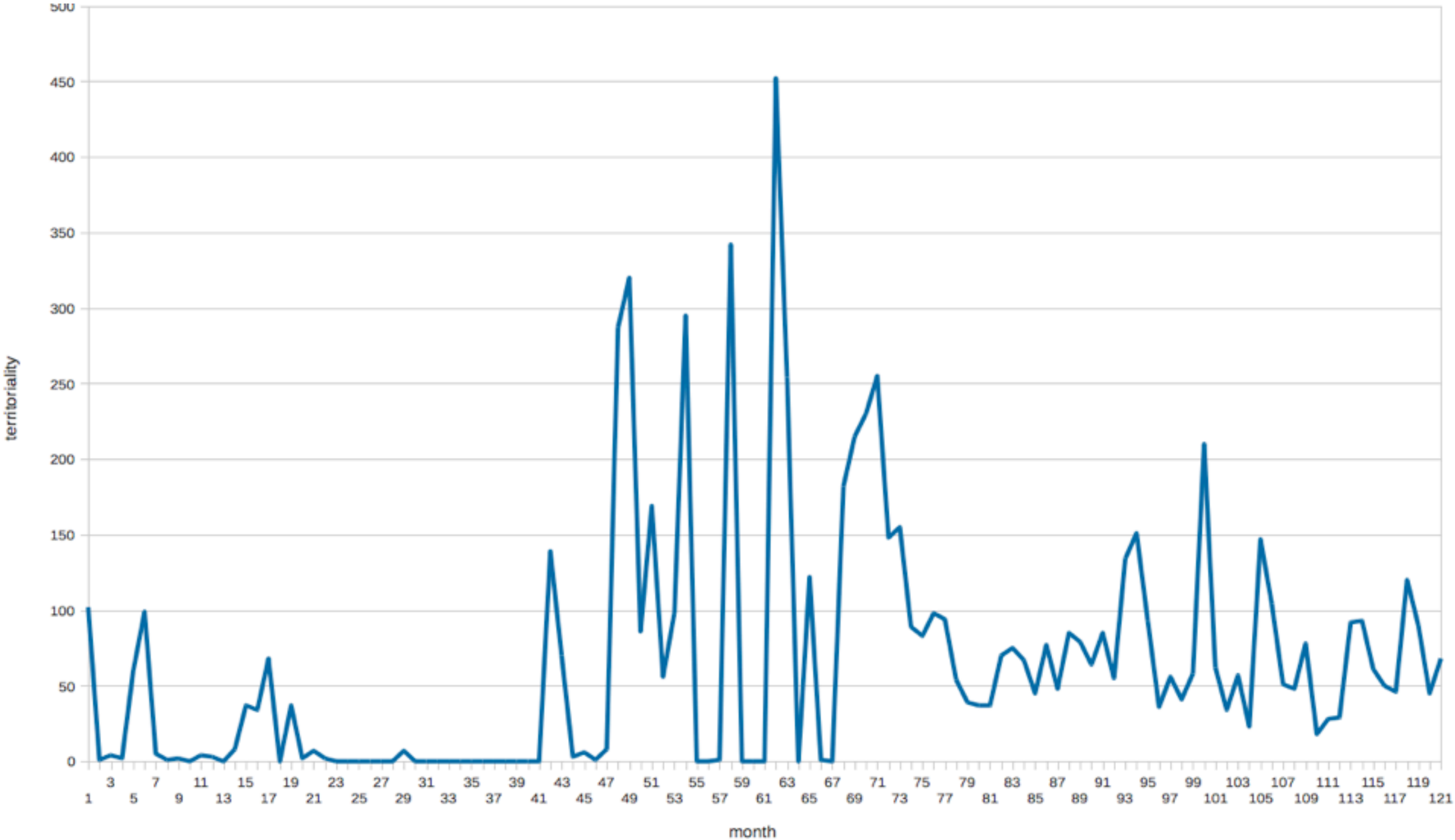
FLOSSMetrics usage : measuring territoriality

- Territoriality covers different notions :
  - The number of files touched by one and only one person during a given period ;
  - The number of files touched by only one person (its owner). If another person touches the file, it becomes a non-territorial file.
  - And other variations ...

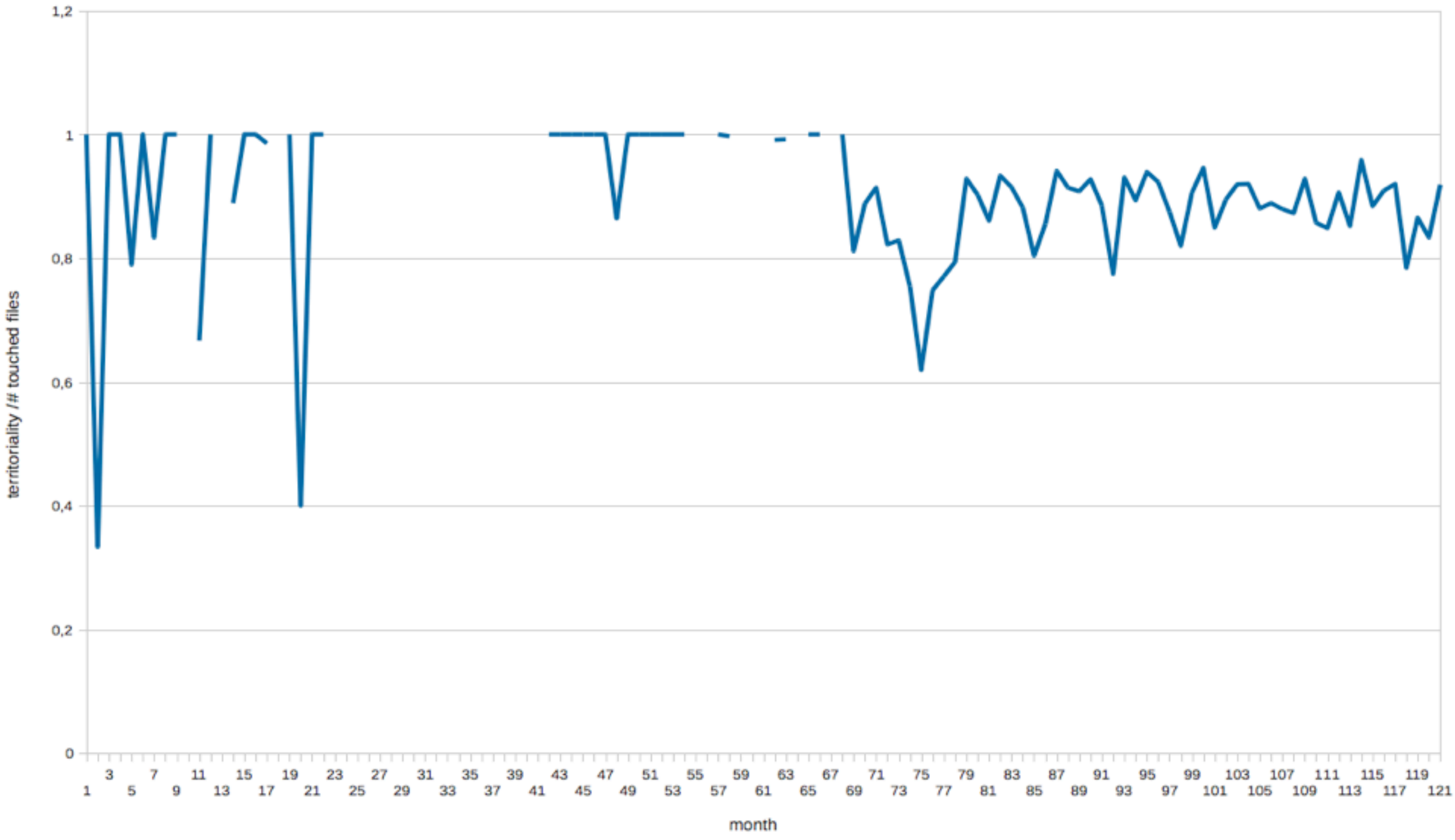
Territoriality

- A high territoriality might be a sign of quality (this is a hypothesis to confirm...)
- Territoriality I gives a *local* (e.g. one-month) view of the source code distribution.
- Territoriality II gives a *global* (from begin to end) view of the source code distribution.

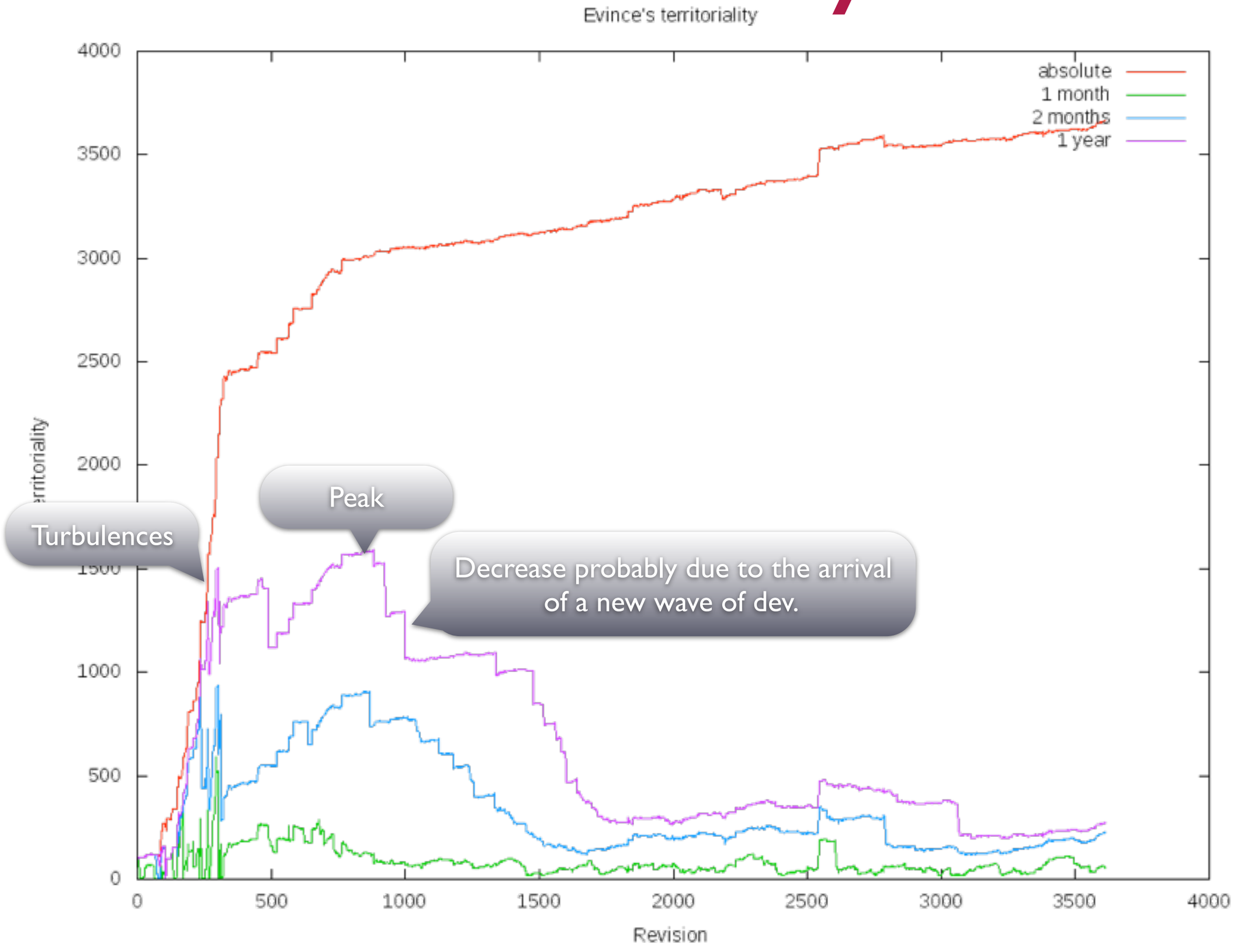
Territoriality 1 (Evince)



Territoriality 1 (relative)



Territoriality 2



Future work

- Integration of mailing lists and bugtracker to refine results and to have a better explanation about the project's behavior.
- Maybe : detect and study legacy software (quality and communities point of view).
- Case study for some popular OSS
- Look for a correlation between the popularity and the quality of an OSS.

Future work

- Territoriality at class / method / package level.
- Other useful «ecosystem» metrics
- Better integration of mailing lists and bug reports.
- Track other relations between software quality and ecosystems.

Questions?

Ideas?