Measuring and mining evolution of software projects

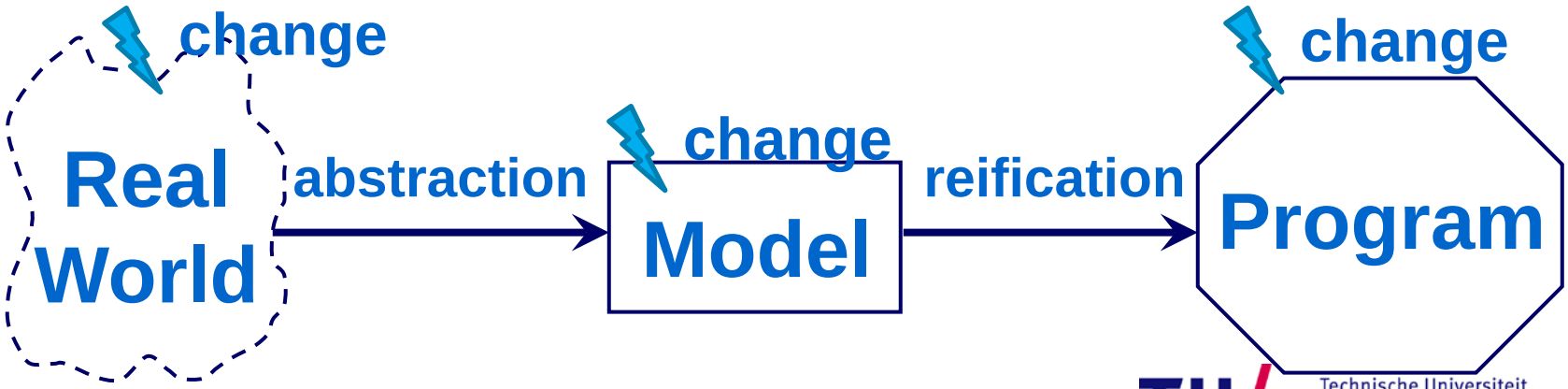
Alexander Serebrenik

TU/e Technische Universiteit Eindhoven University of Technology

Where innovation starts

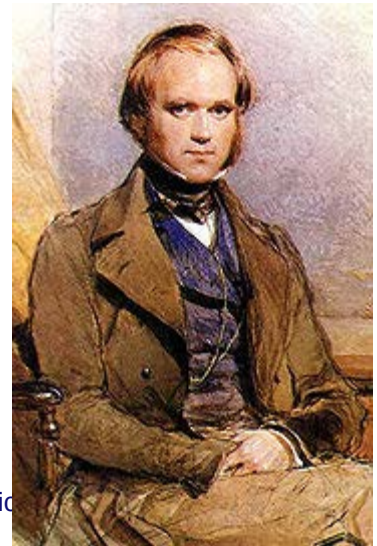
Maintenance: 75-95% costs

- Software is
  - crucial for modern society
  - more complex than any other human artifact
  - subject to change
    - GNOME, 10 years, 1000 developers, 2.5 millions changes
    - Mozilla, 6 years, > 100 developers, > 1 million changes



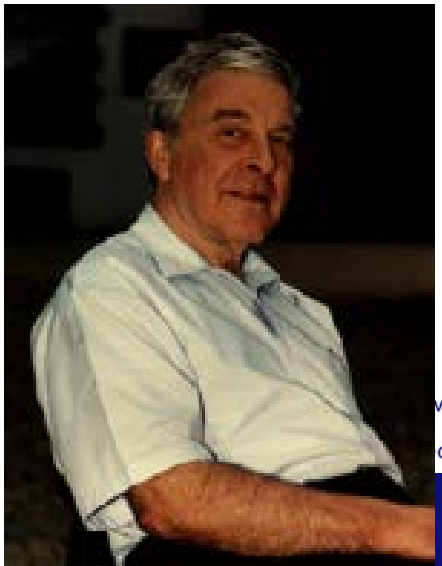
Evolution: one change a top of another

Evolution is staged process of progressive change over time in the properties, attributes, characteristics, behaviour of some material or abstract, natural or artificial, entity or system



Charles Darwin

Meir Manny Lehman



Why do we want to study software evolution?

- **Software = the weakest link (often)**
- **Evolution "in general" makes things more complex**
  - **Science:**
    - What is the nature of software evolution?
      - Psychology, sociology and organization theory, economics, law…
  - **Engineering:**
    - Where did the things go wrong?
      - incorrect, too complex, out of sync with other artefacts
    - Where can/will the things go wrong?
      - prediction, week spot identification, …
    - What can we do to prevent the things from going wrong?

Where did the things go wrong?

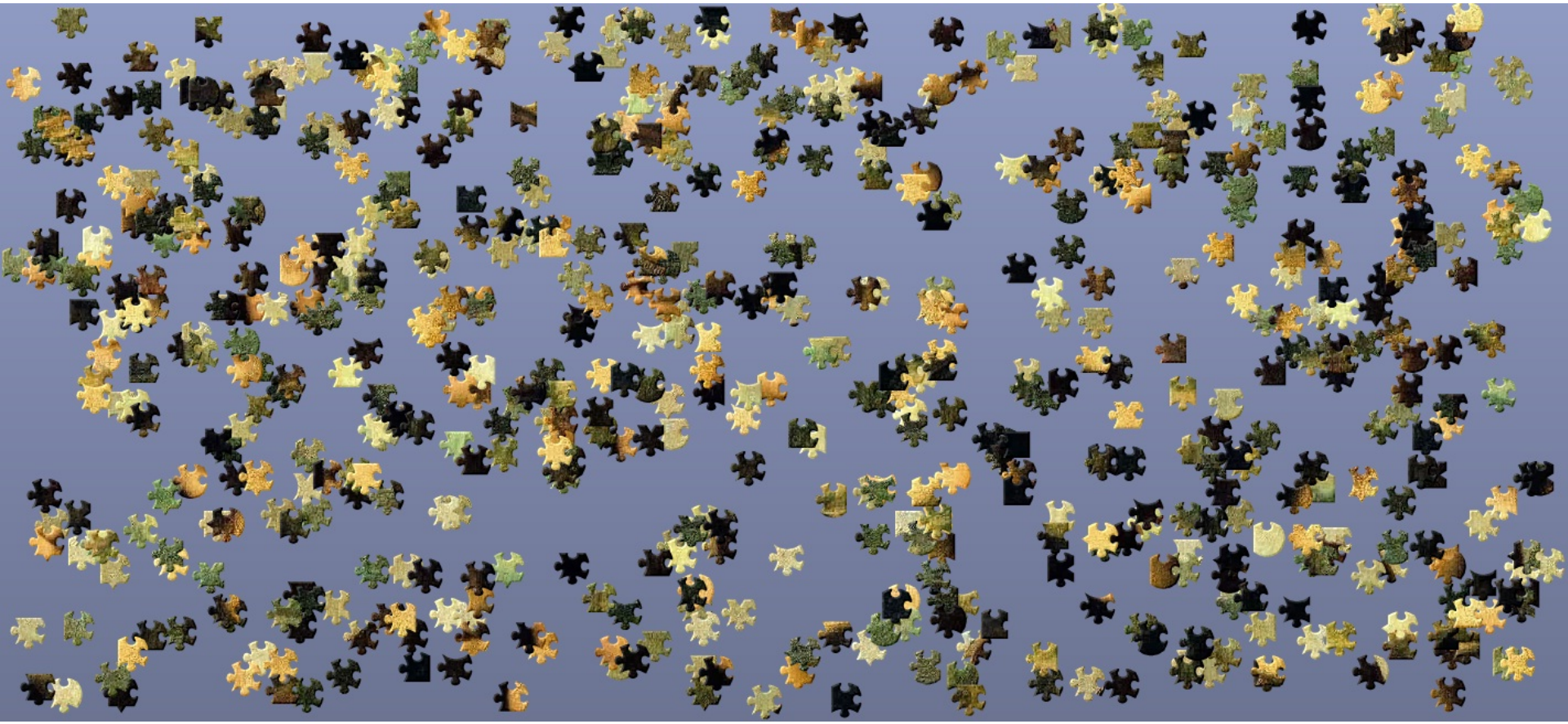
- **What was wrong?**
  - incorrect, too complex, out of sync with other artefacts
  - Metrics:
    - "You can't control what you can't measure"(*DeMarco*)
- **When did it happen?**
  - last month, before the release, when fixing a bug
  - Repository mining

Measuring With Metrics

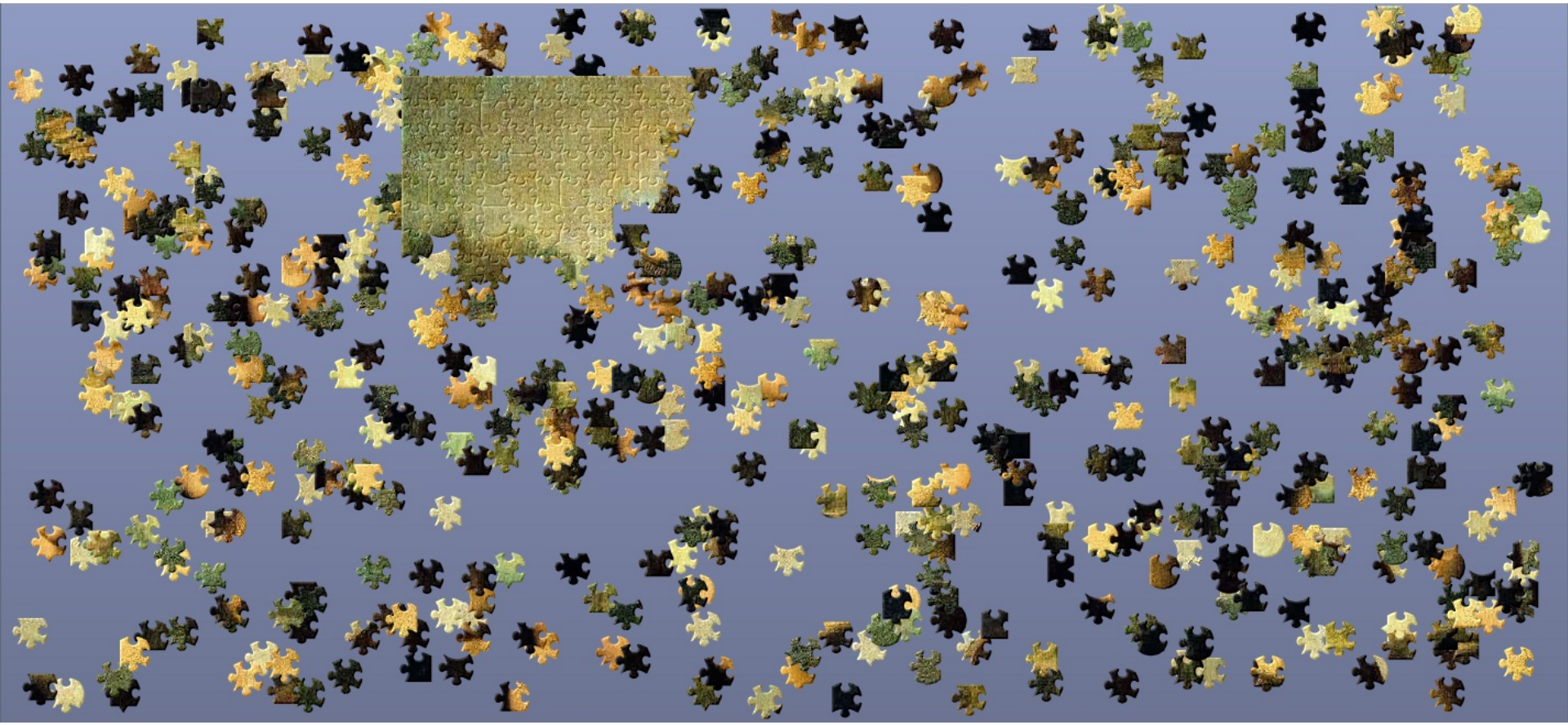
Software metrics: Examples

- Size
- Complexity
- Inheritance
- Comments
- Churn
- Coupling/cohesion
- Abstractness
- …

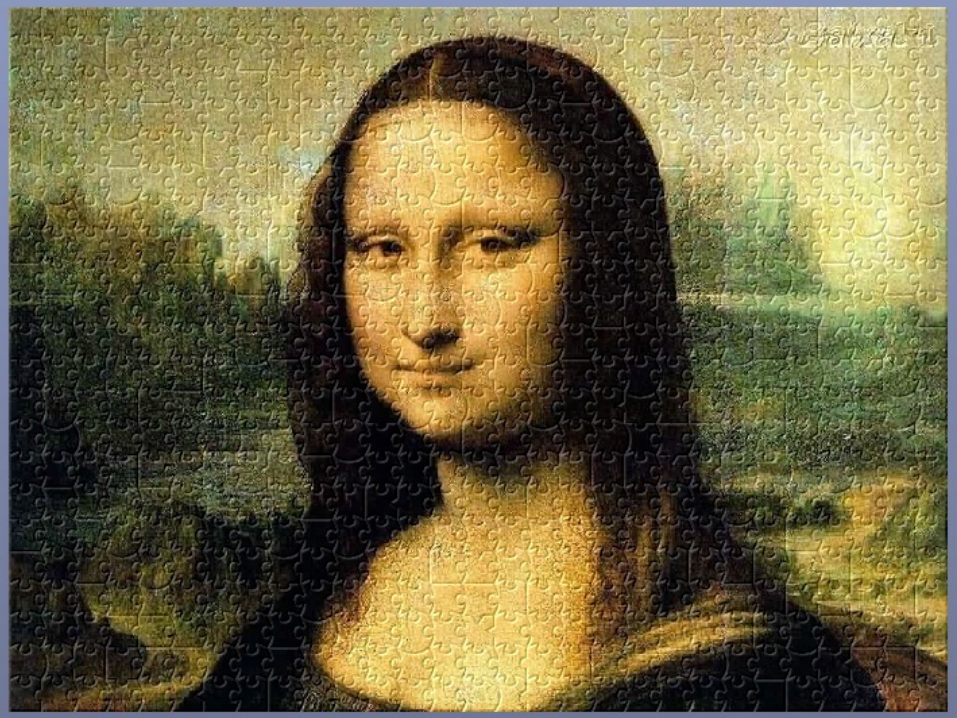
Metrics are usually computed at a *low* level: classes, methods, …



Multitude of data values obscures a general picture of the *system* maintainability



That we are actually interested in!



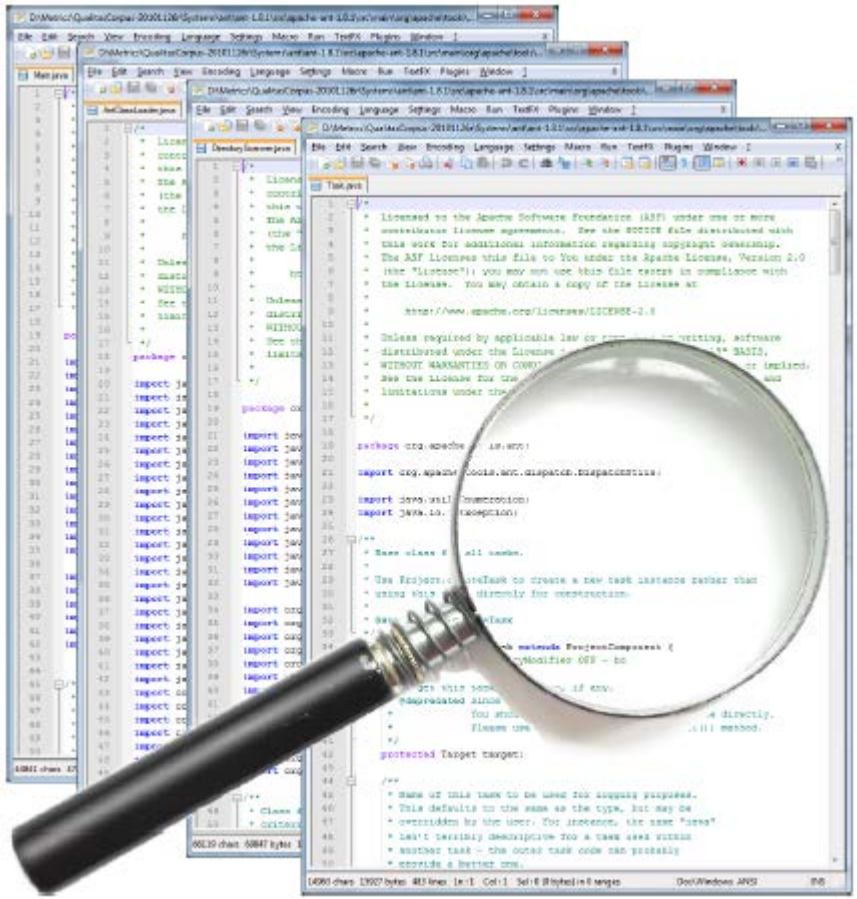
We need

aggregation

techniques to get the whole picture

Two kinds of aggregation

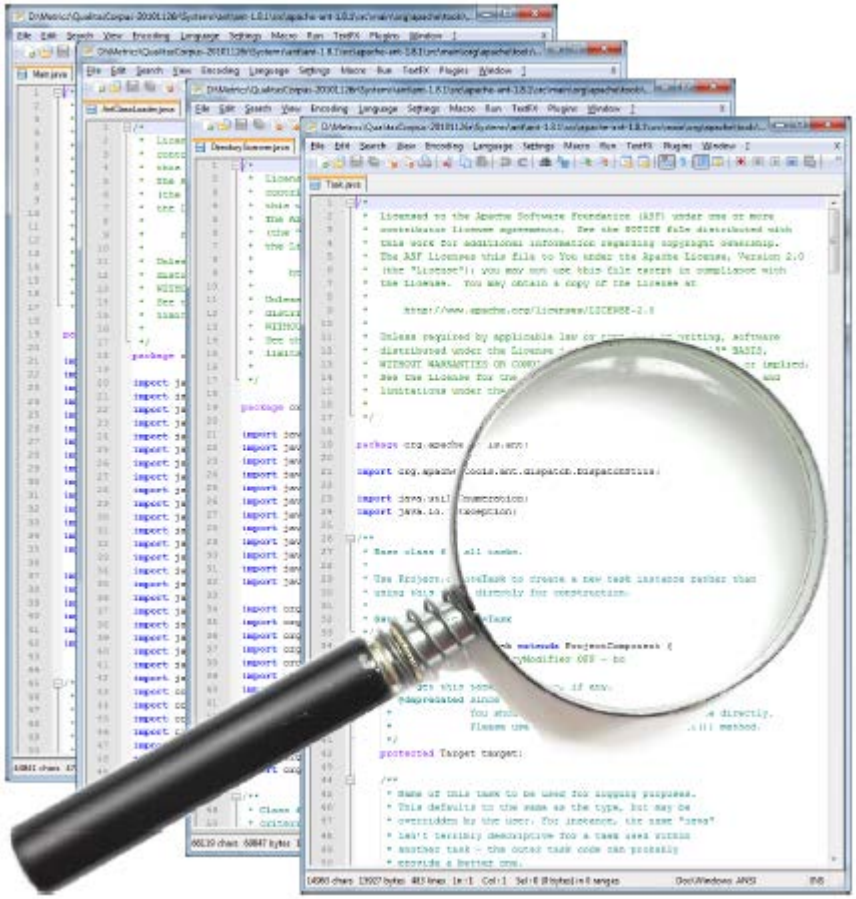
Same metrics, different artifacts



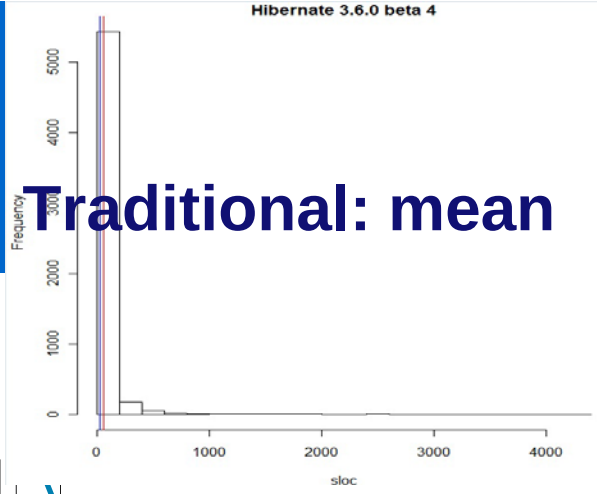
Same artifact, different metrics

Various techniques can be found in the literature

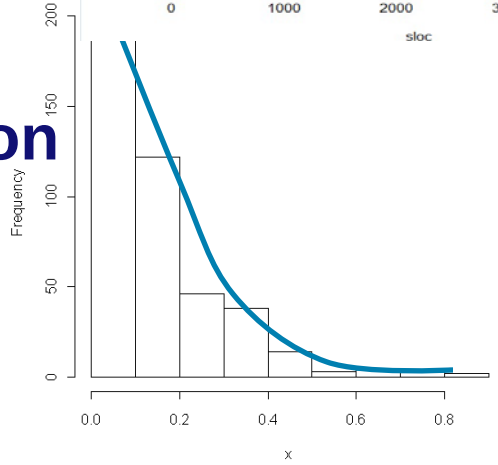
Same metrics, different artifacts



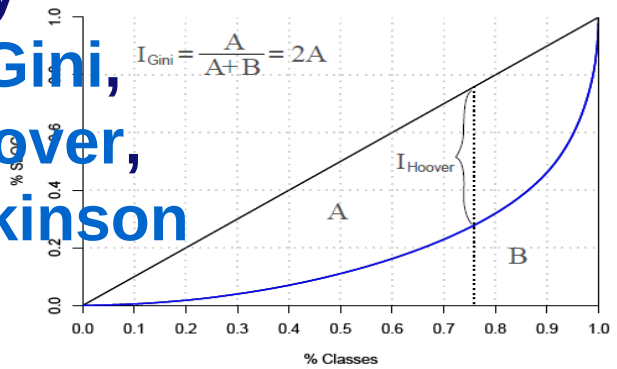
Traditional: mean



Distribution fitting



Econometric inequality indices: Gini, Theil, Hoover, Kolm, Atkinson



Mean

- **3 OS systems: ArgoUML, Adempiere, Mogwai**
- **Correlation SLOC/bugs**
  - Kendall's τ: ranked, does not assume normality
  - SLOC/class
  - Bugs: #

	Mogwai
Ke	0.197
p	< 0.01

By No Means

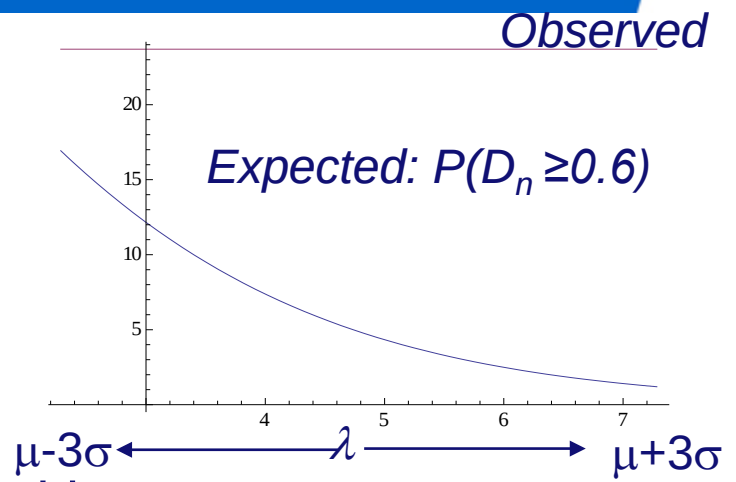
- Inconsistent results!
- Expresses central tendency, unreliable for skewed distributions

[Vasilescu, Serebrenik, v.d. Brand WETSoM'11 ACM]

Distribution fitting

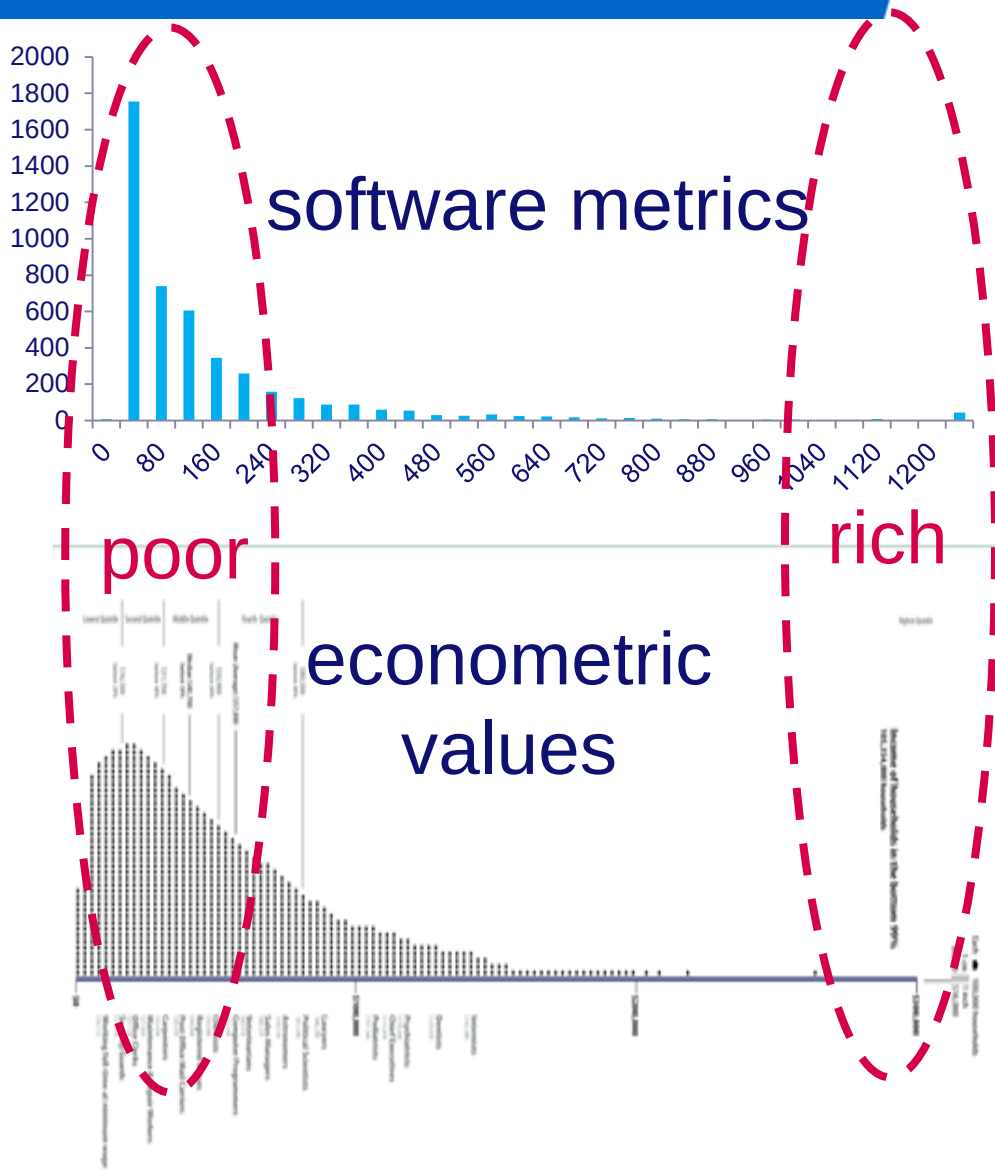
- **Useful:** estimation of excess values
- **Case study**
  - 21 Java OSS
  - D_n: abstractness/instability balance
- **Distribution**
  - Similar to exponential, parameterized with λ
  - λ of diff benchmarks are normally distributed

  [Serebrenik, Roubtsov, v.d. Brand ICPC'09 IEEE]
- **Fitting: involves guessing the distribution family**
  - Controversial for SLOC: log-normal or double Pareto?
  - Even more error-prone for more complex metrics
  - Avoid, if possible…

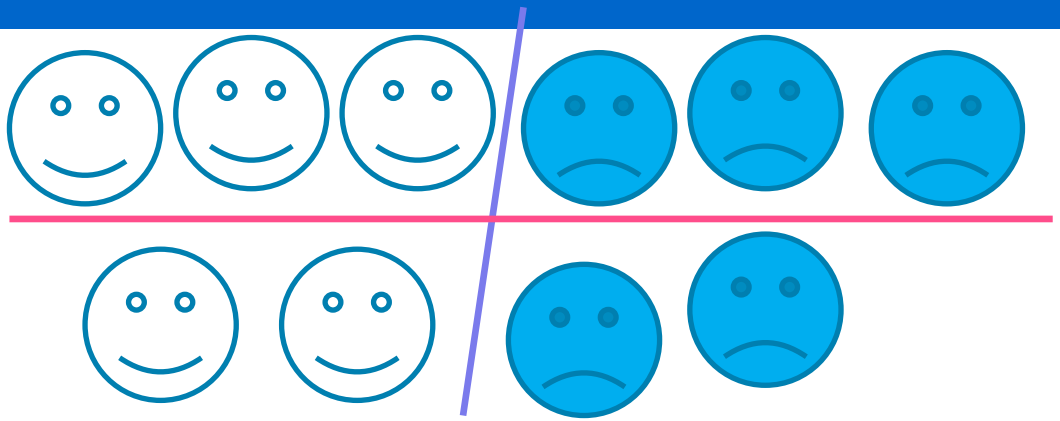


Econometric inequality indices

- **Measures of inequality**
  - wealth, expenditure, income
  - SLOC, #Classes, function points
- **Calculation is metrics-independent**
- **Some support explanation of inequality:**
  - region, education, gender
  - prog. lang., domain, maintainer



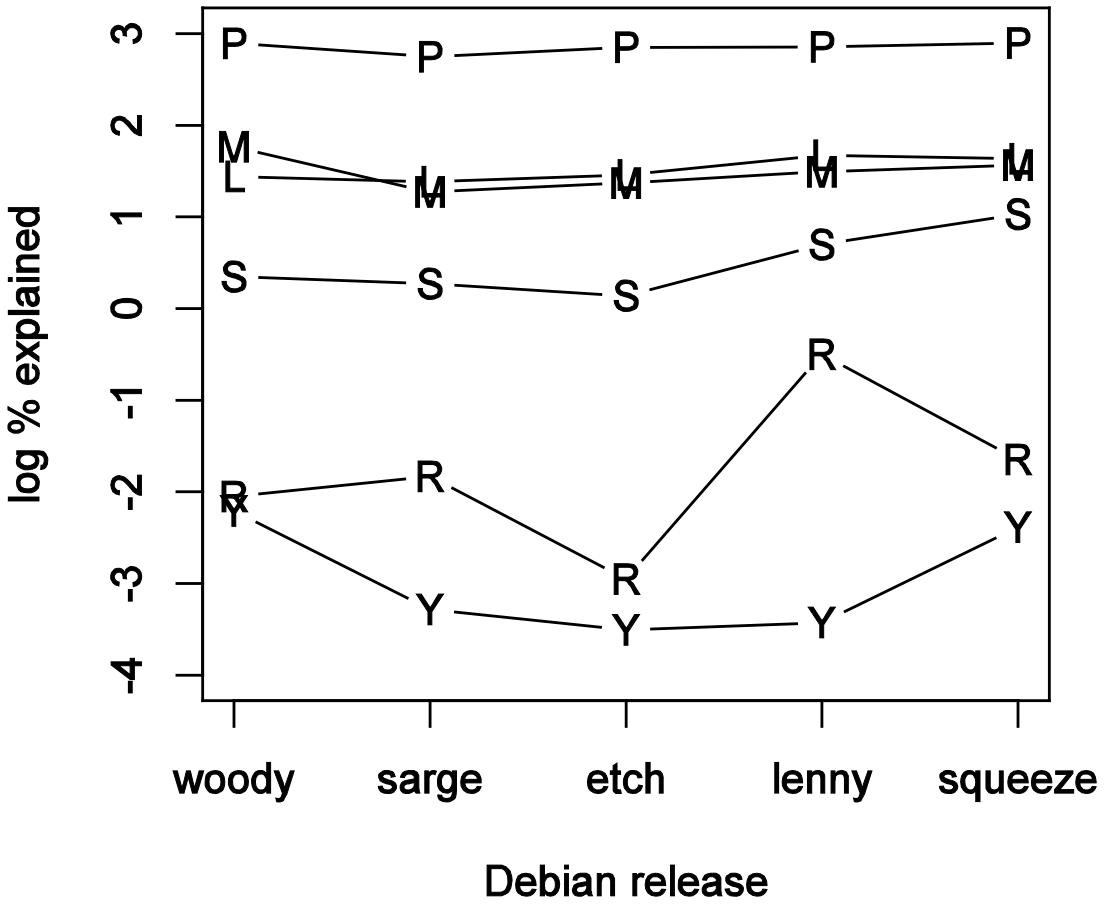
Explanation of inequality: Intuition



- Why are some people wealthier than others?
- Why are some files larger/more complex than others?
- Partition individuals in groups
  - Partition = explanation
  - Inequality within the groups and between the groups
  - Better explanation: more inequality between the groups
  - Lila is better than red

Explanations: which one is better?

- **Theil index**
- **Debian Linux**
  - 1469062 files
  - Metrics: SLOC per file
- **Partitions (lenny)**
  - Package (17.5%)
  - Maintainer (5.3%)
  - Language (4.5%)
  - Section (2%)
  - Repository (0.6%)
  - PriorityY (0.03%)



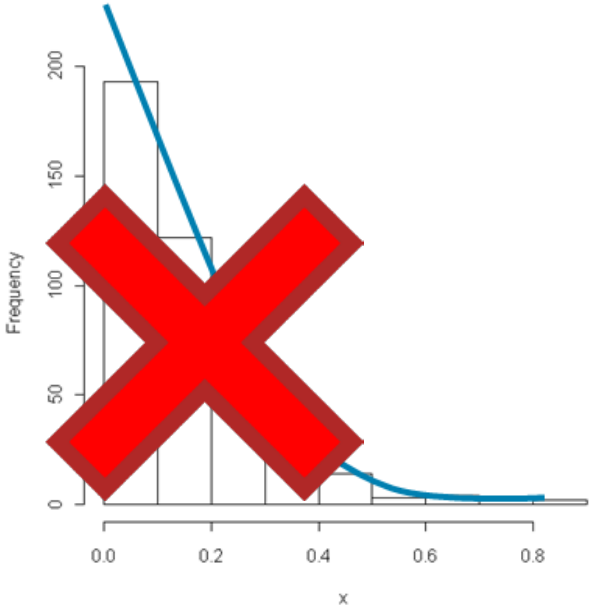
[Serebrenik, v.d. Brand ICSM'10 IEEE]

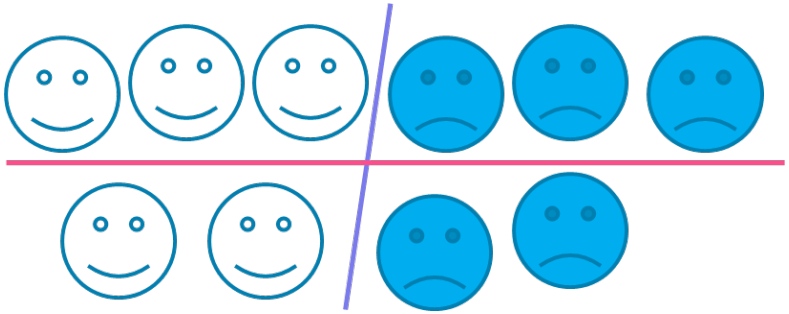
Inequality indices: Which one is better?

- **Theil, Gini, Hoover, Atkinson, Kolm, …**
  - Theil, Gini, Hoover and Atkinson agree
    - Any can be chosen from the correlation point of view
    - Advantages: decomposable (Theil), fixed range (Gini, Hoover), applicable to negative values (Gini, Hoover)
  - Kolm and mean agree
    - for SLOC, not for more advanced metrics
    - Kolm is a better alternative

[Vasilescu, Serebrenik, v.d. Brand ICSM'11 IEEE]

Aggregation techniques: Summary





Econometric inequality indices

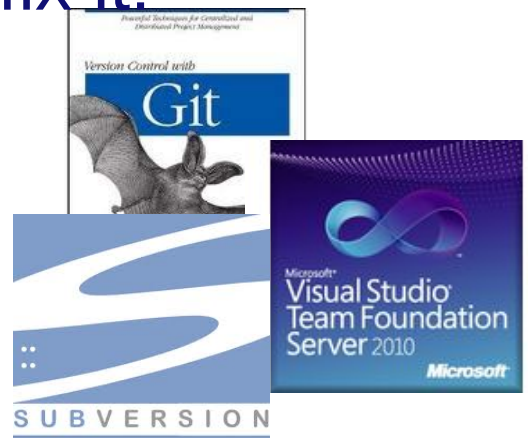
Mining Repositories

It is all about communicati

Test #14352
fails sometimes

The error should be somewhere here…
What does this code do?

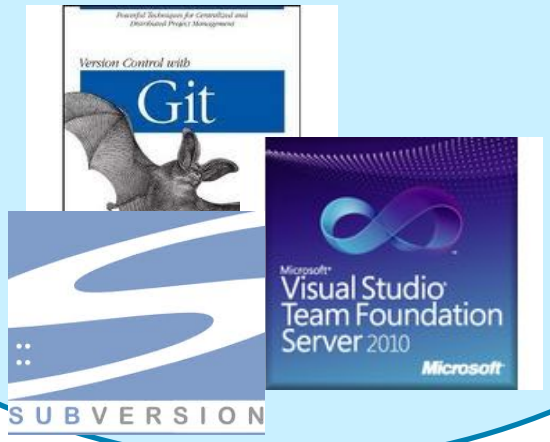
I know how to fix it!



Software repositories re mation

How can the repositories serve you?

Software repositories



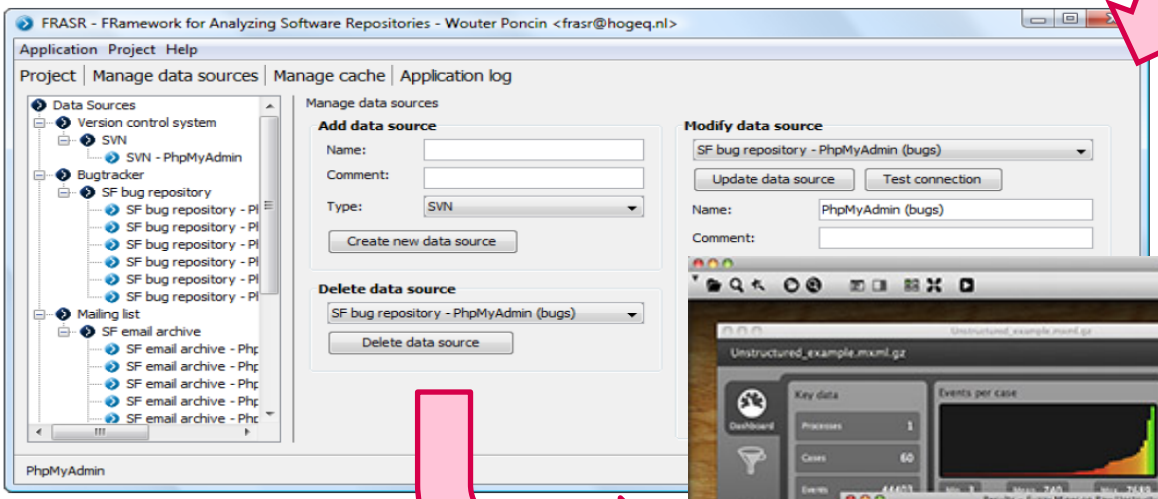
How can the repositories serve you?

- Is the documentation up-to-date?
- How fast are the bugs resolved?
- Who is responsible for
  - Bugs
  - Overtly complex code
  - Code guidelines violations?
- What parts are covered by tests?

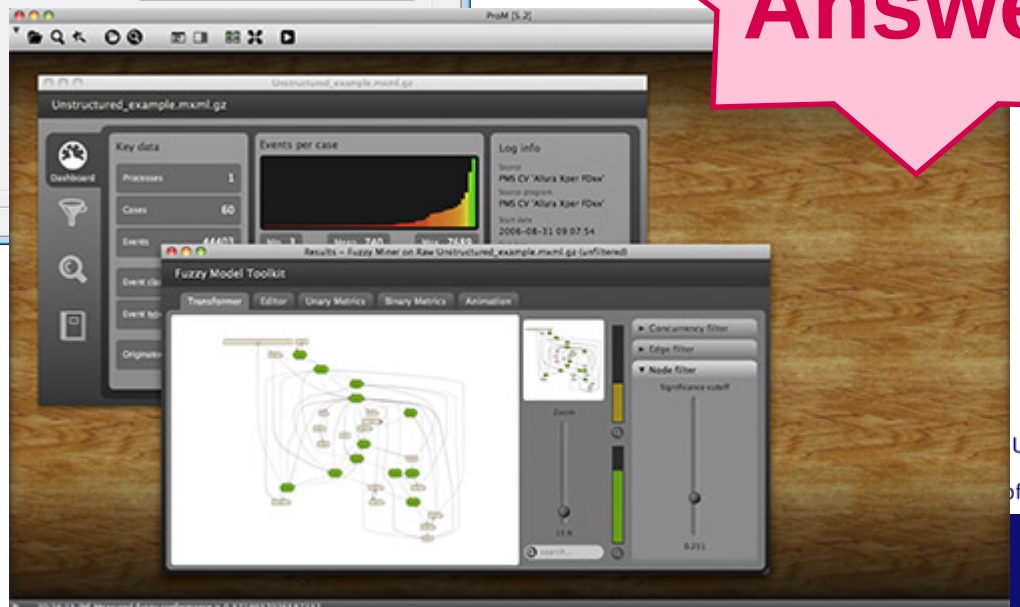
How does it work?

Sw Eng Quest.

Sw Eng Answer



Combined log



Existing work…

- **Existing repository mining works often limited to**
  - one repository or one software engineering question
  - we need a generic framework
- **Process mining**

Business processes	*Software processes*
One data source	Multiple data sources
"Natural case": • association of different events • claim id, person id, vehicle id	Many different options • files, developers, topics…
Explicit events	Implicit events • is the mail relevant to the bug report?
Unique data representation	Different representations in different data sources

Our studies so far

- **Open-Source software:**
  - developer roles
  - use of Bugzilla (intended vs. actual)

    [Poncin, Serebrenik, v.d. Brand CSMR 2011 IEEE]

- **Student capstone projects**
  - adherence to guidelines
  - quality of the development process
  - developer roles

    [Poncin, Serebrenik, v.d. Brand OOPSLA Comp. 2011 ACM]

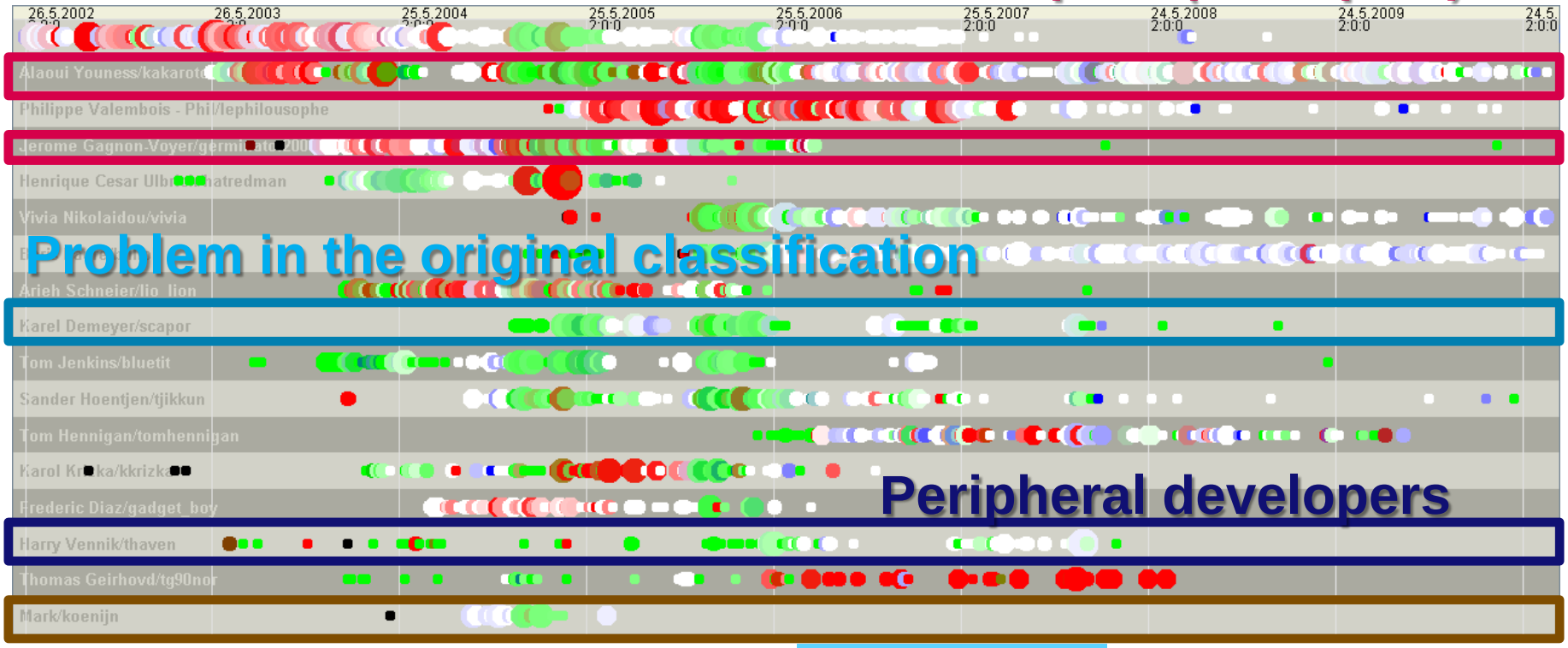
Case study 1: Developer roles

- **Question:** Classify developers according to their roles
- **Classification** of Nakakoji et al. IWPSE 2002: 8 roles
  - Core member involved for a relatively long period and made significant contributions to the development and evolution of the system
- **Case study**
  - **aMSN**: instant messaging application
    - 38 million downloads, 20th most popular at SourceForge
    - February 26, 2002 – July 9, 2010
    - 7 bug repositories: 3137 bug reports
    - 3 mail archives: 34947 messages
    - Subversion: 12062 commits

Case study 1: Results
ProM Dotted Chart visualization

Case study 1: Results

Core developers (examples)



Problem in the original classification

Peripheral developers

File added
Bug opened
Mail
Bug other
File modified
Bug reporter

Case study 1: Classification

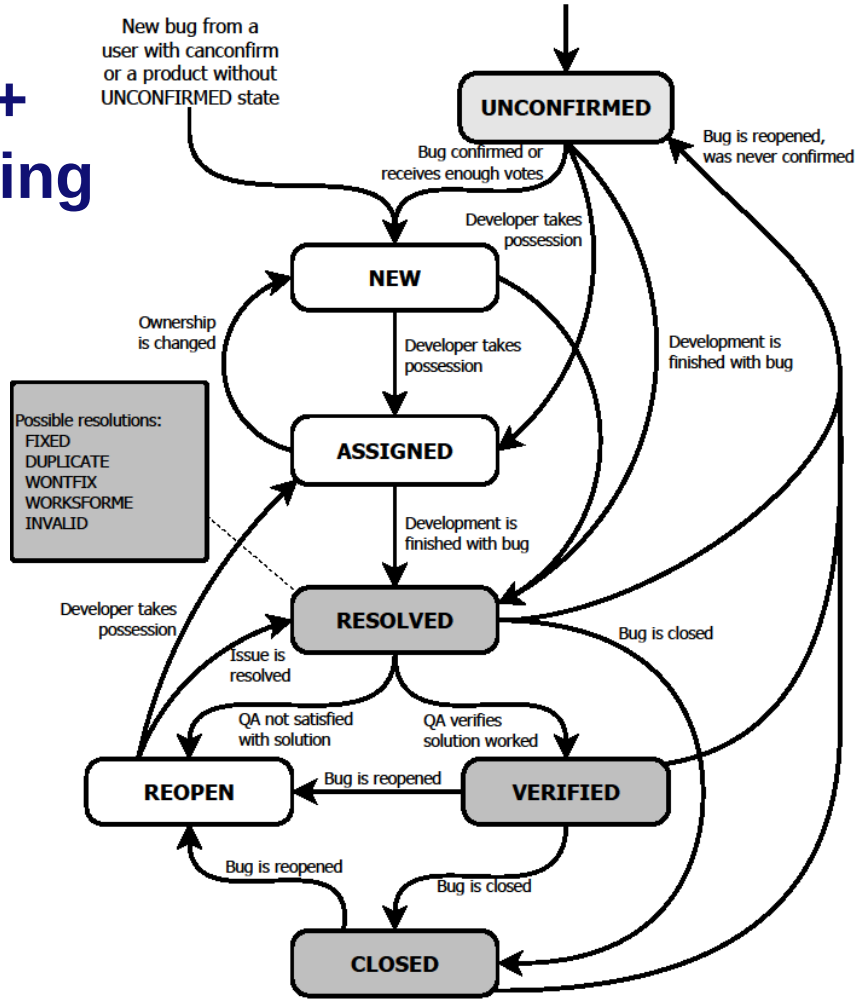
Role	***#developers***
Bug reporter	1443
Bug fixer	3
Peripheral developer	29
Active developer	6
Core member	7
Project leader	3
Other	234
Total	1725

Bugs are usually fixed by peripheral developers

Only ticket-commented or mail-reply

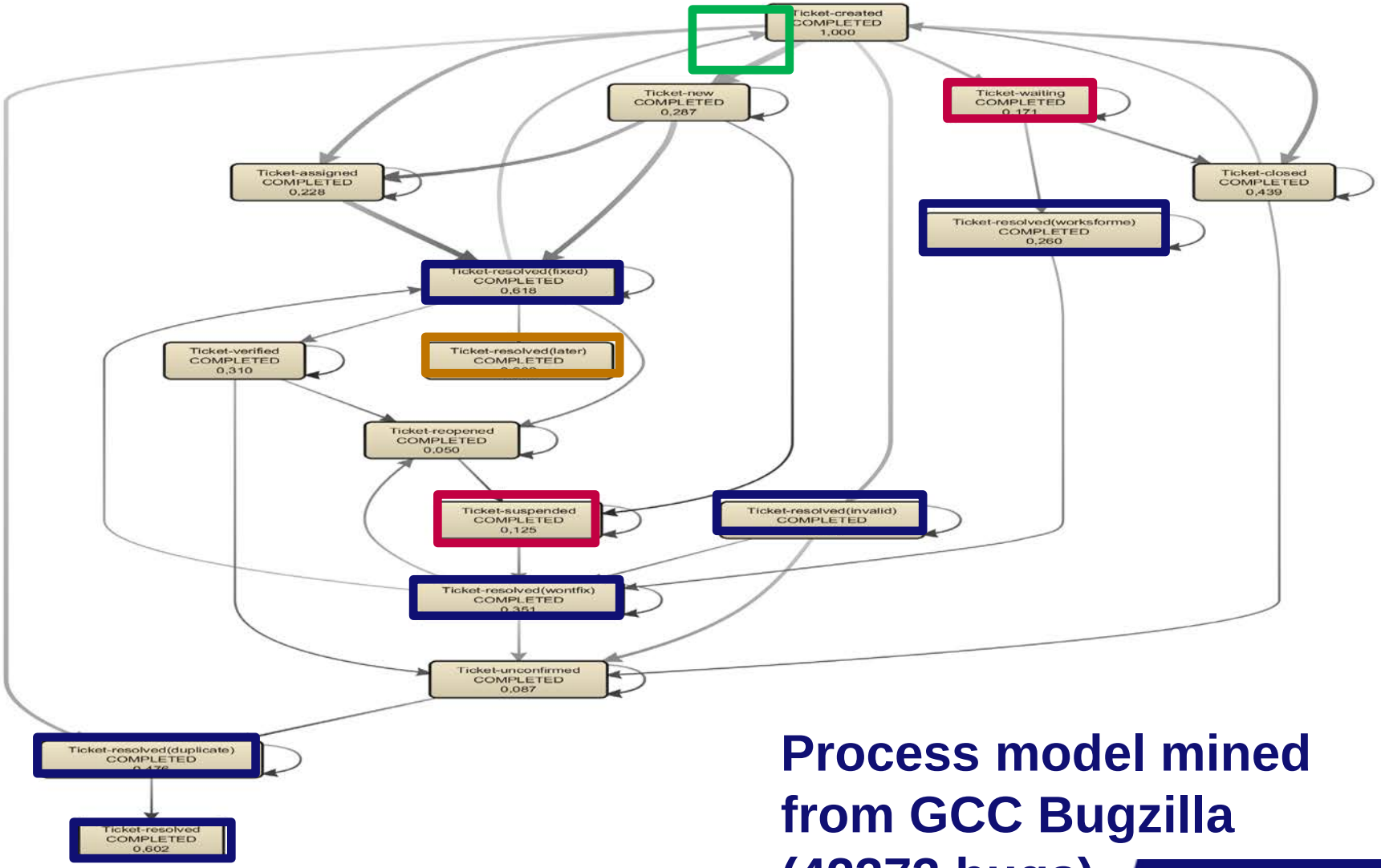
Case study 2: Bug life cycle in Bugzilla Theory according to the Bugzilla Guide

One source + process mining (mining)



S.E. question: Is Bugzilla used the way it is supposed to be?

Case study 2: Bug life cycle in Bugzilla Practice vs. Theory



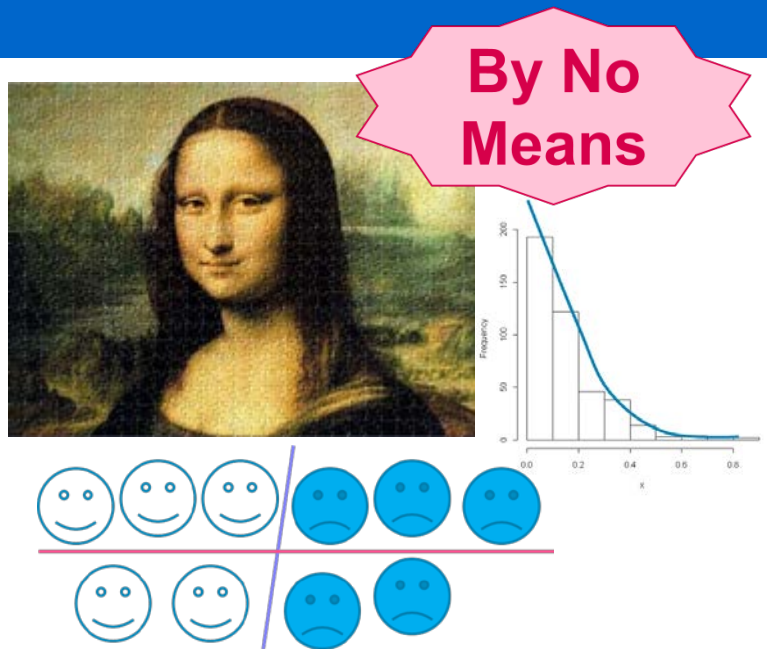
Process model mined from GCC Bugzilla (42373 bugs)

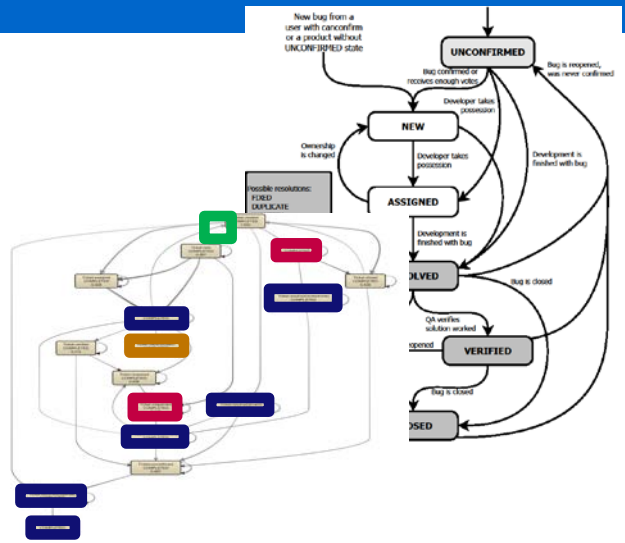
Mining…

- **Information is available in software repositories**
  - Just waiting to be mined

- **Numerous opportunities and chances**

Conclusions





Measuring + Mining

Interested? Join us!