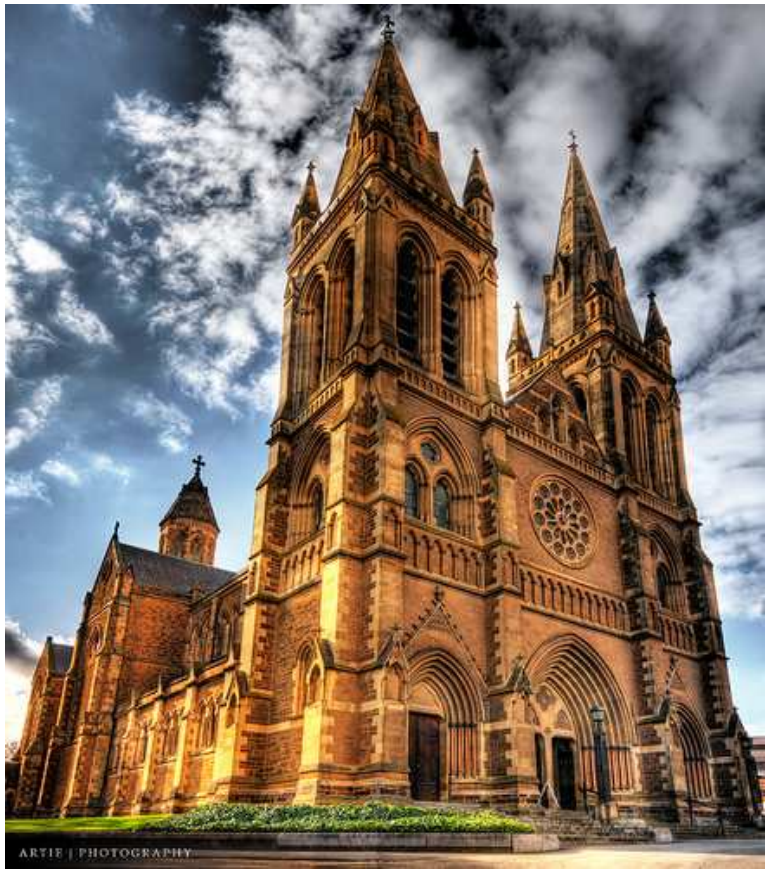
ACCU 2009:
Stop the Software Architecture Erosion

Bernhard Merkle
Central Unit Research & Development
Software-Engineering
SICK-AG Waldkirch

mailto: Bernhard.Merkle@sick.de



Wide product and technology range

: Industrial Sensors

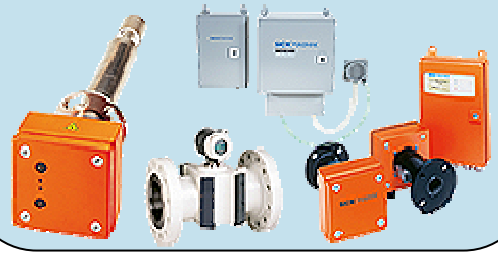
: Advanced Industrial Sensors

: Encoder

: Industrial Safety Systems

: Auto Ident

: Analyzers & Process Instrumentation



DARPA: Urban/Grand Challenge, Google SV

SICK

Traffic and Luggage Control

Introduction:

SICK

Levels of Static Analysis

- Code, Design, Architectural
- Examples

Architectural Analysis

- Use Cases
- Tool Support,
- Examples
- Pros/Cons

Summary

Possible levels of Static Analysis:

SICK

Goal: find, avoid Problems, Increase QA (and measure it)

Micro-Level

- Code
- e.g: =, ==, { },

Macro-Level

- Class-Design
- e.g: by reference, String concat, Exception-Handling

Architecture-Level:

- Layers, Graphs, Subsystems, Compoments, Interfaces
- e.g: Coupling, Dependency, etc…

Tool-support for each level…

Micro-Level / Code:

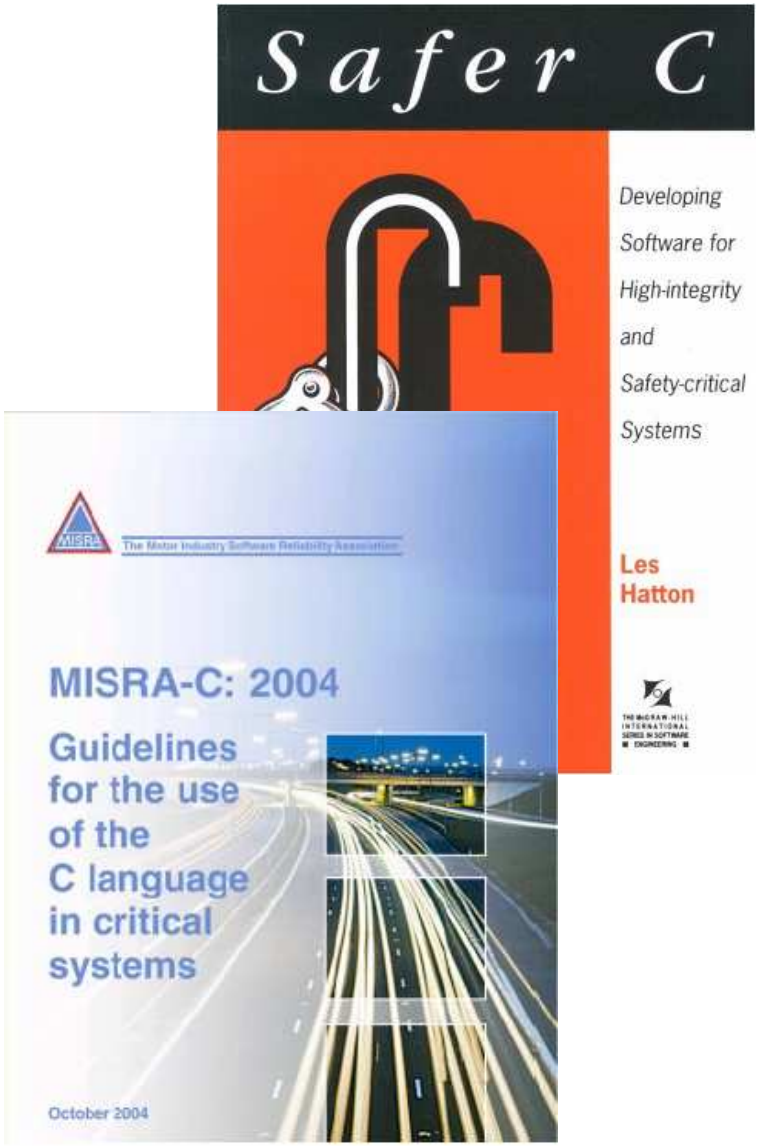
SICK

Language itself

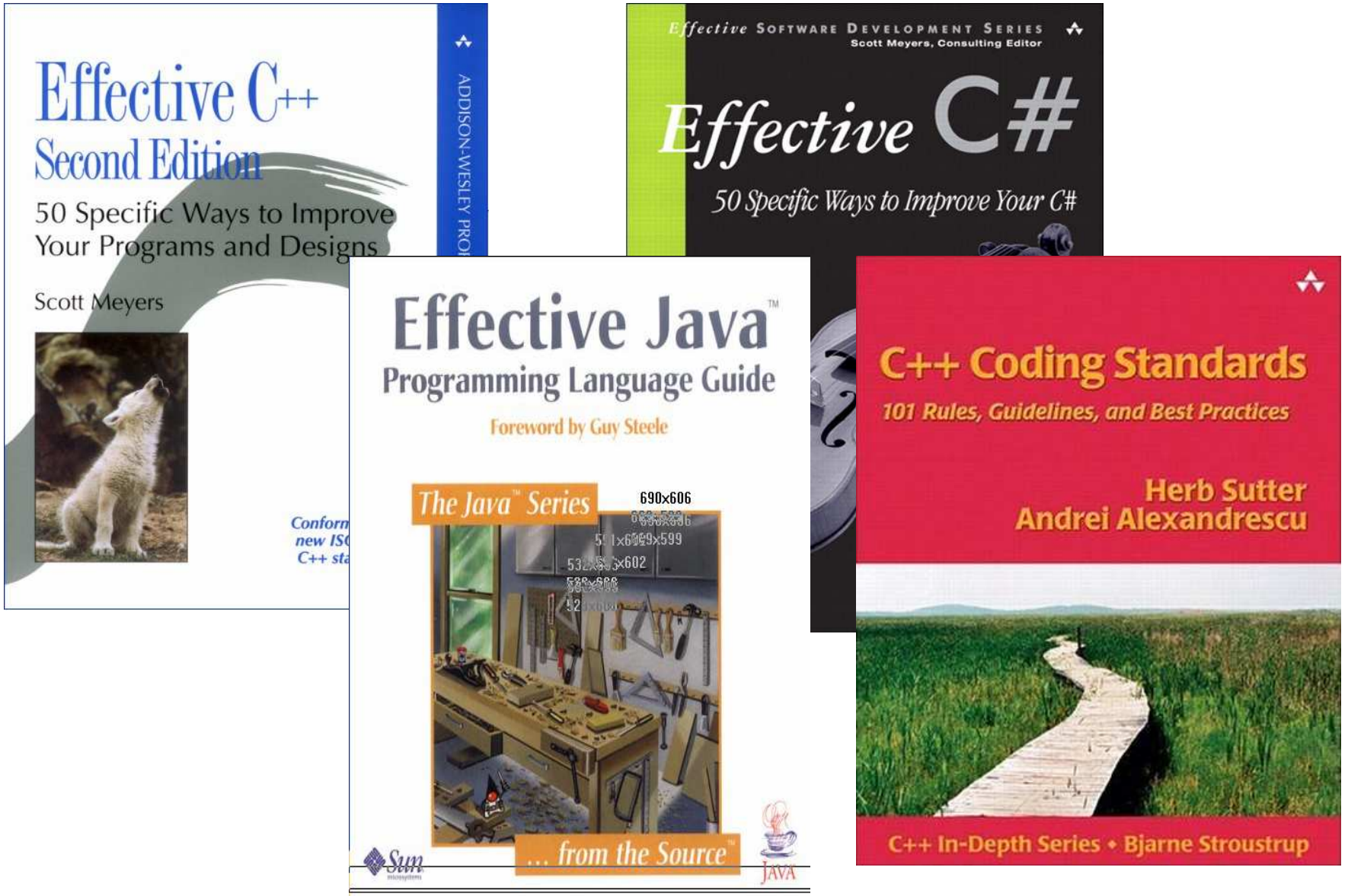
- Appendix F/ANSI or G/ISO
  - Unspecified behaviour
  - Undefined behaviour
  - Implementation-defined behaviour
  - Locale-specific behaviour

Usage of Language

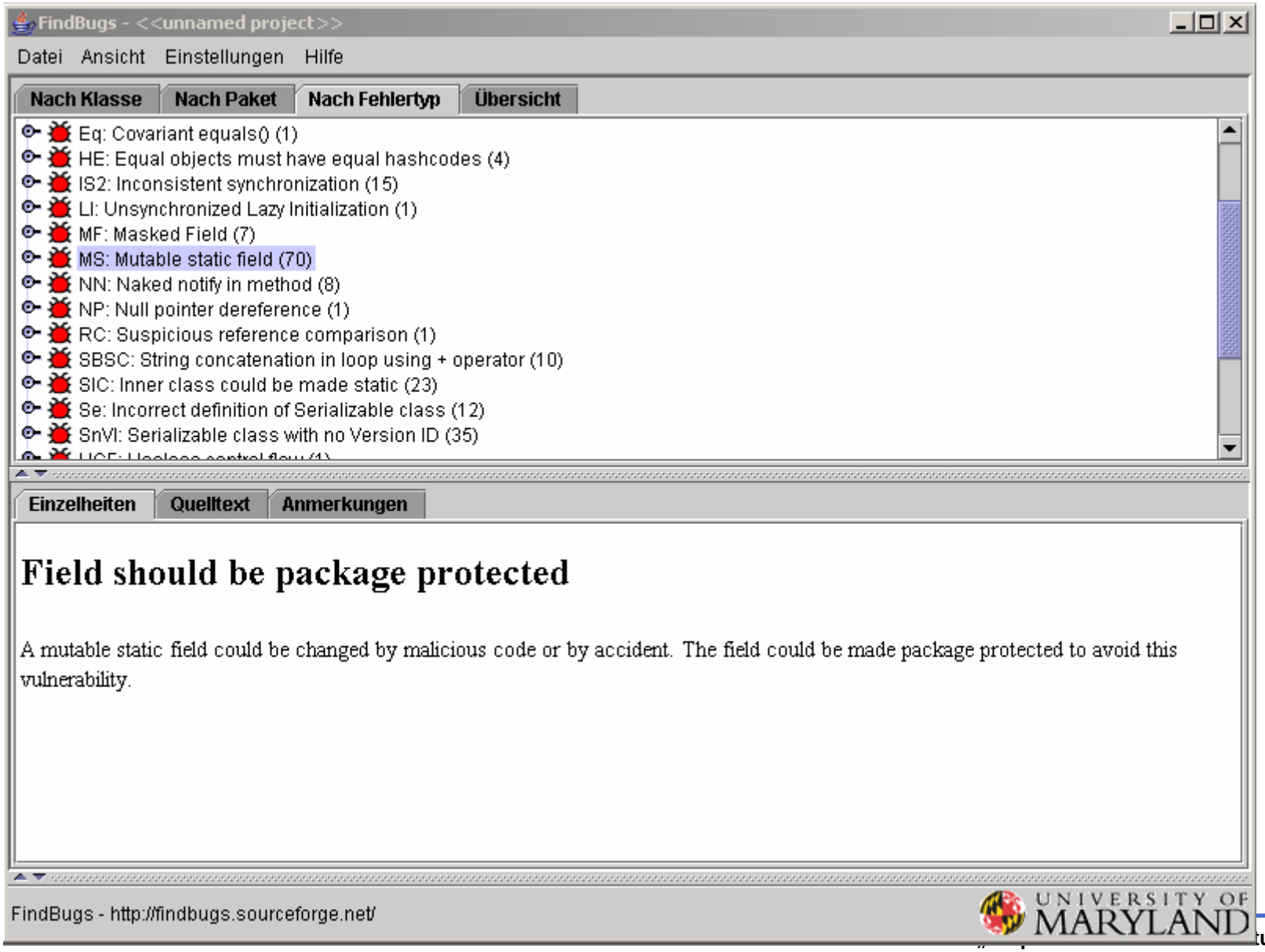
- Misplacement
- Omission and Addition
- Unexpected behaviour
- Complexity



Macro-Level: Design + Coding Guidelines

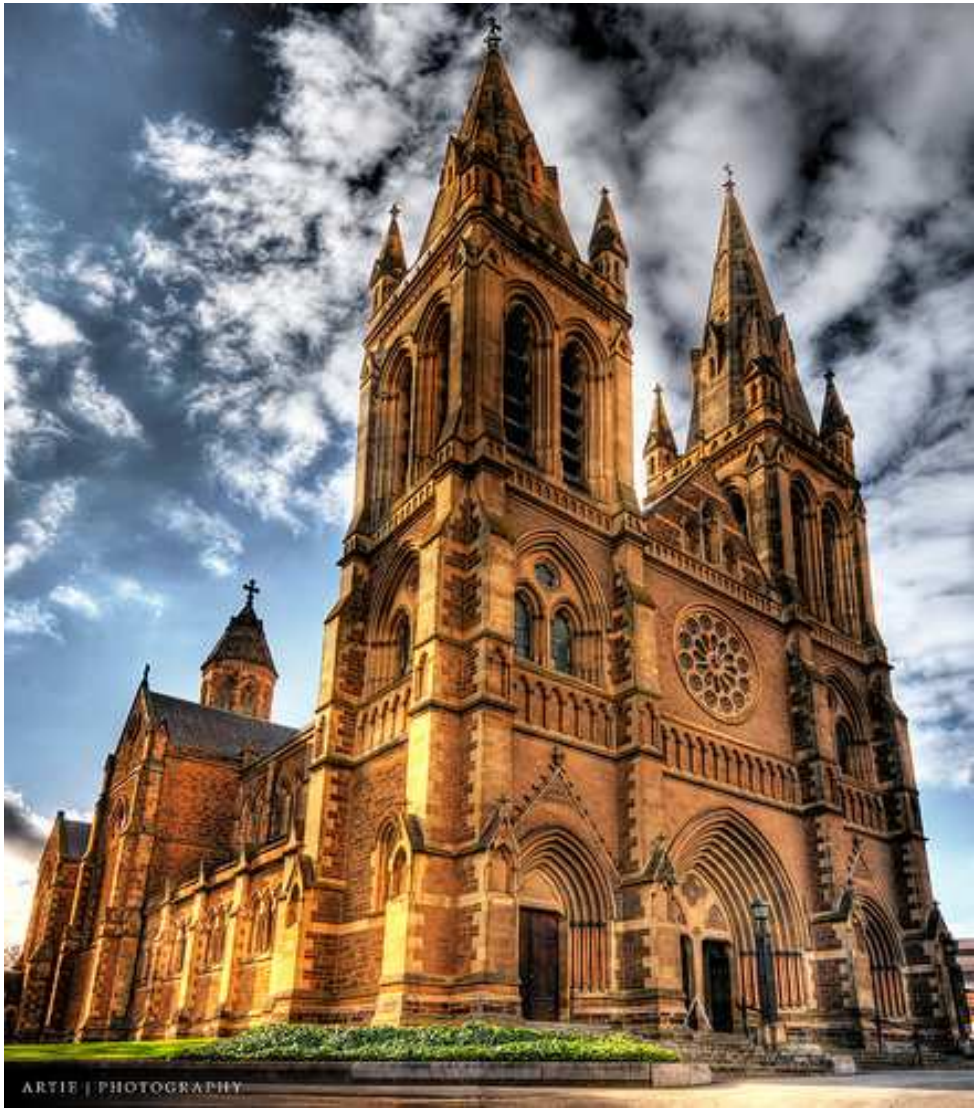


Tool Example Java: findbugs



Software, Architecture, Erosion

SICK



Software-Architecture: Definitions

IEEE 1471-2000:

- The fundamental organization of a system,
- embodied in its components,
- their relationship to each other and the environment,
- and the principles governing its design and evolution.

Kruchten: captured in two documents:

- *Software Architecture Document*
- *Software Design Guidelines*

Architecture Erosion ALWAYS happens

SICK

- **Time pressure** leads to abbreviations, which are not replaced later by clean solutions
- **Prototypes** become products
- **Hacks**. Architectural restrictions are ignored or worked around
- **Lack of understanding the "should architecture"** leading to extensions and modification which obey only micro- and macro architecture. (e.g. outsourcing)

- It is **impossible to detect manually** architectural violations in a nontrivilal software system

Architectural Erosion

“Sometimes the developers manage to maintain this purity of design through the initial development and into the first release.

More often something goes wrong.The software starts to rot like a piece of bad meat”.

Uncle Bob: “Agile Software Development”

Architecture Analysis

STATIC Analysis Approach

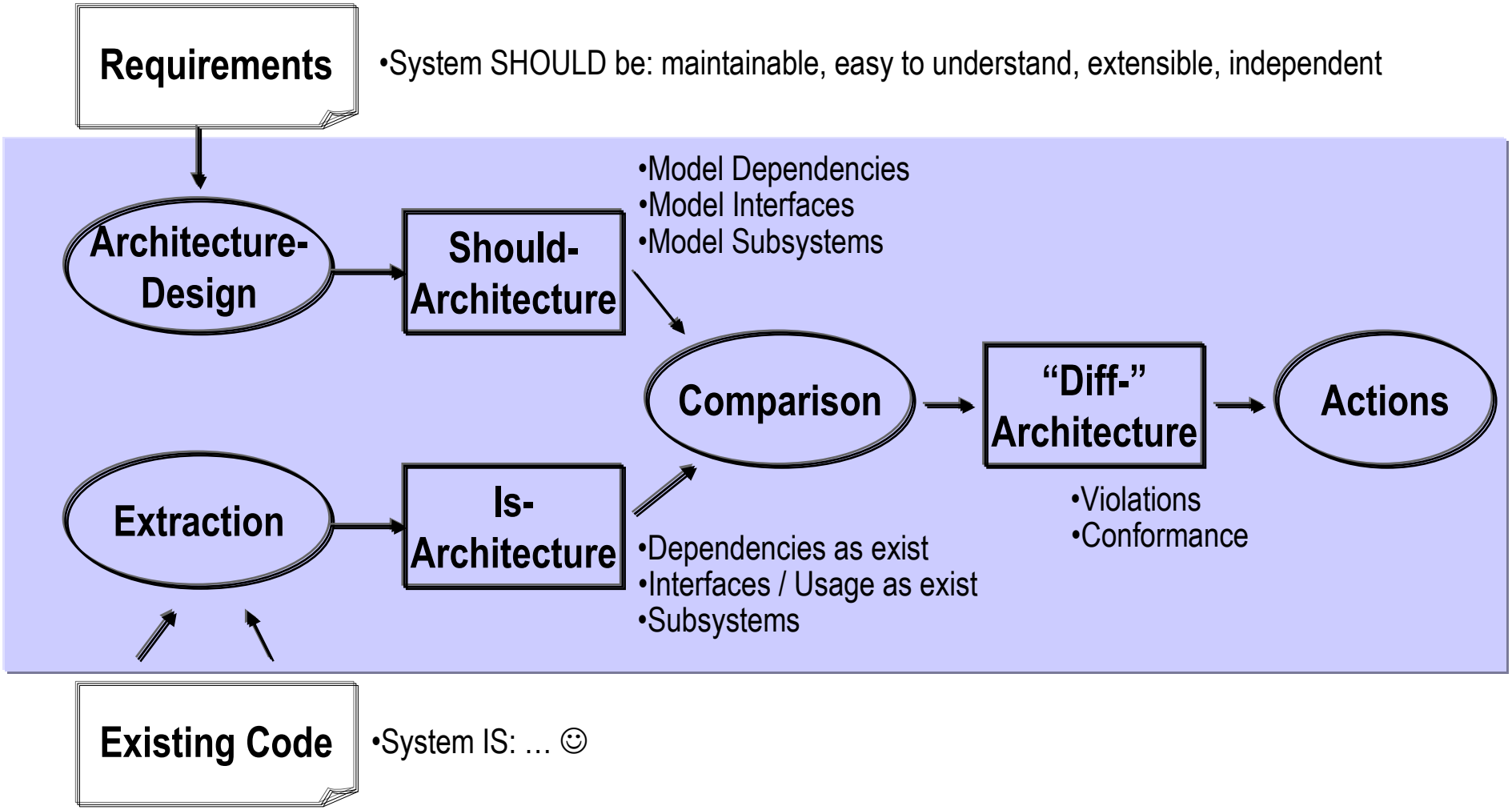
- well known (lint like)
- Reflection, LoadLibrary

With Tool support

- Pro:
  - automatic, consistent,
  - rule enforcement
  - Bad smell detection
  - Metrics to measure
- Cons:
  - no external Quality
  - Amount of messages

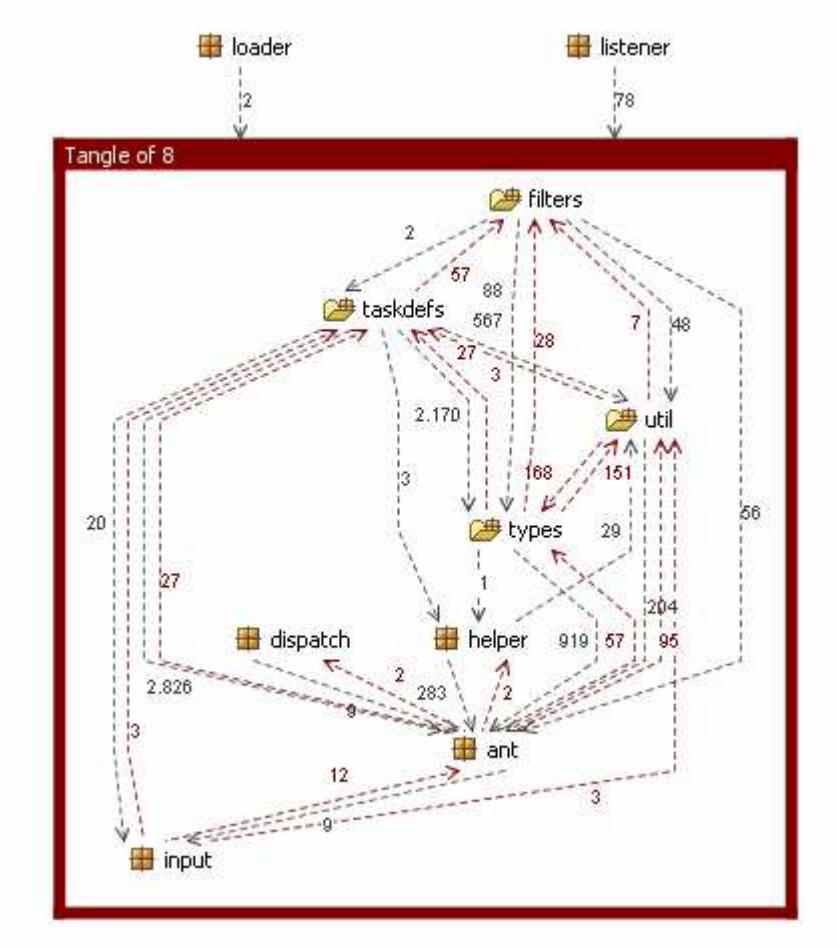
- The Pro is much stronger compared with Code-Lints !!!

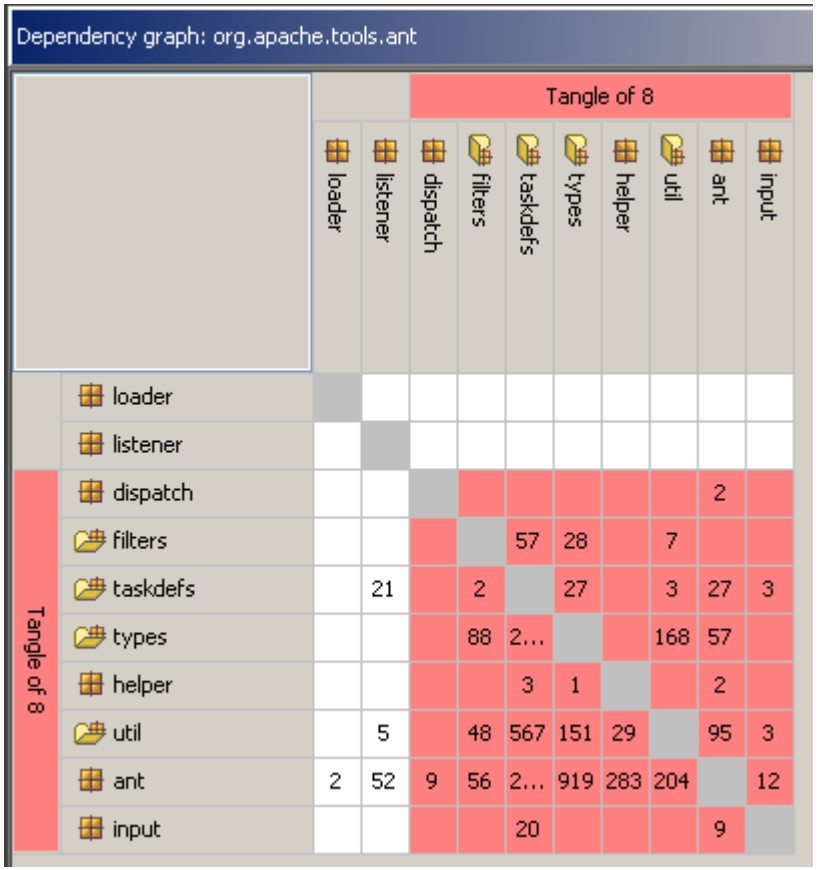
Consistency-Analysis



Displaying results:

Graphical and/or numerical

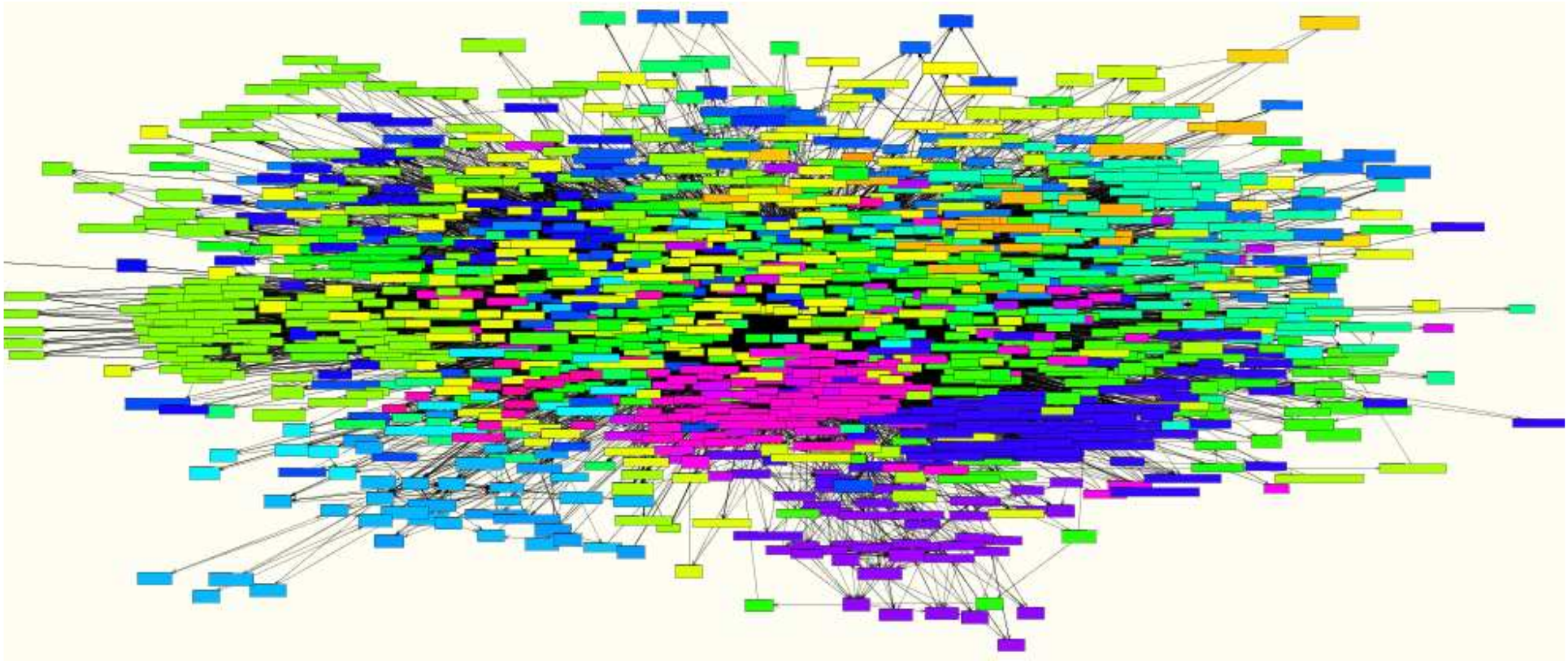




Rating of Architecture

SICK

- JDK 1.5:… 1315 classes in 229 packages all depend on each other !!!
- classes.zip, rt.jar (BIG BALL OF … ;-)



Still managable ?

SICK

Monitoring changes, trends (QA)

SICK

Level Subsystem, Package, File, Class, Operation etc.

- New artefacts
- New dependencies
- New Architecture violations

Early, betimes correction of viloations

Monitoring

- Trendreports
- “outsourcing” projects

Tools for Architecture-Analysis

Products:

- Sotograph
- Bauhaus
- Structure101

- SonarJ
- Lattix
- Klocwork K7

- Others: Semmlecode, CodeCrawler, SeeSoft, XRadar,…

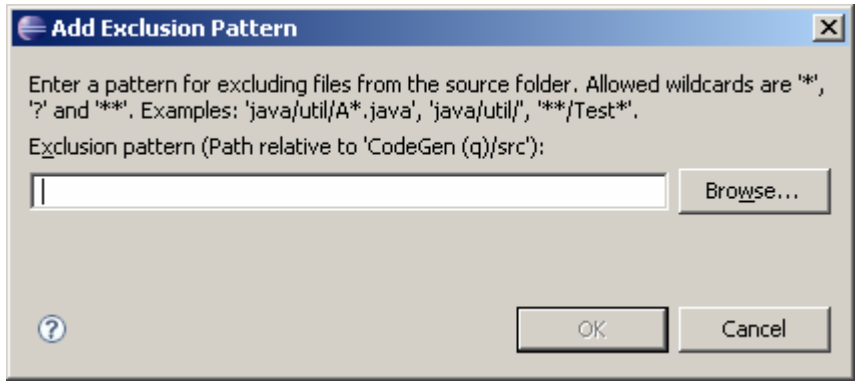
Podcast: http://se-radio.net/podcast/2008-10/episode-115-architecture-analysis

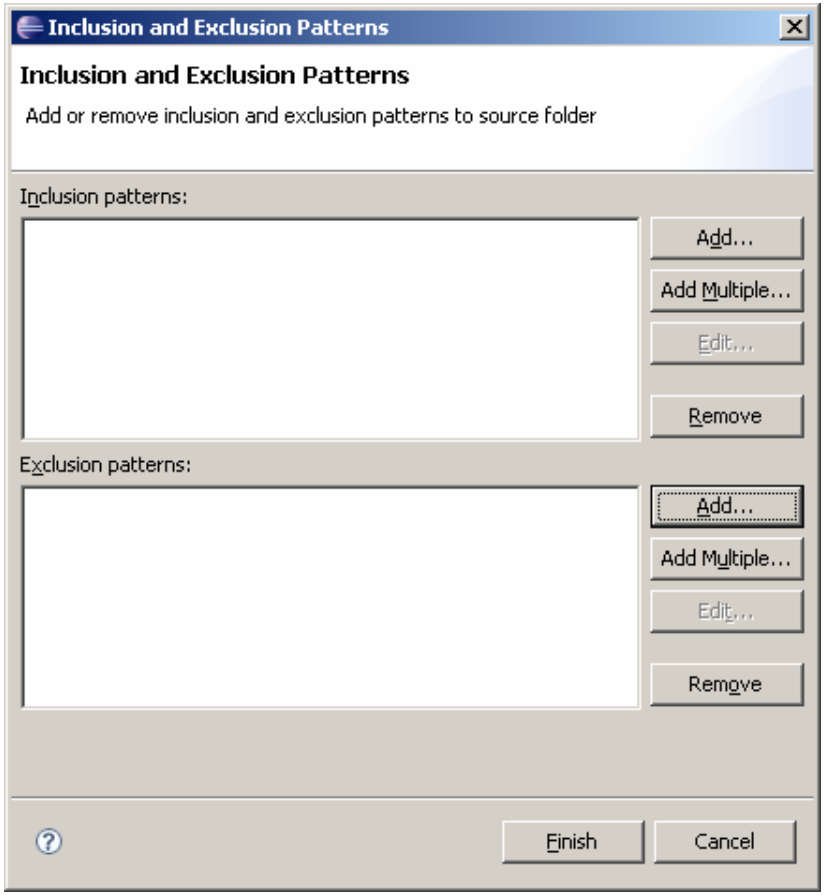
Basic Approaches

SICK

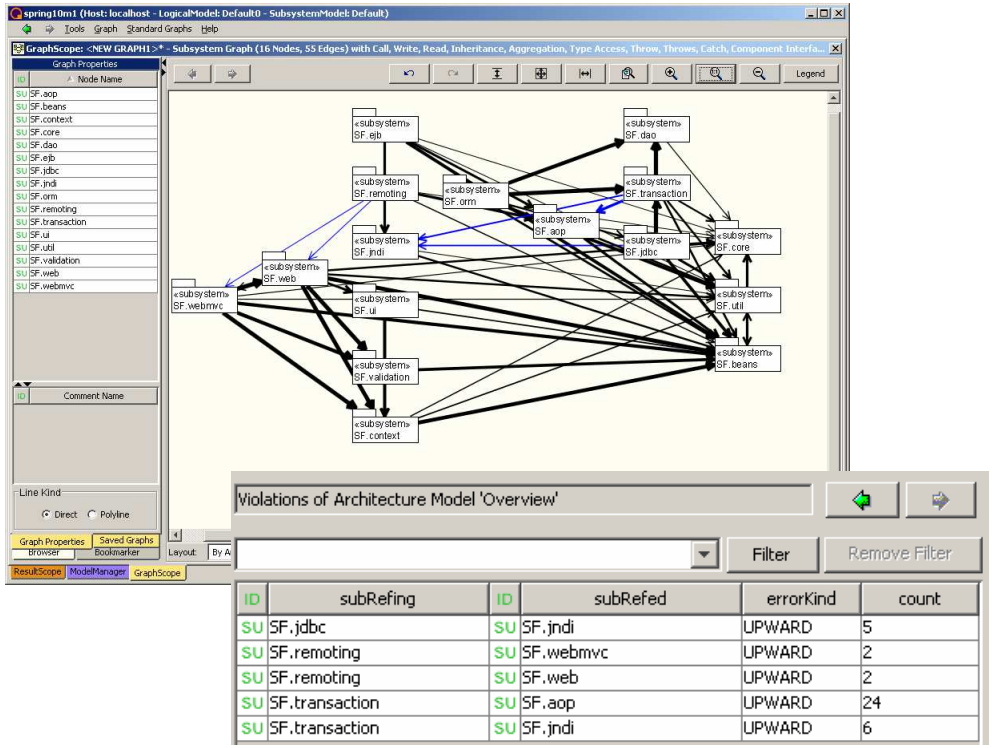
Basic approaches

- Your makesystem…
- makedepend, jdepend
- RE code into UML model
- Eclipse (Java Build Path)

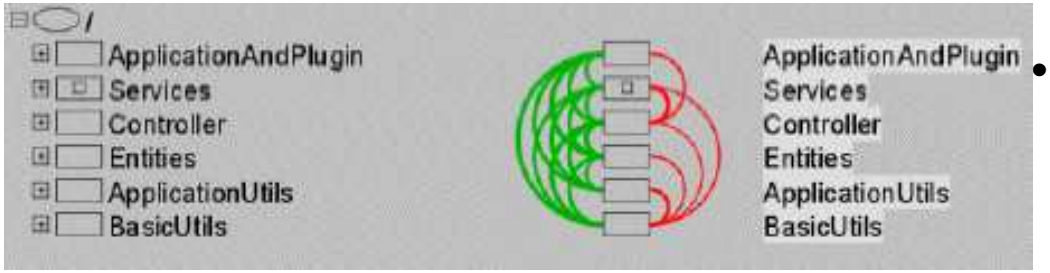




Sotograph: Overview

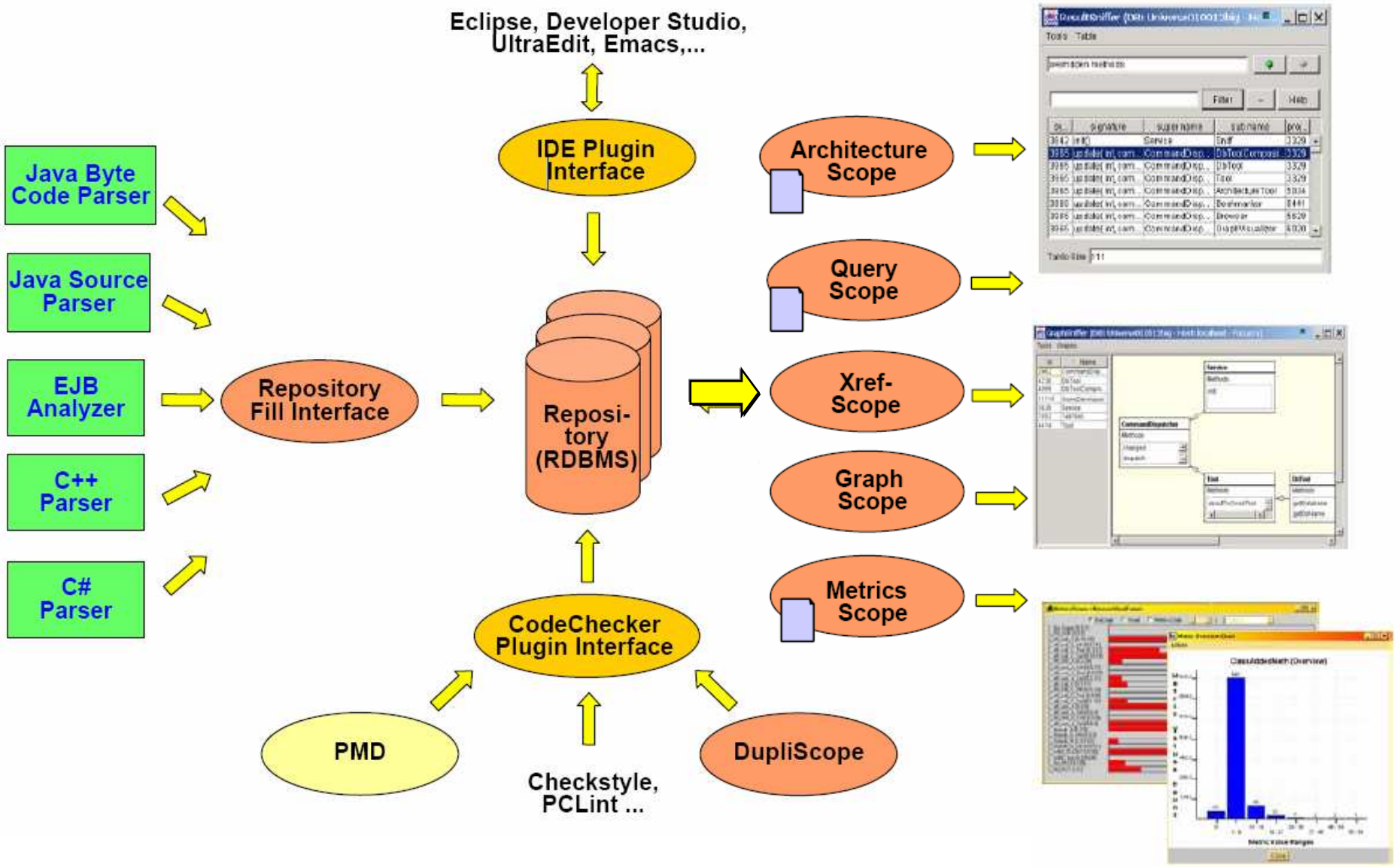


- VERY powerfull
- Infos via Table + Graph
- Strong layout algorithms
- Known since 2003
- Mysql DB, open schema
- GUI Client, Web Report
- About 200+ Metrics
- Arbitrary User queries
- Trend Analysis
- Virtual Refactoring
- Java, C++, C#, source parser

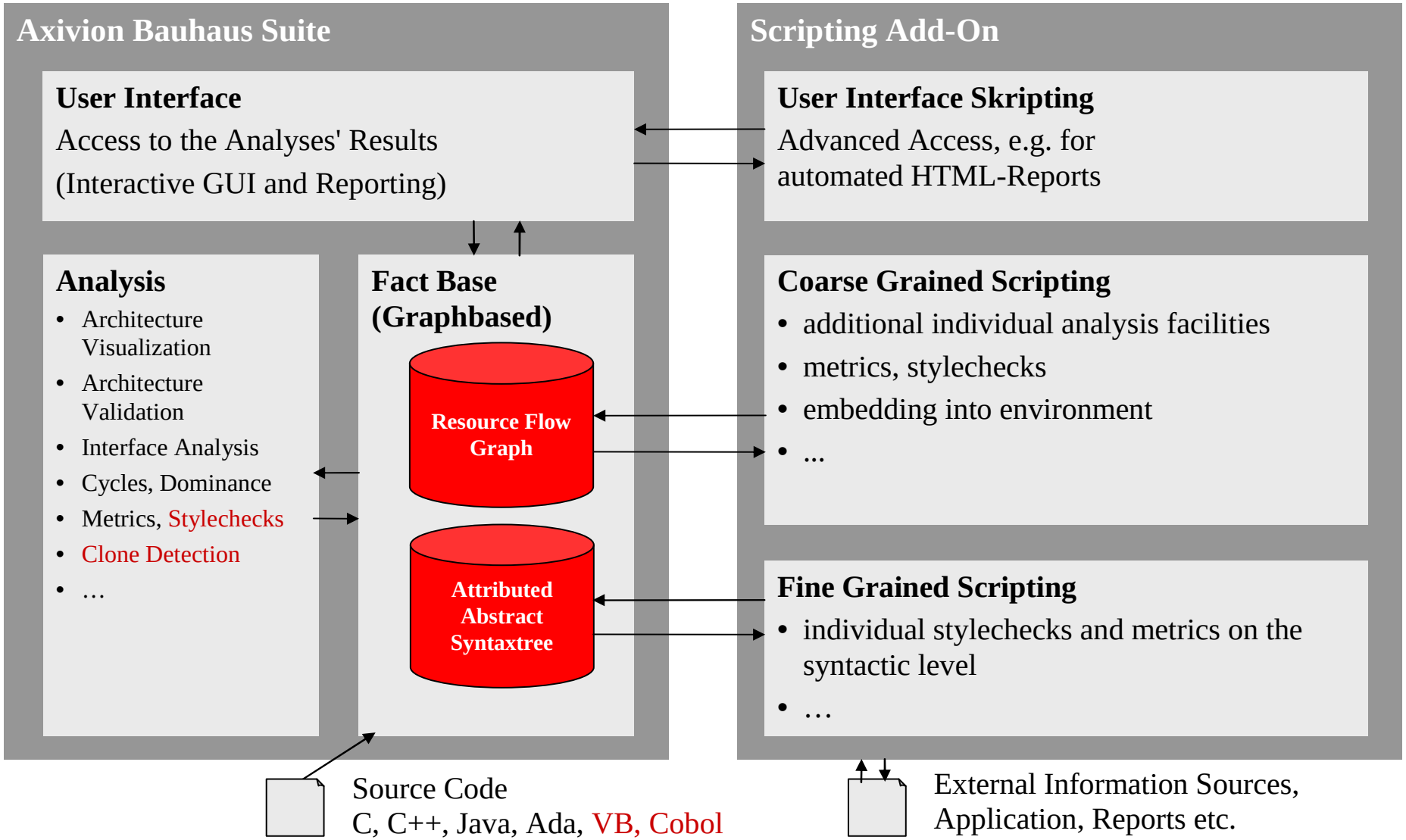


- SotoArch since 2007

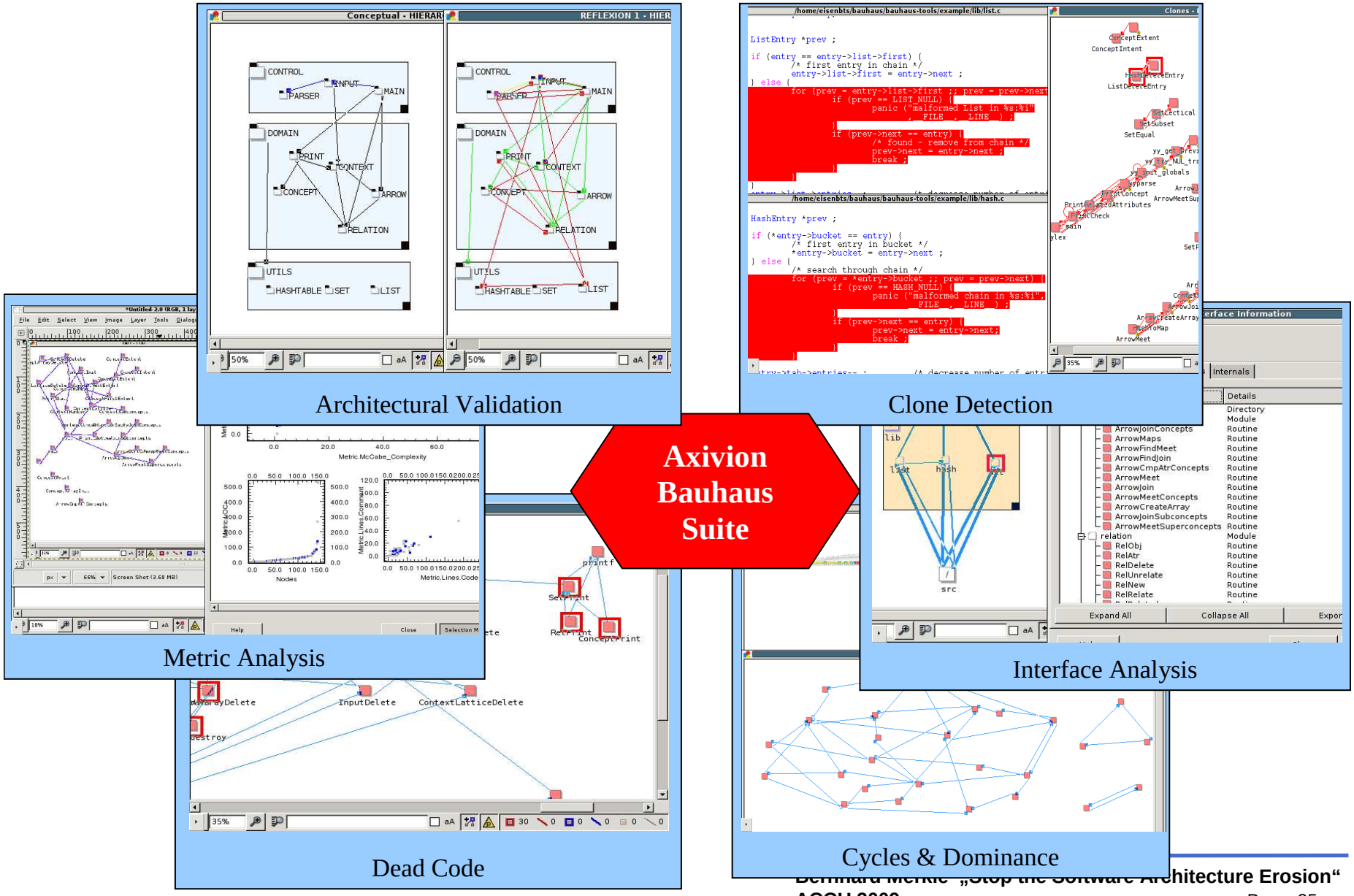
Sotograph


Eclipse, Developer Studio,
UltraEdit, Emacs,...
IDE Plugin Interface
Java Byte Code Parser
Java Source Parser
EJB Analyzer
C++ Parser
C# Parser
Repository Fill Interface
Reposi-tory (RDBMS)
Architecture Scope
Query Scope
Xref-Scope
Graph Scope
Metrics Scope
CodeChecker Plugin Interface
PMD
Checkstyle, PCLint ...
DupliScope

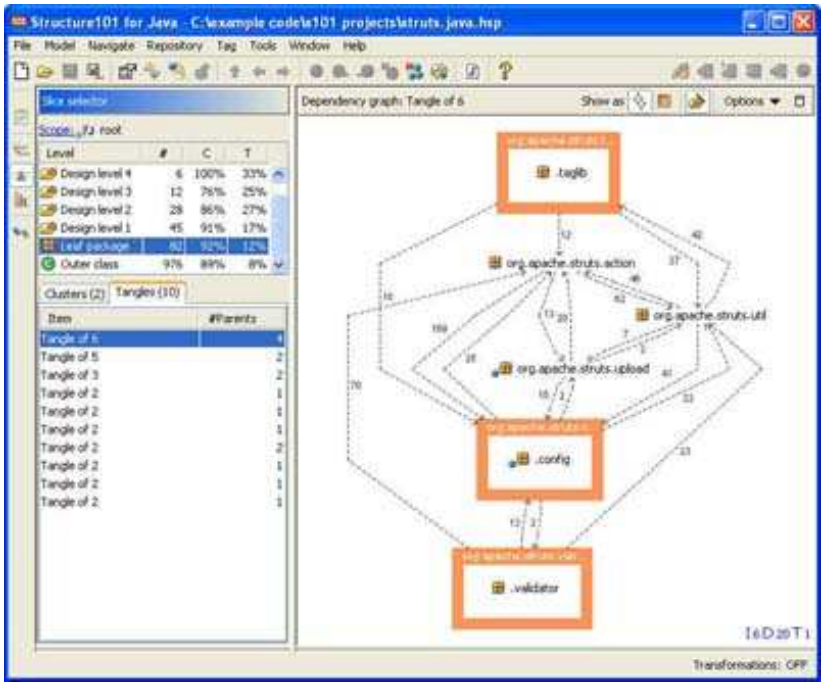
Axivion Bauhaus Suite



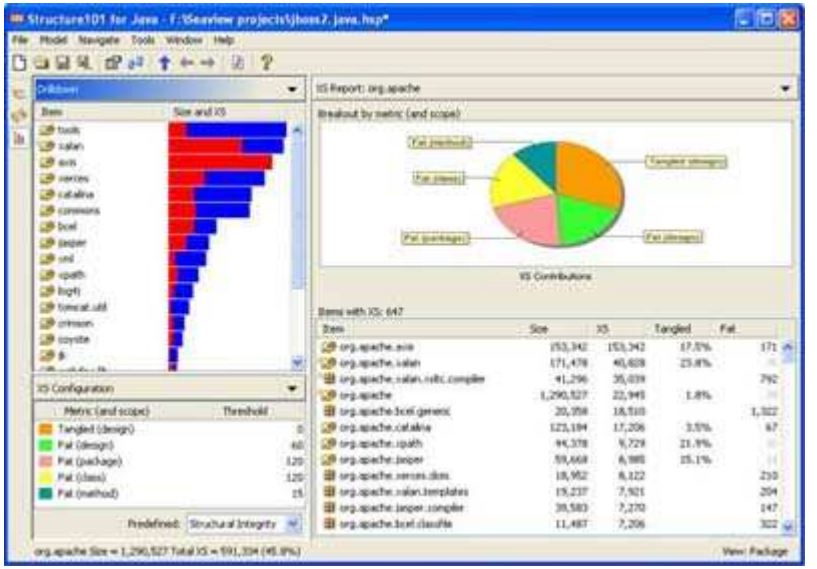
Axivion Bauhaus Suite



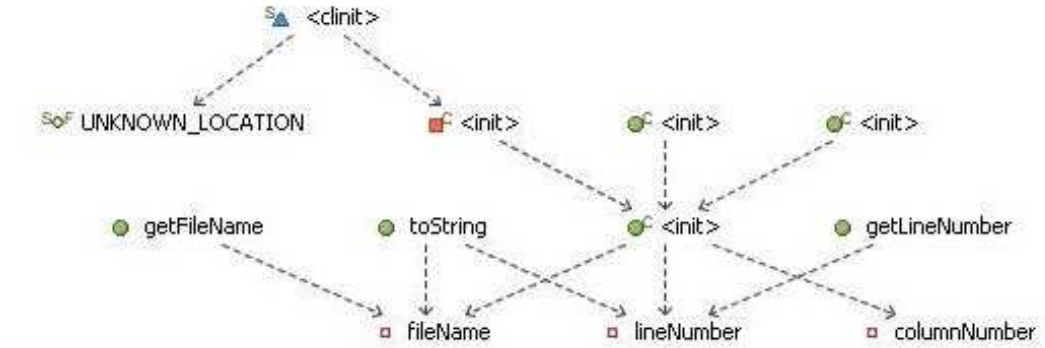
Structure101: Overview

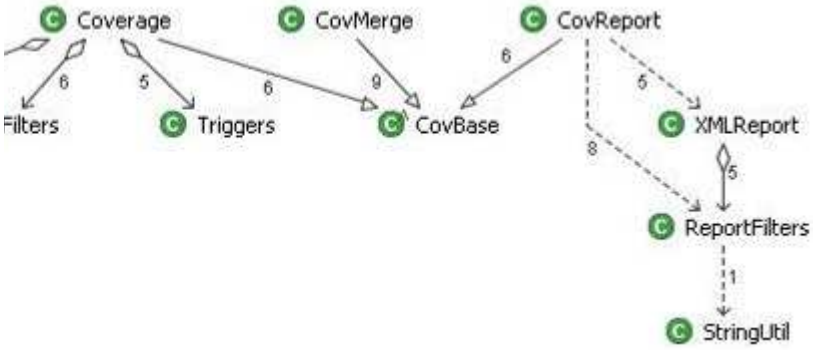


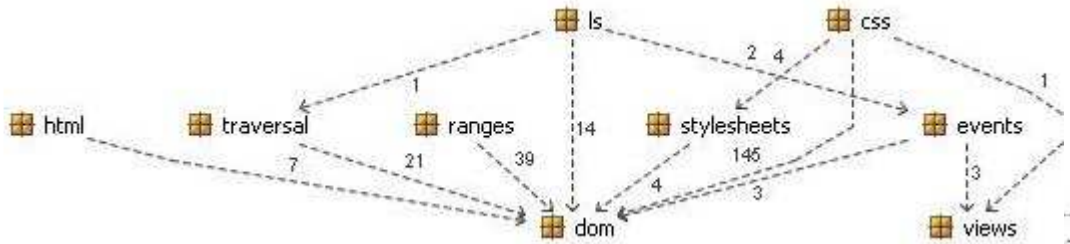
- Java, C++, Ada
- Generic version
- Infos via DSM + Graphs
- Known since 2005
- Repository/DB server
- Fat-Client, Web
- Lightweight approach

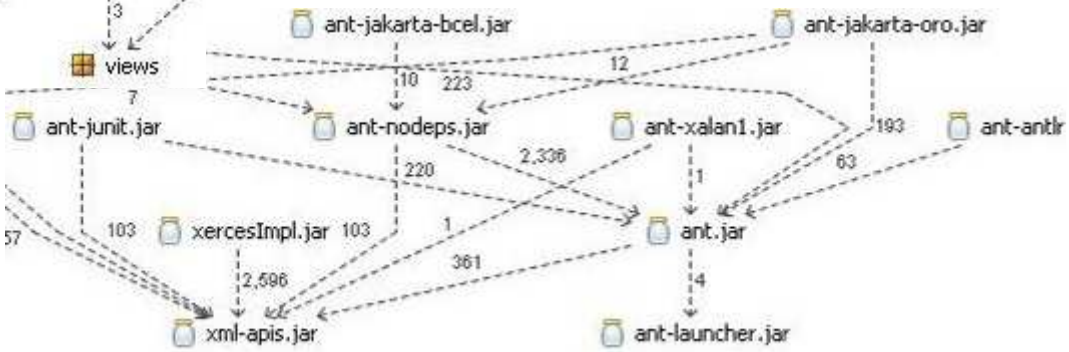


Structure101: Dependency Management









Usability

Build/analysis time

Integration in IDE, own GUI

Continuous build, commandline,

Analysis results, thrill down

Granularity (dependency types)

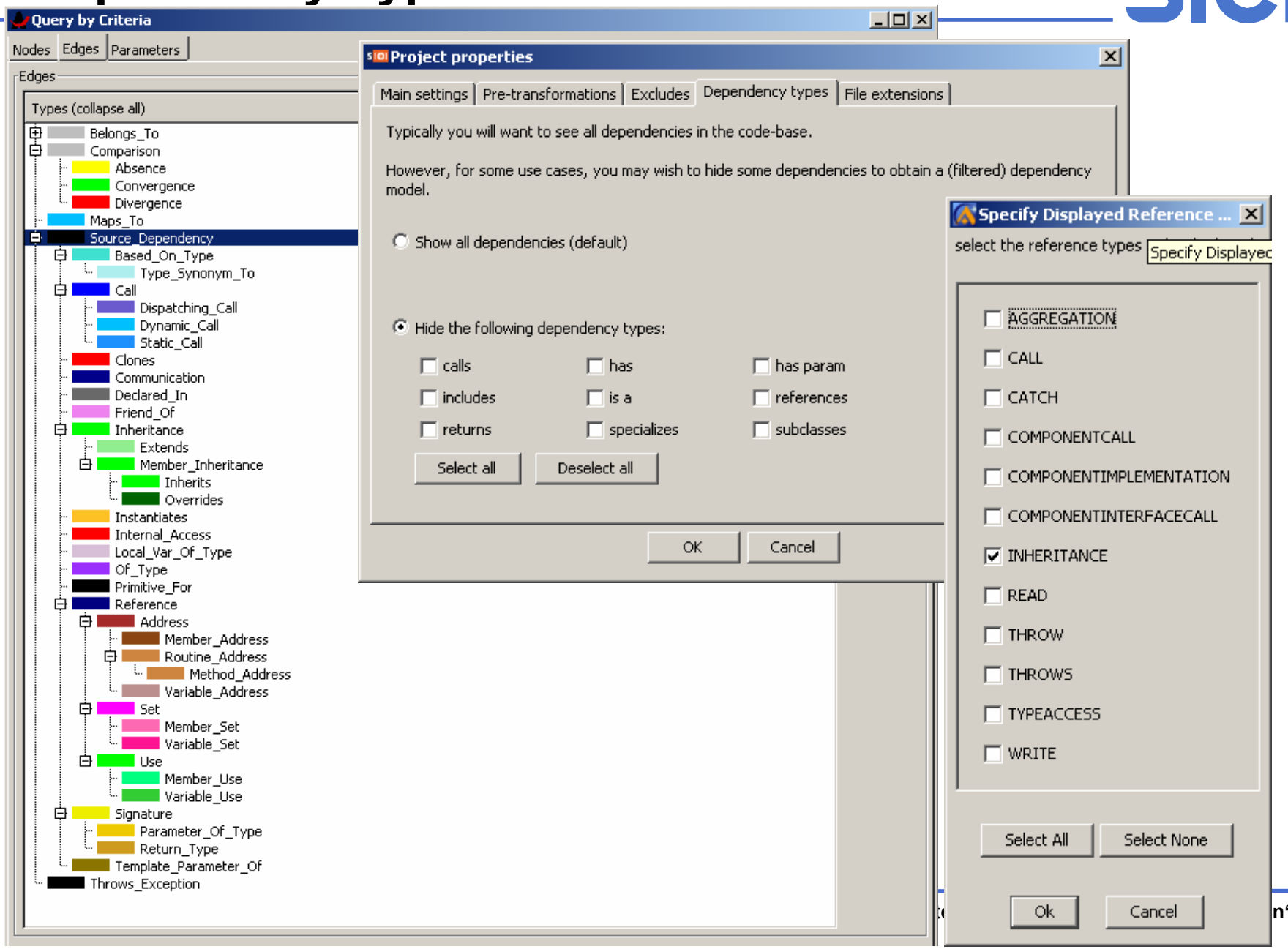
Modeling styles (API, regexp, Graphical)

Usage by developer, architect

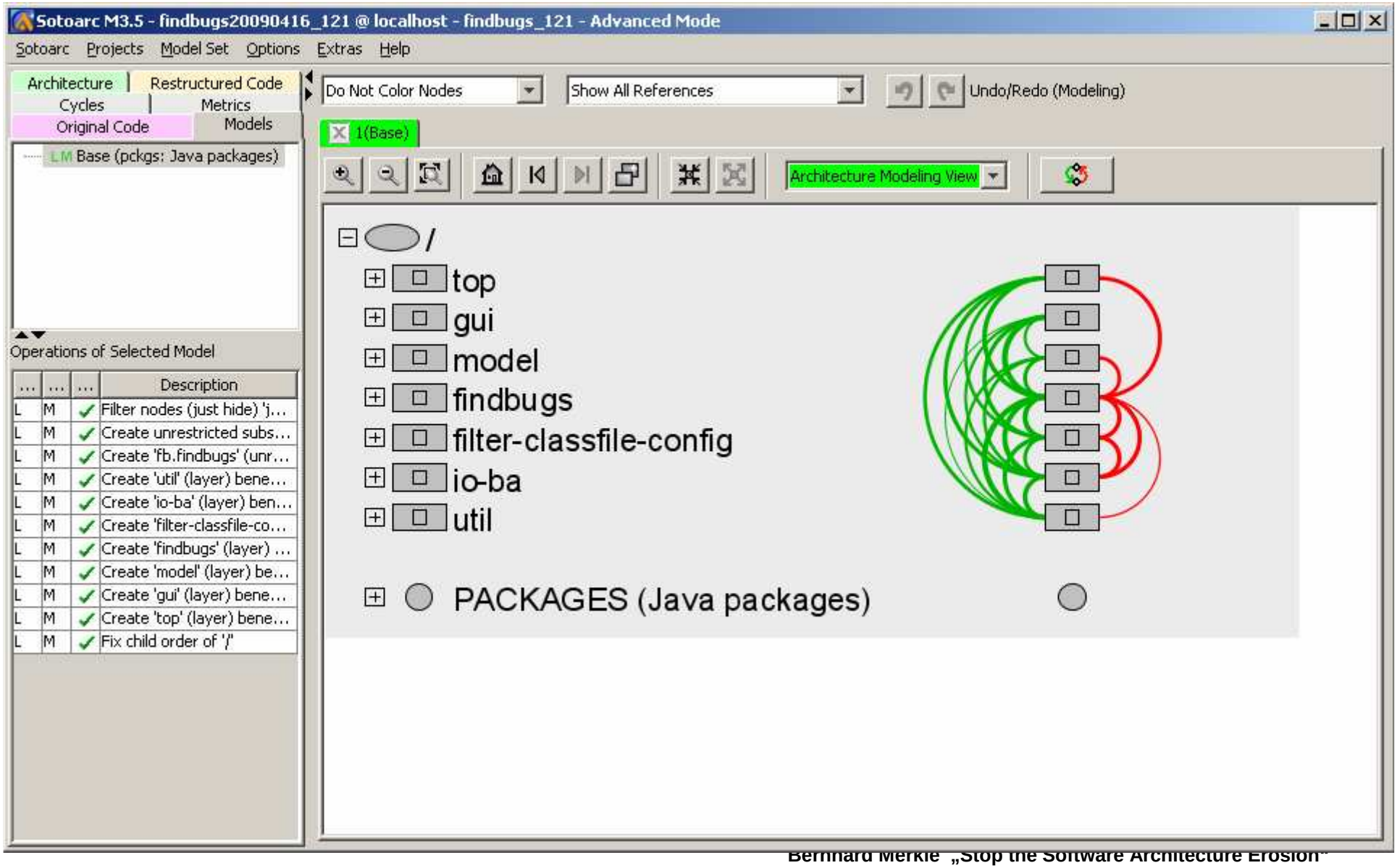
Report, Trenddata, Web-Export

Complexity, Price, Setup

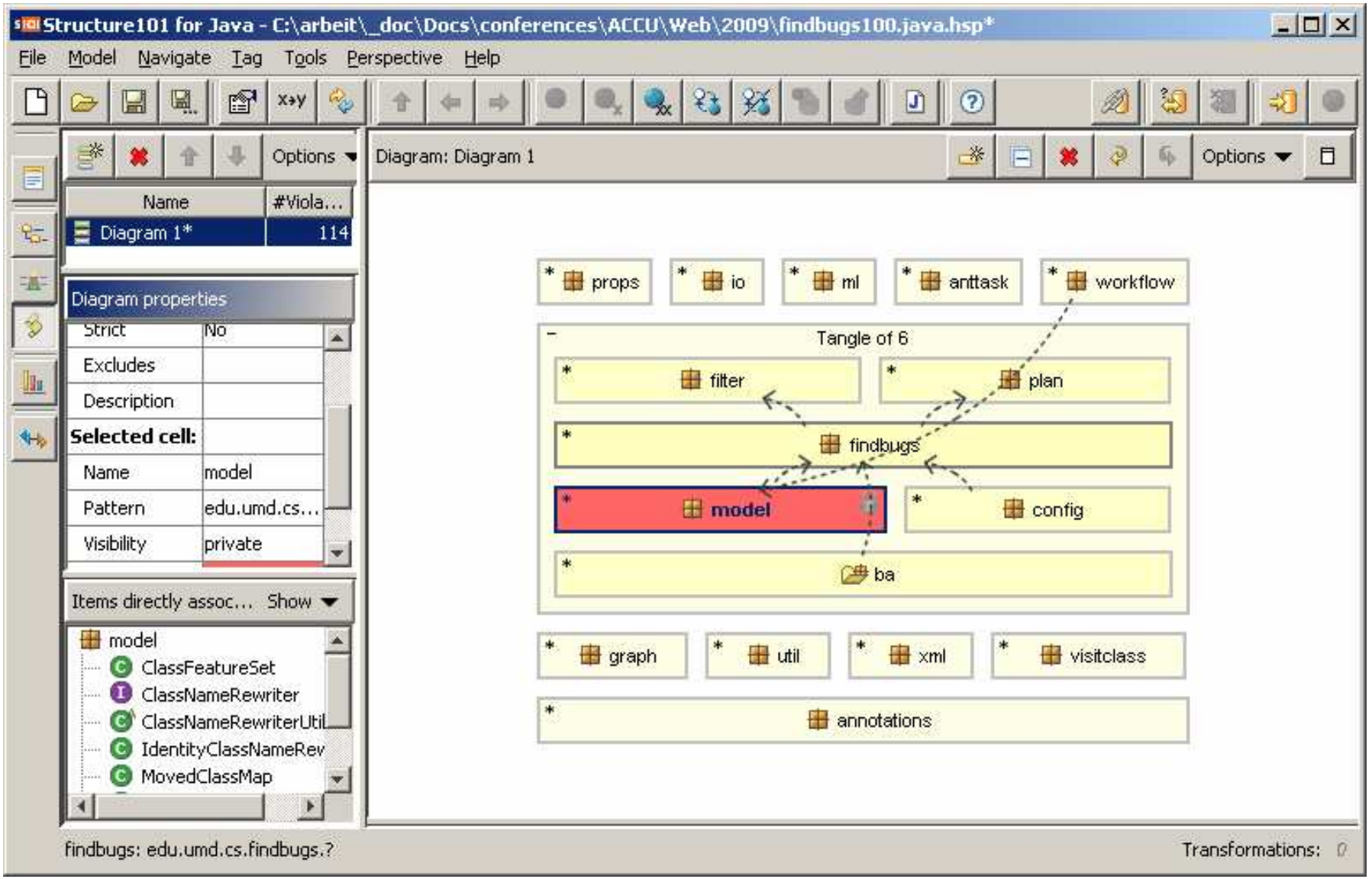
Dependency Types



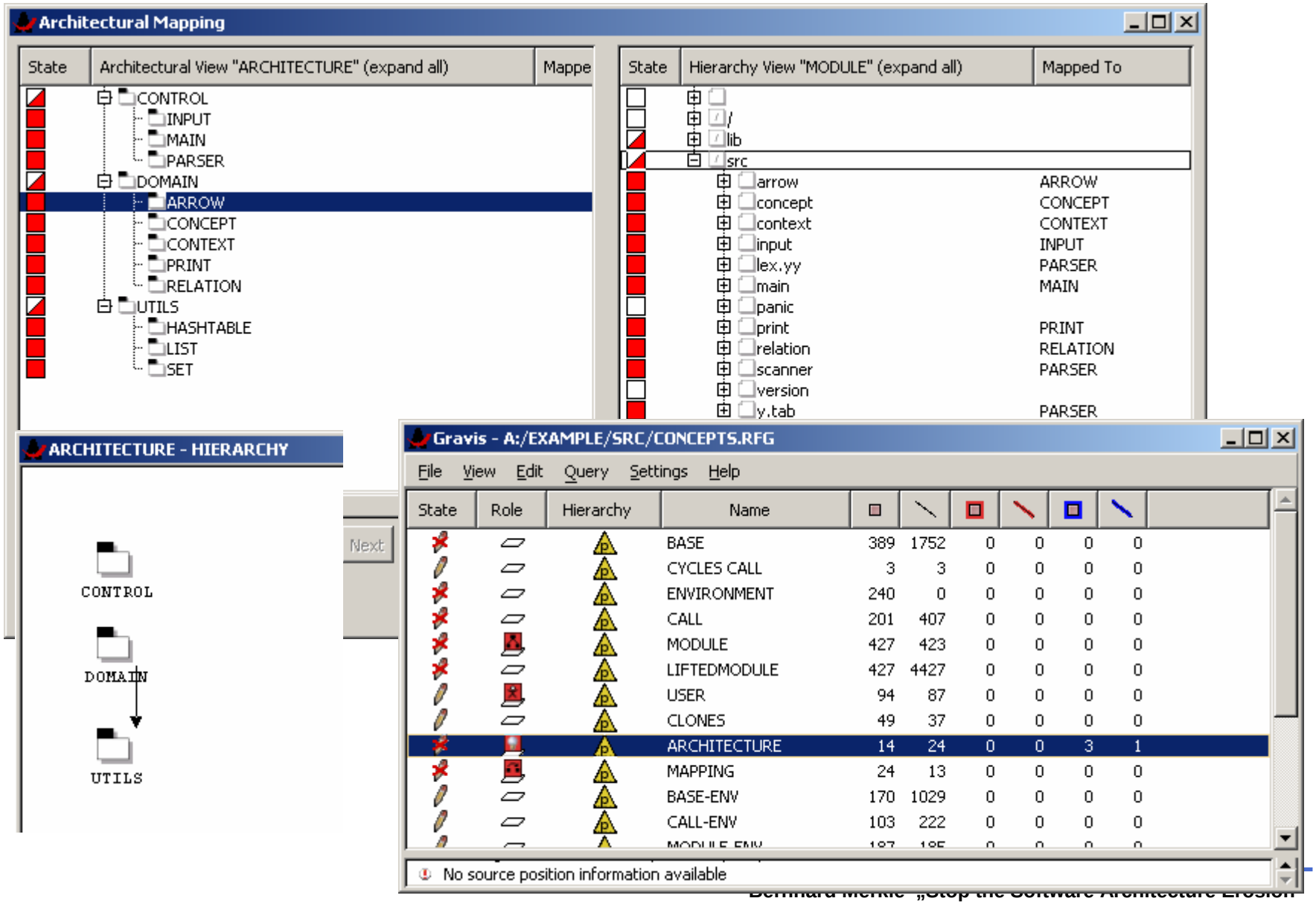
Modeling Approach: SotoArc



Modeling Approach: Structure101



Modeling Approach: Bauhaus

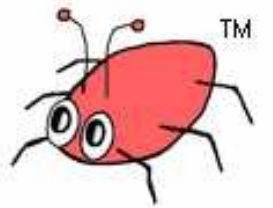


Examples for Architectural Analysis

SICK

-ant
  - easy to use make system for java

-findbugs
  - code/design level analysis tool
  - Implementation of Effective Java book (Joshua Bloch)

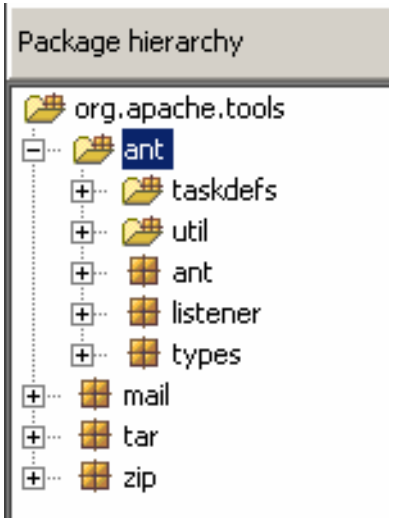


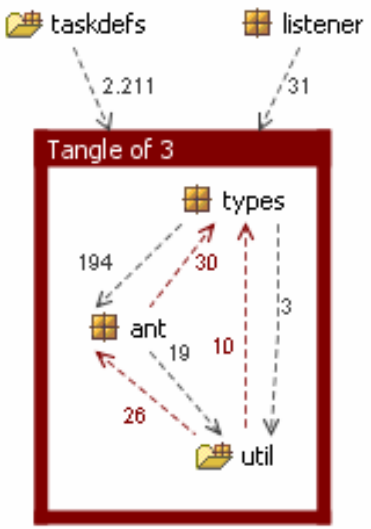
-boost
  - collection of (meta) template libraries
  - close relation to C++(0x) standardization

-poco
  - cross platform library (embedded development)

-flightgear
  - open source flight simulator

ANT 1.4.1: Architectural Analysis



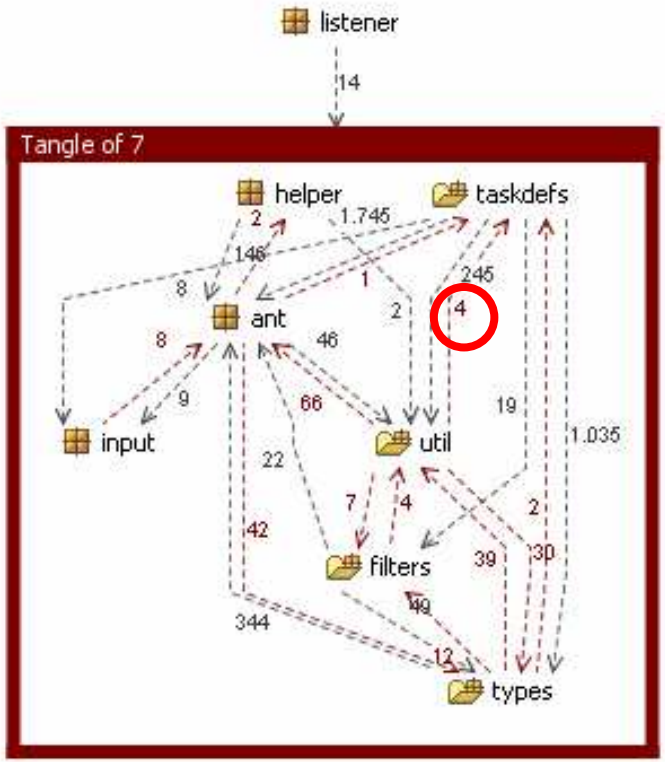


ANT 1.5.2: Architectural Analysis

SICK

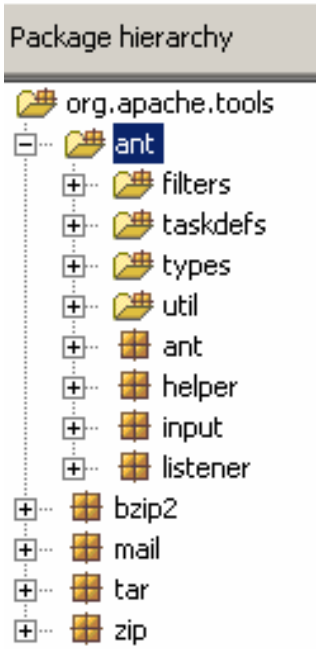
Package hierarchy

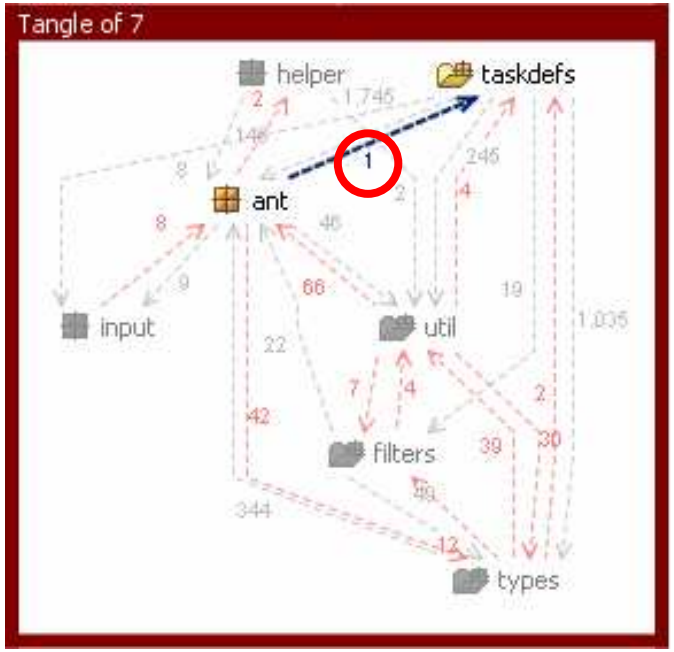
- org.apache.tools
  - ant
    - filters
    - taskdefs
    - types
    - util
    - ant
    - helper
    - input
    - listener
  - bzip2
  - mail
  - tar
  - zip

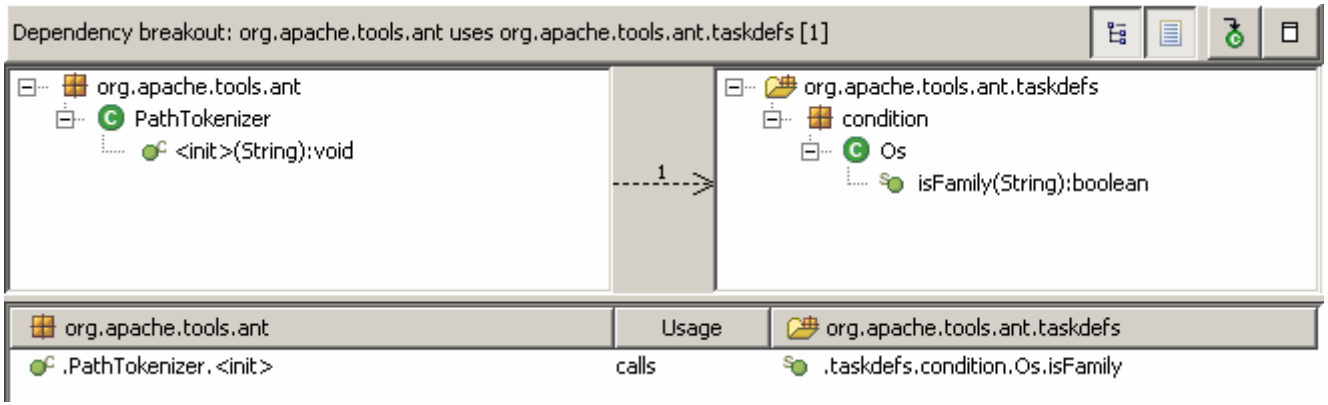


<APACHE ANT>

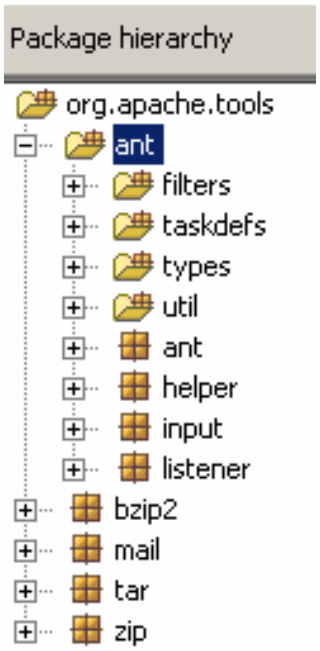
ANT 1.5.2: Architectural Analysis

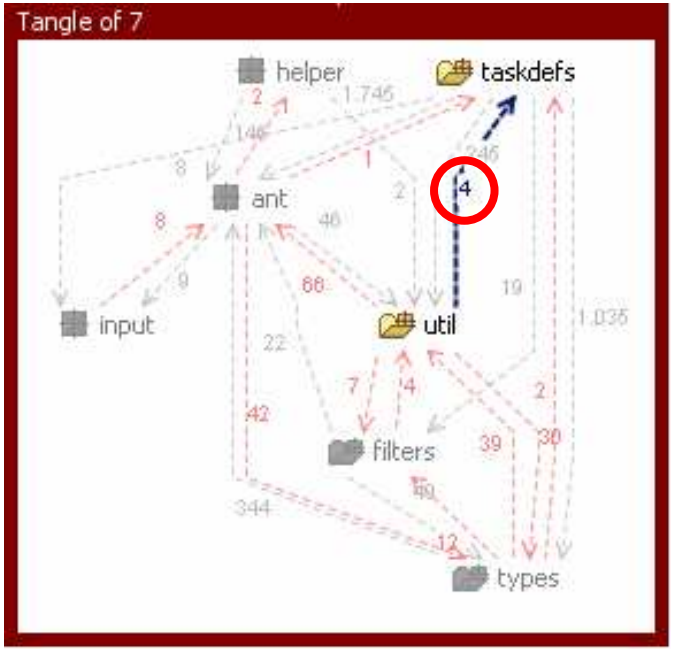


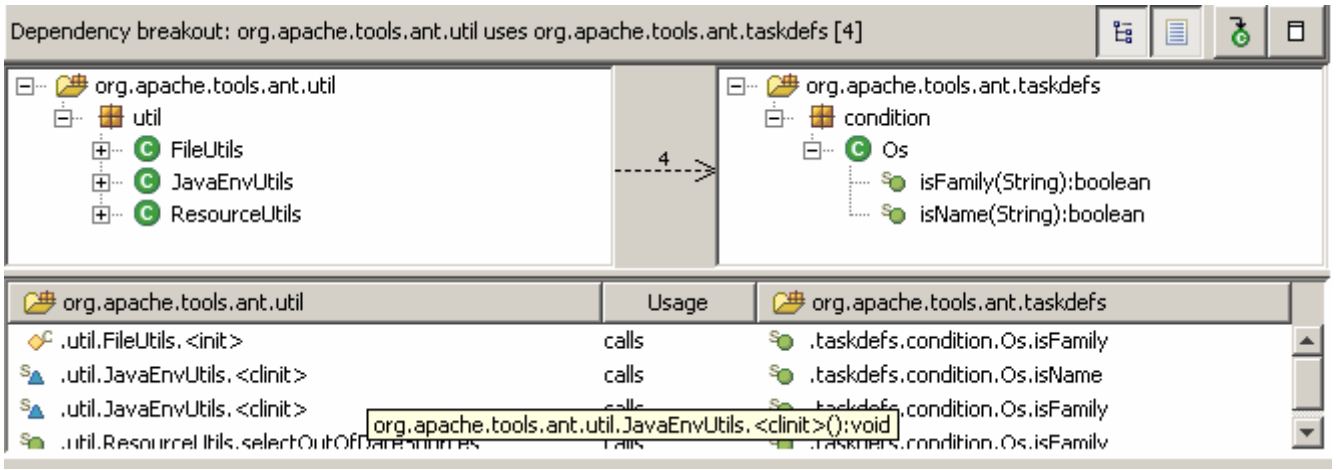




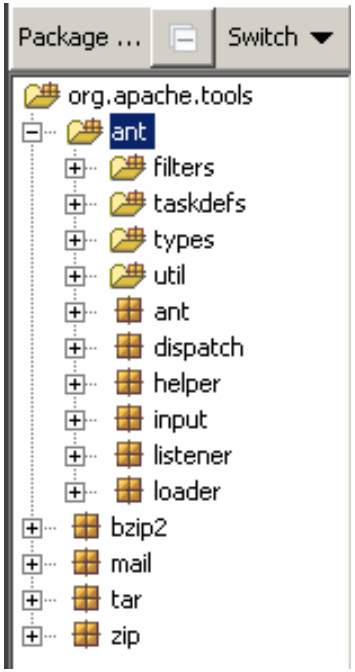
ANT 1.5.2: Architectural Analysis

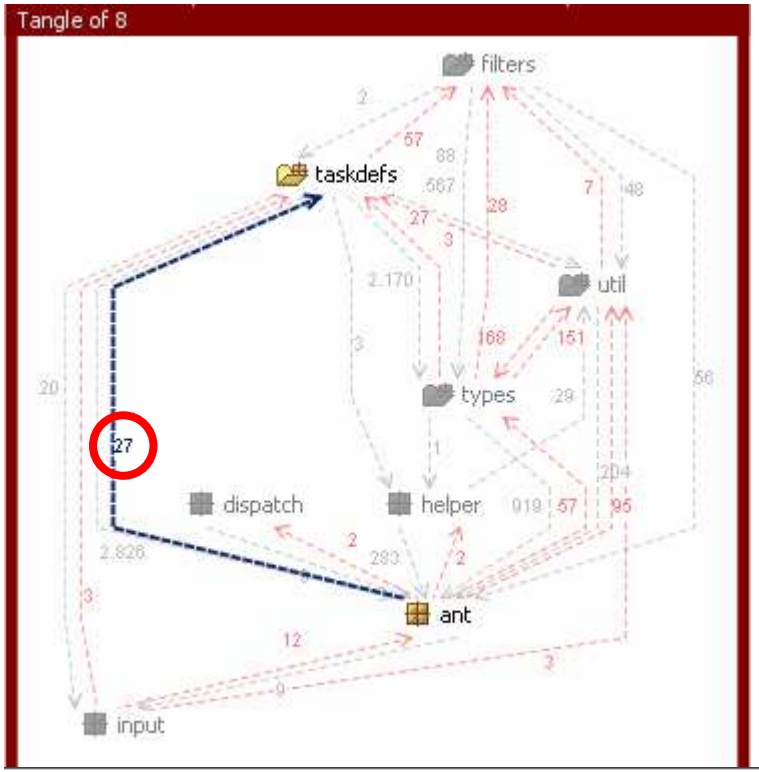


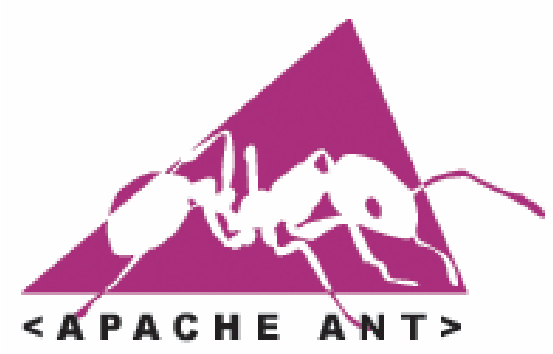


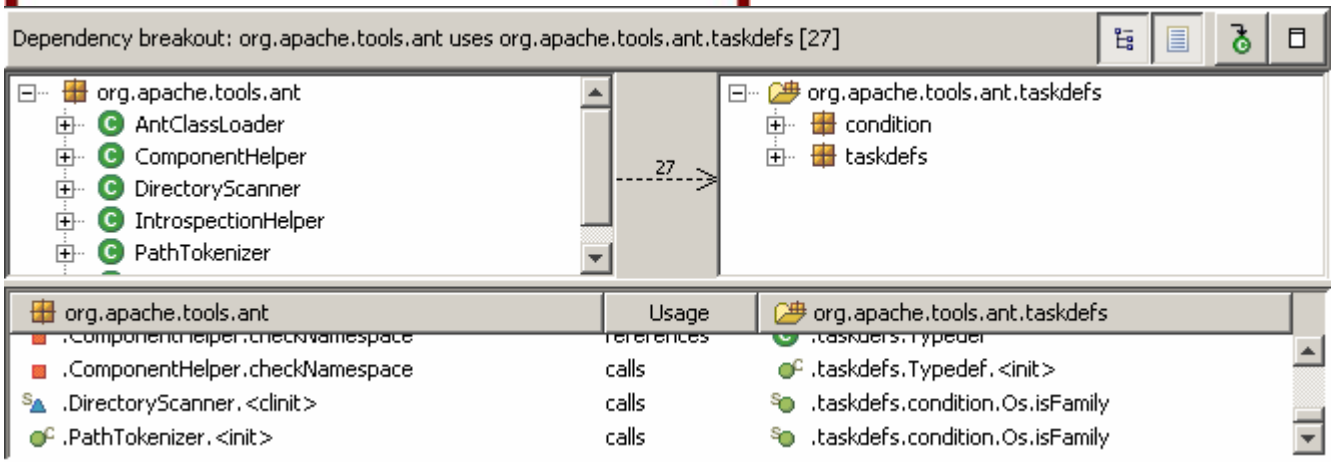


ANT 1.7.1: Architectural Analysis

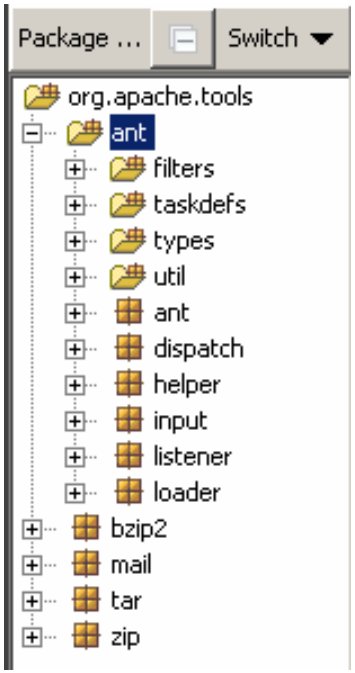


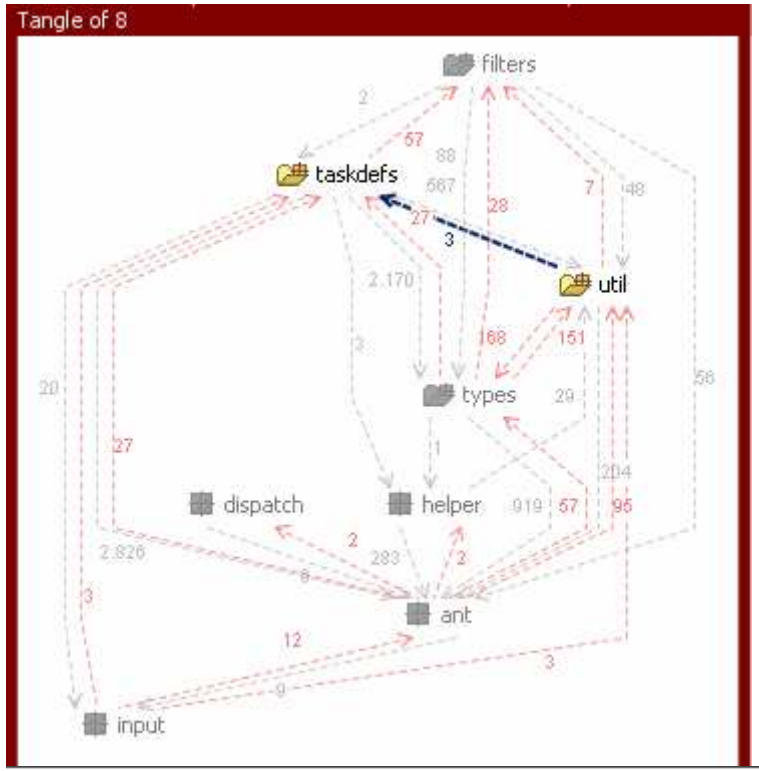


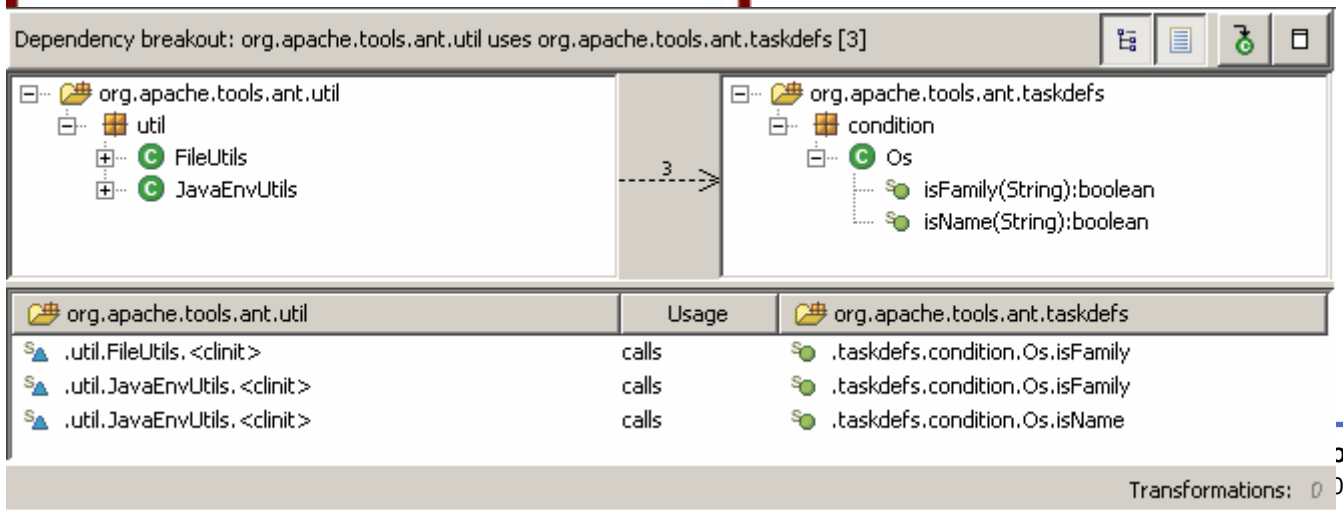




ANT 1.7.1: Architectural Analysis







Still manageable ?

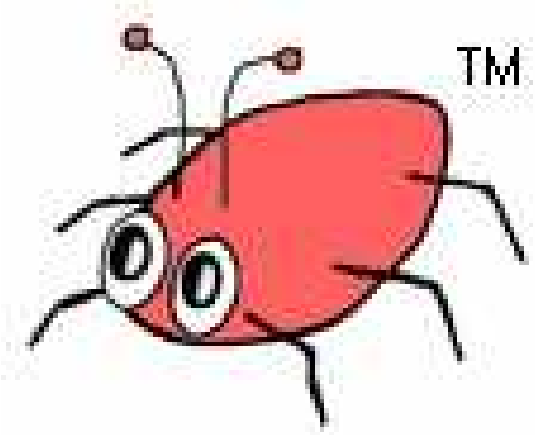
SICK

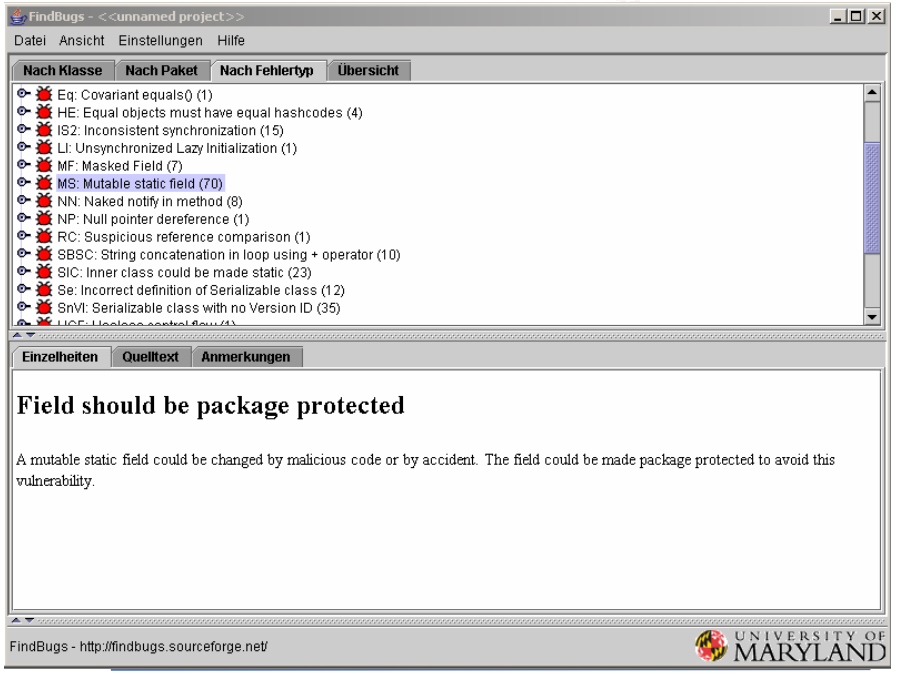
Aim of Architectural Analysis

SICK

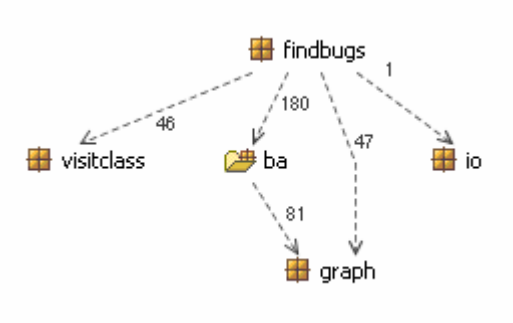
-Findbugs

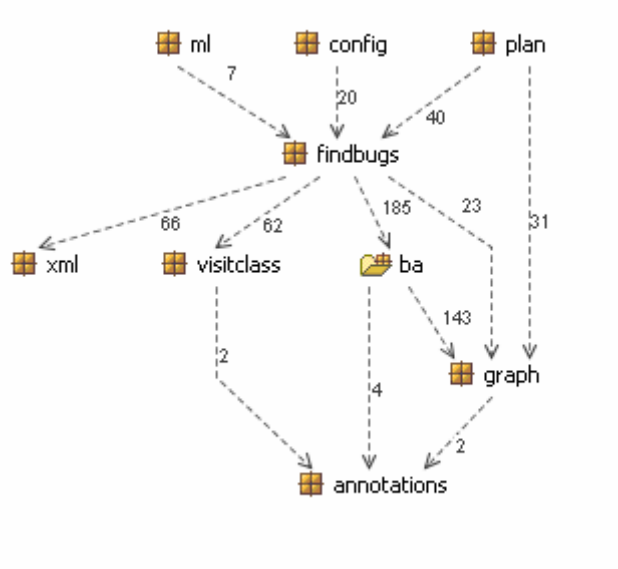
- do codelevel lints care about architecture ?
- Is there a architecture ?
- Is there a erosion ?
- Do AA-Tool work well ?





Findbugs: the first years

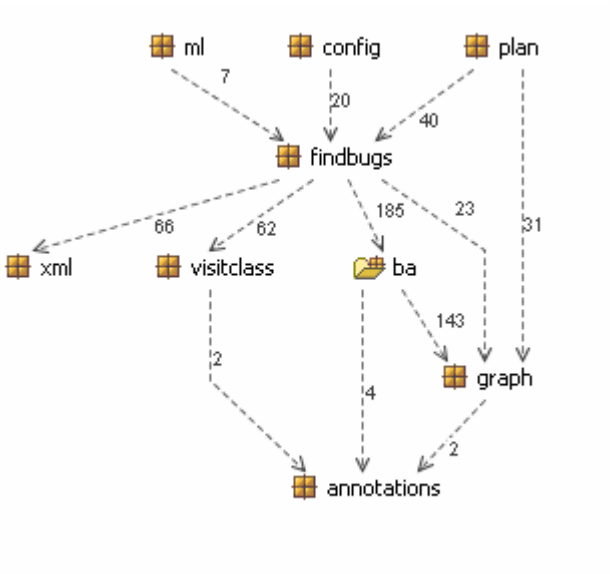




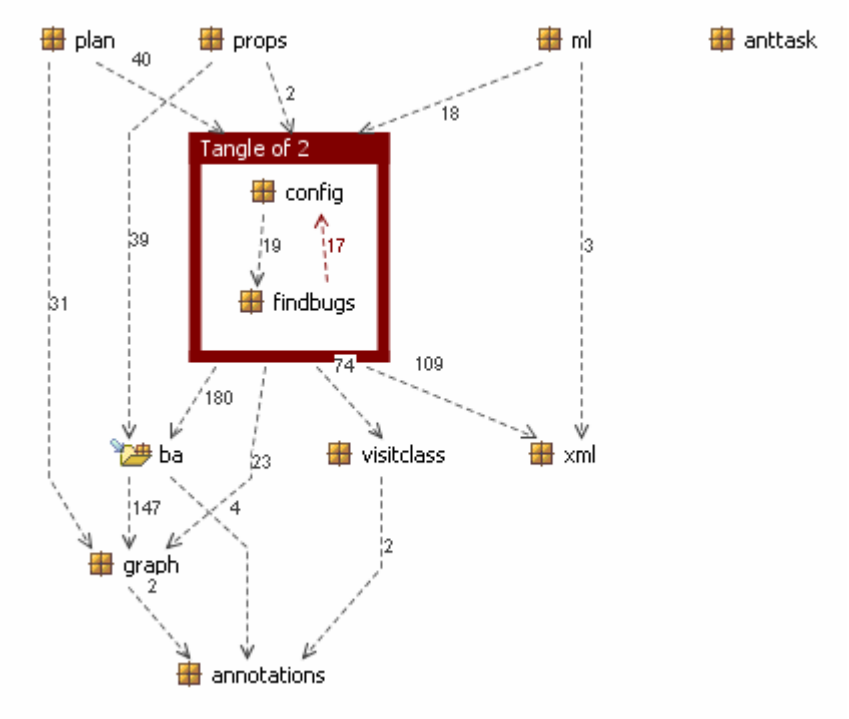
0.7.2 (03/2004)

0.8.6 (10/2004)

Findbugs: the first years

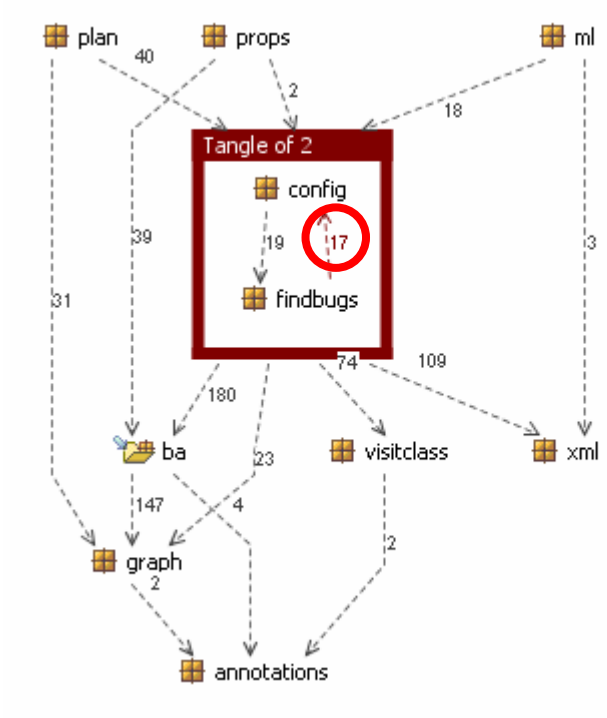


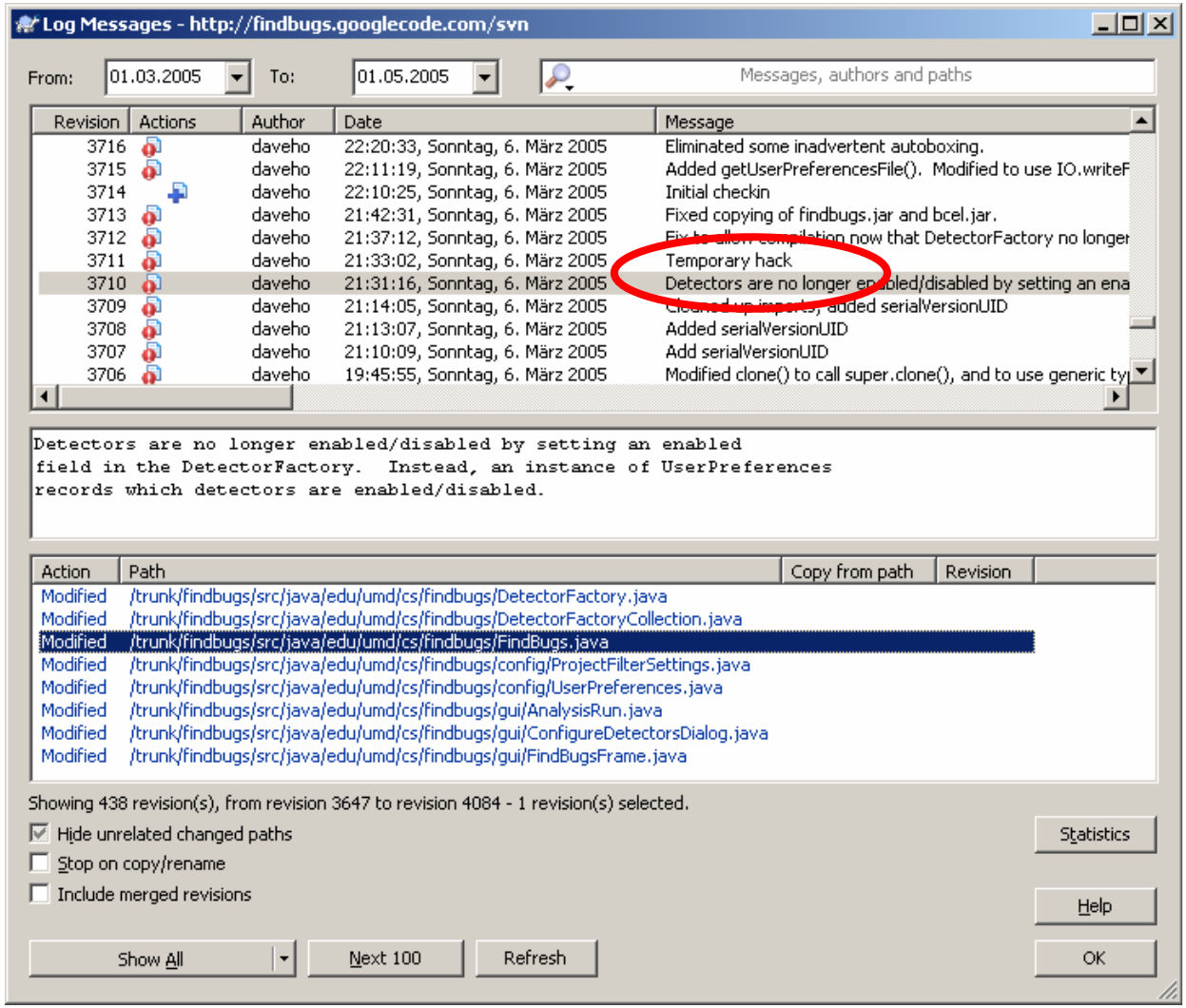
0.8.6 (10/2004)



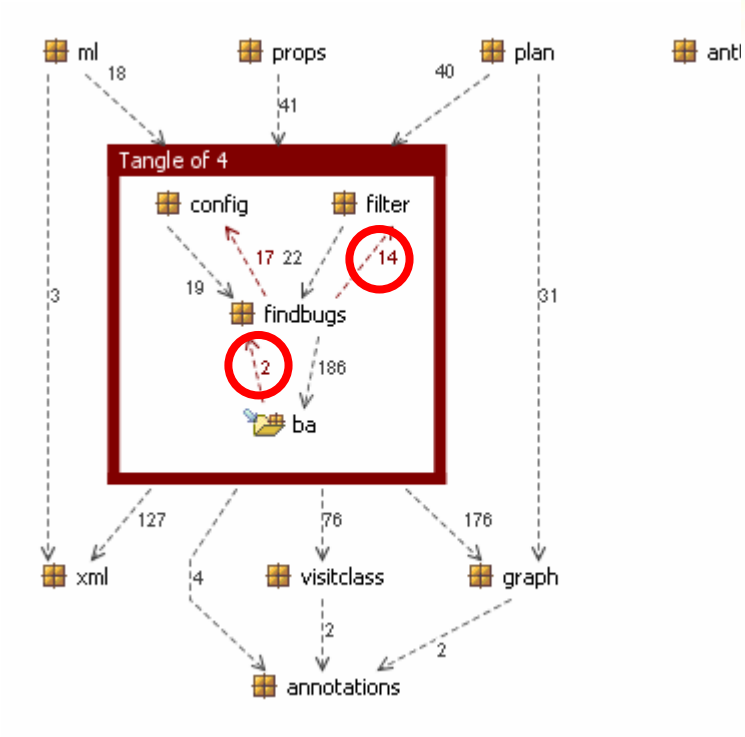
0.8.7 (05/2005)

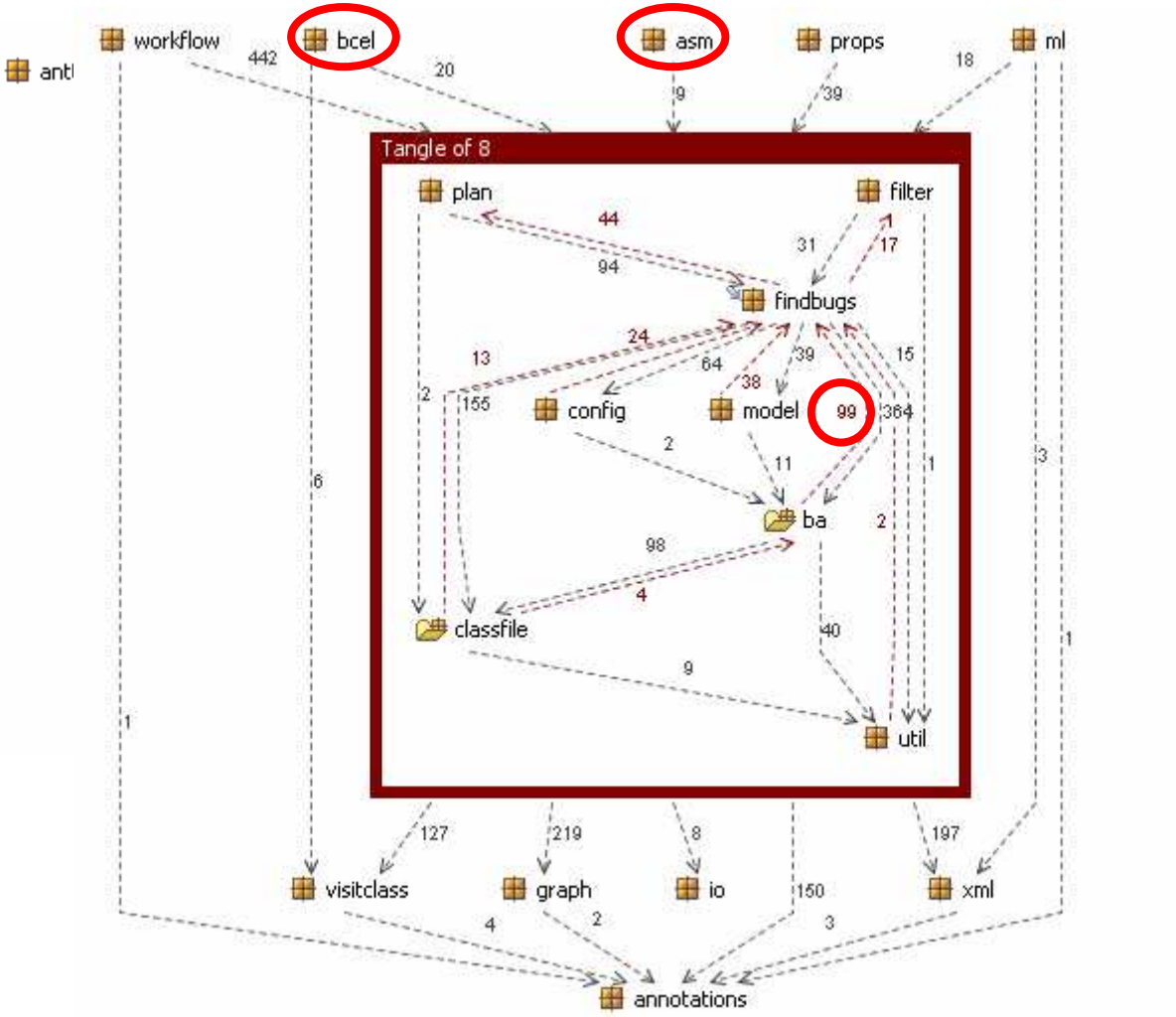
Findbugs 0.8.7: Architectural Analysis





Tangle increase…

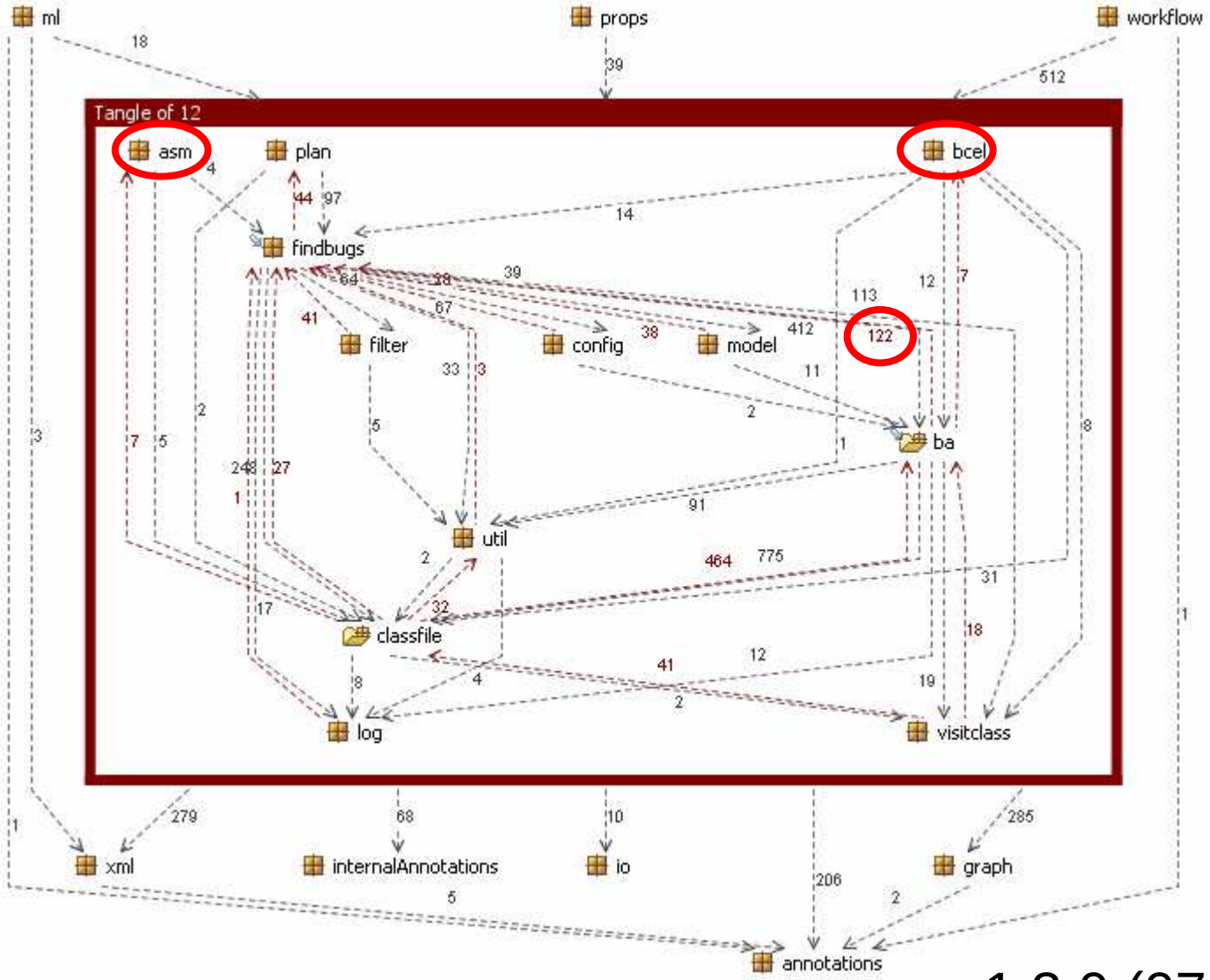




0.8.8 (05/2005)

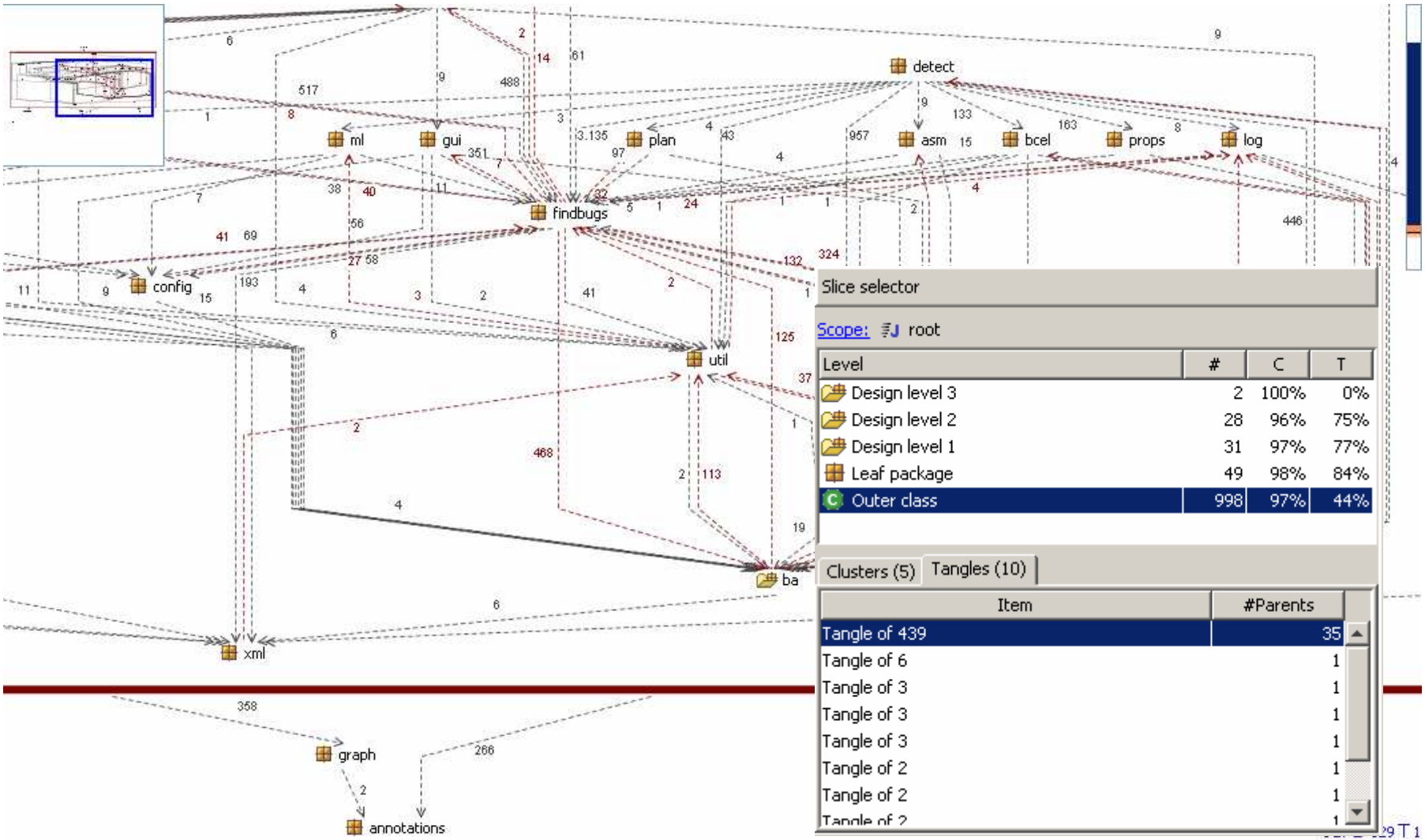
1.0.0 (06/2006)

Tangle increase even more…



1.3.0 (07/2007)

ONE BIG Tangle…



1.3.8 (03/2009)

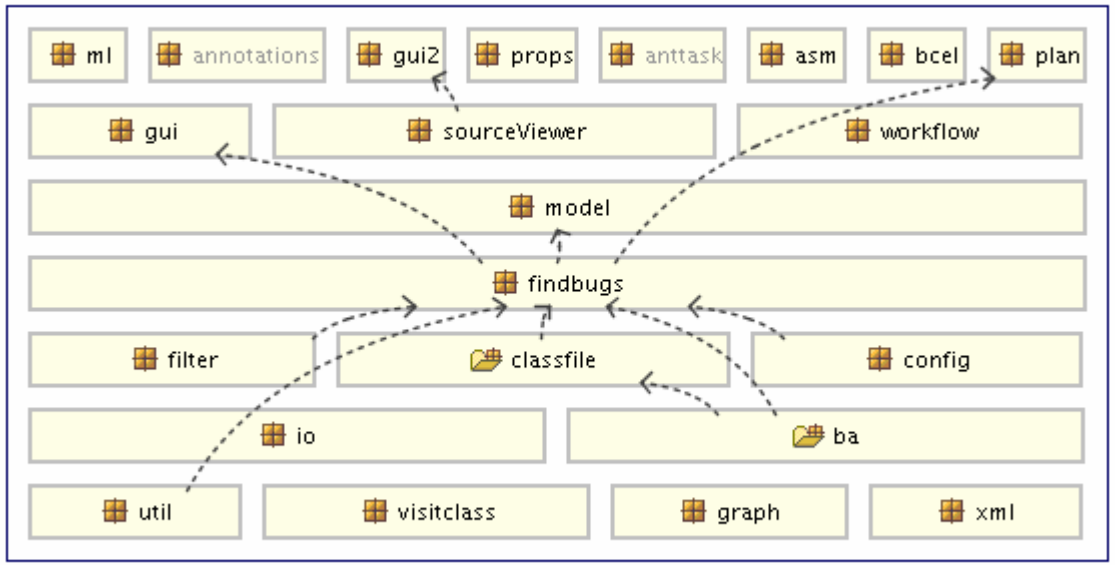
Findbugs: and the next level of checking ;-)

SICK

Findbugs 1.2.1: Architectural Analysis

SICK

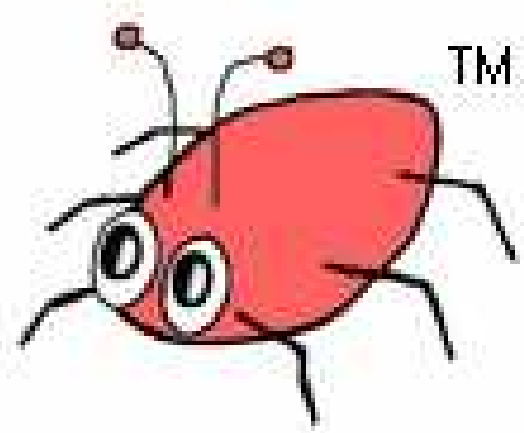
Diagram 1



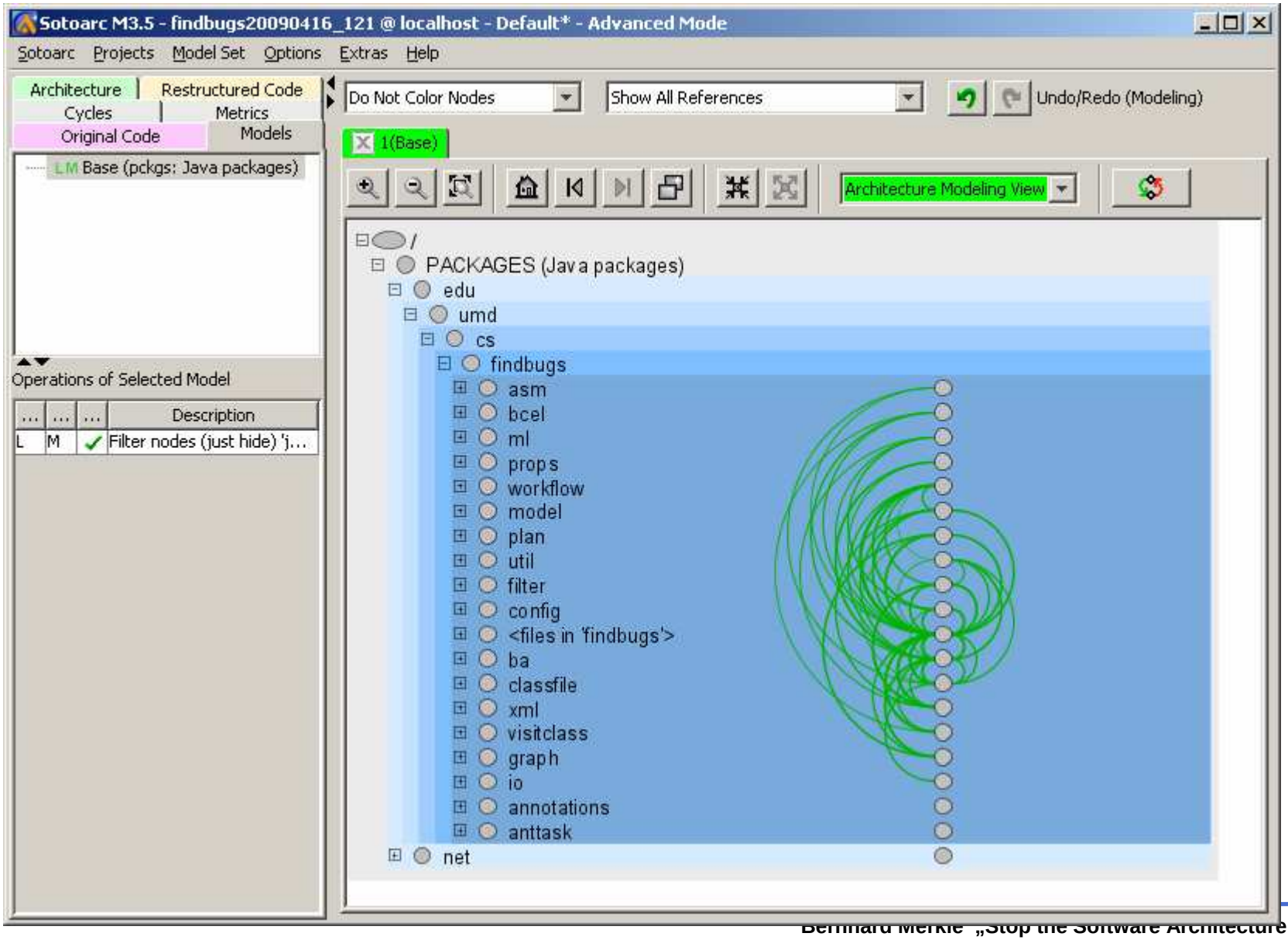
Description: *Subsystem breakout for 'edu.umd.cs.findbugs'*

Violations

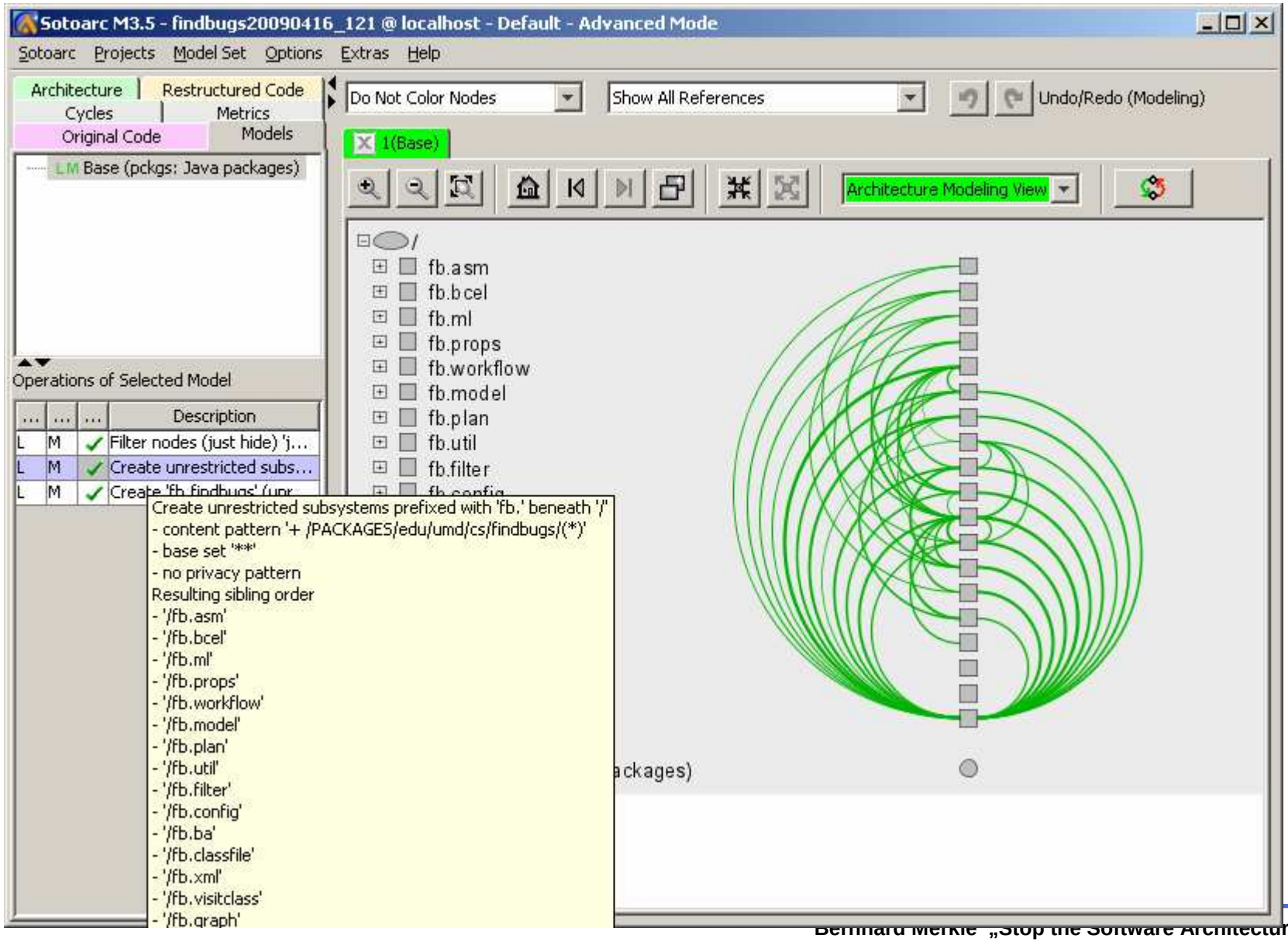
Source	Target	#Violations
sourceViewer	gui2	7
findbugs	plan	37
findbugs	gui	7
findbugs	model	30
filter	findbugs	36
classfile	findbugs	12
config	findbugs	19
ba	findbugs	99
ba	classfile	78
util	findbugs	2
	Total	**327**



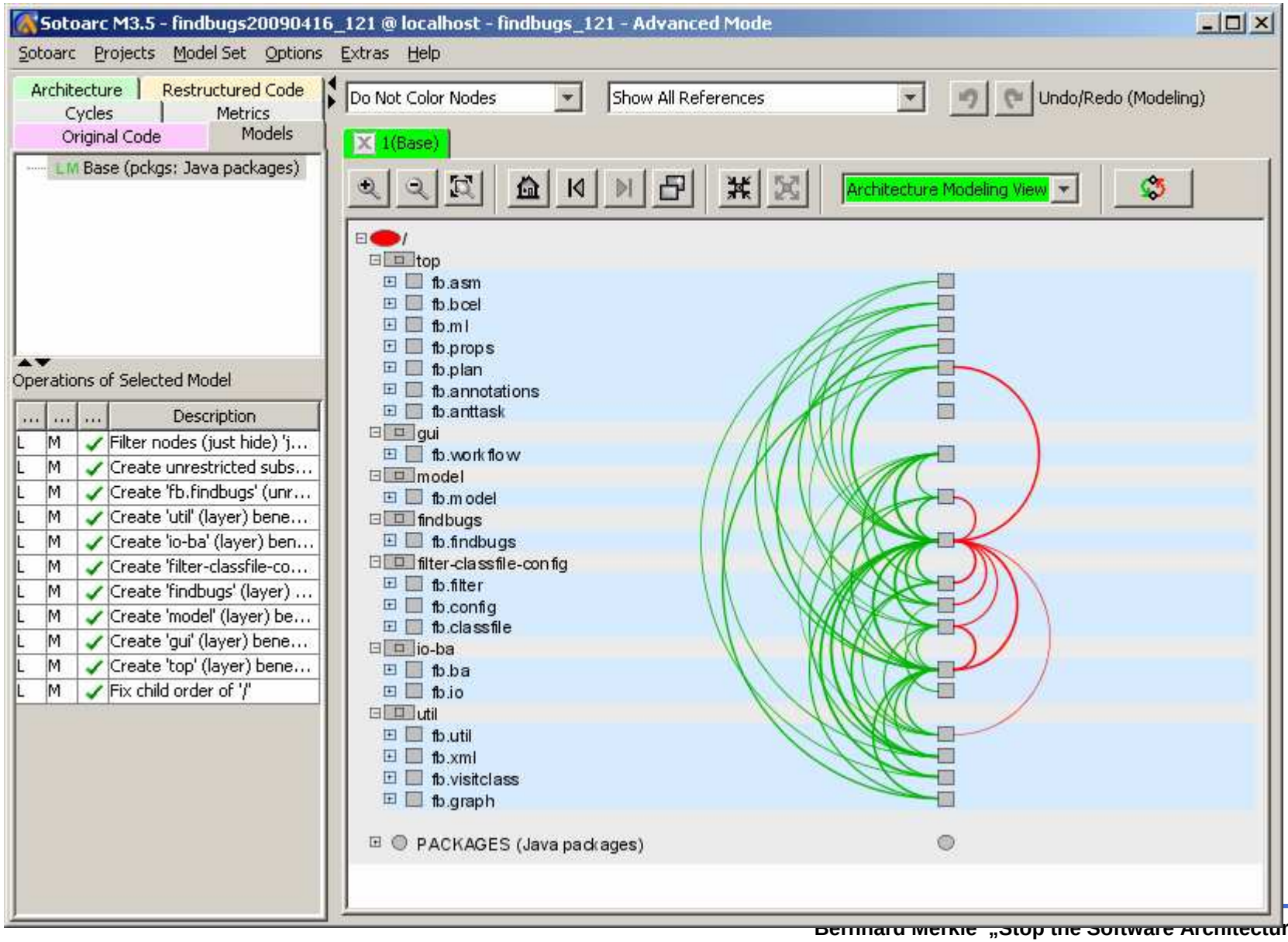
Modeling Subsystems:



Modeling Subsystems:

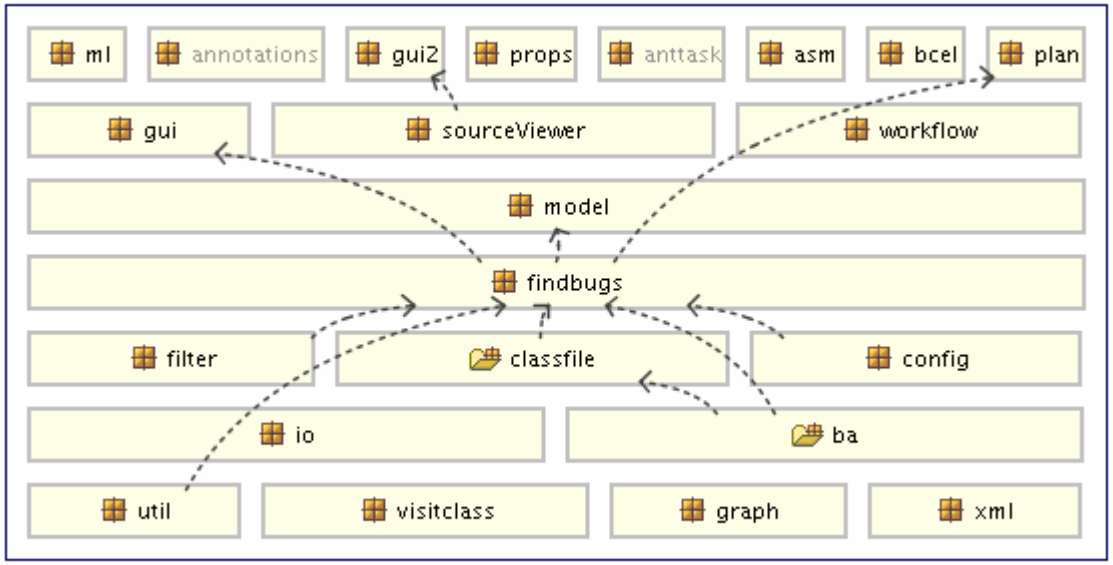


Modeling Subsystems:



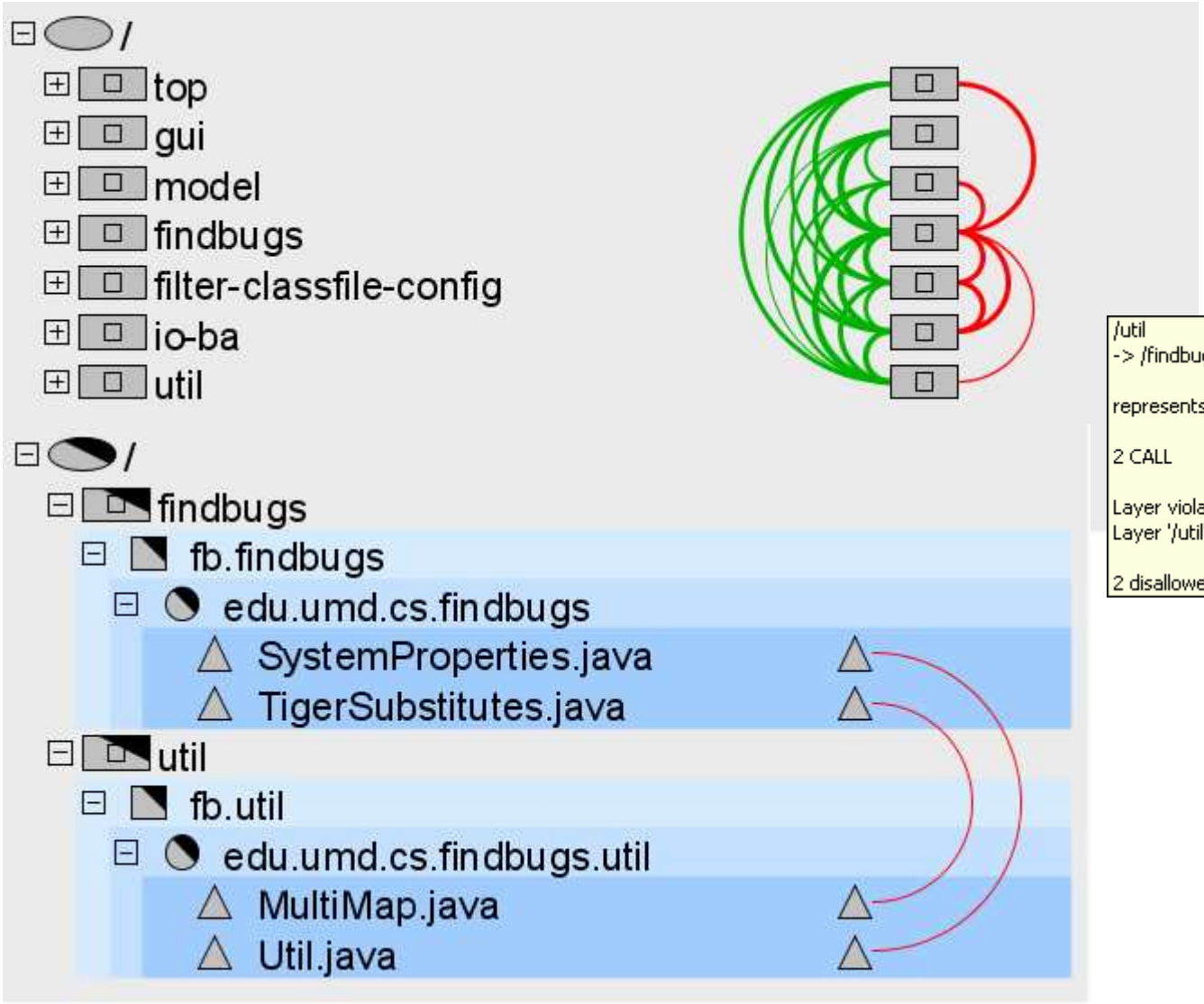
Fixing Architectural Violations

Diagram 1

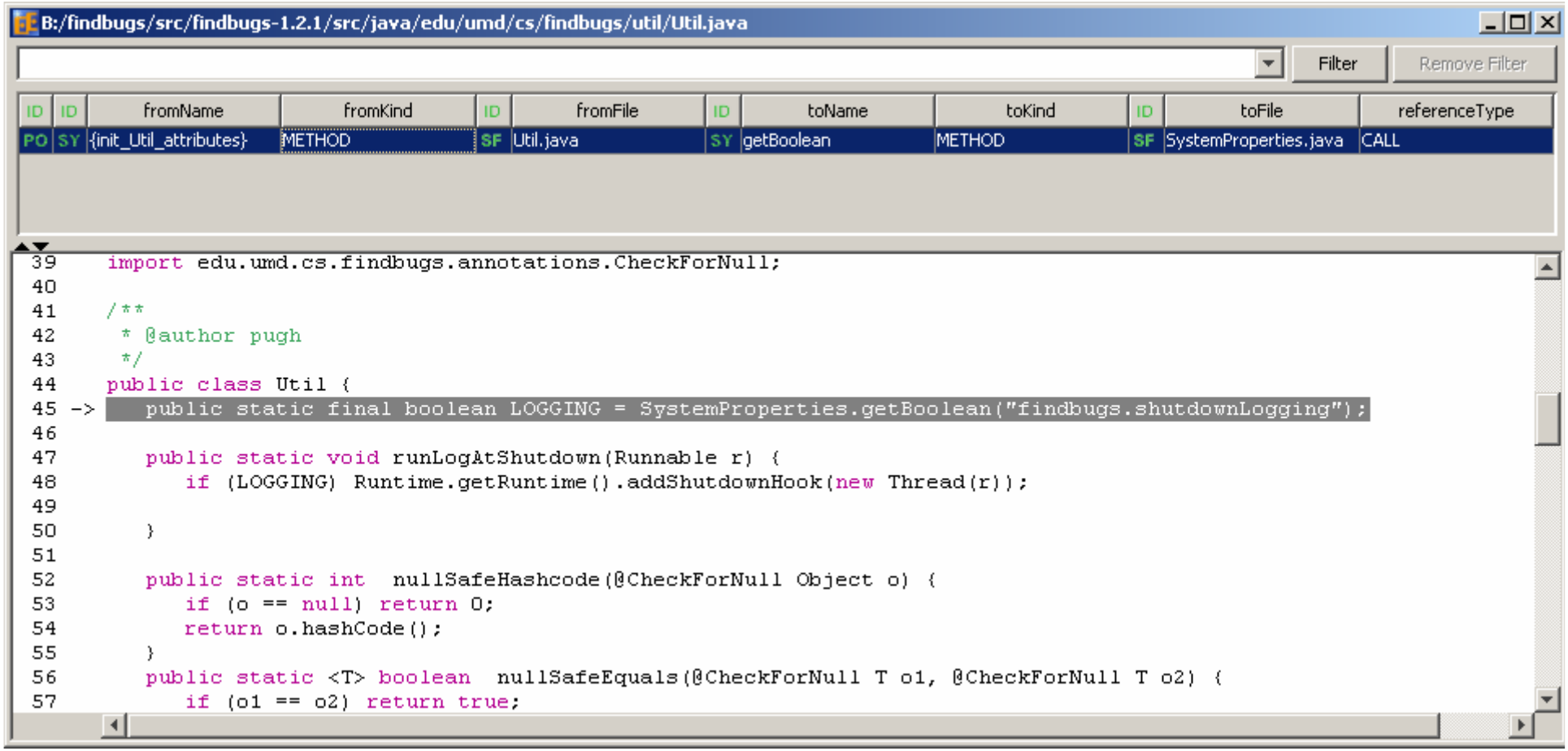


Description: *Subsystem breakout for 'edu.umd.cs.findbugs'*

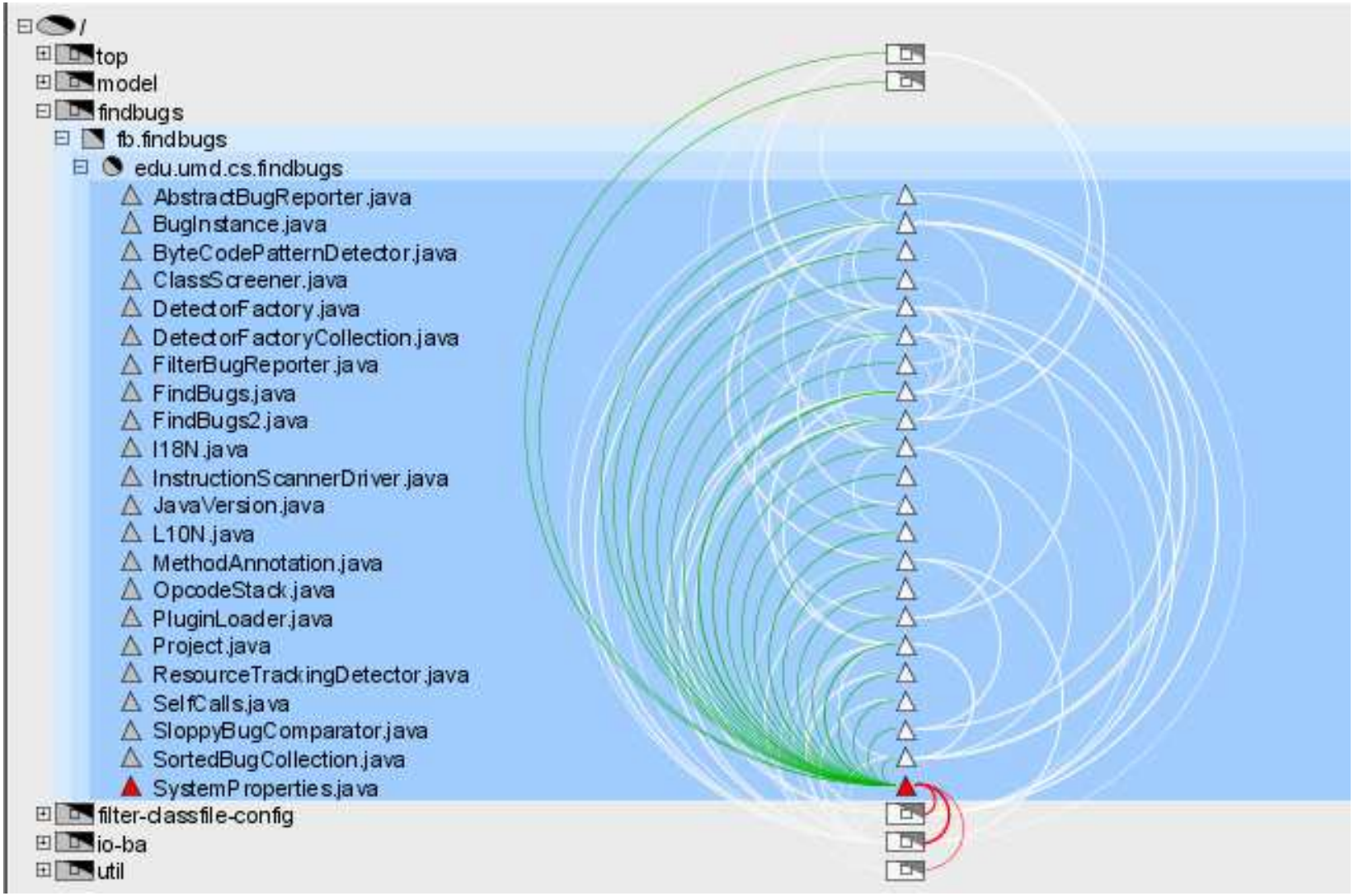
Fixing Architectural Violations



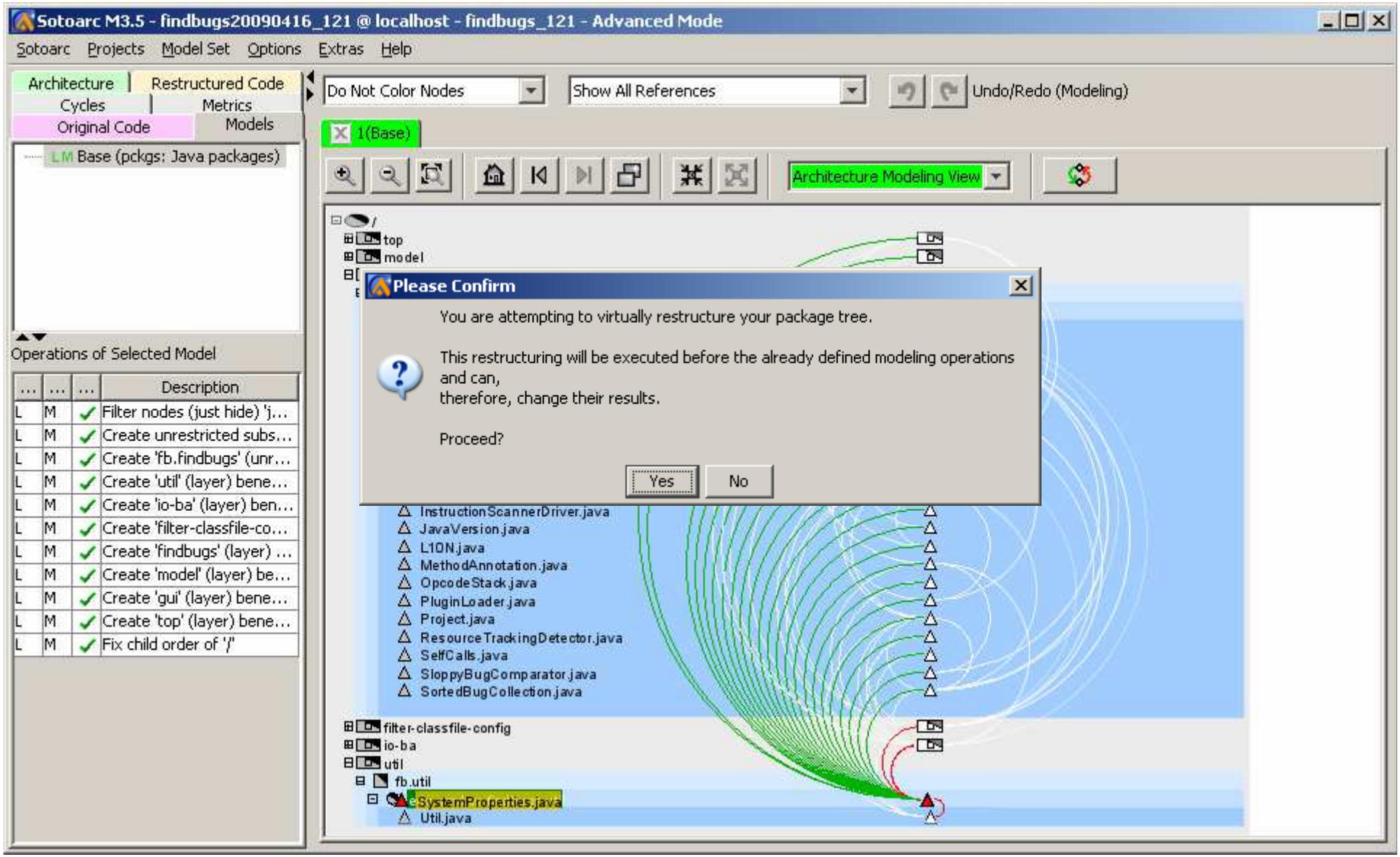
Fixing Architectural Violations



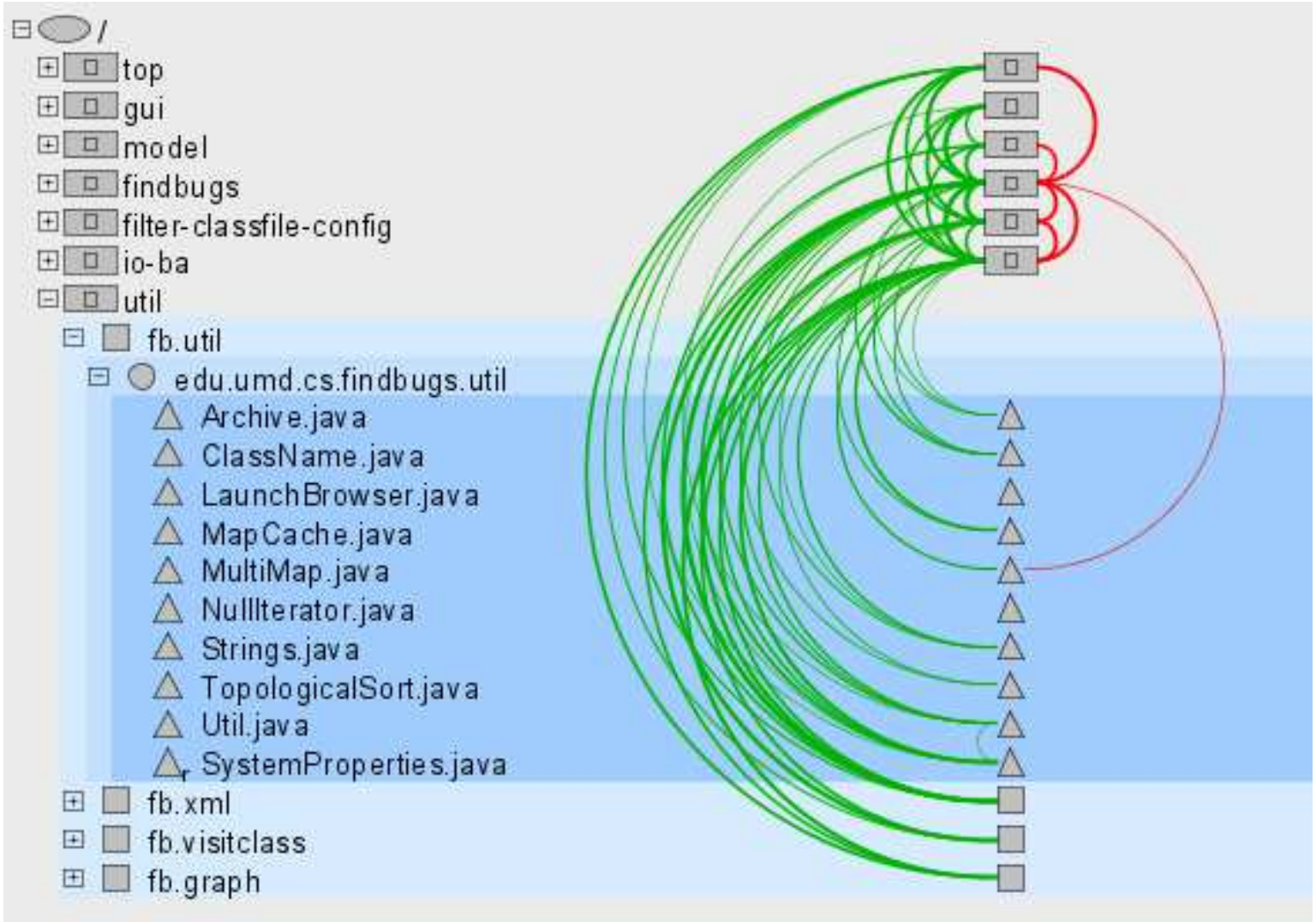
Fixing Architectural Violations



Fixing Architectural Violations



Fixing Architectural Violations



Aim of Architectural Analysis

SICK

- Boost

1. **do AA-Tools work with heavy templated code ?**

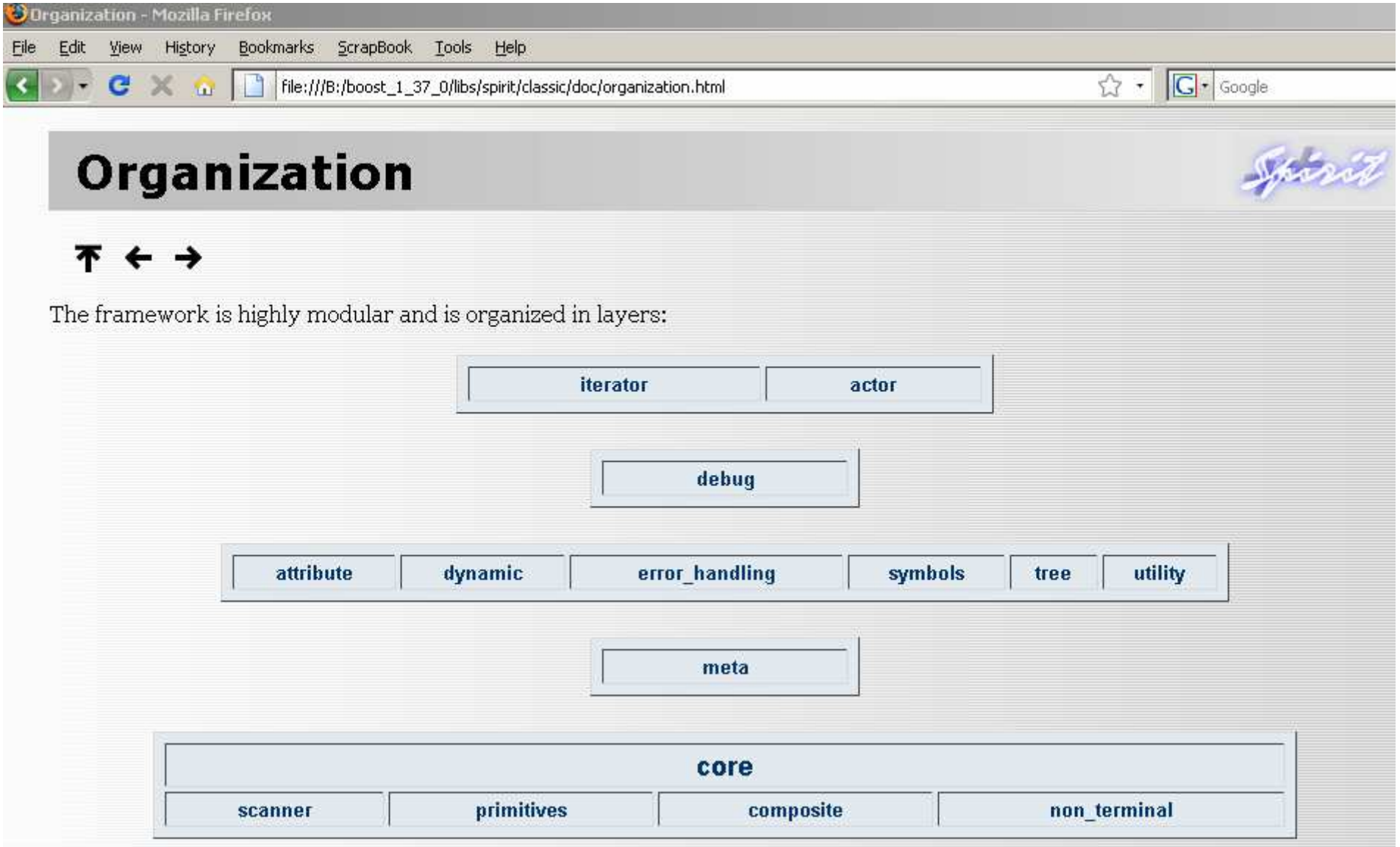
2. **find out if Boost has some architectureal violations**: e.g. dependencies
   1. from **utility** to other libraries
   2. from **somelib** to "heavy" libraries like **mpl**
   3. which access the internal implementation details, e.g. (aux_, detail, impl)
   4. which violate the existing documented architecture,

Boost: Checking dependencies ?

- Deliver “units”, “iterator_facade”, etc.

- bcp.exe
  - bcp --list boost\iterator\iterator_facade.hpp > deps.txt
  - about 300 header files
  - about 20 times mpl usage
  - remove some capability,
    - traits pod, const, etc
    - no need to include std mpl…
    - no support for iterator_writability_disabled<>

Boost Spirit



Build Setup

- Usually capture process
- header only (90%)
- Idea
  - Compile test cases (libs)

```
$ cd boost_1_37_0
$ find boost/mpl -name "*.hpp" | grep -v aux_ > includes.txt
$ for i in `cat includes.txt` ; do printf "#include <$i>\n";
done >mplmain.cpp

$ cafeCC -BB:\boost_1_37_0 -I. -shared -w -no_strip -o
mpl.dll.iml mplmain.cpp

$ iml2rfg -lifted_module_view mpl.dll.iml mpl.dll.rfg
```

SA: Modeling Boost

L	M	✓	Create 'BOOST.SPIRIT' (unrestricted subsystem) benea...
L	M	✓	Move 'IteratorActor', 'Debug', 'AttributeDynamicErrorSy...
L	M	✓	Fix child order of 'BOOST.SPIRIT'
L	M	✓	Create 'BOOST.UTILITY' (unrestricted subsystem) bene...
L	M	✓	Move 'utility' to 'BOOST.UTILITY'
L	M	✓	Create 'BOOST.SMART_PTR' (unrestricted subsystem) ...
L	M	✓	Move 'smart_ptr' to 'BOOST.SMART_PTR'
L	M	✓	Create 'BOOST.CONVERSION' (unrestricted subsystem)...
L	M	✓	Move 'conversion' to 'BOOST.CONVERSION'
L	M	✓	Filter nodes (just hide) 'optional.hpp'
L	M	✓	Filter nodes (just hide) 'config.hpp'
L	M	✓	Filter nodes (just hide) 'unordered_map.hpp'
L	M	✓	Create 'BOOST.FOREACH' (unrestricted subsystem) be...

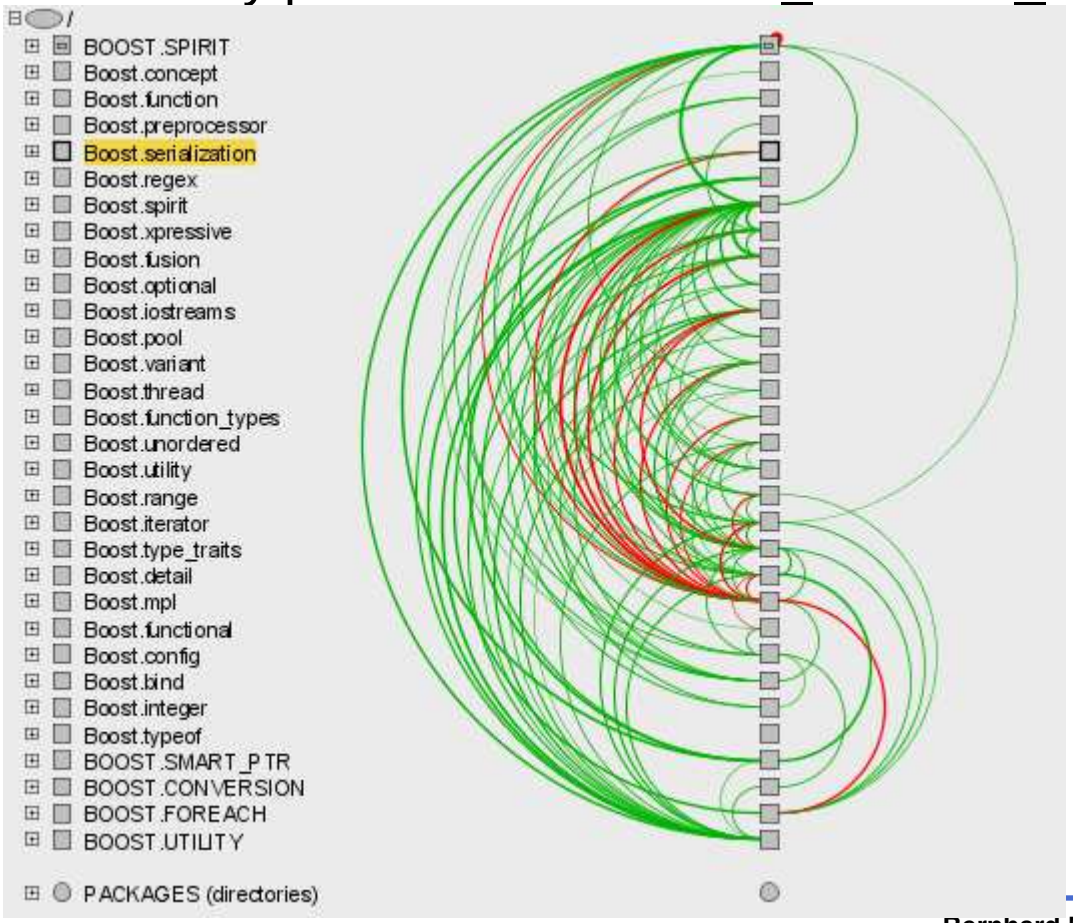
- BOOST.SMART_PTR
- BOOST.CONVERSION
- BOOST.FOREACH
- BOOST.UTILITY

...	...	...	Description
L	R	✓	Create 'utility' (container) beneath PACKAGES
L	R	✓	Move 'assert.hpp', 'call_traits.hpp', 'checked_delete.hp...
L	R	✓	Move 'integer_traits.hpp' to 'integer'
L	R	✓	Move 'range.hpp' to 'range'
L	R	✓	Move 'regex.hpp', 'regex_fwd.hpp' to 'regex'
L	R	✓	Move 'type_traits.hpp' to 'type_traits'
L	R	✓	Move 'iterator_adaptors.hpp' to 'iterator'
L	R	✓	Move 'preprocessor.hpp' to 'preprocessor'
L	R	✓	Create 'smart_ptr' (container) beneath PACKAGES
L	R	✓	Move 'intrusive_ptr.hpp', 'scoped_array.hpp', 'scoped_...
L	R	✓	Move 'mem_fn.hpp' to 'bind'
L	R	✓	Move 'function.hpp' to 'function'
L	R	✓	Create 'conversion' (container) beneath boost
L	R	✓	Move 'implicit_cast.hpp', 'lexical_cast.hpp' to 'conversion'
L	R	✓	Move 'blank.hpp', 'blank_fwd.hpp' to 'mpl'
L	R	✓	Move 'none.hpp', 'none_t.hpp' to 'optional'
L	R	✓	Move 'memory_order.hpp' to 'smart_ptr'
L	R	✓	Move 'aligned_storage.hpp' to 'detail'
L	R	✓	Move 'array.hpp' to 'utility'
L	R	✓	Create 'foreach' (container) beneath boost
L	R	✓	Move 'foreach.hpp' to 'foreach'
L	R	✓	Move 'ref.hpp' to 'bind'

SA: Modeling Boost

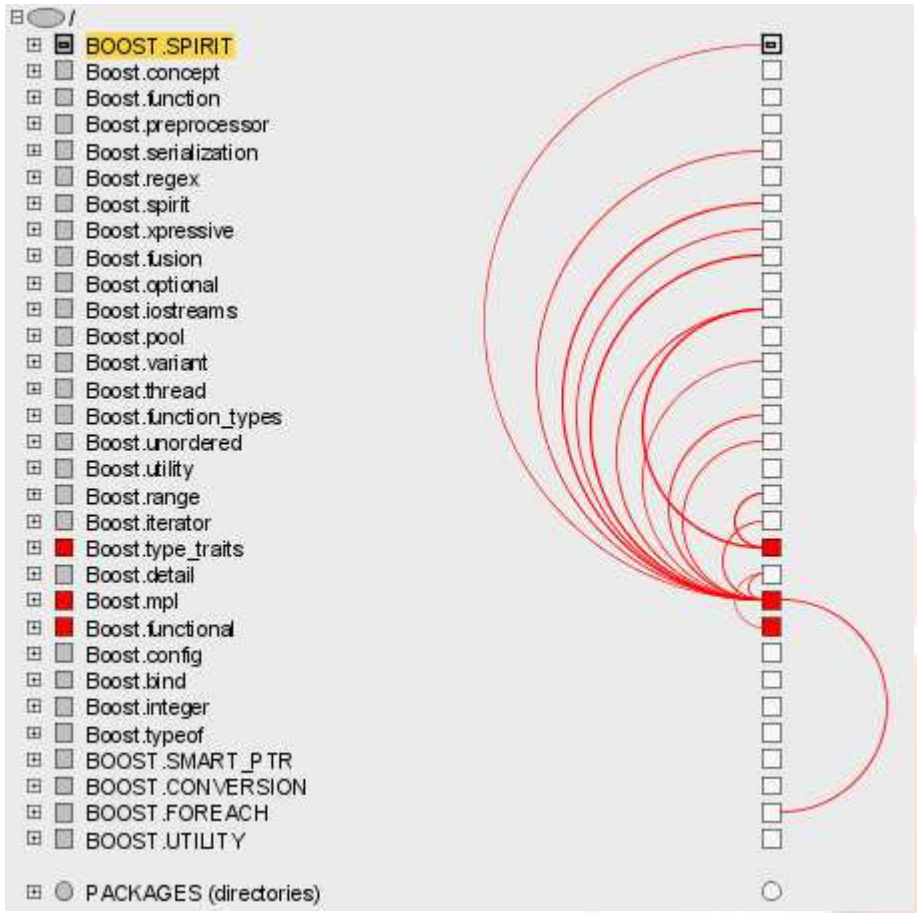
SICK

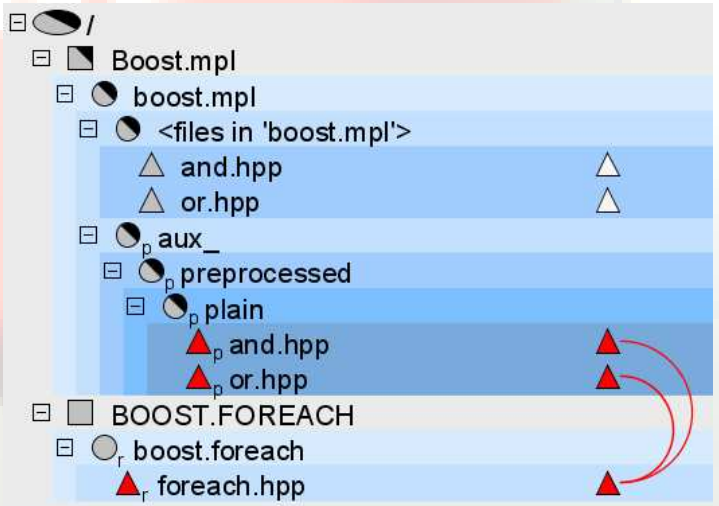
- Create unrestricted subsystems with name X for packages under (Packages)/boost/(X)
- Privacy pattern used + **/aux_ + **/aux_/** + **/detail + **/detail/**



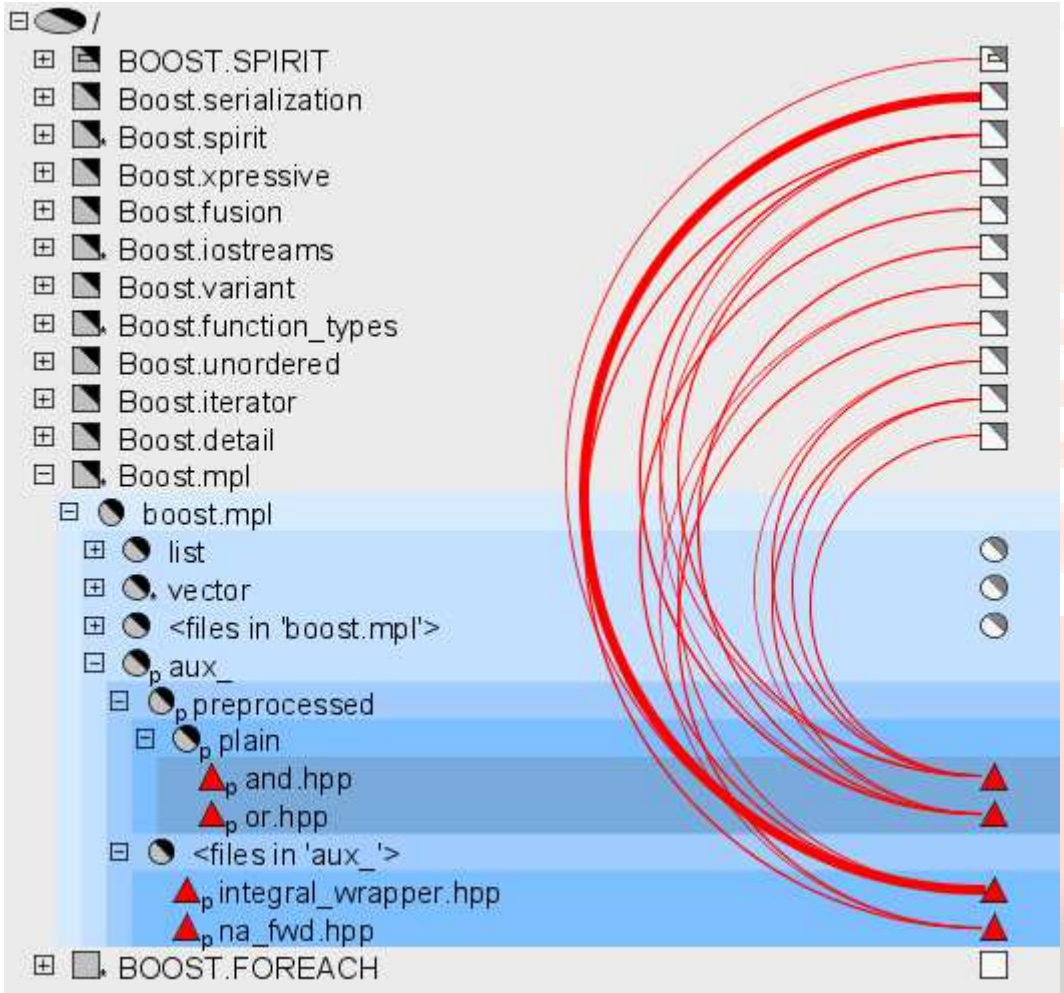
SA: Modeling Boost

SICK

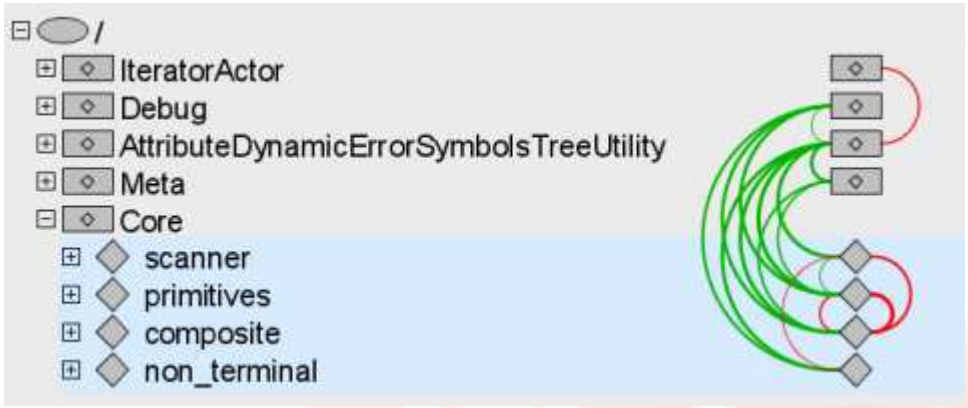


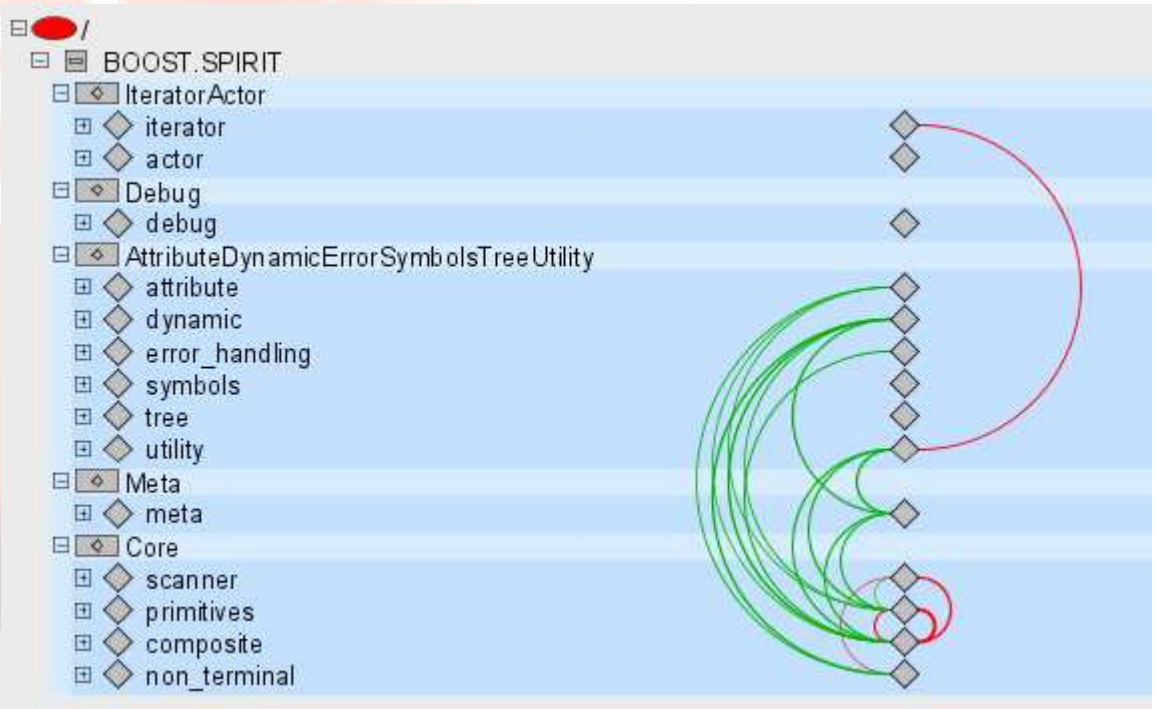


SA: Modeling Boost

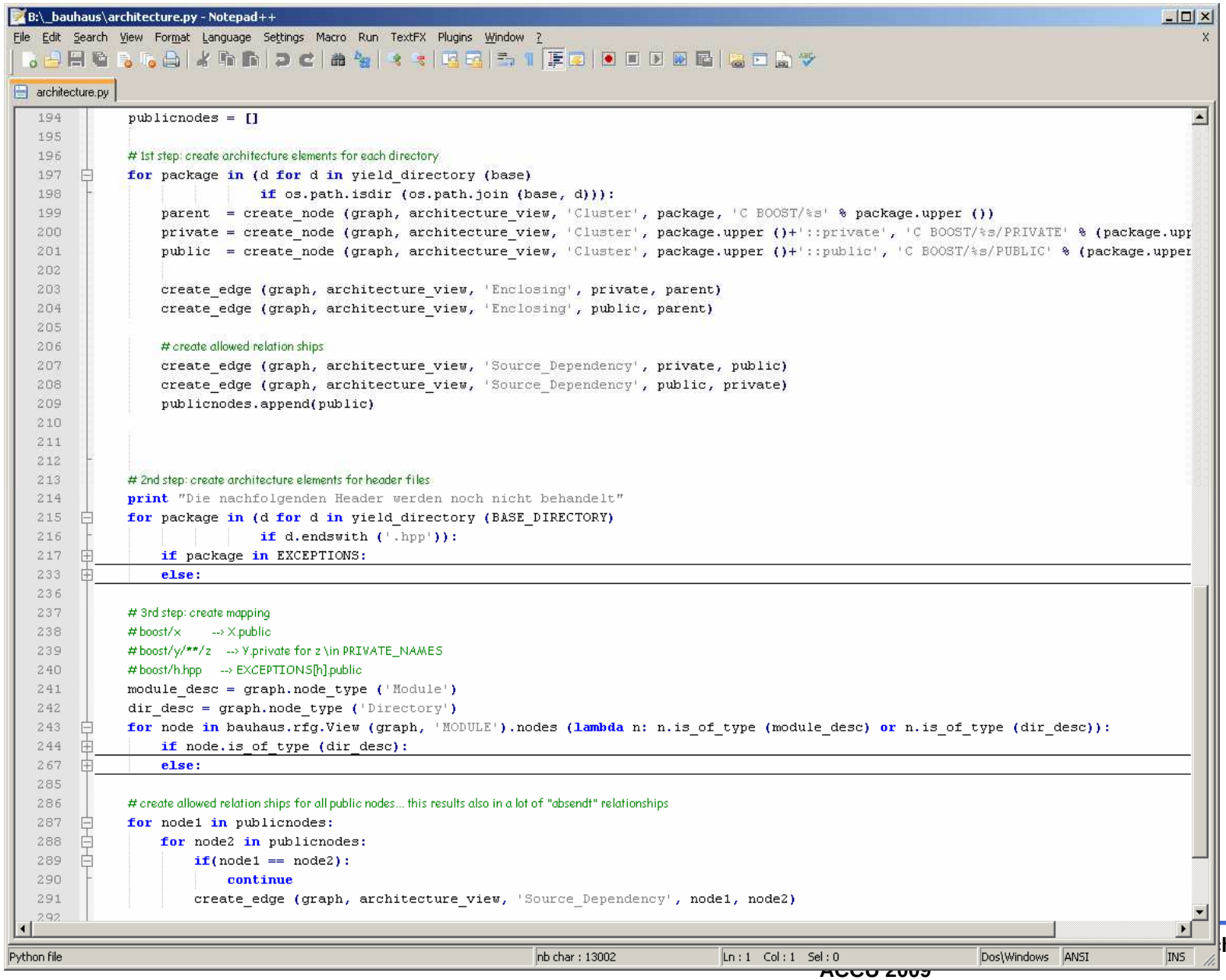


SA: Modeling Spirit



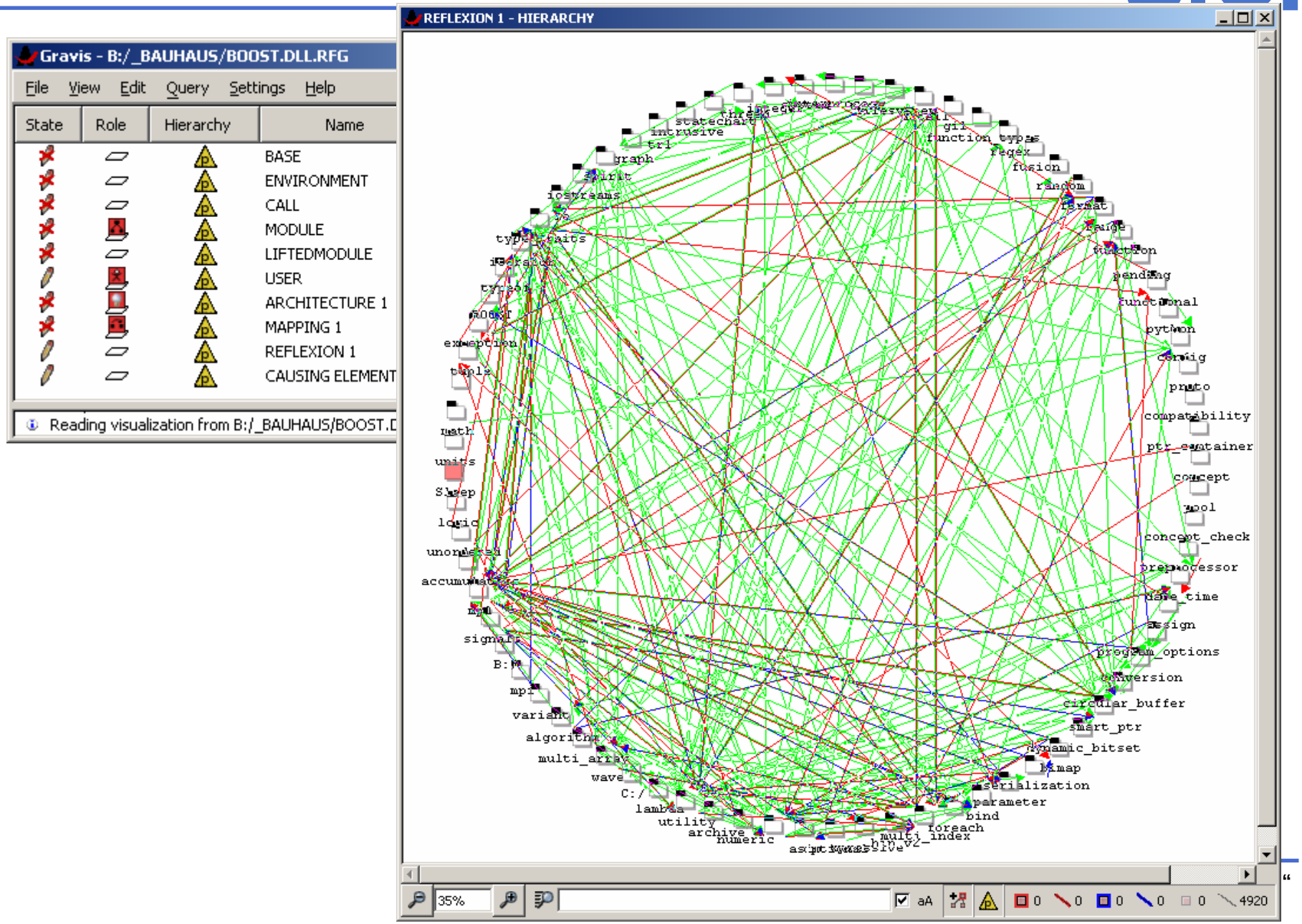


Bauhaus: Modeling



Bauhaus: Modeling

SICK



No (real) violations found in MPL stuff…

SICK

Factorial example:

```
template <unsigned N>
struct Factorial
{
  static const unsigned value = N * Factorial<N-1>::value;
};

template <>
struct Factorial<0>
{
  static const unsigned value = 1;
};

void main()
{
  int y = Factorial<2>::value;
}
```

Template Dependencies:

SICK

- Explicit specialization :
  - Factorial<0> is an explicit specialization of Factorial<N> #2
  - Factorial<1> is a specialization of Factorial<N> #1
  - Function main uses the specialization Factorial<2> #3
  - Factorial<2> is a specialization of Factorial<N> #3
  - Factorial<2> uses the specialization Factorial<1> #1
  - Factorial<1> uses the specialization Factorial<0> #1

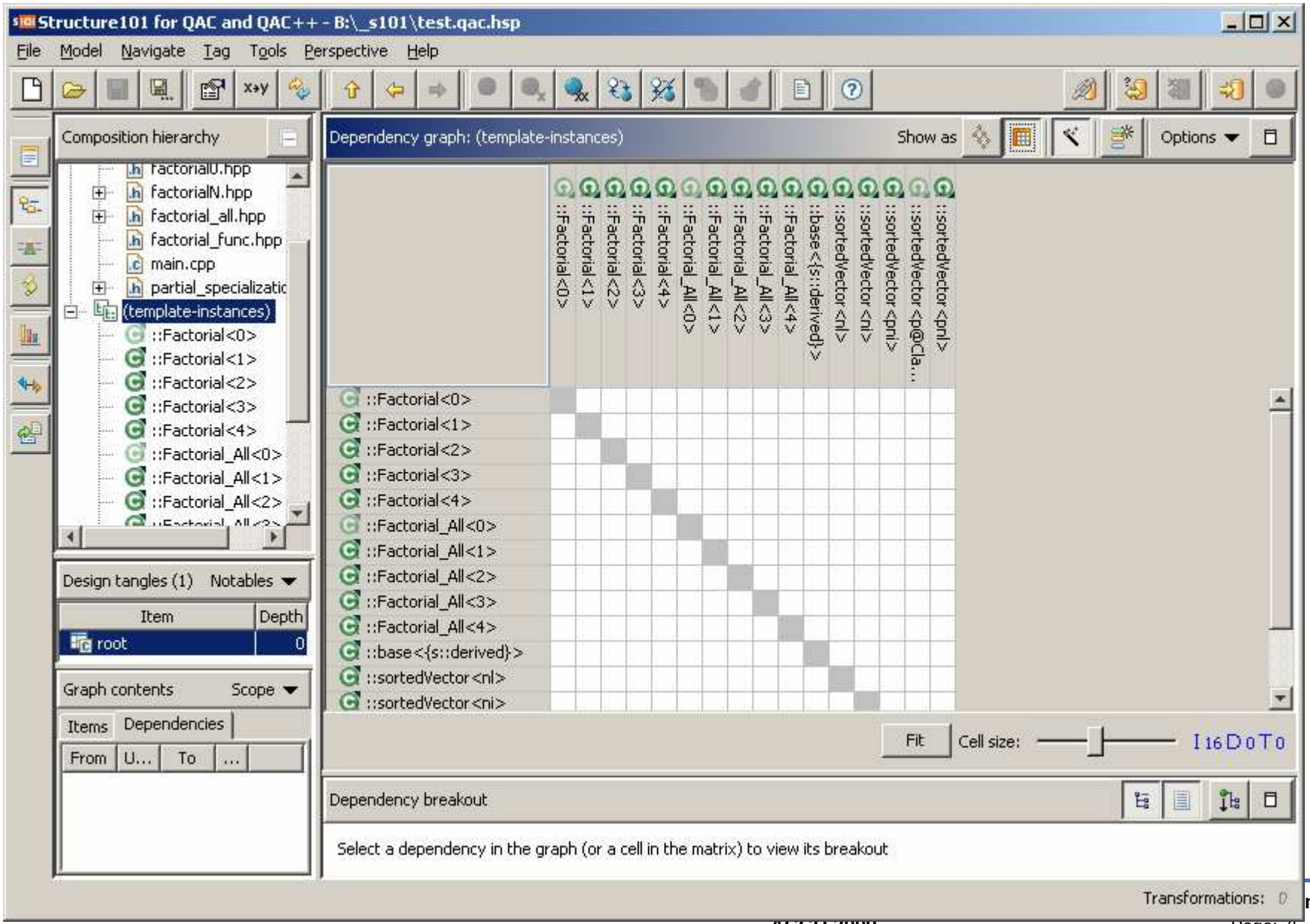
Template Dependencies:

SICK

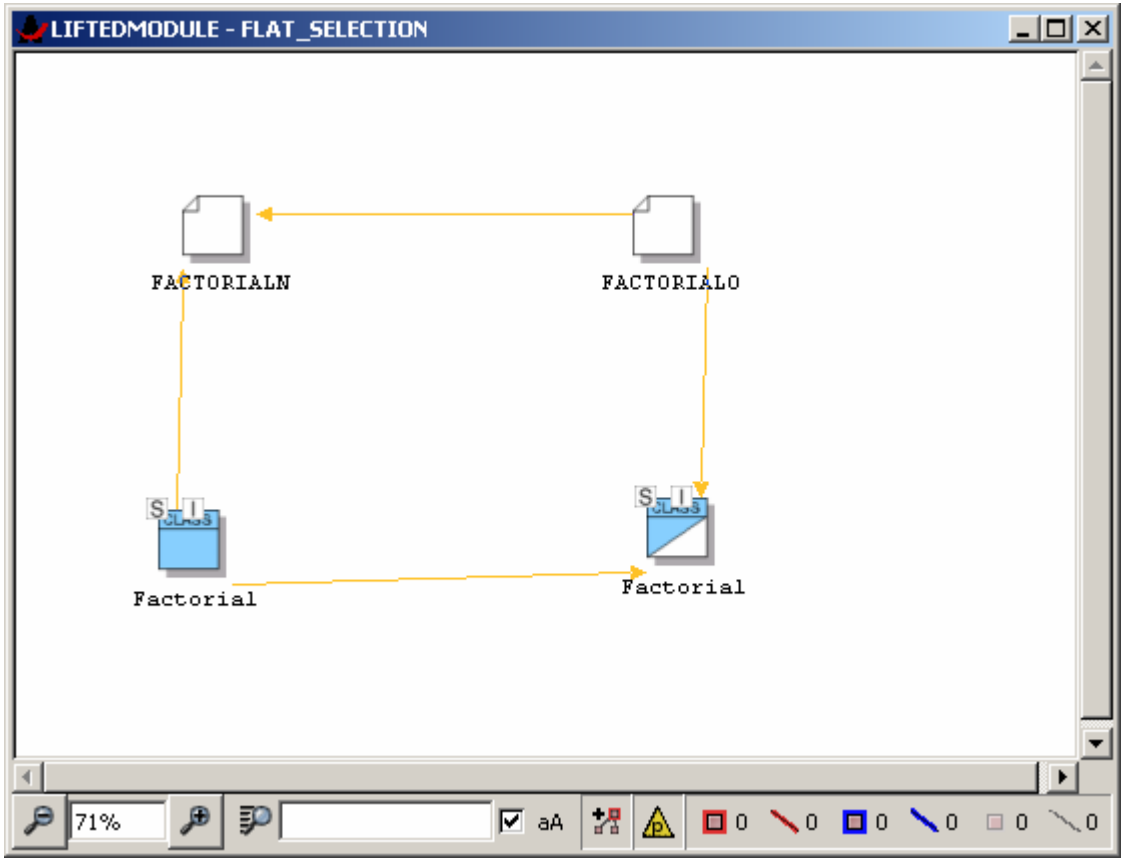
Partial specialization:

```
template <typename T>
class sortedVector
{...}

template <typename T>
class sortedVector<T *>
{...}


sortedVector<int >     i1;
sortedVector<int *>    i2;
sortedVector<long>     j1;
sortedVector<long *>   j2;
```

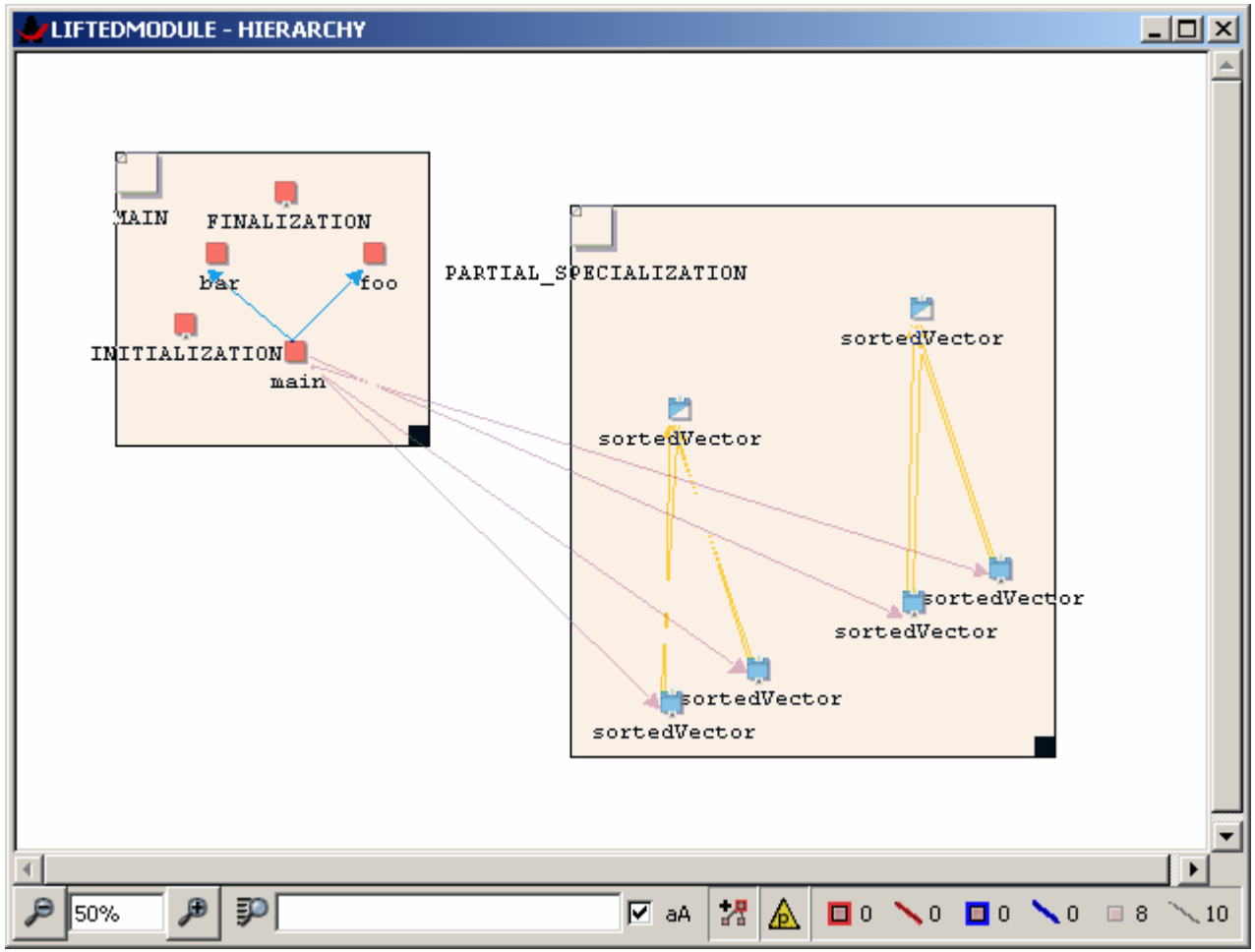
Template Dependencies:



Template Dependencies:



Template Dependencies:



Template Dependencies and Tool support

SICK

Decuducable dependencies

- Code
  - Usage/Call graph at runtime
- Templates:
  - Usage/Call graph at COMPILE time

- Problems
  - Compiler vs. Analysis tools
  - Optimization vs. verbose information
- C++ Parsers
  - EDG optimizes information away (< EDG 4.0)
  - QA/C++ PRL has some more information

Boost MPL and Architectural Analysis Tools…

SICK

POCO: Architectural Analysis

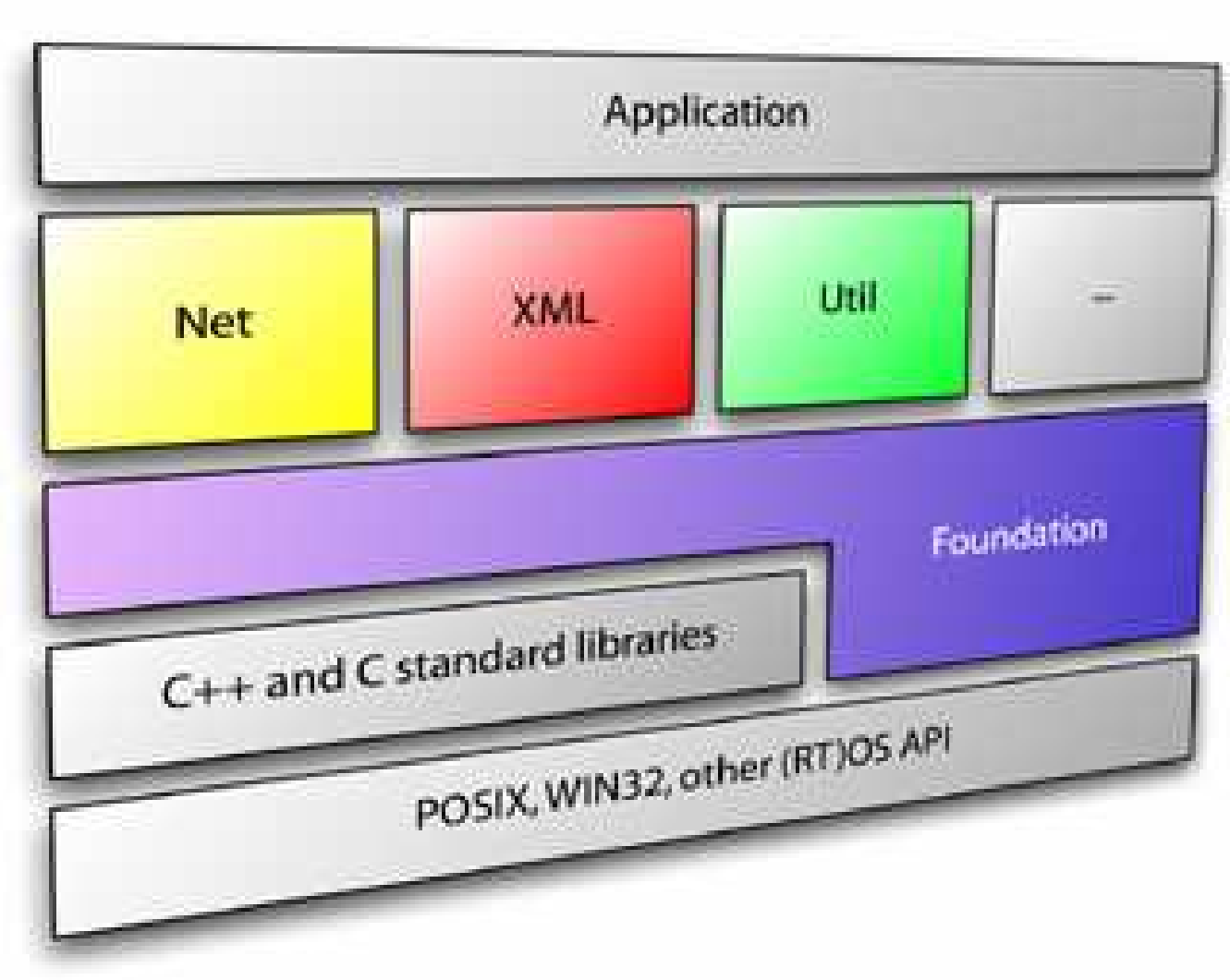
SICK

Features

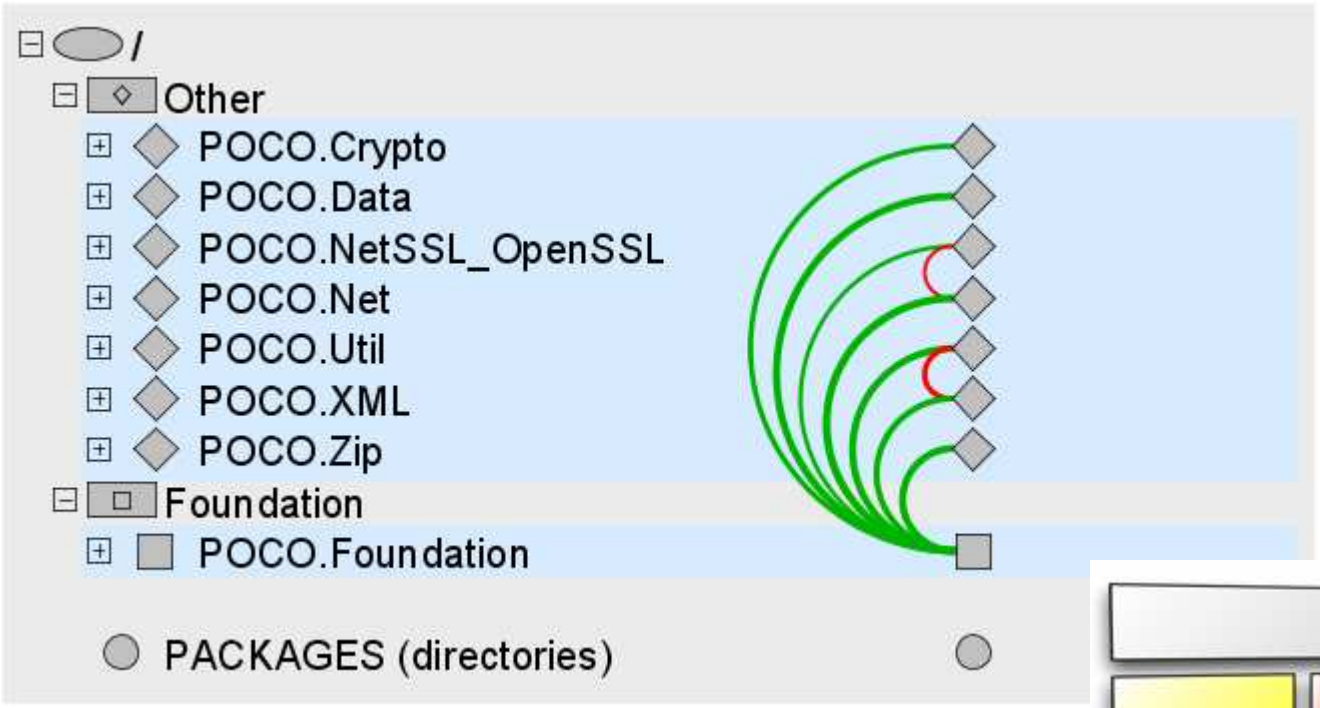
- threads, MT abstractions
- streams and filesystem access
- shared libraries and class loading
- powerful logging and error reporting
- security
- network programming
- XML parsing and generation
- configuration file and options handling
- SQL database access

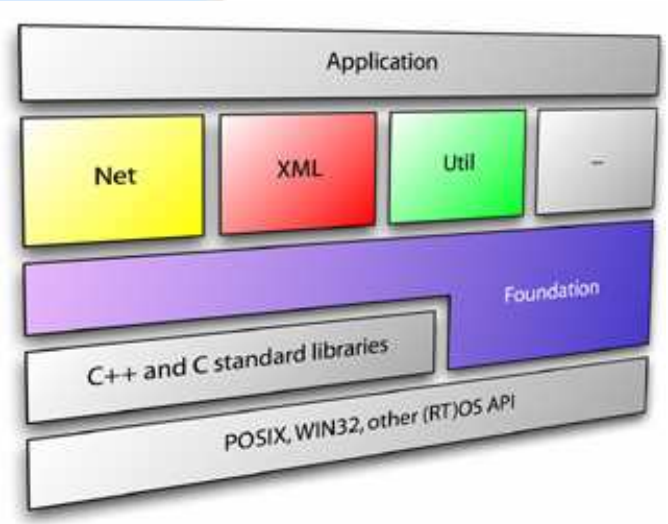
Written by Guenter Obiltschnig , a ACCU member and friend

POCO : Should Architecture

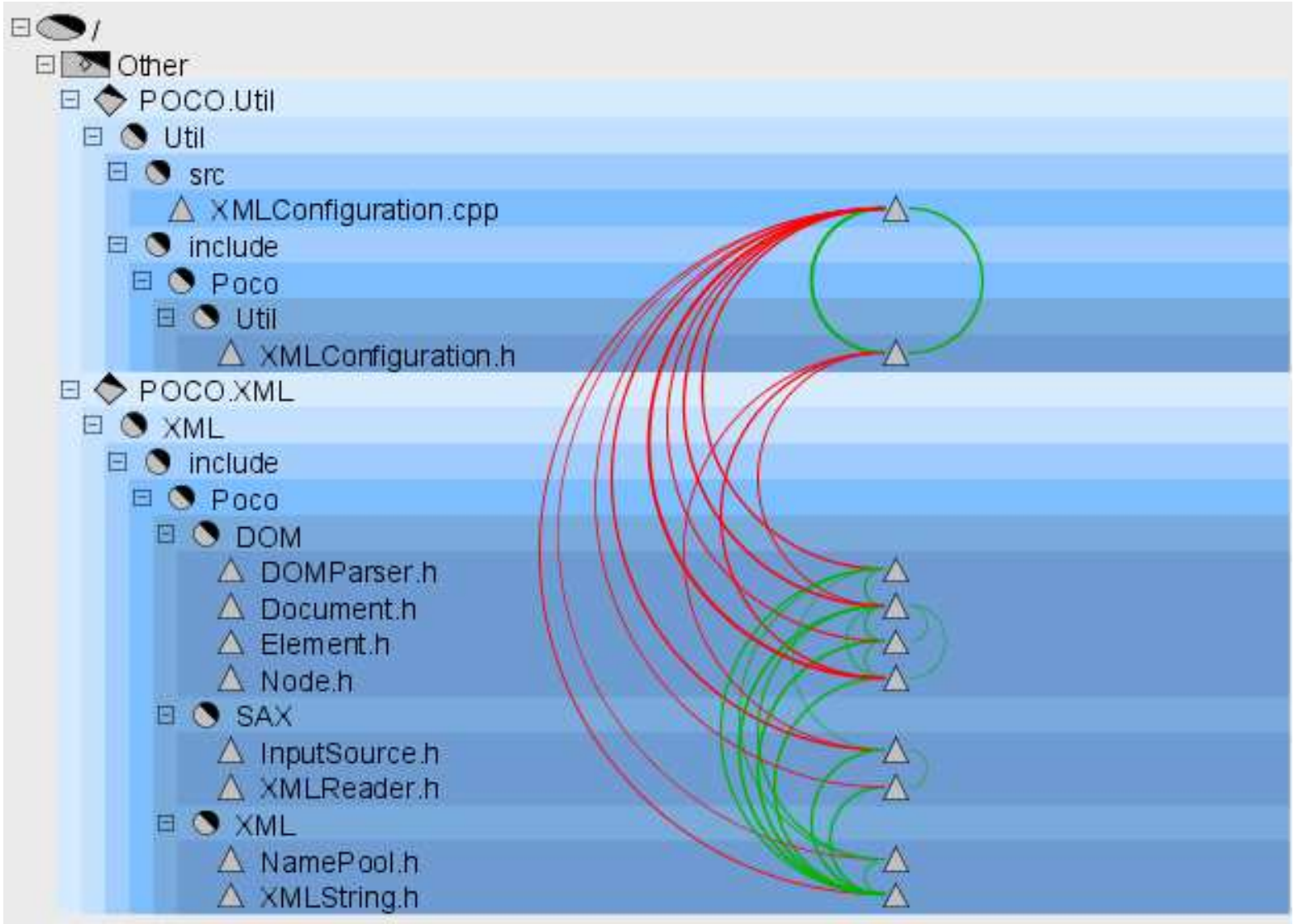


POCO 1.3.3: Overall Layering





POCO 1.3.3: Violation Util → XML



POCO Usage

MARCH 11, 2009

POCO on Tiny Hardware

Filed under: C++, Development, Fun by guenter at 20:52

We recently got our hands on a Digi Connect ME 9210, one of the smallest Linux-capable embedded computers in the world. The system, which is just a bit larger than an Ethernet RJ-45 socket, is based on an ARM9 CPU running at 75 MHz. With 4 MB of Flash and 8 MB or DRAM, the system is powerful enough to run POCO-based applications. For example, we ported the Mindstorms/iPhone controller application from the demo we showed at Embedded World in Nuremberg to the Connect ME, and it runs great. Well, porting is a bit overstating, as we merely had to build a new Flash image for the Connect ME, and update the application's config file. Well, a 75 MHz ARM9 CPU provides enough power to run an application with a built-in web server powering an Ajax-enabled website. Also, the performance improvements for the 1.3.4 release help a lot to make the application work great. Additionally, the 1.3.4 release will introduce some minor changes to help reduce the executable size of statically linked applications. For example, it is possible to build the Util library without XMLConfiguration support, which prevents the XML library from being linked in, cutting about 500K from the executable size.

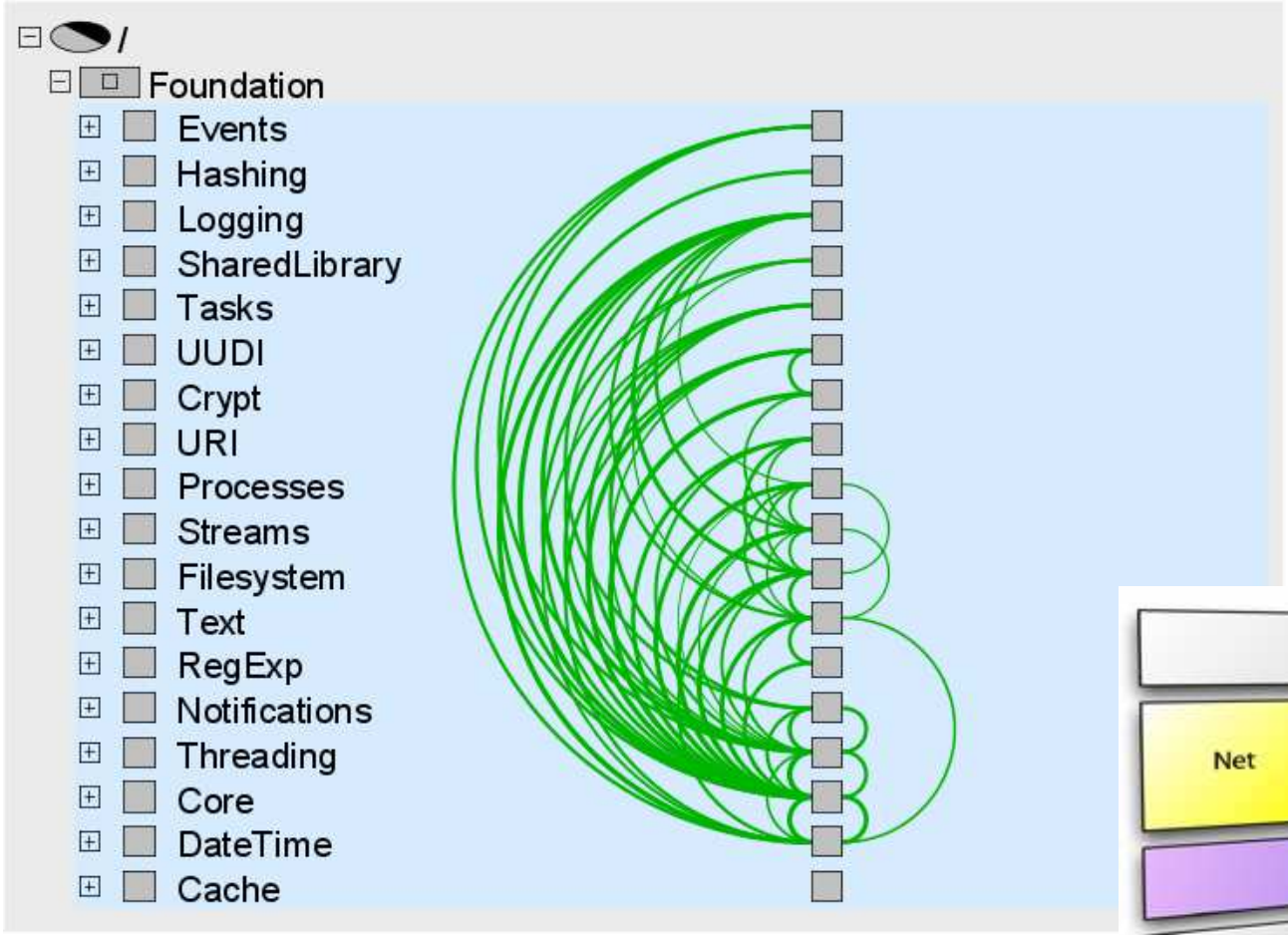
Working with this little device has been a lot of fun, and we are looking forward doing some cool projects with it.

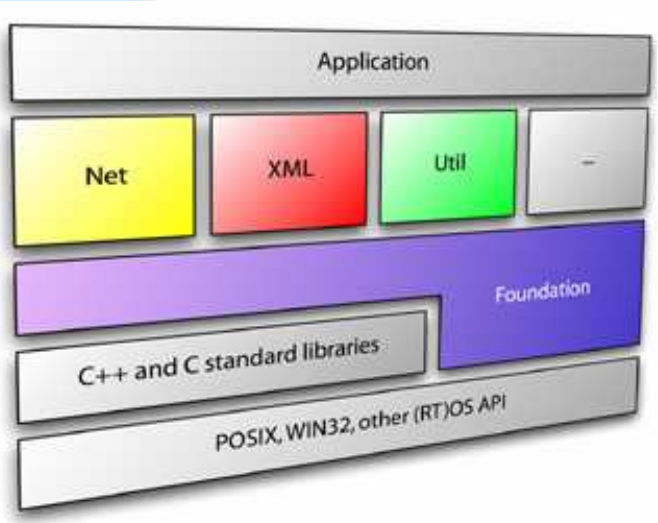
Poco: Architectural Analysis

SICK

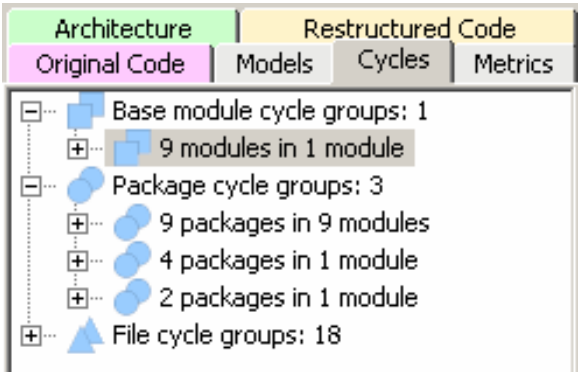
POCO : Foundation Dependencies

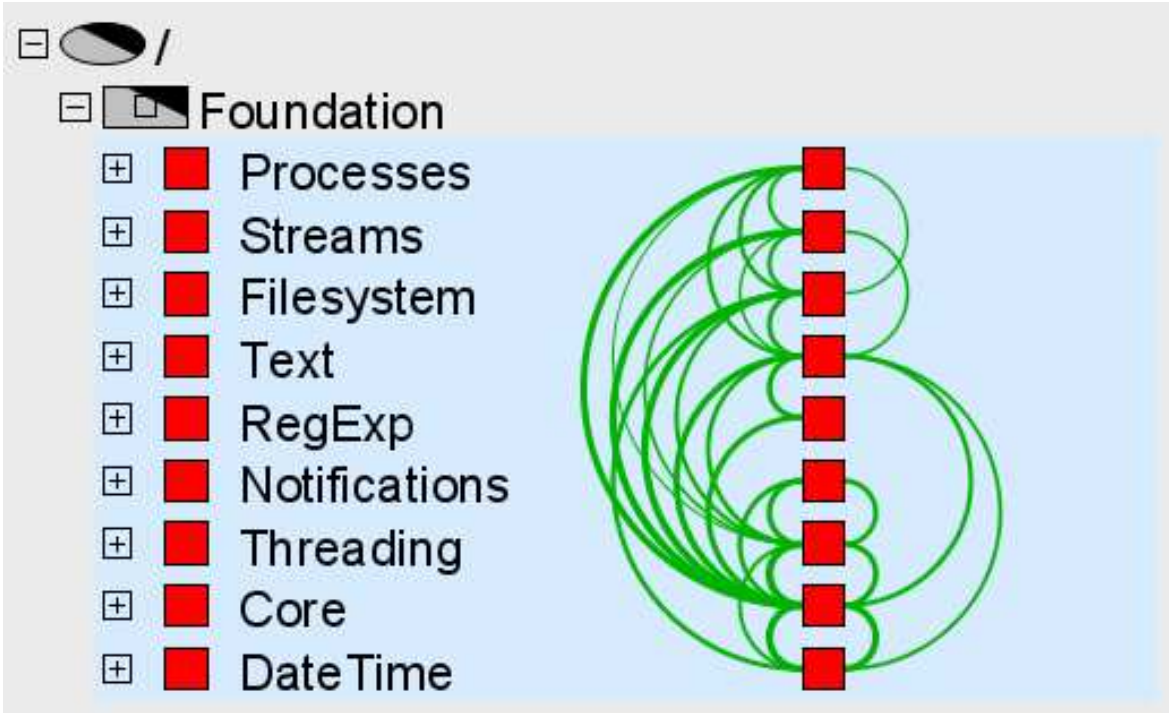
SICK





POCO : Foundation Cycles





Some erosion…

SICK

FlightGear:

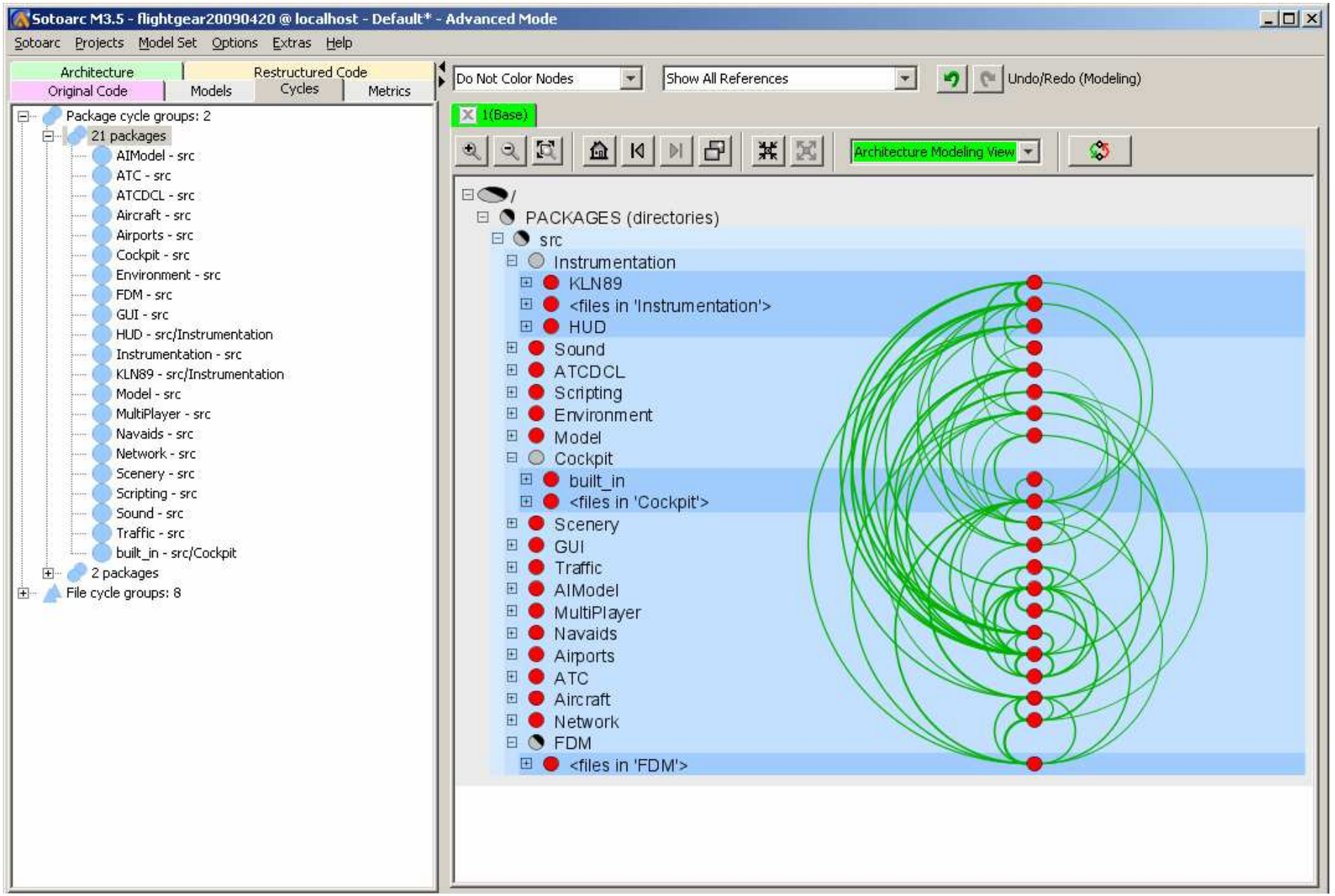

Copyright © www.flightgear.org
NG
NAVY


EQA


Copyright © www.flightgear.org

FlightGear: Architectural Analysis

SICK



FlightGear: Architectural Analysis

SICK

URLs, References:

My Podcast at se-radio.net:
http://se-radio.net/podcast/2008-10/episode-115-architecture-analysis

Thanks to:

- Axivion/Bauhaus:
  - Thomas Eisenbarth
  - Bernhard Berger
- Headways/Structure101:
  - Ian Sutton
  - Paul Hickey
- Programming Research:
  - Richard Corden
  - Fergus Bolger
- Software Tomography: SotoArc
  - Thomas Schoen
  - Heinrich Rust