Software Evolution
Work / Bench / Marks

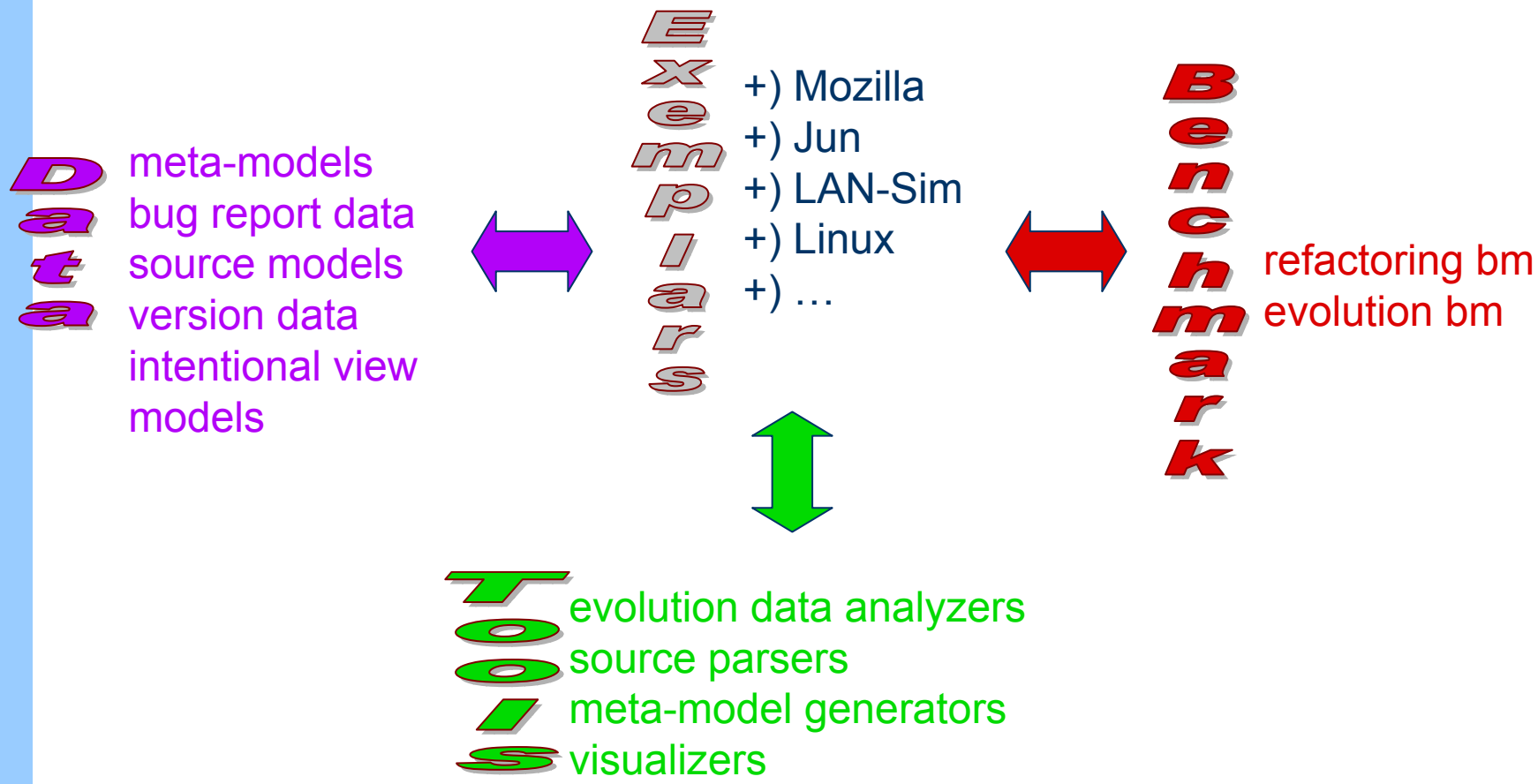
Harald Gall
Martin Pinzger

Technical University of Vienna
University Zürich, Ifi

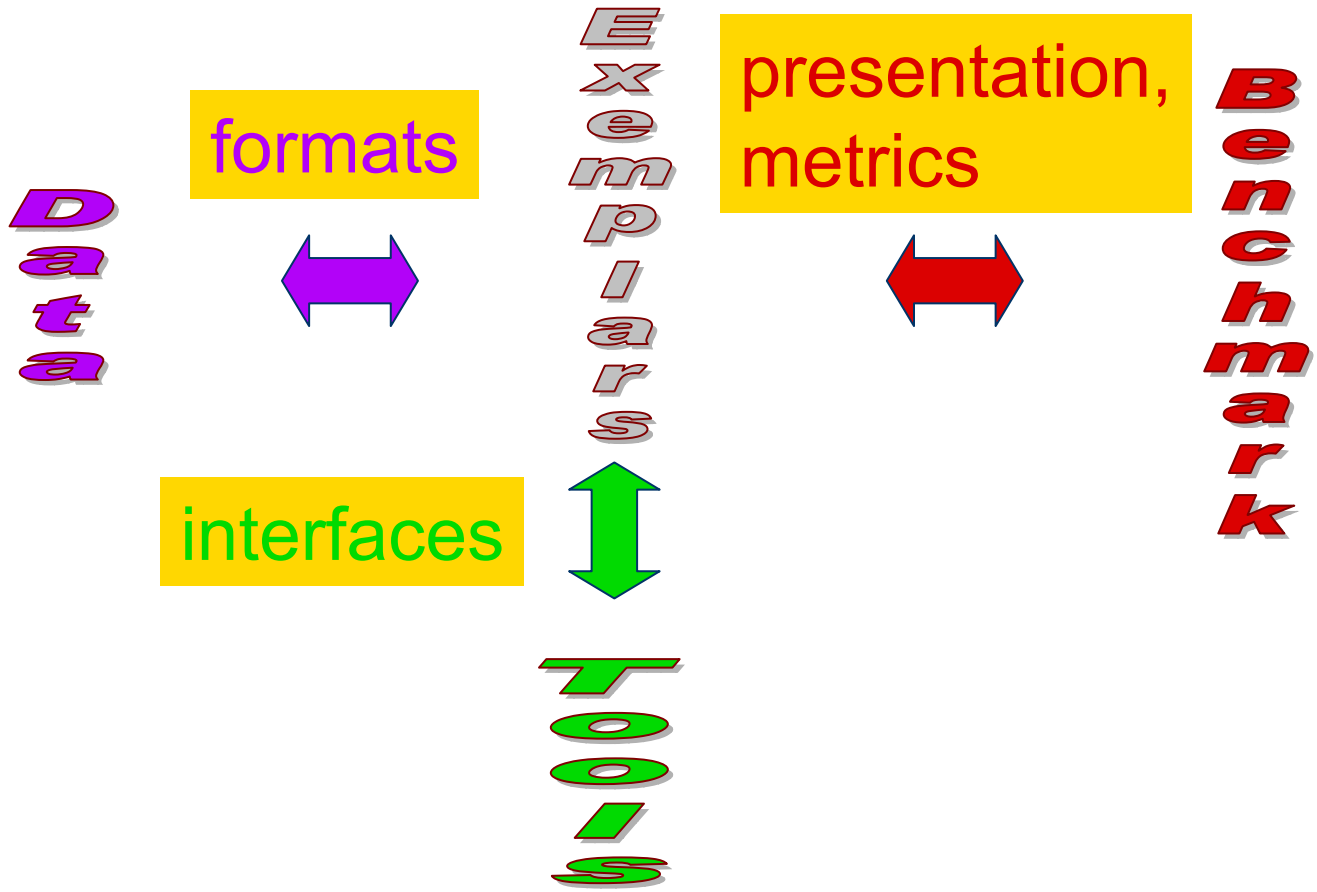
www.infosys.tuwien.ac.at

Release Meeting, Strasbourg, Feb 2004

Vision: evolution wizard



Let's talk about …



Objectives for the day

- The goal is to work on
  - meta-models, schemas, source models,
  - tools and algorithms, time warping, visualization,
  - techniques to promote data and result interchange, benchmarking, tool description, etc.
  - or whatever can be used as a basis for different tools that rely or work on evolution analysis.

Agenda suggested

- (a) A short introductory talk
- (b) A short presentation (max 7 min) by each Release network partner to about their cooperation activities and results (since we are a network the plan is to focus on "cooperation activities"):
  - what has been done in the cooperation,
  - what has been achieved (prelim results),
  - what is open,
  - future plans for the cooperation
- The slides will be synthesized into one presentation at the end of the day.

Envisioned outcome

- Produce a short report about the achievements in this area of research.
  - This should include all the papers written/accepted in that area.
  - Based on that we could produce a nice technical report as a kind of "RELEASE deliverable" including open issues and future work (as a kind of short-term research agenda).
- At the end we could synthesize the results in an overview short report (1-2 pages). (maybe for SE)

Coop reports

- UBe – Vienna (TUV)
- Sannio – TUV
- Lisbon – Leicester
- UMH –UA
- UCL – UMH – UNL – ULB
- UBe - UTT
- …

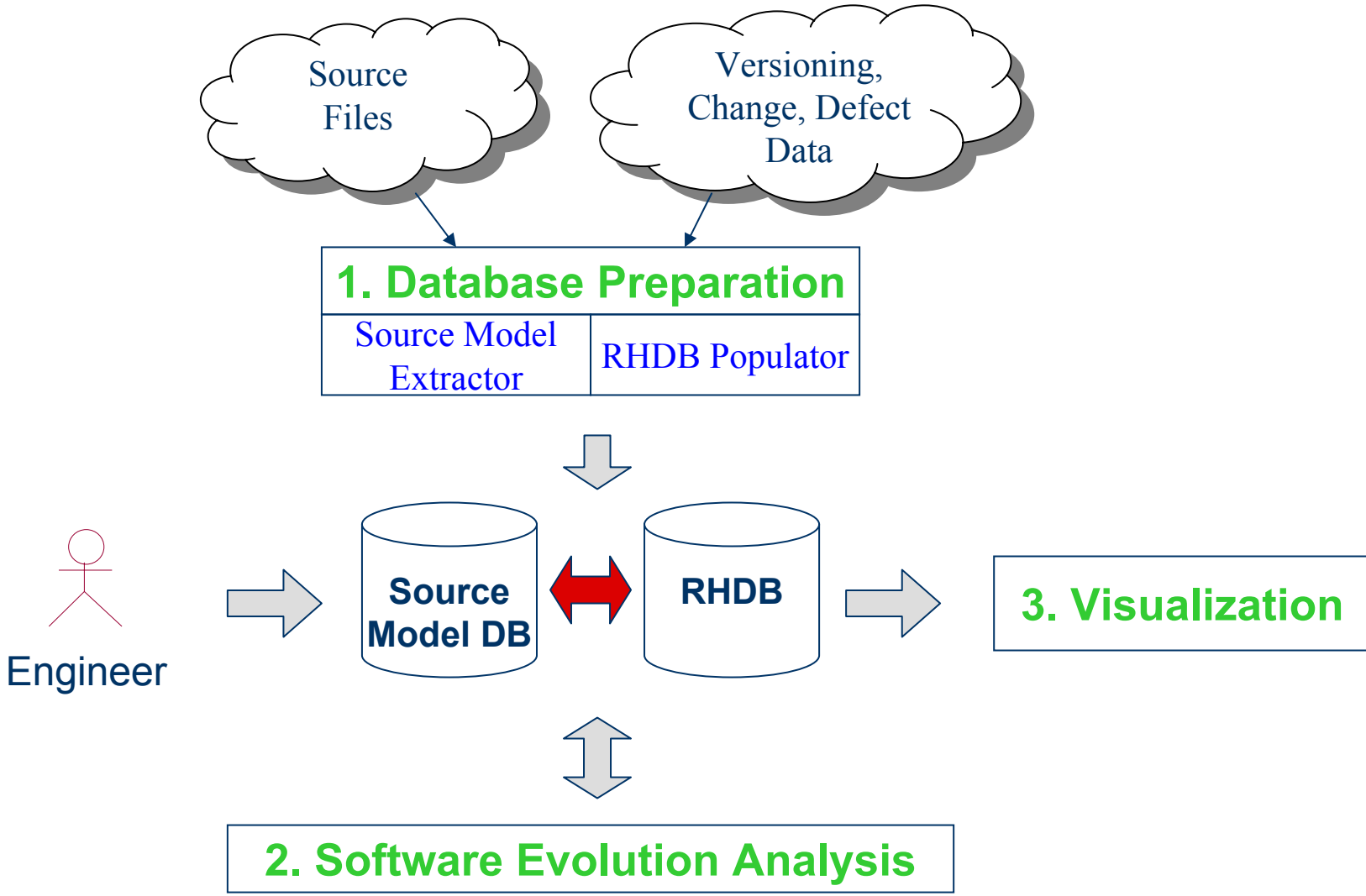
Benchmarks

- Exemplars
  - Mozilla, Jun, LAN-Simulator, …
- Tools
- Meta-models

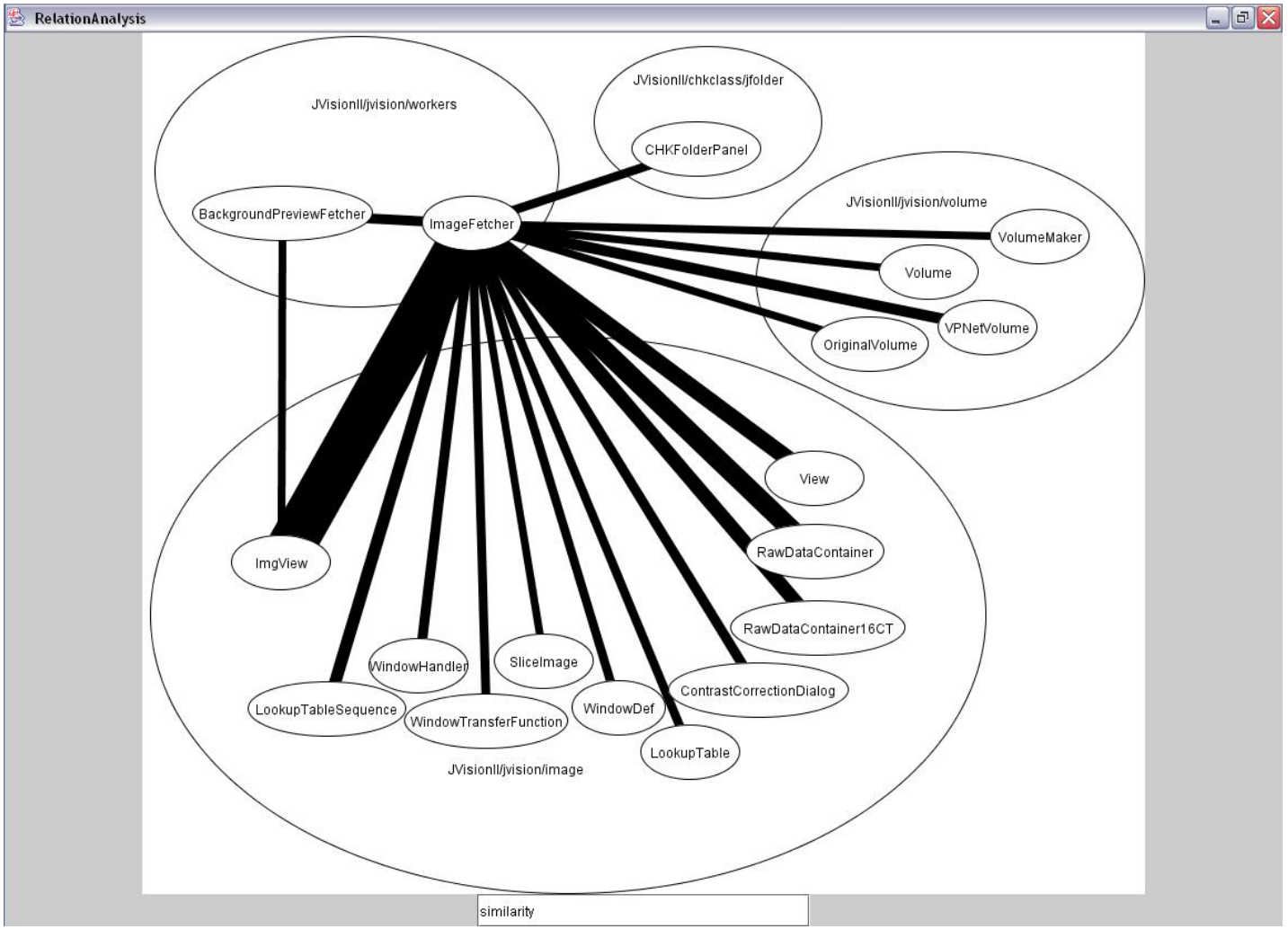
system	lang	#rel	bug-db	
Mozilla	C/C++	20+	yes	
Jun	Java			
LAN-Sim	C?			
Linux	C	400+	yes	

Software Evolution Analysis @ TUV (& UniZh)

SEA evolution analysis process



Change dependent coupling Viewer



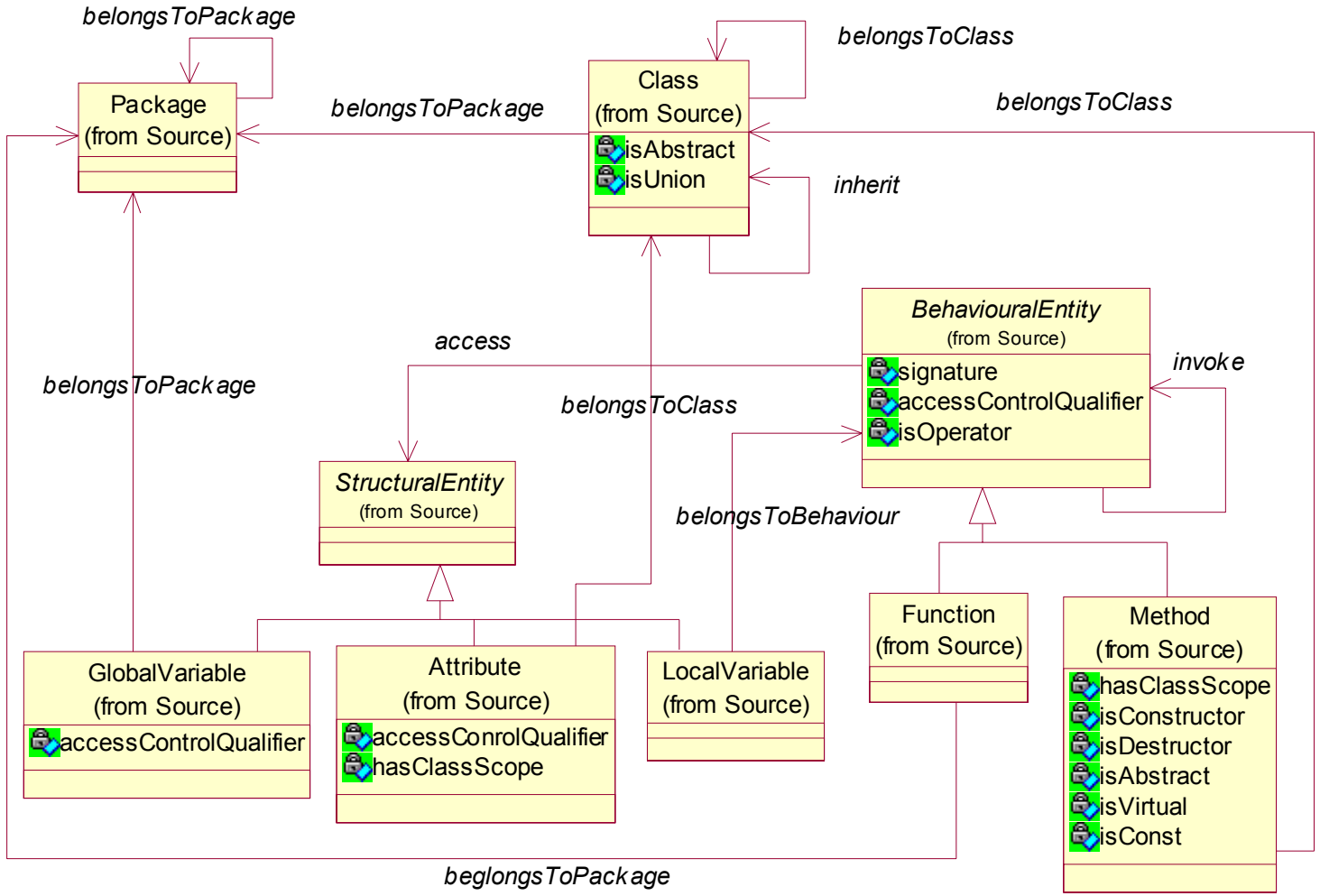
Data preparation

- Source Model Database
  - Parsing and lexical analysis techniques
  - Source model entities and relationships ← source files
    - e.g. classes, methods; inherit, invoke
  - Source model = directed attributed graph
  - **TUV-FAMIX** meta model
- Release History Database (RHDB)
  - Modification Reports (MR) ← versioning systems
  - Problem Reports (PR) ← bug reporting systems
    - patch information

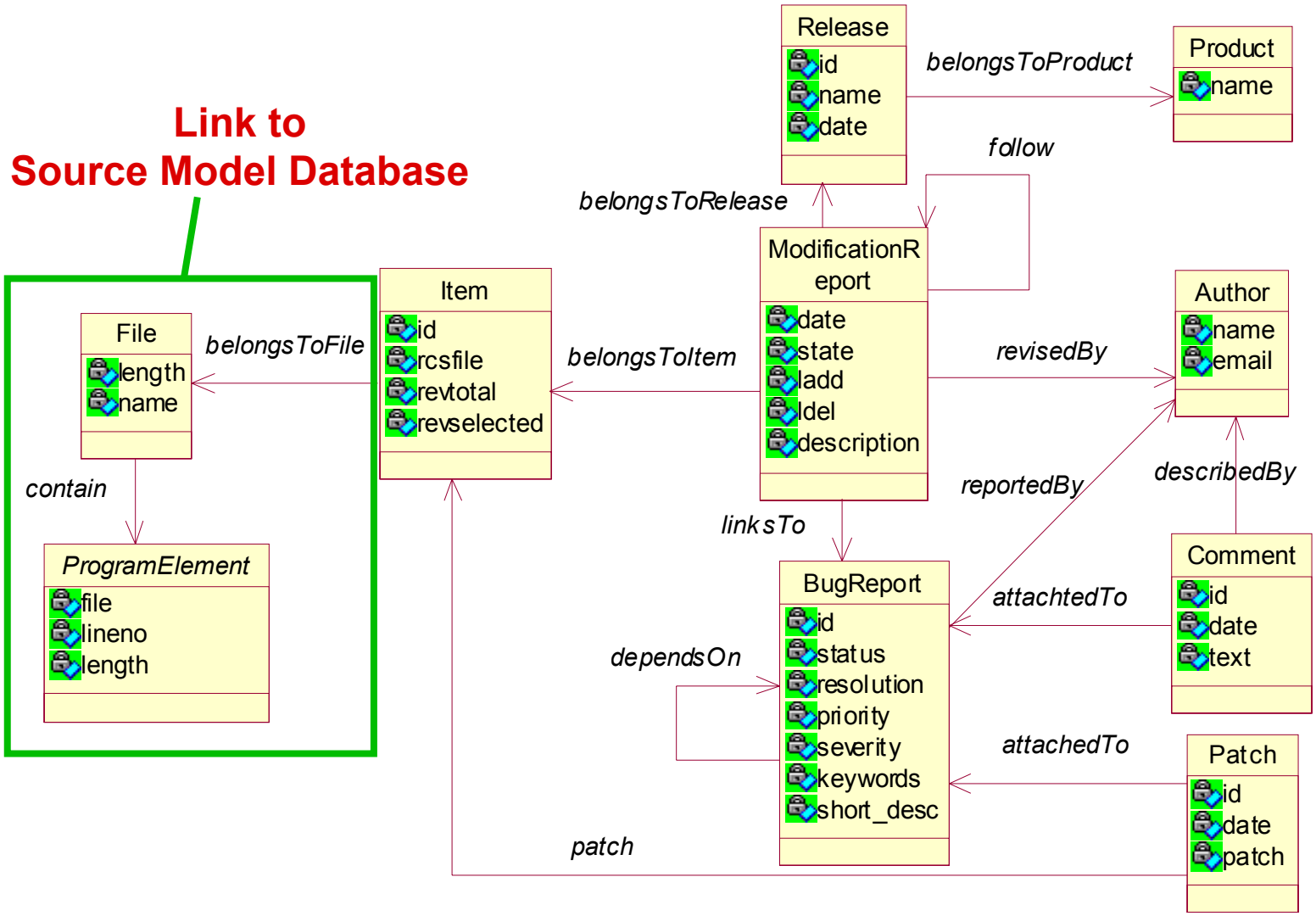
TUV-FAMIX meta model

- Source code meta **model for object oriented programming languages**
  - Based on FAMIX
- Defines entity, relationship types together with their attributes
  - Semantics of entities and relationships
- **Basis for integration** of analysis tools
  - Source code analysis
  - Architecture recovery

TUV-FAMIX (excerpt)



Release History Database



Integrating versioning, change, and defect data

- **Files** as links between source model and release history data
  - **Abstract** source model information up to file-level

  -> Files that implement an architectural property
- Query RHDB to extract **hidden dependencies** and **logical couplings** between these files
  - „Two files are **depended** if there exists a bug report that references both files“
    - Both files had to be touched to fix a certain bug
  - „Two files are **logically coupled** if both files have been checked in to the repository at the same time“

Source model extraction

- Currently focus on C/C++ and Java
- Parsing C/C++
  - Can get complex
    - preprocessor statements, templates, types, ...
  - In cooperation with Uni Sanio (Giuliano Antoniol)
    - extracting UML-like model
    - converter to TUV-FAMIX
- Java
  - Using Eclipse JavaDT API

SM storage format

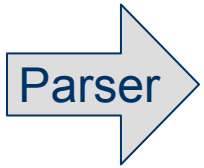
- Files - Rigi Standard Format (RSF)
  - Used to specify **directed attributed graphs**
    - **Nodes** (e.g. File, Class, Method, Attribute, ...)
    - **Edges** (e.g. include, inherit, associate, invoke, ...)
    - **Attributes** (lineno, accessControlQualifier, isAbstract, ...)

    -> entity & relationship types defined by the TUV-FAMIX
  - <verb> <subject> <object>
    - **Node:** type CA Class
    - **Edge:** invoke fa() fb()
    - **Attribute:** lineno CA 25

Source model example FAMIX-RSF

```
// **** A ****
class A {
    B pB;
    int sumB() {
       return pB.getX() + pB.getY();
    }
}

// **** B ****
class B {
    int x, y;
    int getX() { return x; }
    int getY() { return y; }
}
```




```
type A Class
type A::pB Attribute
type A::sumB() Method
belongsToClass A::pB A
belongsToClass A::sumB() A

type B Class
type B::x Attribute
type B::y Attribute
type B::getX() Method
type B::getY() Method
belongsToClass B::x B
belongsToClass B::y B
belongsToClass B::getX() B
belongsToClass B::getY() B

associate A B
access A::sumB() A::pB
invoke A::sumB() B::getX()
invoke A::sumB() B::getY()
```

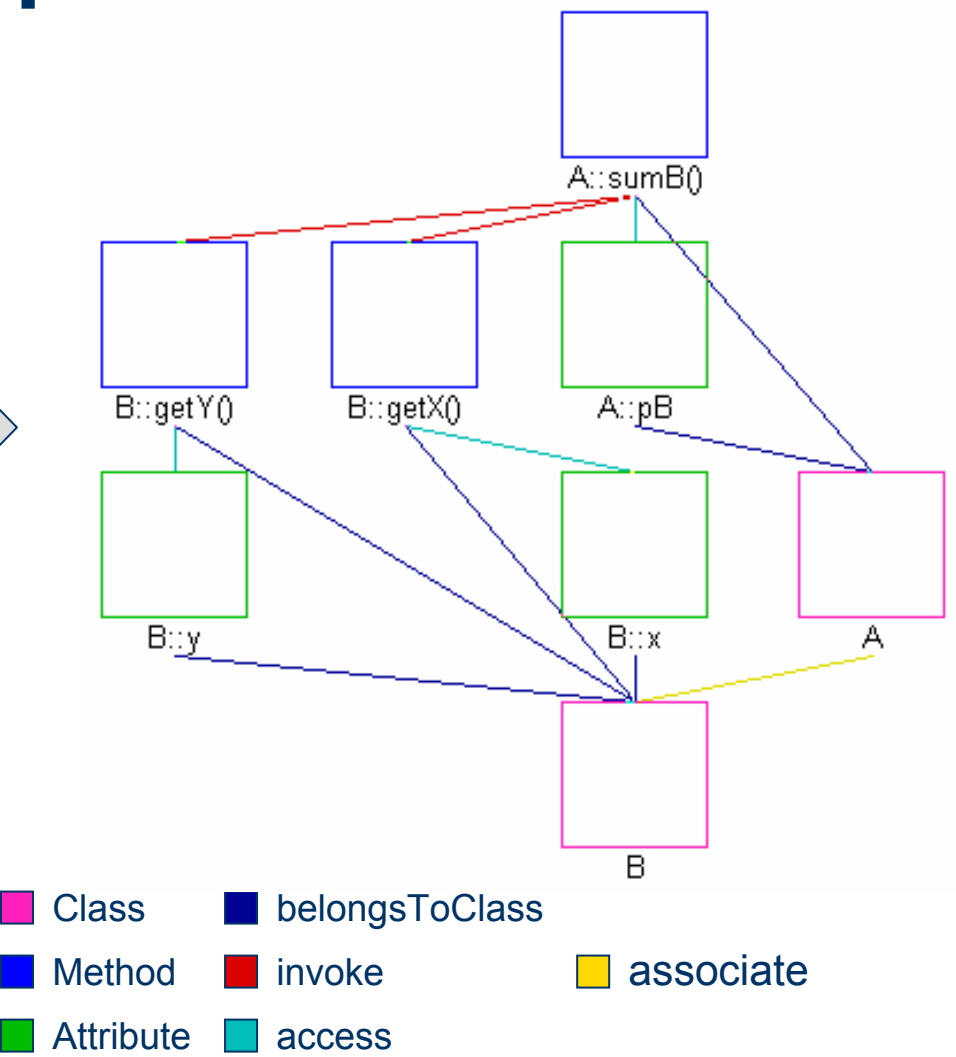
RSF graph representation

type A Class
type A::pB Attribute
type A::sumB() Method
belongsToClass A::pB A
belongsToClass A::sumB() A

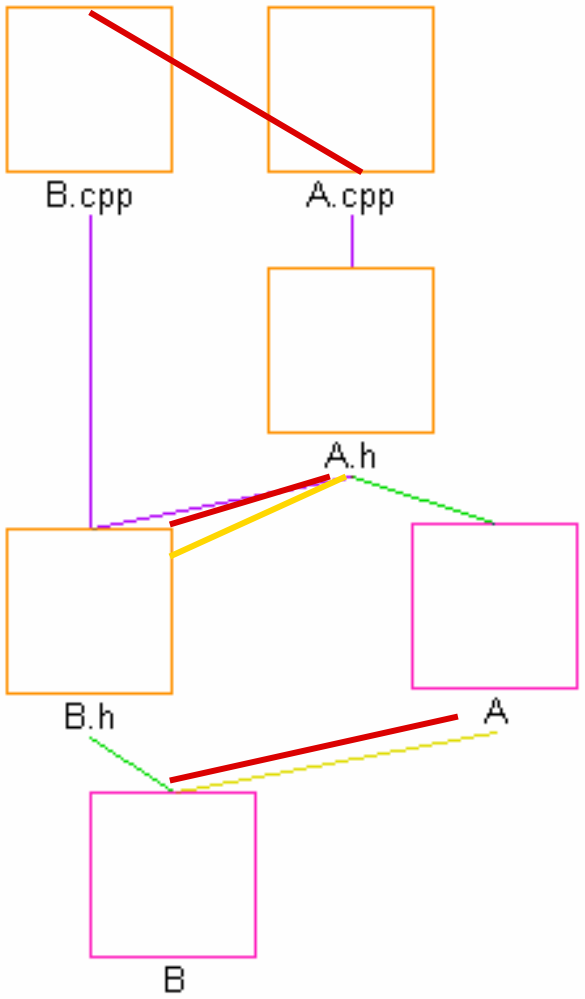
type B Class
type B::x Attribute
type B::y Attribute
type B::getX() Method
type B::getY() Method
belongsToClass B::x B
belongsToClass B::y B
belongsToClass B::getX() B
belongsToClass B::getY() B

associate A B
access A::sumB() A::pB
invoke A::sumB() B::getX()
invoke A::sumB() B::getY()

Rigi



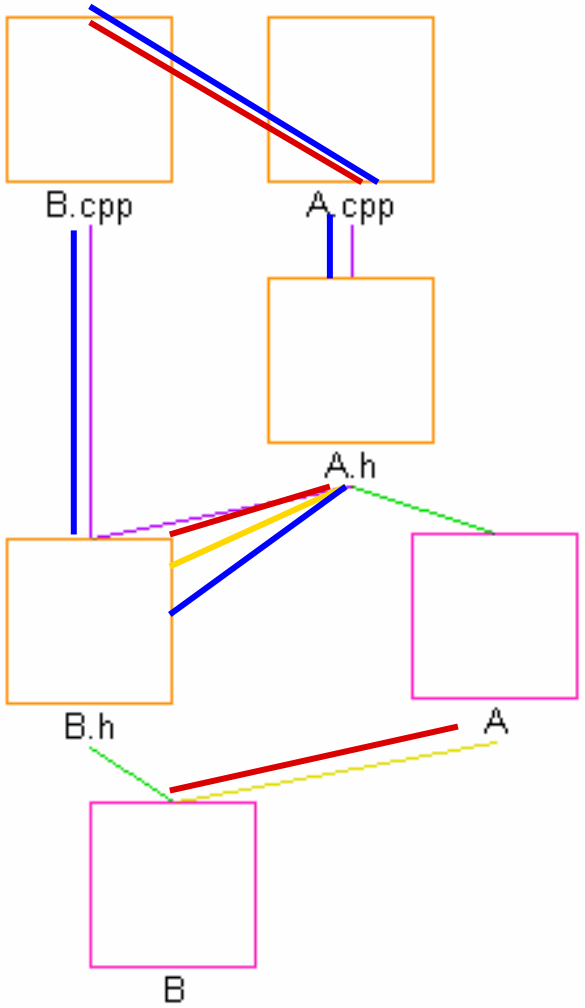
Information abstraction



- Method/function level -> class level -> file level
- Compute scores to characterize weight of abstracted relationships

File include invoke
Class contain associate

Integrating evolution data



- On file level integrate logical couplings and hidden dependencies relationships

-> Verify source model relationships

File include invoke logical_coupling
Class contain associate

Results

- Higher-level views on the implementation of software systems
  - file/module-level, directory/subsystem-level

  -> **facilitates reasoning about the implemented design**
- Integration of evolutionary data
  - Verification of source model relationships by versioning, change and defect data

  -> **highlight modules and subsystems that often are changed together (change impact analysis)**

Implementation

- Data preparation
  - Source Model Database
    - Parsers: Antoniols parser, Imagix-4D
    - Lexical analyzers: mpgrep (prototype), perl scripts, grep
    - FAMIX-RSF Exporters (Imagix-4D, Antoniol data)
  - Release History Database
    - Shell and Perl scripts, Java programs for RHDB queries
- Source model abstraction
  - Relational algebra tool (Grok)
  - Graph querying tool (Crocopat)

Implementation (cont.)

- Data integration
  - Conversion of RHDB query results to RSF (script)
  - Combination of RSF files (script)
- Visualization
  - **Rigi**
    - querying and editing graphs in FAMIX-RSF format
  - **Graphviz**
    - only for visualizing graphs
    - incorporate metrics and report data (node size, weight of edges)
  - **Relation Analysis Viewer (TUV)**

Conclusions

- TUV-FAMIX meta model
- Release History Database
- Integration of source model data, versioning, change and defect data
  - Contains **static relationships** but also **logical couplings** and **hidden dependencies** between modules and subsystems

-> More complete view on the implementation of a software system that allows reasoning about software architecture and evolution.

Next steps

- Refine meta model
  - Investigate other versioning and bug reporting systems
    - e.g. Rational ClearCase, Rational ClearQuest, MS Visual SourceSafe
- Extend links between source models and release history data to source model entities
  - Analyze cvs diffs and patches

  -> Source model entities affected by a change or bug fix
- Integrate time-slice approach to visualize the evolution of architectural properties