Mining Software Repositories

Studying the Past to Survive the Future

Dr. Gregorio Robles

grex@gsyc.escet.urjc.es

Universidad Rey Juan Carlos

http://libresoft.urjc.es

TU München, June 6th 2007

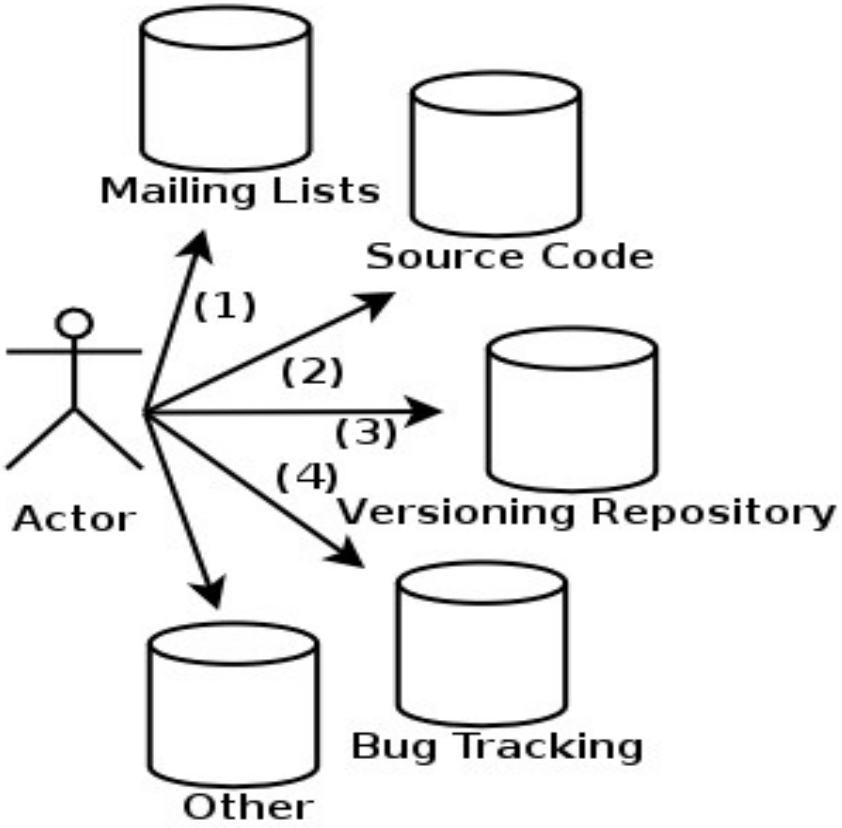
Premise of MSR

- Empirical and systematic investigation of repositories will shed new light on the process of software maintenance/evolution by uncovering
  - pertinent information,
  - relationships
  - or trends

Why so much MSR research?

- In the past, MSR was almost subjected on industrial systems:
  - few software systems (and domains)
  - hampered by lack of historical data
- Today, major possibilities due to public available data sources from Open Source repositories
  - 100,000+ projects in SourceForge.net
  - Some high-quality projects (Apache, Eclipse...)

Publicly Available Data Sources



Purpose of MSR studies

- Two (broad) types of purposes:
- “Market-based” question
  - if A occurs then what else occurs on a regular basis?
- Prevalence question
  - For instance, how many and which of the functions are reused?

Representation

- Type:
  - physical (file and line numbers)
  - logical (syntax and semantics)
- Granularity
  - System (even super-system)
  - Files
  - Classes
  - Lines

Methodologies

(a)Indirect (or external) measurement (or *changes to properties*)

- Each version is extracted, properties computed on each version separately and then individually compared
- software complexity, defect density,etc.

(b)Mechanisms or facts that take a software system from one version to the next (*changes to artifacts*)

- Specific differences among versions

Evaluation

- A wide spectrum enables to conduct empirical studies to evaluate approaches
- But with luxury comes the additional responsibility of selecting appropriate project repositories (!!)
- The use of mined repositories may change from project to project

Assessment

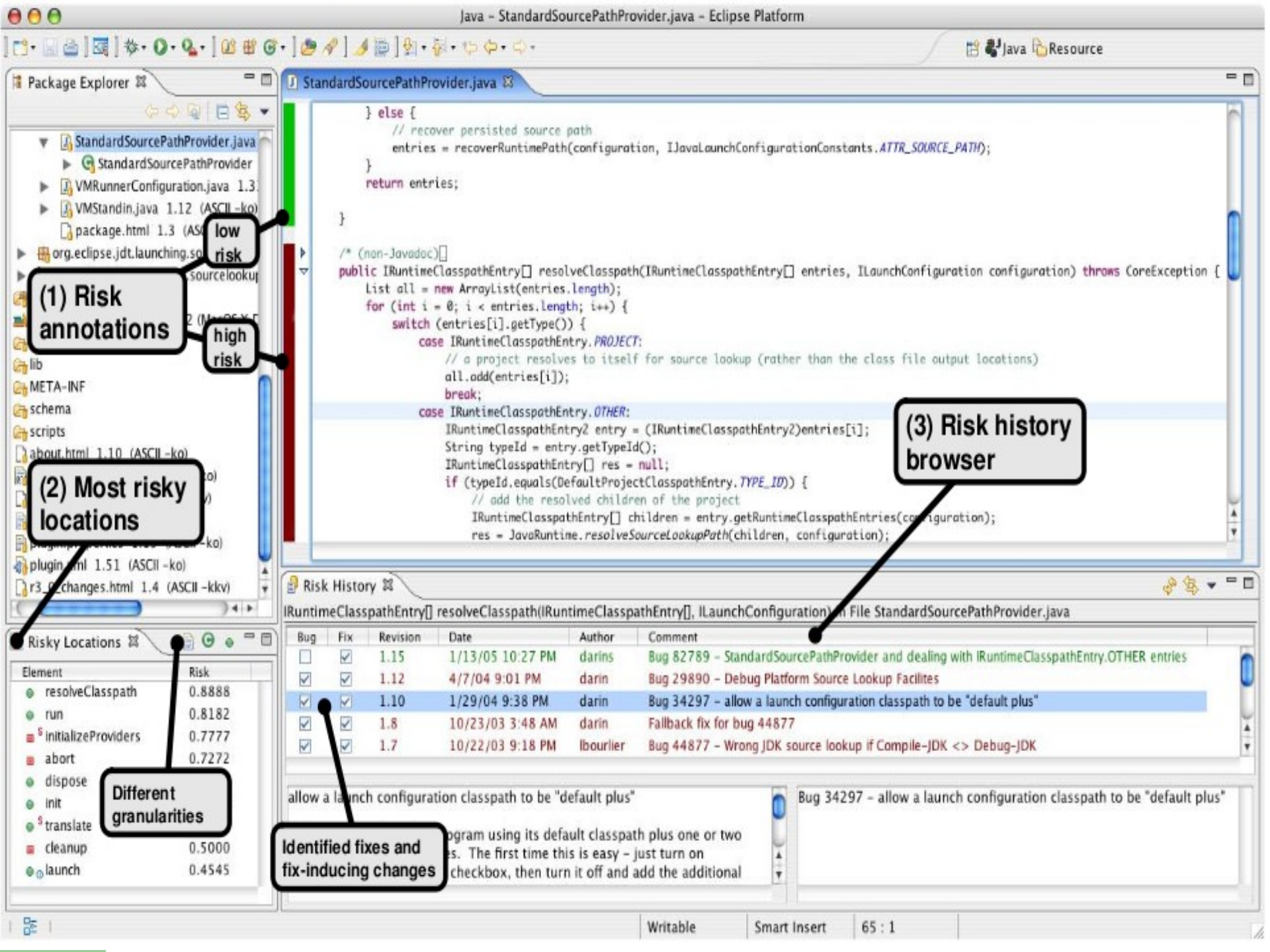
- Precision
  - How much of the information found is relevant?
- Recall
  - How much of the relevant information is found
- Also, evaluation of probabilistic models
  - Future bug or change predictions, etc.

Trends in MSR

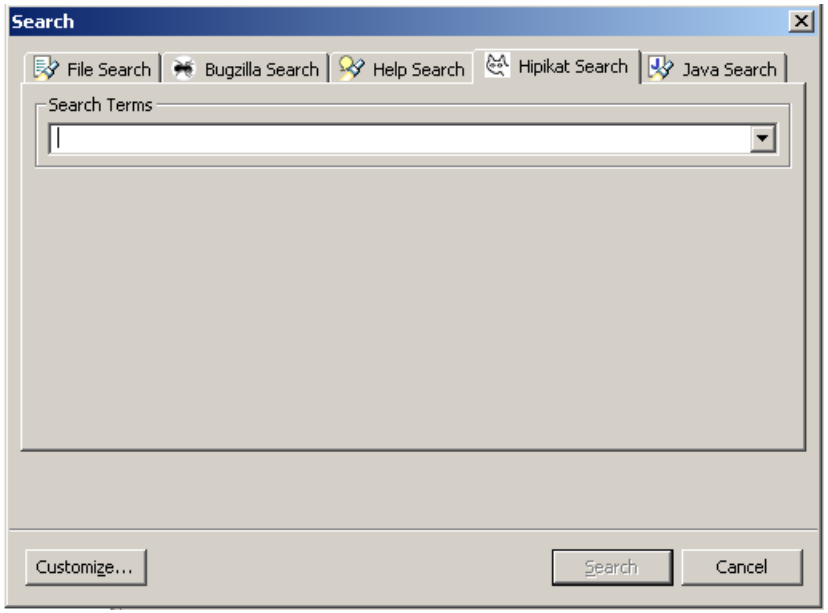
- Development assistance
  - Tools, environments…
- Management assistance
  - Reports, controlling, feedback...
- Evaluation/assessment of software systems
  - esp. of external software tools

Development assistance

- Integrate knowledge into development tools. Future environments should (Zeller):
  - mine patterns from program + process
  - apply rules to make predictions
  - provide assistance in all development decisions
  - adapt advice to project history


Java - StandardSourcePathProvider.java - Eclipse Platform
Package Explorer
StandardSourcePathProvider.java
low risk
high risk
(1) Risk annotations
(2) Most risky locations
(3) Risk history browser
Risky Locations
Element
Risk
resolveClasspath 0.8888
run 0.8182
initializeProviders 0.7777
abort 0.7272
dispose
init
translate
cleanup 0.5000
launch 0.4545
Different granularities
Risk History
Bug
Fix
Revision
Date
Author
Comment
1.15 1/13/05 10:27 PM darins Bug 82789 - StandardSourcePathProvider and dealing with IRuntimeClasspathEntry.OTHER entries
1.12 4/7/04 9:01 PM darin Bug 29890 - Debug Platform Source Lookup Facilites
1.10 1/29/04 9:38 PM darin Bug 34297 - allow a launch configuration classpath to be "default plus"
1.8 10/23/03 3:48 AM darin Fallback fix for bug 44877
1.7 10/22/03 9:18 PM lbourlier Bug 44877 - Wrong JDK source lookup if Compile-JDK <> Debug-JDK
Identified fixes and fix-inducing changes
Writable
Smart Insert
65 : 1

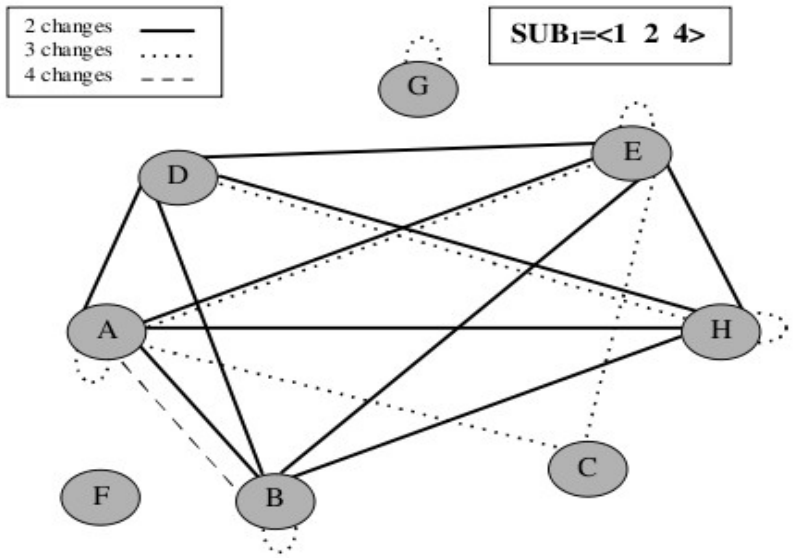
New developer assistance



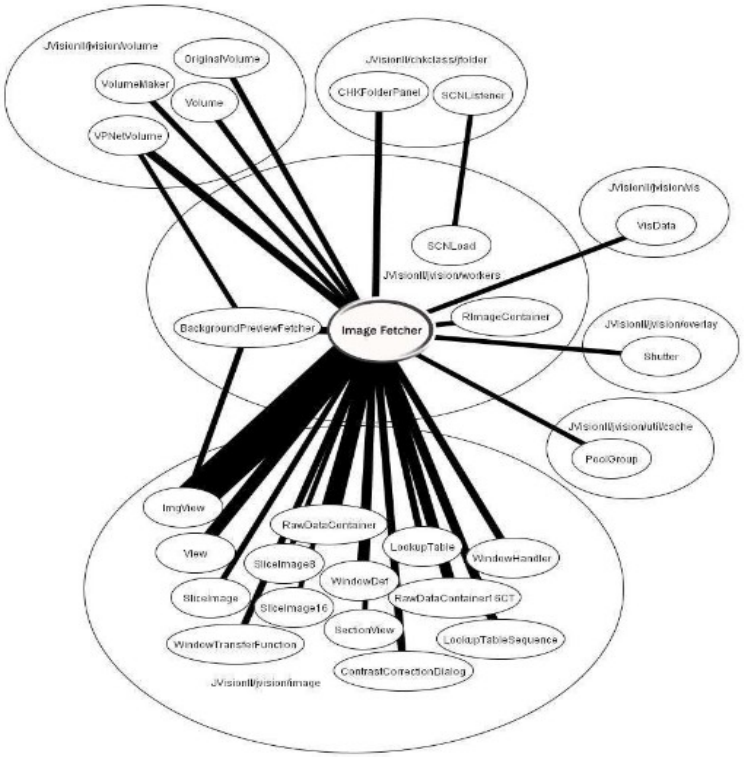
- By means of the Hipikat tool
- Artifacts created by the project are integrated into a *project memory*
- Developers can query for relevant artifacts

Logical couplings

- Coupling among subsystems (Gall et al.)
- Which classes change together?
- How many classes changes occurred across subsystems?



Change *smells*

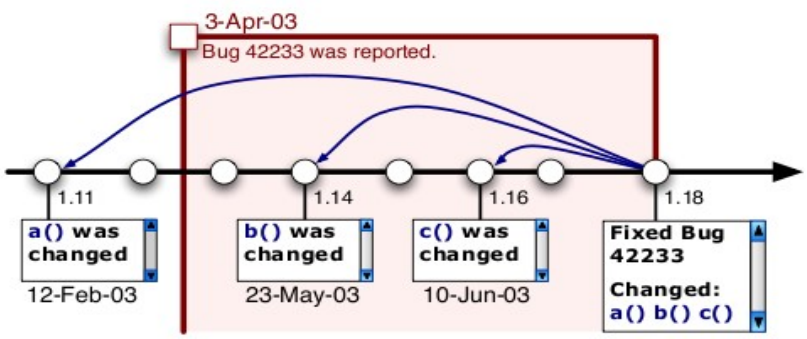


- Strength of logical couplings (Weissgerberger)
- Based on an analogy with “bad smells”
- Man-in-the-middle
- Suggests refactorings

More about refactorings

- Identifying for incomplete refactorings in versioning systems (Weissgerber et al.)
- Are refactorings less error prone than other changes?
  - Yes, they are! (surprise!)
  - But many refactorings at a time introduces many subsequent changes.

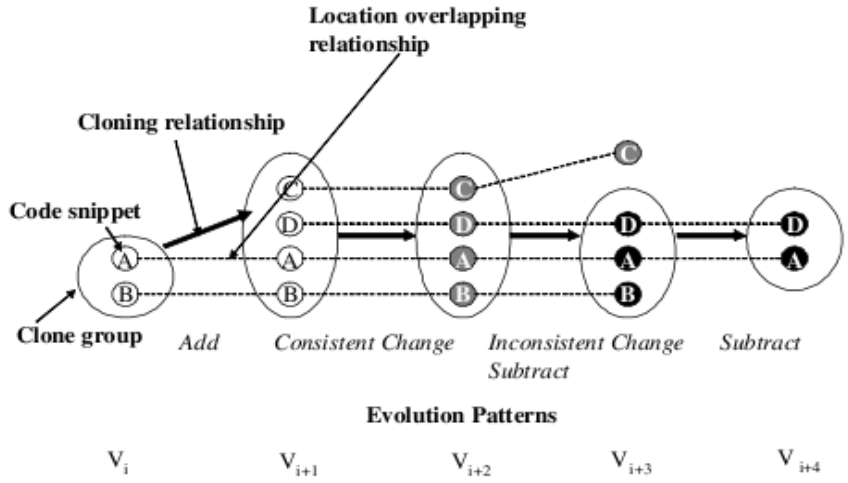
Fix-inducing changes

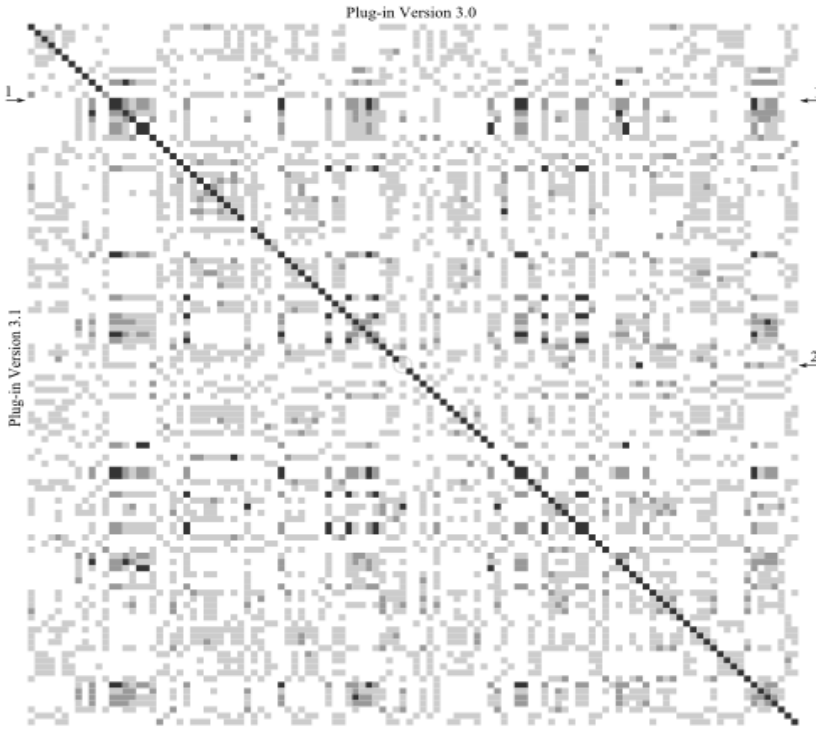


- Fixes that introduces new bugs (Zimmermann et al.)
- Don't program on Fridays! (Saturdays is also a bad day)

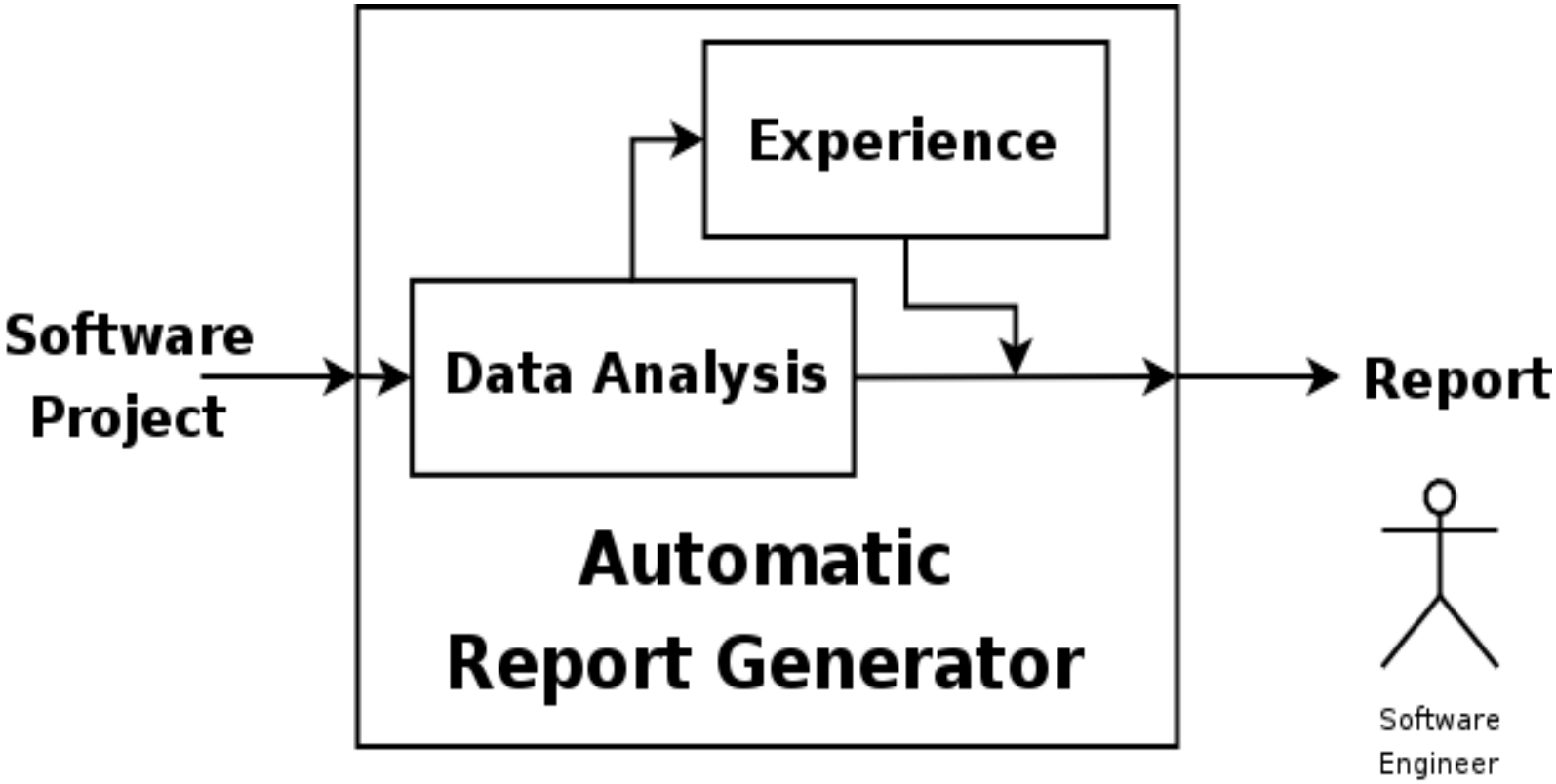
Cloning, similarities, etc.

- Similar classes
- Clone genealogy





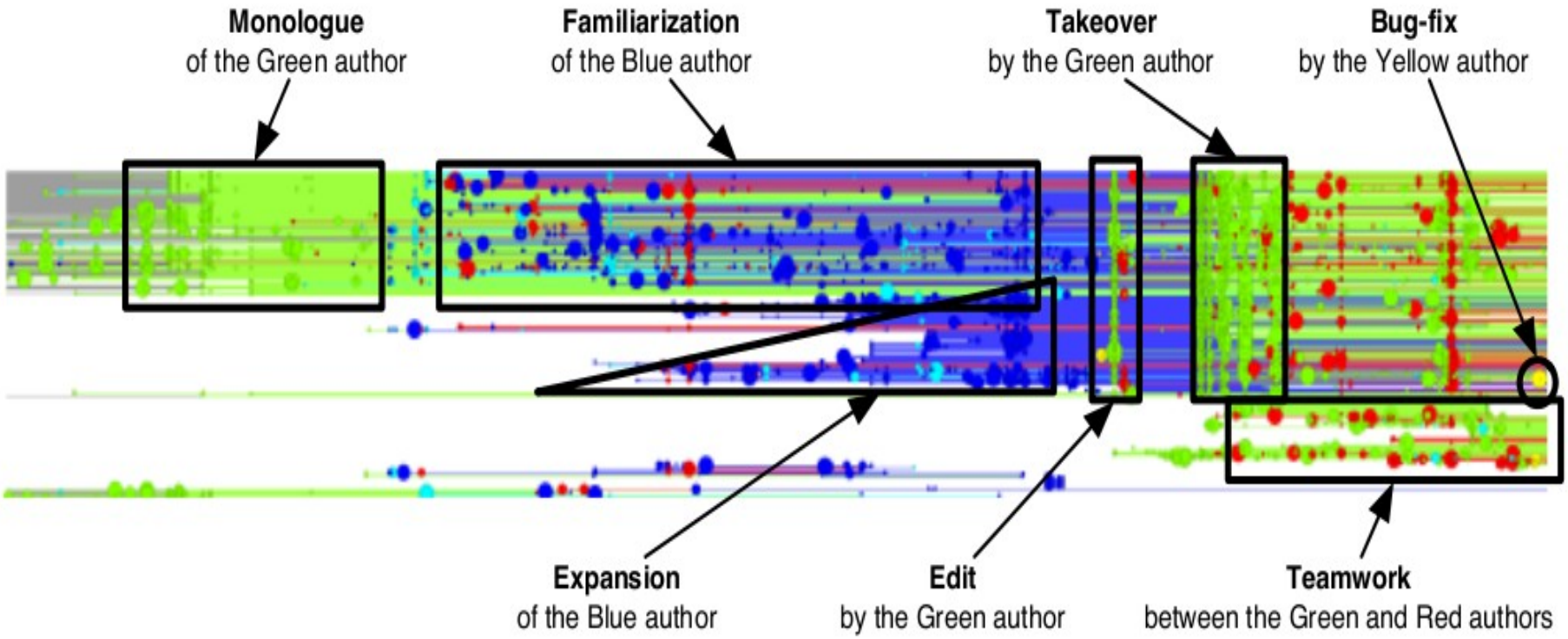
Management Assistance



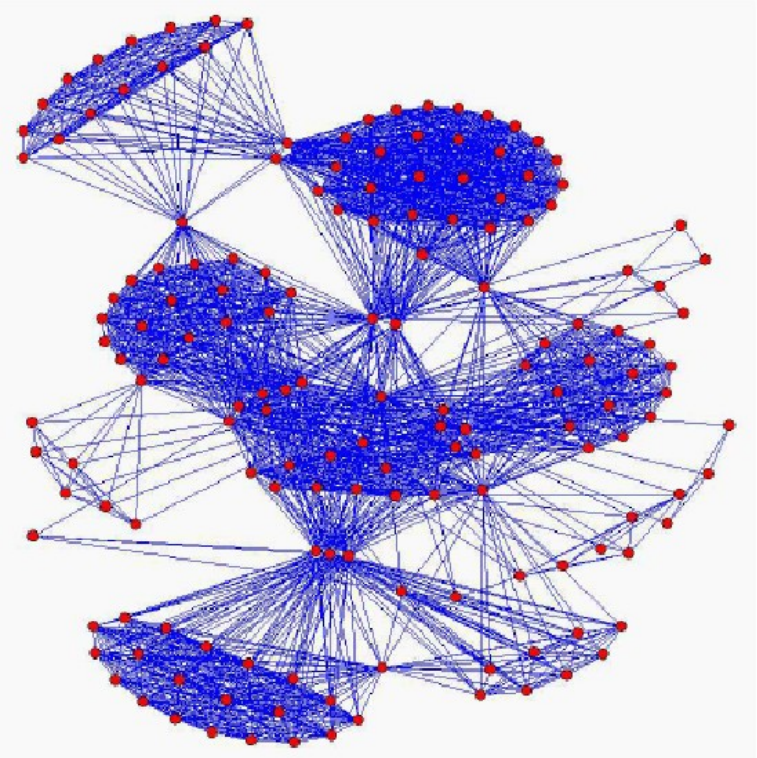
Who should fix this bug?

- Help in the automation bug assignments using machine learning techniques
- Depends on the management policies
  - Do we want specialization or broader knowledge of our maintenance team?

Co-changing files and authors[]

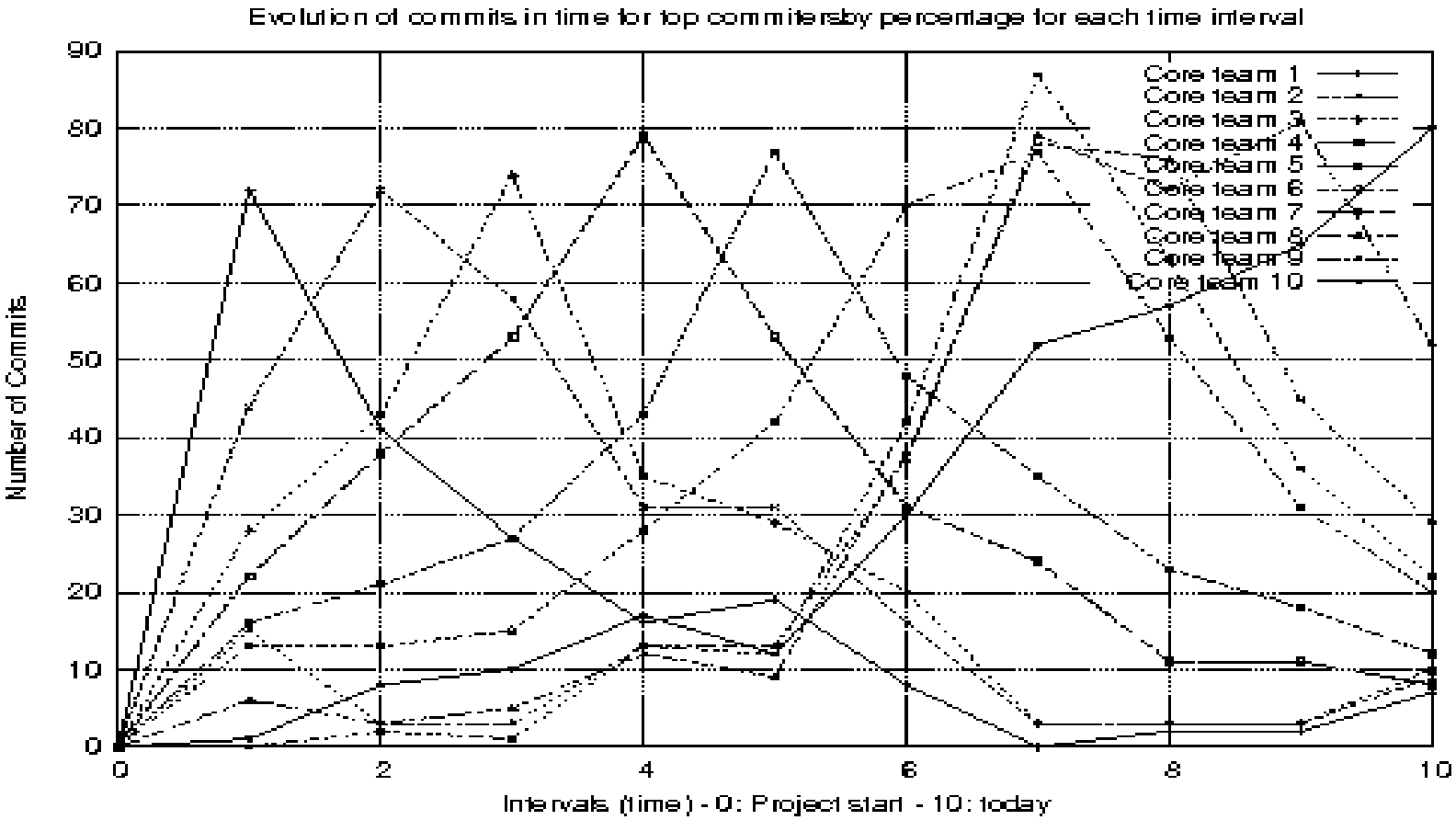


Social Network Analysis

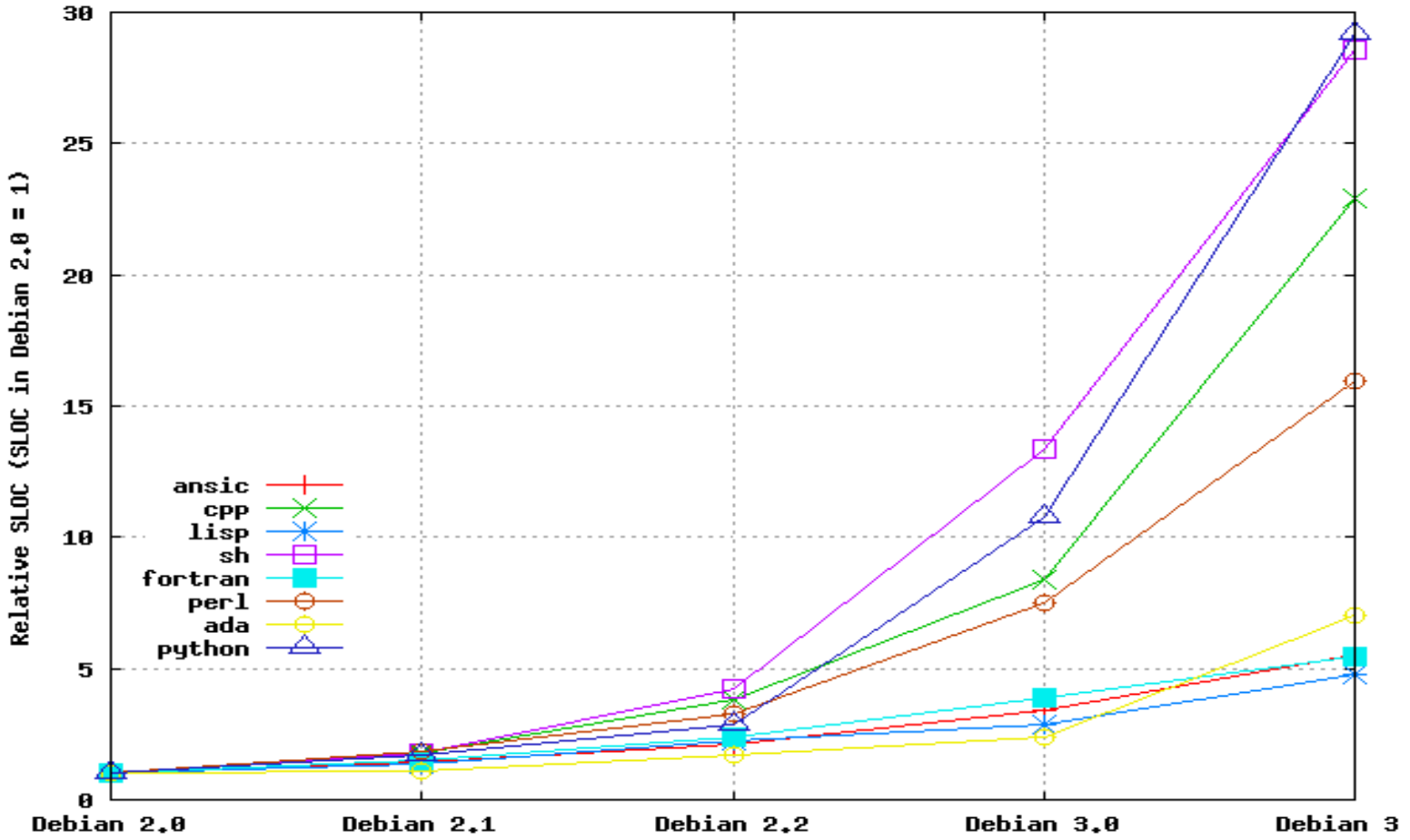


- Interactions among developers
- Who knows about this?
- Developer territoriality (German)

Developer generations [R06a]



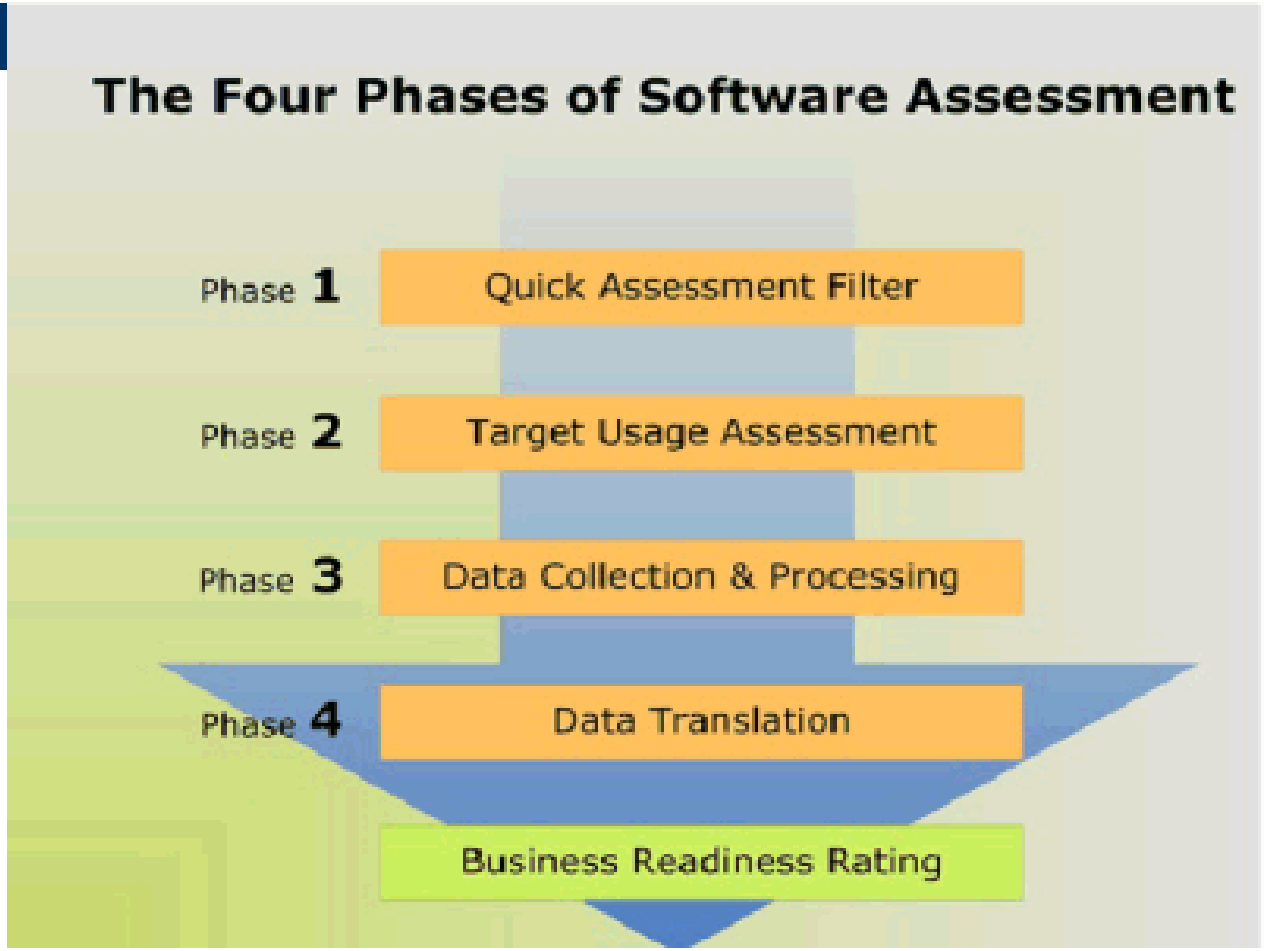
Programming Languages [R06]



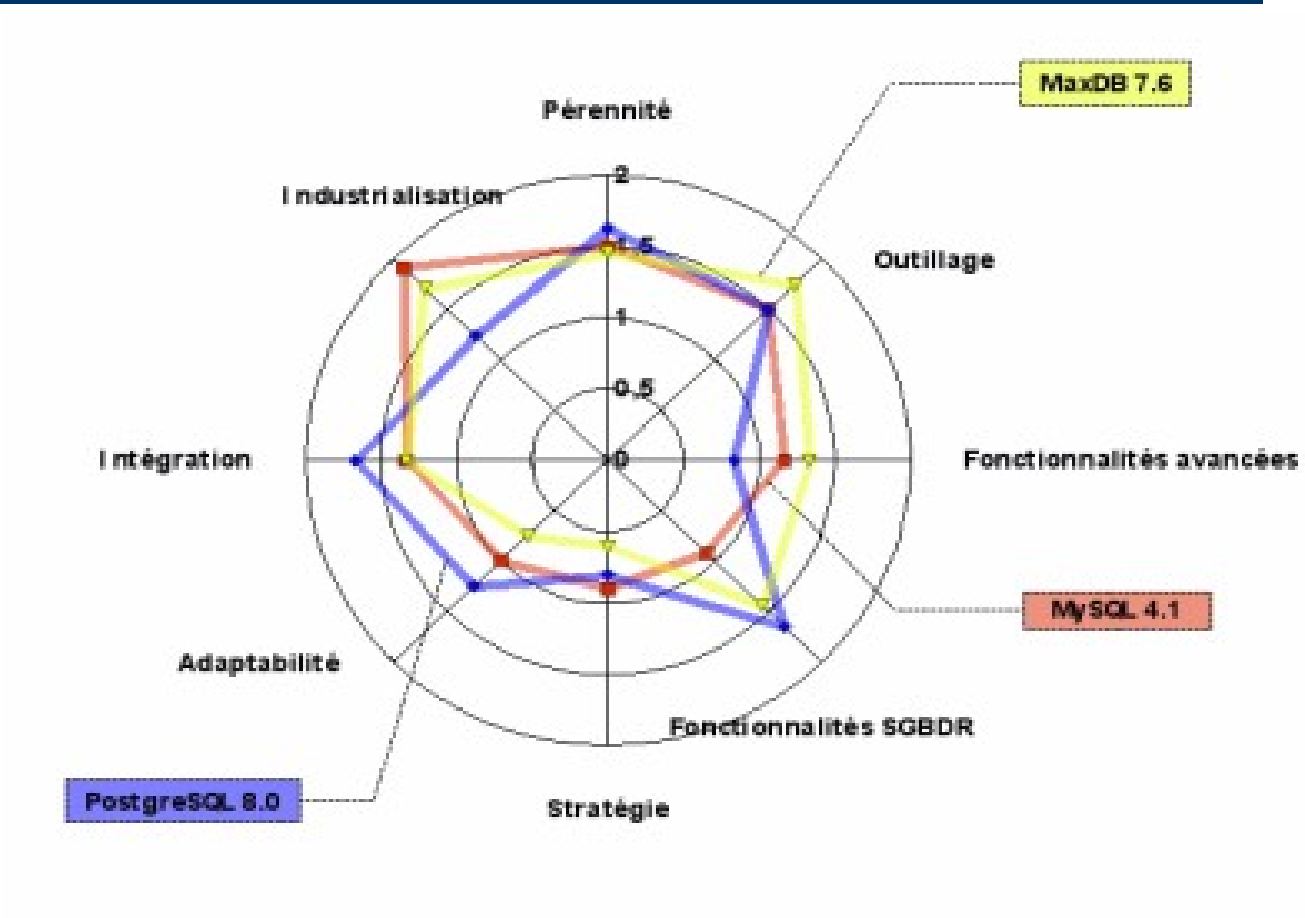
Evaluation of software systems

- Especially focused on Open Source Systems
  - Project is unknown, but we want to deploy it
- Trust, reliability, quality...
  - Many of these do not depend only on the current state of the source code!

Frameworks: OpenBRR [W06]



Frameworks: QSOS [QSOS06]



Limitations of MSR

- Information sources depicting high-level abstractions (design and architecture models) not directly available
- Who's experience is a “good” experience?
  - Validation
- Integration of various data sources is complex
- Never complete information
- …

Final Summary

- MSR is a field with much activity in recent times!
- We have seen the main trends in MSR research
- Yet, it is still in an immature phase to affect development...

Any questions?

Thanks for your attendance and interest!

More information is available at
http://libresoft.urjc.es

Acknowledgements

- Much, if not all, of the first slides has been taken from:
  - “A survey and taxonomy of approaches for mining software repositories in the context of software evolution”
  - Huzefa Kagdi, Michael L. Collard, Jonathan I. Maletic
  - Journal of Software Maintenance and Evolution: Research and Practice, Volume 19, Issue 2, March/April 2007