ACCU 2007

Linting Software Architectures

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

mailto: Bernhard.Merkle@sick.de
mailto: Bernhard.Merkle@googlemail.com

Some Background, and the plan…

- About…
  - myself
  - SICK
- The plan for this talk
  - Software-Architectures
    - Terms, Definitions, etc
  - Checking Architectures
    - Different Kinds of Architecture-Analysis
    - Tools for Architecture-Analysis
  - Experiences, Discussion… ;-)

Linting Software-Architectures

- Why should we care ?
  - In lots of Projects, Architecture declay happens
  - We are not alone, as we‘ve prominent representatives… ;-)

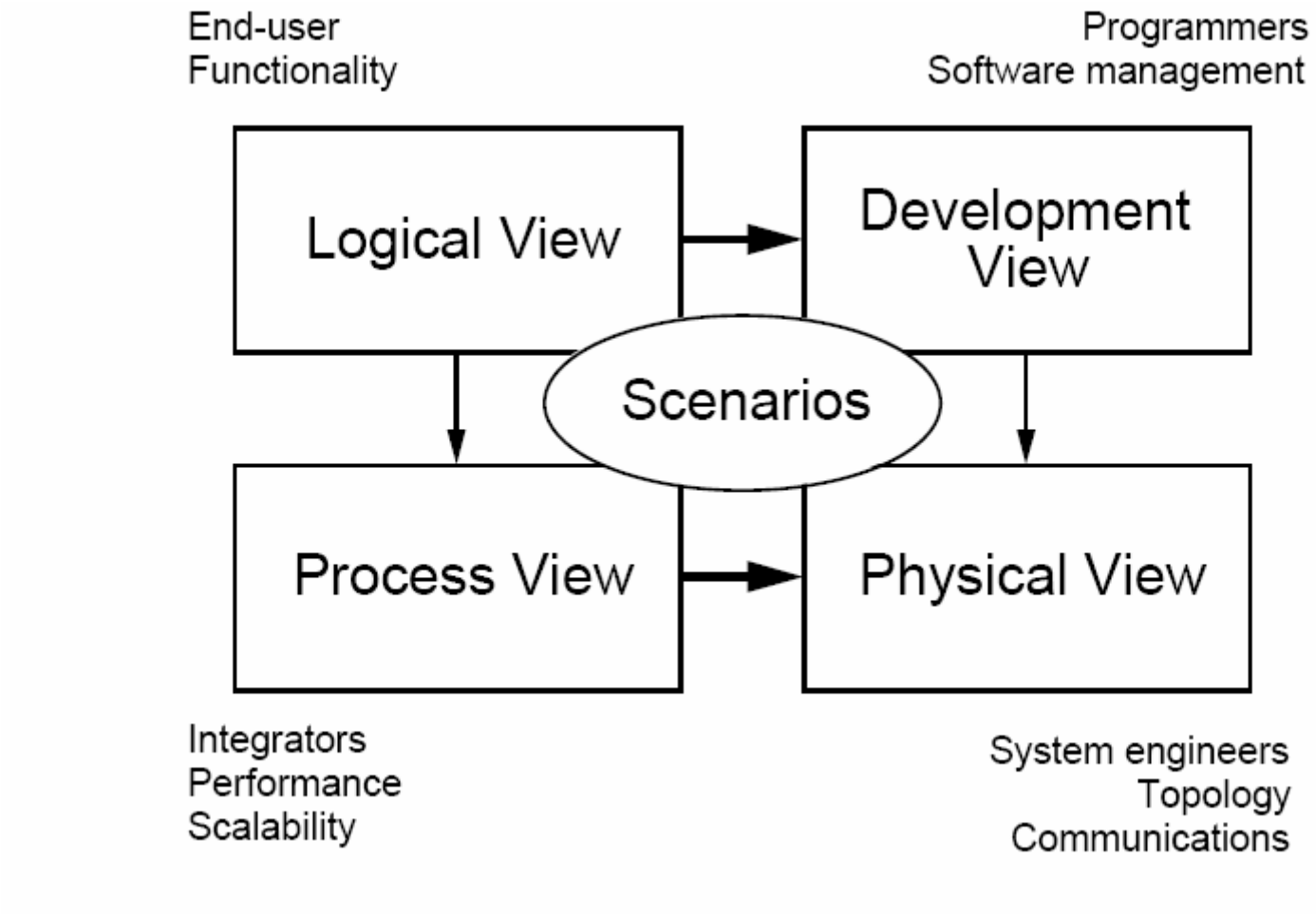
Software-Architecture: Definitions

- 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.

- Booch, Rumbaugh, Jacobson 1999
  - ... the set of significant decisions about the organization of a software system ...
  - … is the highest level of technical design for a software system: It is driven by your key concerns

Views on a Software-Architecture

- 4+1 View Model (Kruchten, 1995)



Documenting a Software-Architecture (Kruchten)

- captured in two documents:
  - *Software Architecture Document*
  - *Software Design Guidelines*

  - *respected to maintain the architectural integrity of the system*.

- Documents are important, but they **are** Documents (enforce ? ;-)

→ Some kind of Automatic Rulechecking

MDSD (Model Driven Software Development)

- Aproach:
  - Architectural Design IS IN the model (and Application !?)
  - Executable Model (MDA, UML+CodeGen, UML-VM)
  - Source: Model, Target: Application (→ Forward Engineering)

- Open Items:
  - Reverse-/Roundtrip-Engineering ?
  - UML too general: DSL ? (Meta-Modeling Support)
  - Important Standards (e.g. in MOF, ASL, QVT) ?
  - Manual Extensions of generated Code
  - Good Integration of “legacy Software” ?

Architecture-Analysis

- Lint == STATIC Analysis
  - hence…some limitations if you do things/tricks at runtime (e.g. Reflection in Java,…)
- With Tool support
  - Pro: automatic, consistent, rule enforcement
  - Cons: Semantic, external Quality
  - The Pro is much stronger compared with Code-Lints !!!

Levels of Static Analysis:

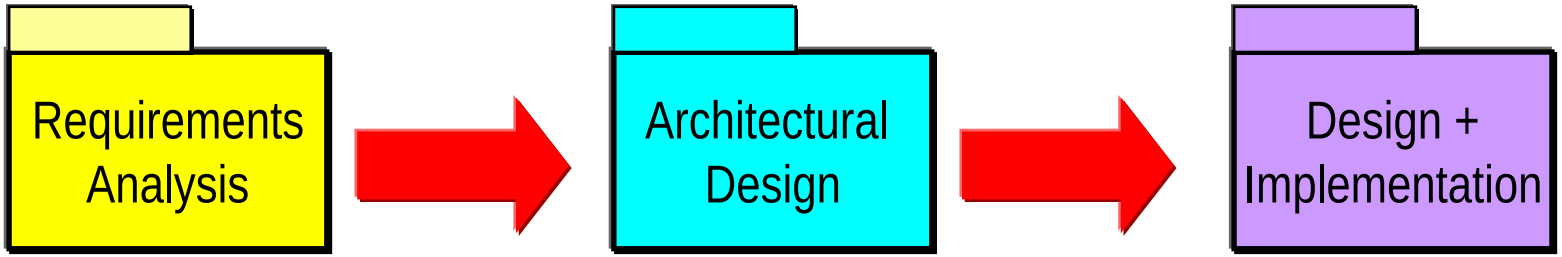
- Goal: (on all Levels)
  - find, avoid Problems, Increase QA (and measure it)
- Micro-Level
  - Code, MIRSA-C
  - E.g: =, ==, {},
- Marco-Level
  - Class-Design, Effective Rules, C++, Java, C#
  - E.g: by reference, String concat, Exception-Handling
- Architecture-Level:
  - Layers, Graphs, Subsystems, Compoments, Interfaces
  - E.g: Coupling, Dependency, etc…

Different kinds of Architecture-Analysis

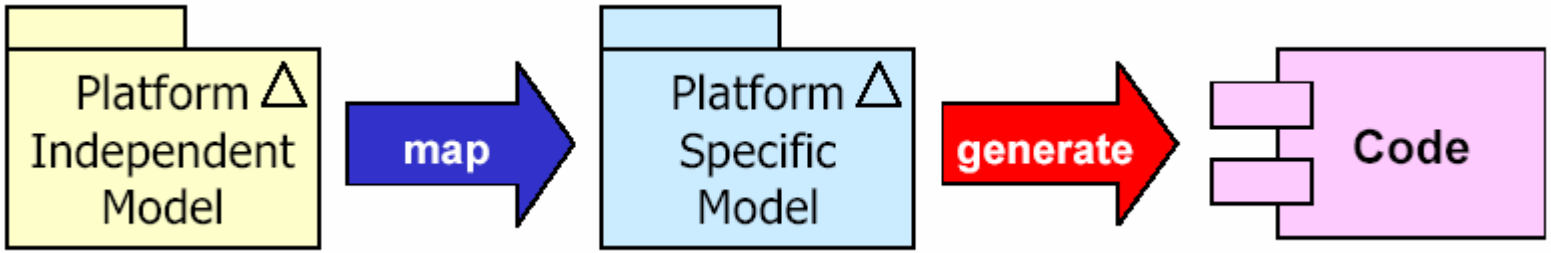
- Consistency-Analysis
- Rating of Architecture
- Discover a Architecture
- Measure real facts (e.g. metrics)
- Monitoring changes, trends (QA)

Consistency-Analysis

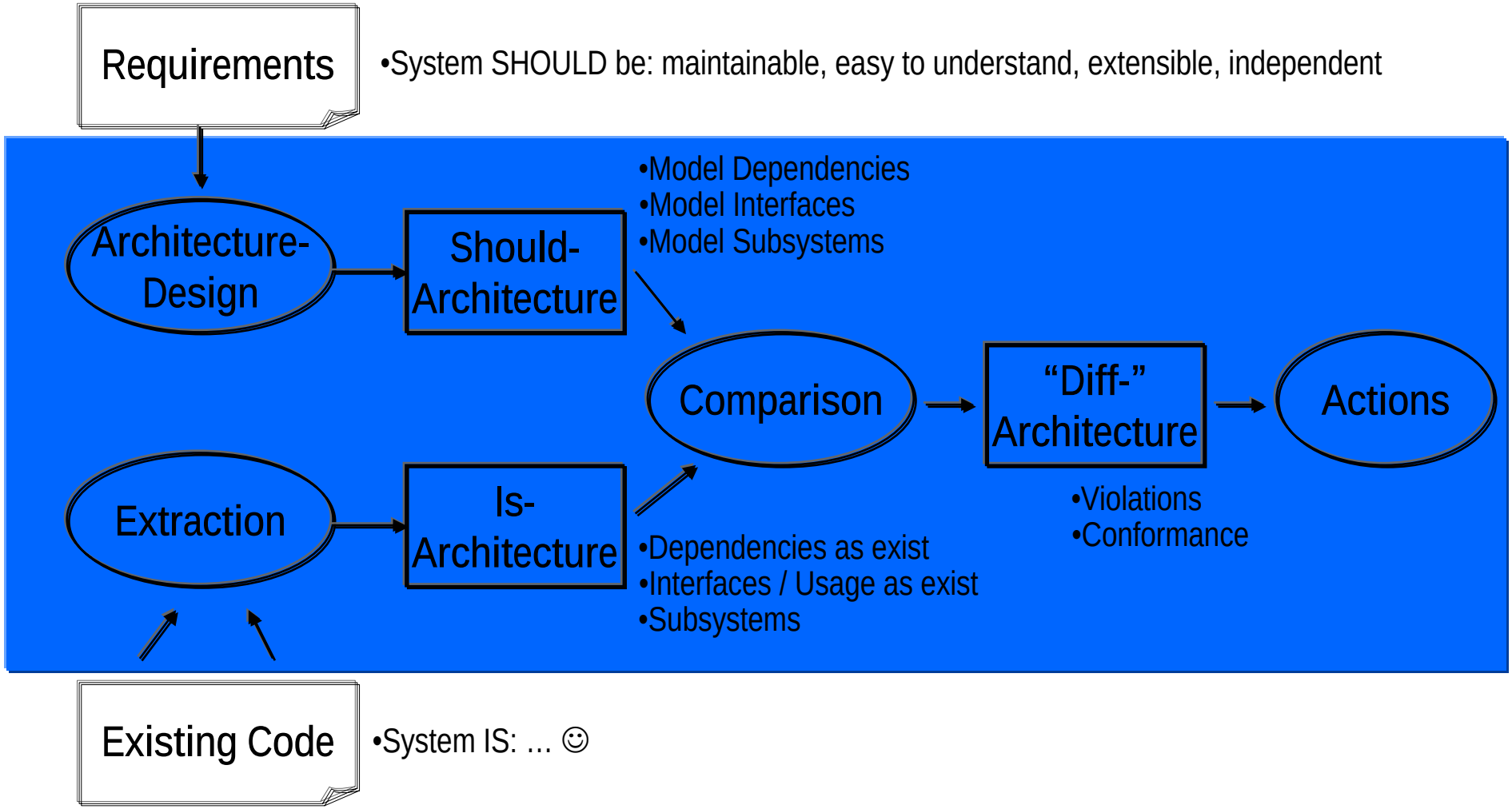
- Aim: No inconsistency



- Dispersion (no toolchain, information loss)
- Declay of Architecture, Rules vilotated, (over project time, various reaons…)
- Deviation Comparison

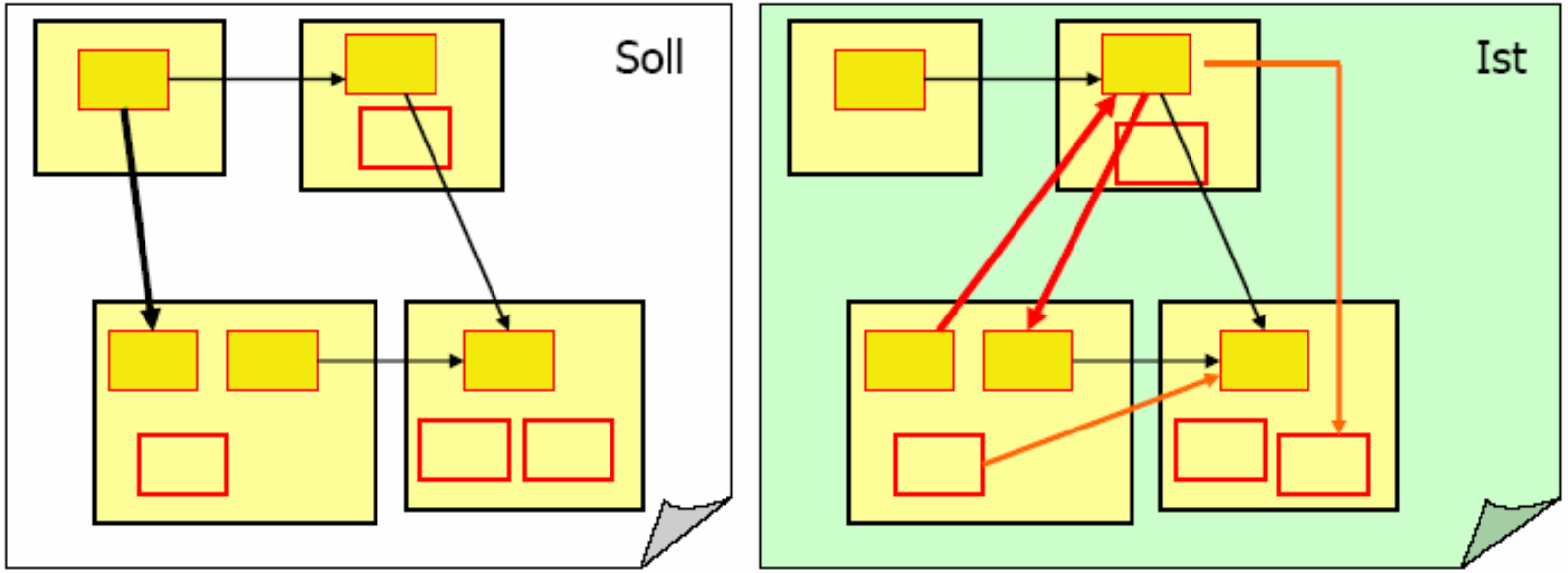


Consistency-Analysis



Consistency-Analysis: Things become VISIBLE

- Results aggregated the right way: (e.g. Subsystem level)



How to cope with violations…

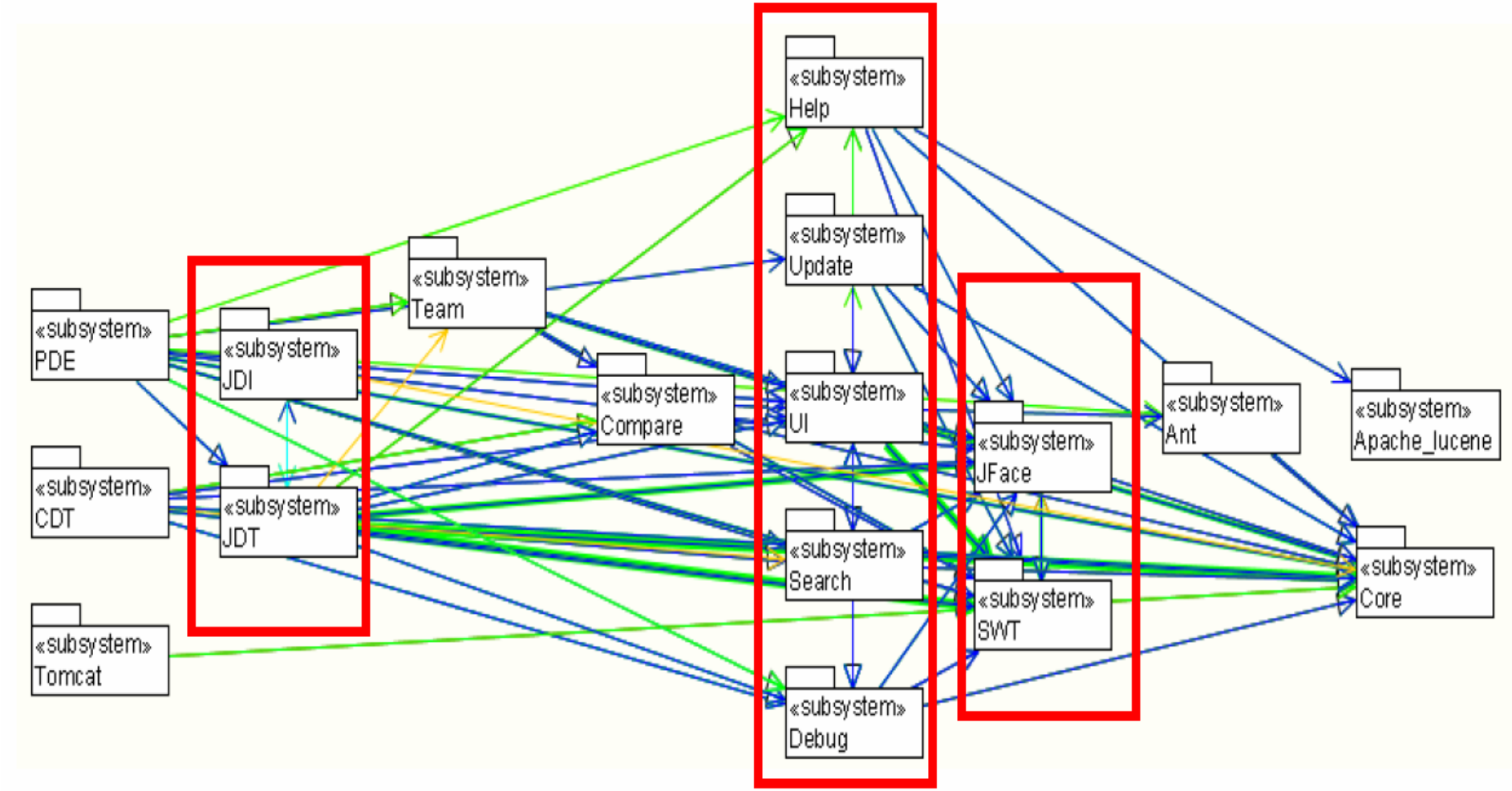
- Identify violations
  - Where
  - Quantity, Quality
  - Heaviness, Impact
- Handling violations
  - Fix possible ? (effort, costs, time)
  - Virtual refactorings, Simulations
  - List with modifications
  - Programmer implements fixes
  - Sometime, “autofix” lint ?… ☺

Rating of Architecture

- NO Rating of external Requirements (Fullfillment)
- Internal Quality (is the focus)
  - Cycles
  - Coupling
  - Stability
  - Anti-Patterns, Bad Smells
- Target:
  - Analyze Problem (and fix) (during project)
  - Compare _different_ Architecture solutions ?

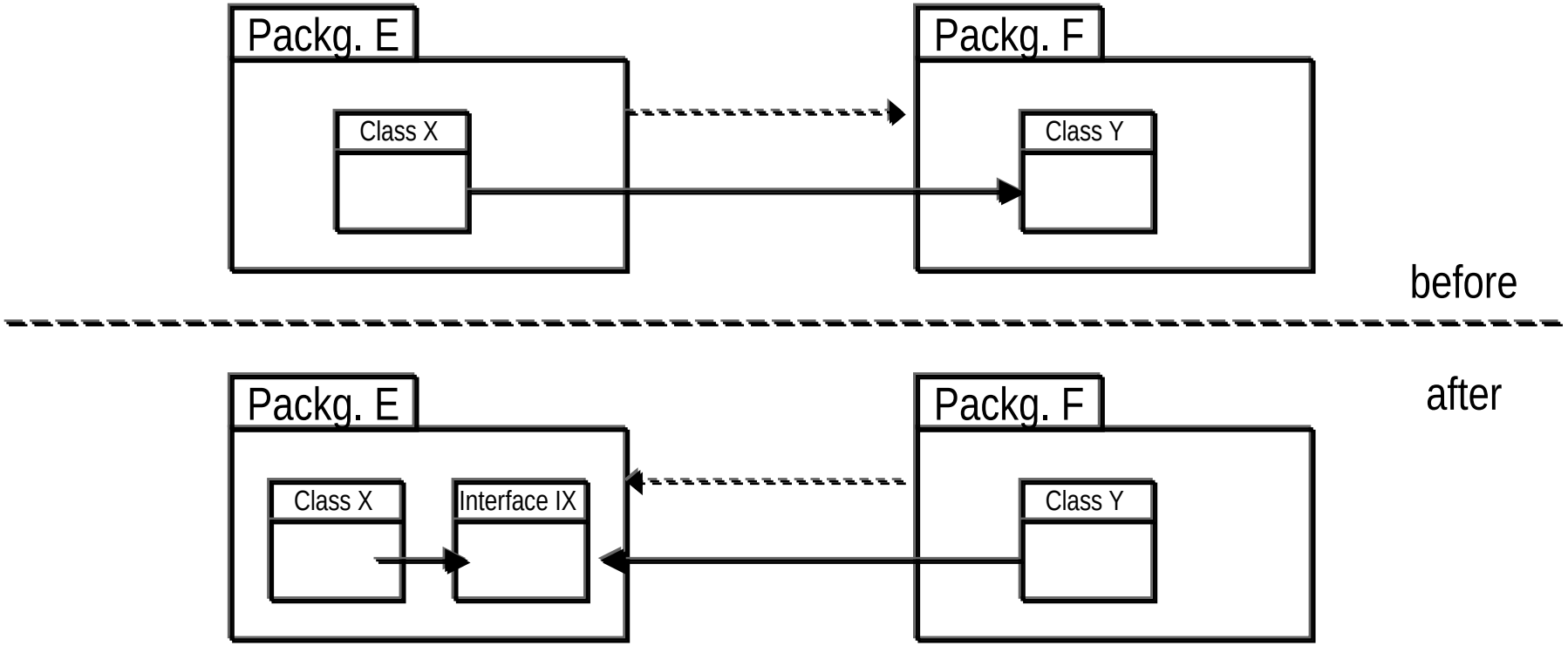
Rating of Architecture: e.g. Cycles

- Handling of Subsystems becomes difficult…



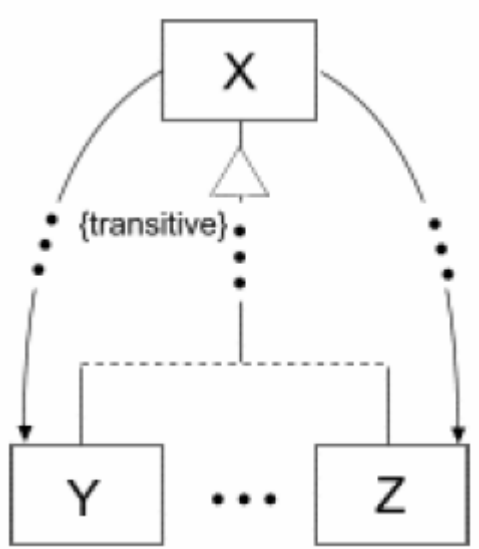
Rating of Architecture: e.g. Coupling

- DIP (Dependency Inversion Principle), R. Martin



Rating of Architecture: e.g. AntiPatterns

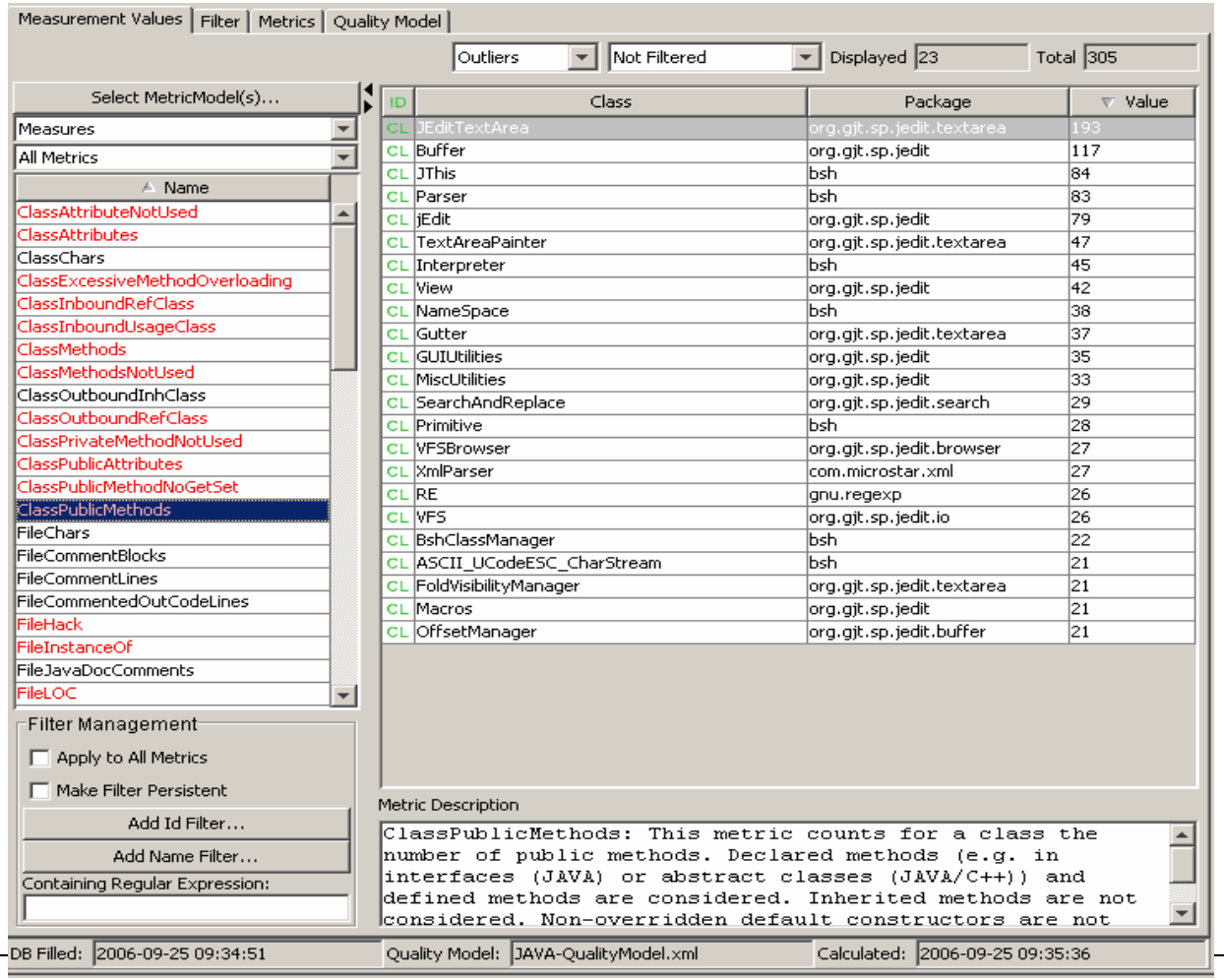
- Dependent BaseClass
  - Type: Design Problem



  - Problem: one of more Methods shall implement different behavior, depending on the type, passed in
  - Context: make “extensible” systems, frameworks
  - Forces: Programming languages offer, instanceof/typeid funcs.
  - Antipattern: Methods of the baseclass, depend on derived classes, e.g. accessing their members, doing switch/case depending on type information

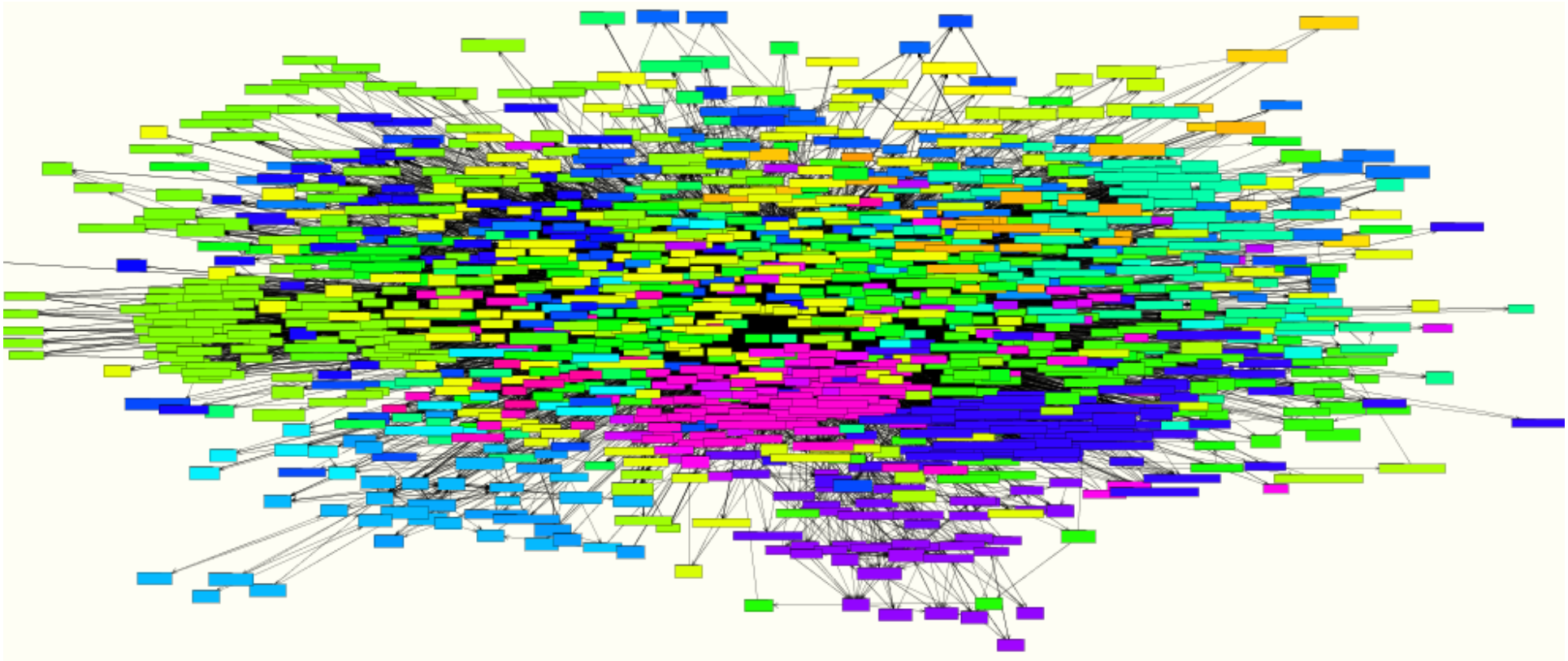
Rating of Architecture: e.g. How to find AntiPatterns

- Dependent Baseclass: 1,5/1000 in Eclipse 2.1, 16/1000 in JDK 1.4.0
- Multiple Interface Inheritance 4/1000 in Eclipse 2.1, 18/1000 in JDK 1.4.0



Rating of Architecture: e.g. How to find AntiPatterns

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



Discover a Architecture (Erosion, prog. understand)

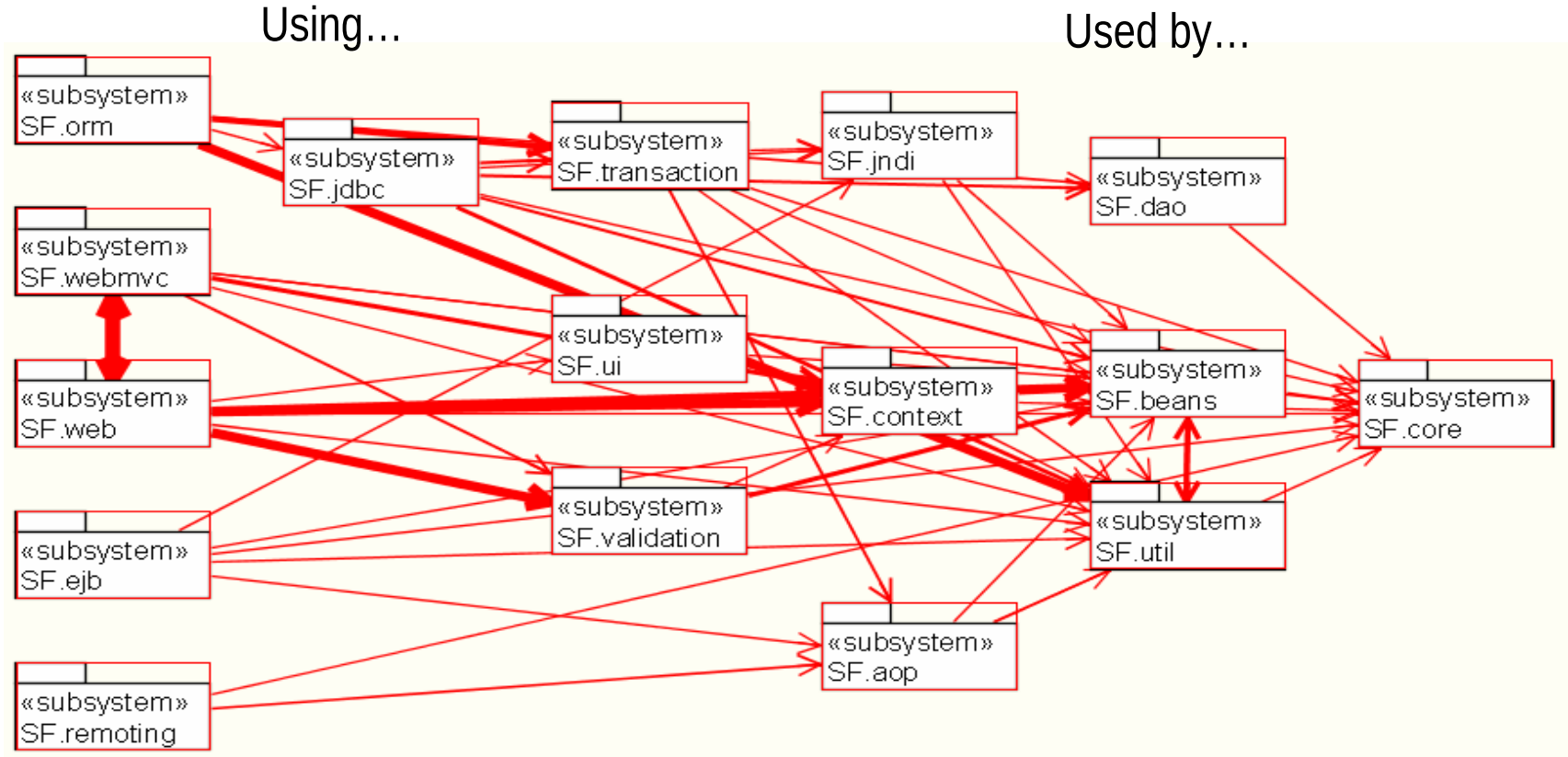
- Visualisation of _existing_ Architecture (Layout !)
  - Architecture often implicit
    - Undocumented
    - new staff in project,
    - Quick Overview of external software
  - Erosion and Analysis
    - Discover central abstractions/key concepts, e.g. Worker-classes
    - Typical Usage of certain artefacts, Patterns
  - Navigation
    - Used from, Using others,…
    - Library dependency ?
    - High-Level Cross Referencer

Discover a Architecture: Questions

- Is there a Software Architecture ?
  - Implicit, explicit
  - Conformance with rules
- Which Architecture Artefacts are there ?
  - Interfaces, Packages, Components, Subsystems, Layers
  - Layer-Architecture, Graph-Architecture,…
- Any Violiations of the Reference/Target-Architecture ?
  - Cycles between xyz…
  - Interface violations between subsystems
  - Bypassing Interfaces

Discover a Architecture: Level of Abstraction

- Topologic sorted layout, only Call-Relationships

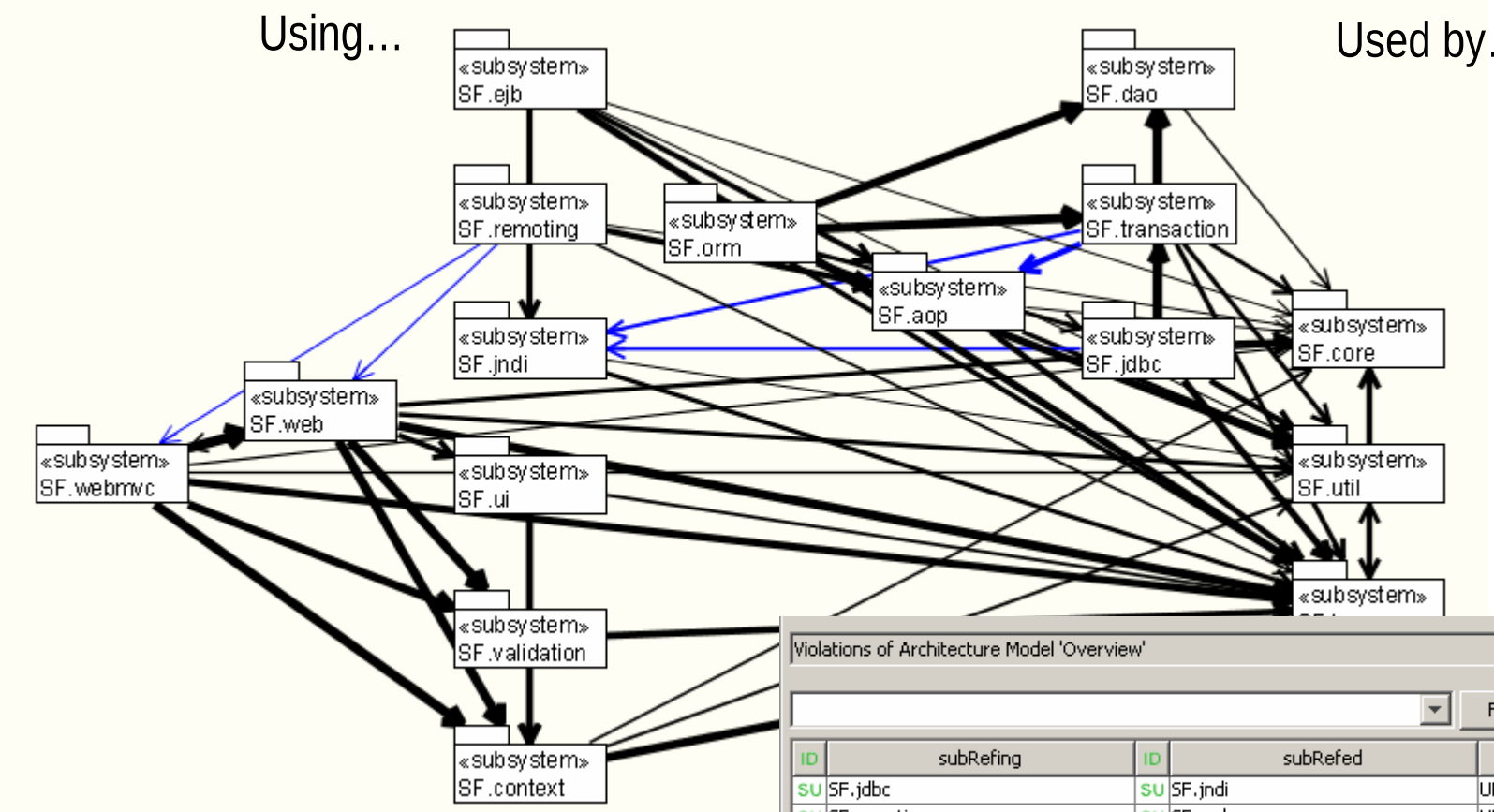


•Business Modules
•Public, exported Methods

•Utility Modules, Infrastruc
•Private, internal Methods

Discover vs.Model a Architecture: Variance comparison

- Arch. sorted layout, only Call-Relationships



Violations of Architecture Model 'Overview'

Filter | Remove Filter

ID	subReFing	ID	subRefed	errorKind	count
SU	SF.jdbc	SU	SF.jndi	UPWARD	5
SU	SF.remoting	SU	SF.webmvc	UPWARD	2
SU	SF.remoting	SU	SF.web	UPWARD	2
SU	SF.transaction	SU	SF.aop	UPWARD	24
SU	SF.transaction	SU	SF.jndi	UPWARD	6

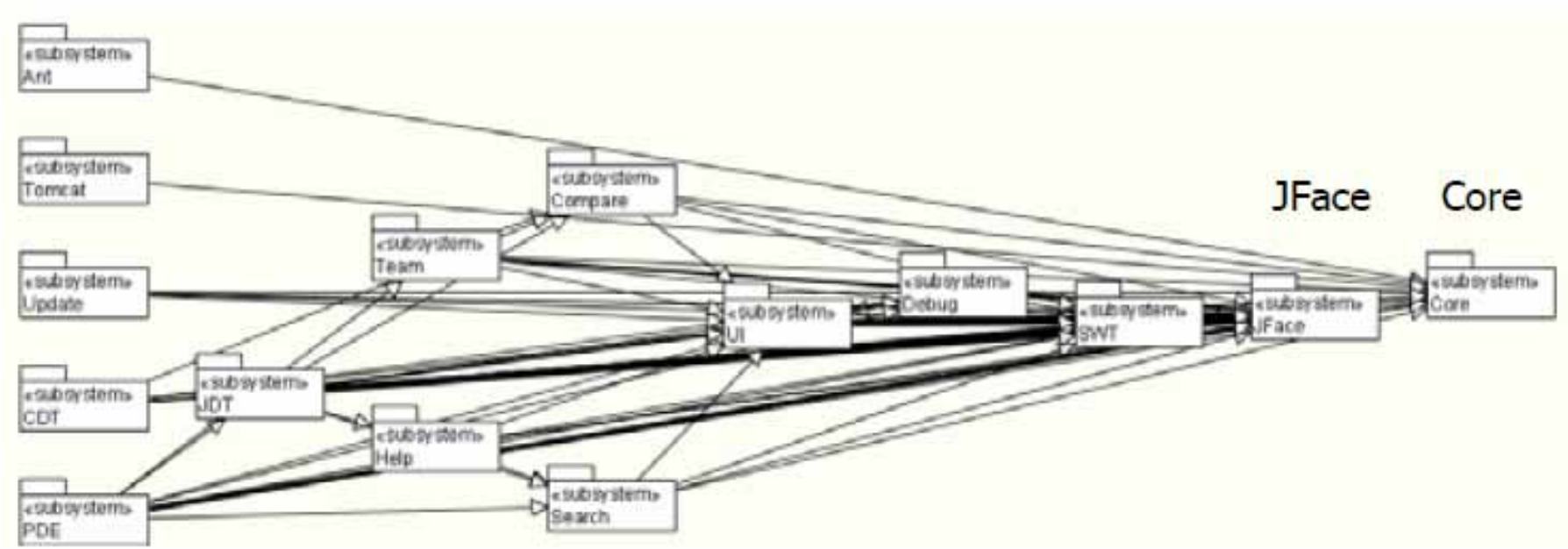
•You SEE the architectural violoations

Discover a Architecture: Level of Abstraction

- Topologic sorted layout, only Inheritance-Relationships

Sub-Classes…

Base-Classes…



•You SEE important Base-Classes….

•E.g from Core:
IWorkspaceRunnable 102x
IAdaptable 100x
IPlatformObject 50x

Measure real facts (e.g. Metrics)

- Metrics are _indicators_ for
  - Quality, Understandability, Maintenance, Error Probability,…
  - Hard facts, measured numbers
- Examples
  - LOC (lines of code)
  - Cyclomatic complexity
  - ACD (average component dependency)
  - Metrics of Robert C. Martin (abstractness, instability etc.)
  - Inheritance depth, overridden/implemented methods,…

Measure real facts (e.g. Metrics)

- Controlled Quantities
  - LOC, #of pakets, files, classes, methods
  - Simple counting of certain artefacts
  - Set a threshold
  - Identify and handle outliers
- Discover candidates which are
  - Sources for bugs, complex, hard to maintain
  - Performance problems
  - Duplicates

Monitoring changes, trends (QA)

- Level Subsystem, Package, File, Class, Operation etc.
  - New artefacts
  - New dependencies
  - New Architecture violations
- Early, betimes correction of viloations
- Monitoring
  - Trendreports
  - “outsouring” projects

Tools for Architecture-Analysis

- Features:
  - Static Analysis → Actual state of Arch
  - Description of Arch Rules → List of violations, deviations

  - Show Dependencies (granularity, number, graph)
  - Simulation of Refactoring, Worklist
  - Metrics
  - Trendanalysis

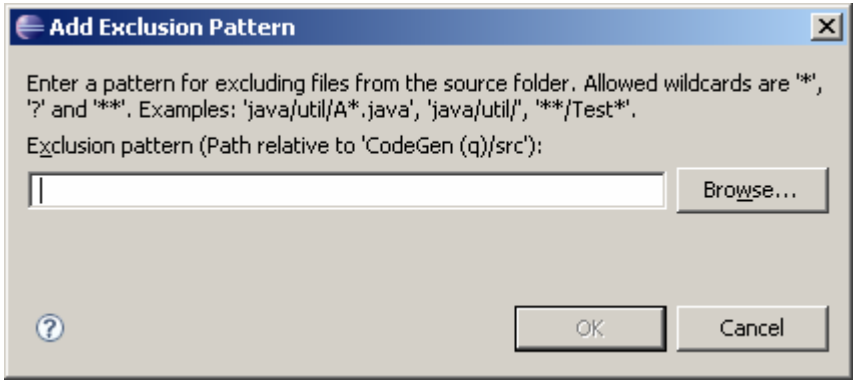
  - IDE-Integration
  - Web-Report
  - Automation, cmd-line

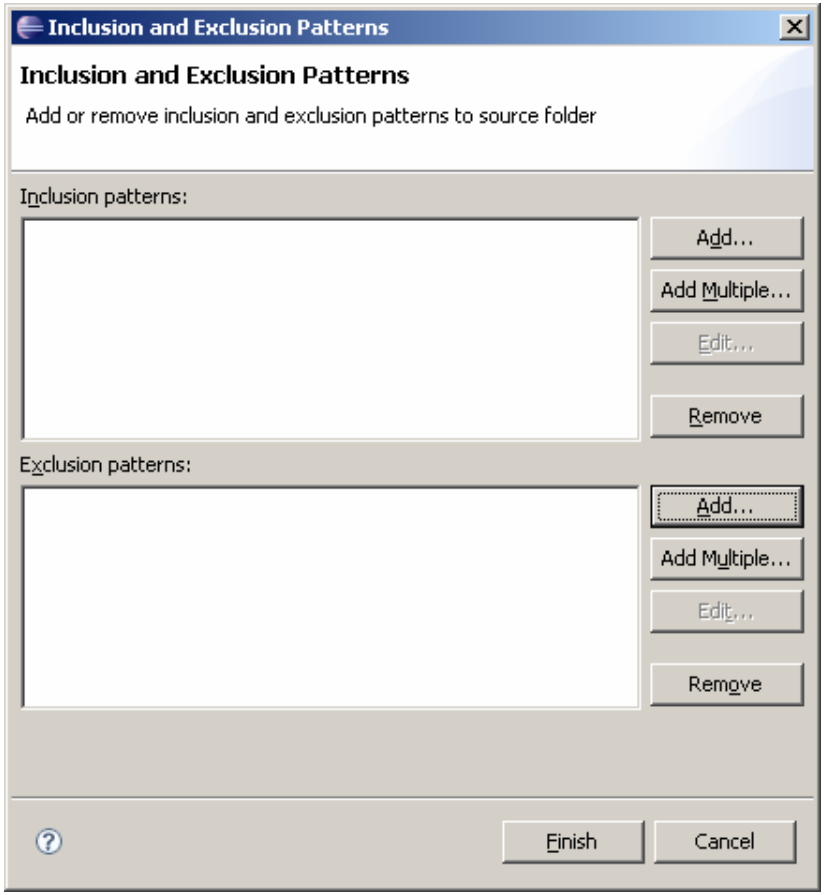
Tools for Architecture-Analysis

- Products:
  - Sotograph: www.software-tomography.de
  - Bauhaus: www.axivion.com
  - SonarJ: www.hello2morrow.de
  - Structure101: www.headwaysoftware.com
  - Lattix: www.lattix.com
  - Klocwork K7: www.klocwork.com
  - XRadar (opensource): www.xradar.org
  - Others: CodeCrawler, SeeSoft, ResourceStandardMetrics

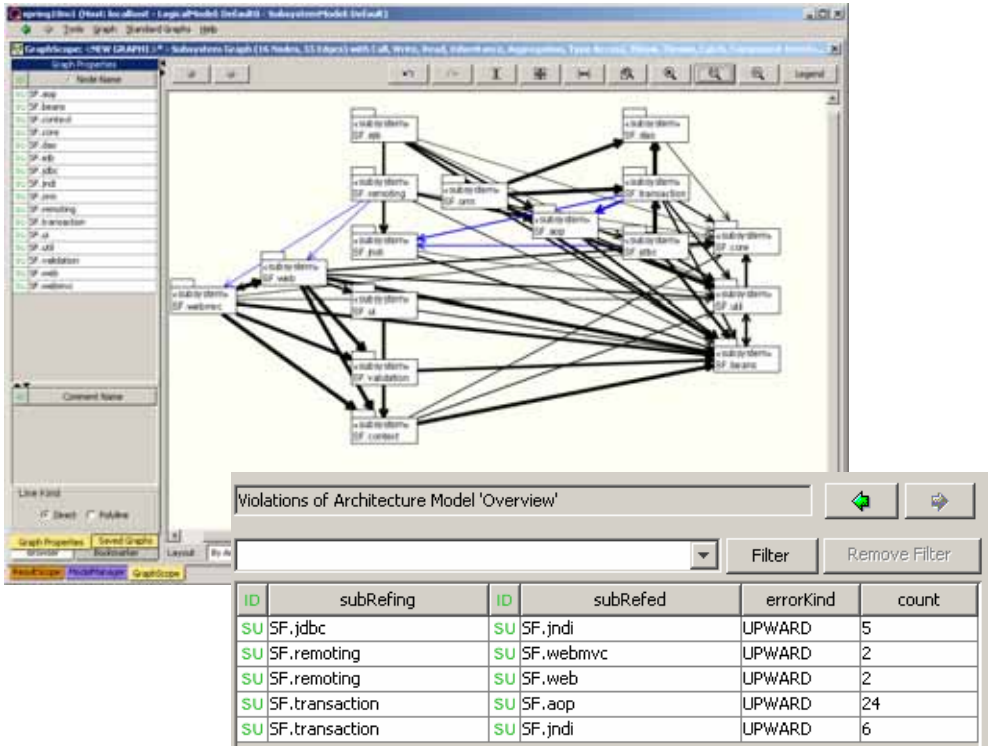
Basic Approaches

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

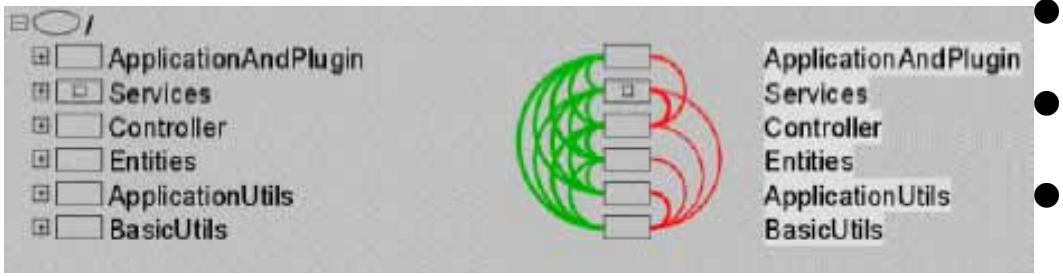




Sotograph: Overview

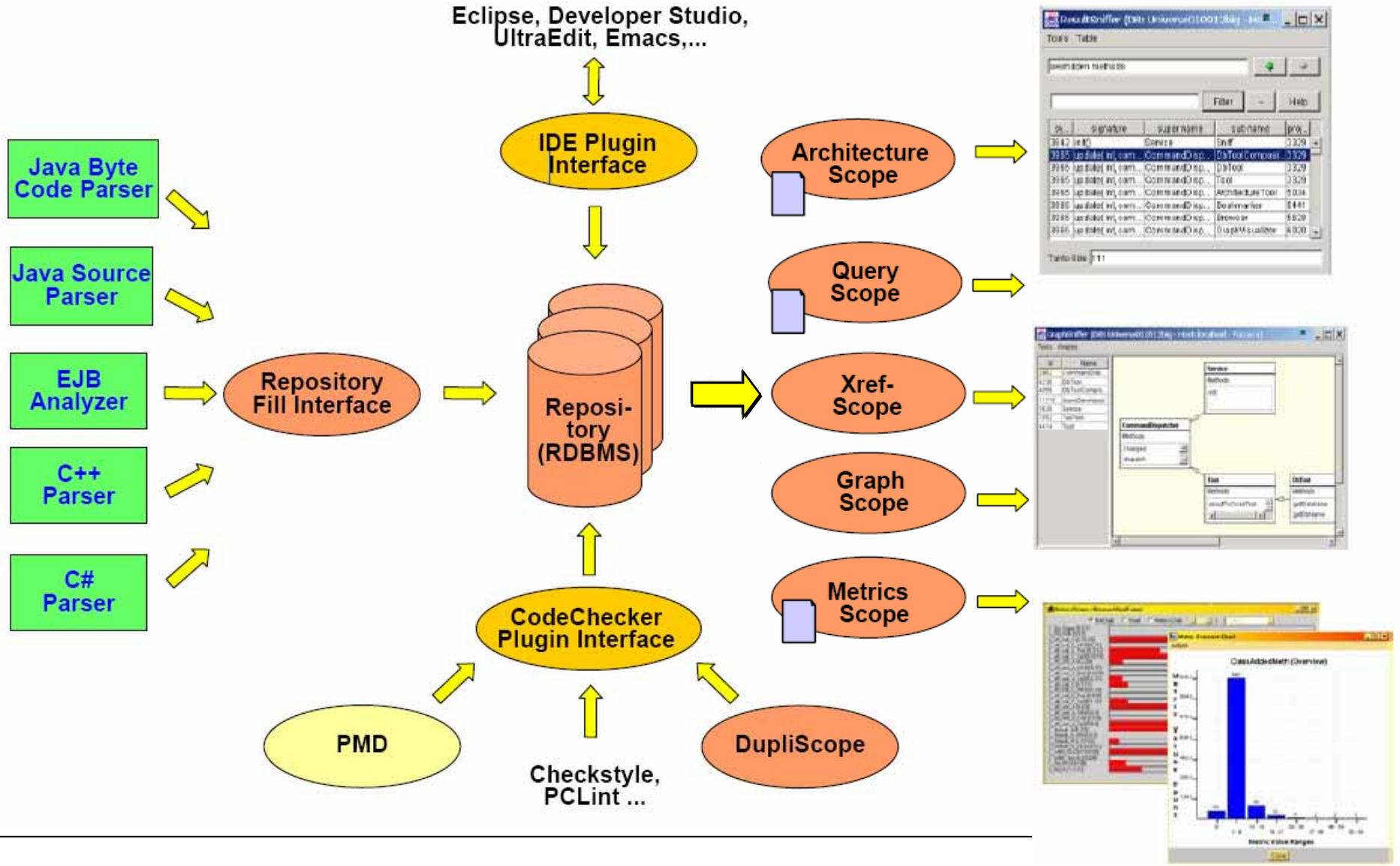


ID	subRefing	ID	subRefed	errorKind	count
SU	SF.jdbc	SU	SF.jndi	UPWARD	5
SU	SF.remoting	SU	SF.webmvc	UPWARD	2
SU	SF.remoting	SU	SF.web	UPWARD	2
SU	SF.transaction	SU	SF.aop	UPWARD	24
SU	SF.transaction	SU	SF.jndi	UPWARD	6

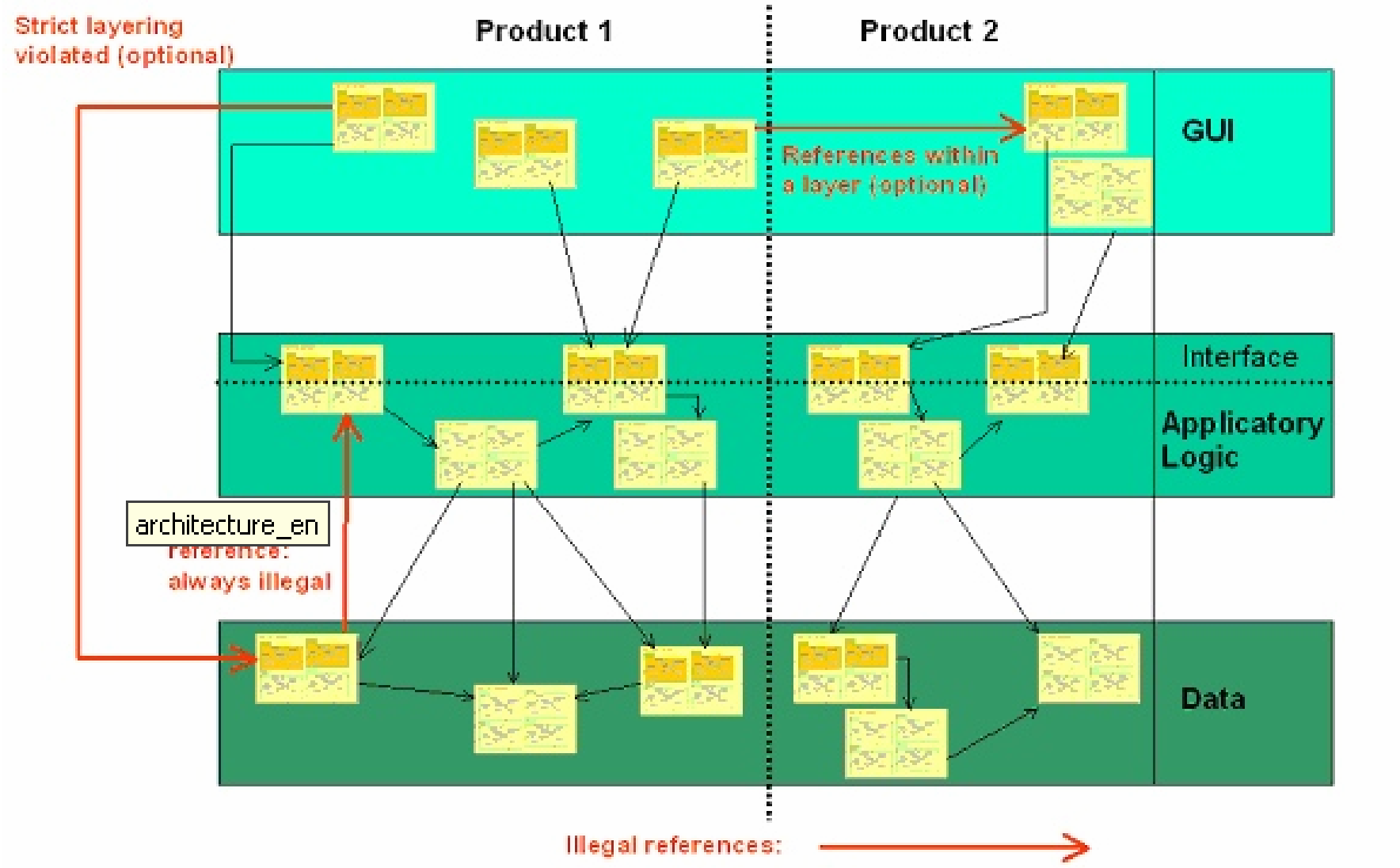


- VERY powerfull
- Infos via Table + Graph
- Cool layout algorithms
- Known since 2003 (NG”SNIFF++”)
- Mysql DB, open schema
- Fat GUI Client, Web Report
- About 200+ Metrics
- Arbitrary User queries
- Trend Analysis
- Virtual Refactoring
- Java, C++, C#, source parser
- Lightweight SotoArch 2007

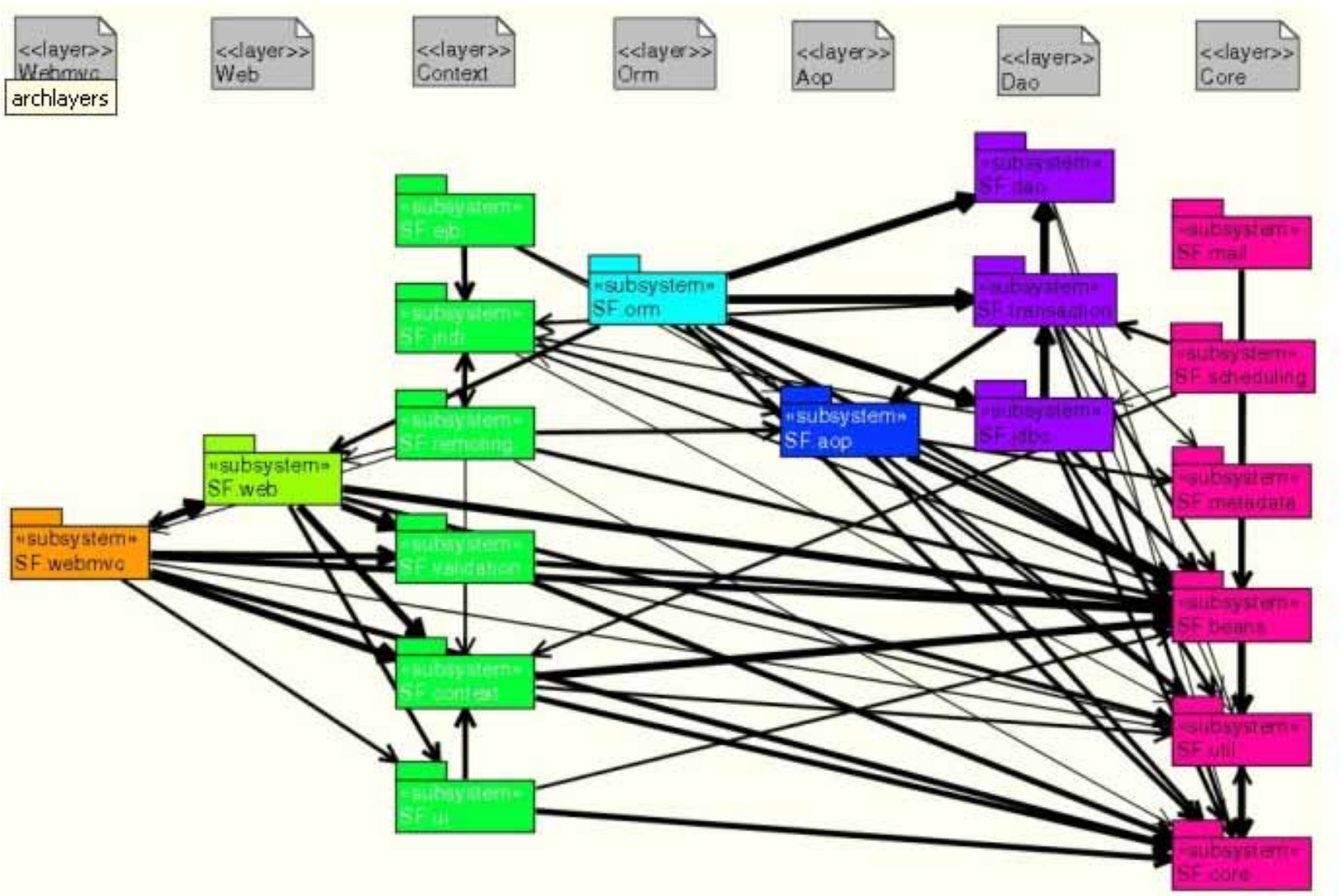
Sotograph


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)
CodeChecker Plugin Interface
PMD
Checkstyle, PCLint ...
DupliScope
Architecture Scope
Query Scope
Xref-Scope
Graph Scope
Metrics Scope

Sotograph: Source and Architecture

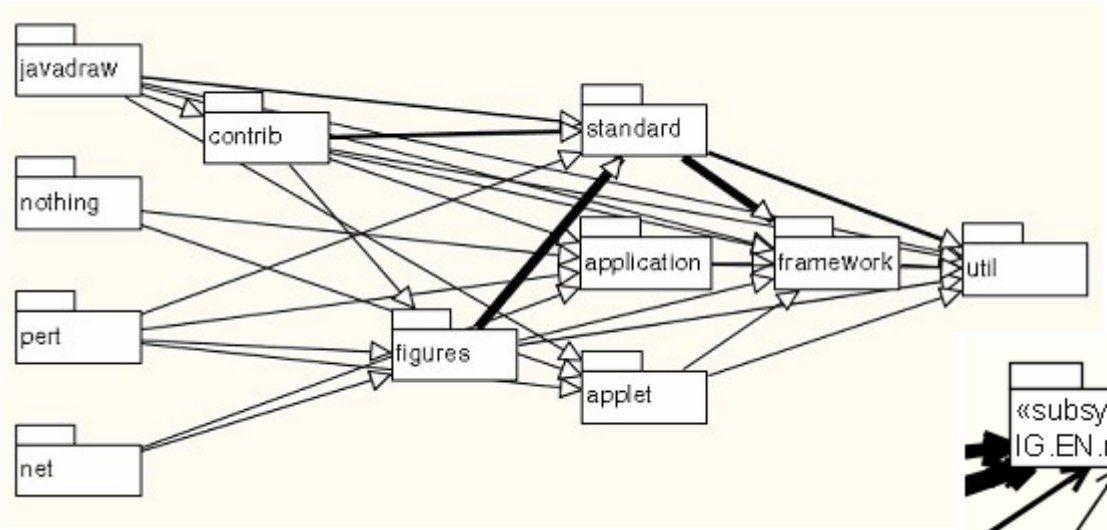


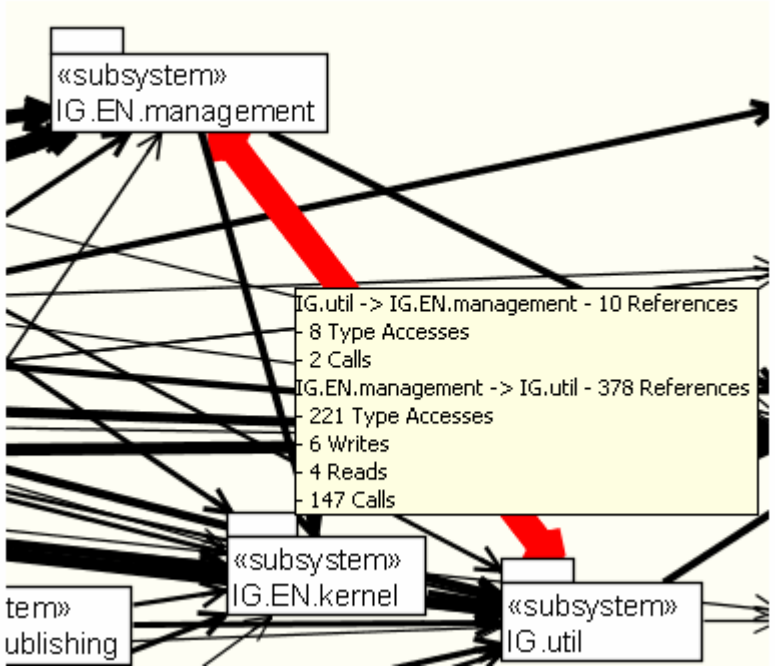
Sotograph: Structure and Relationships



Sotograph: Structure and Relationships

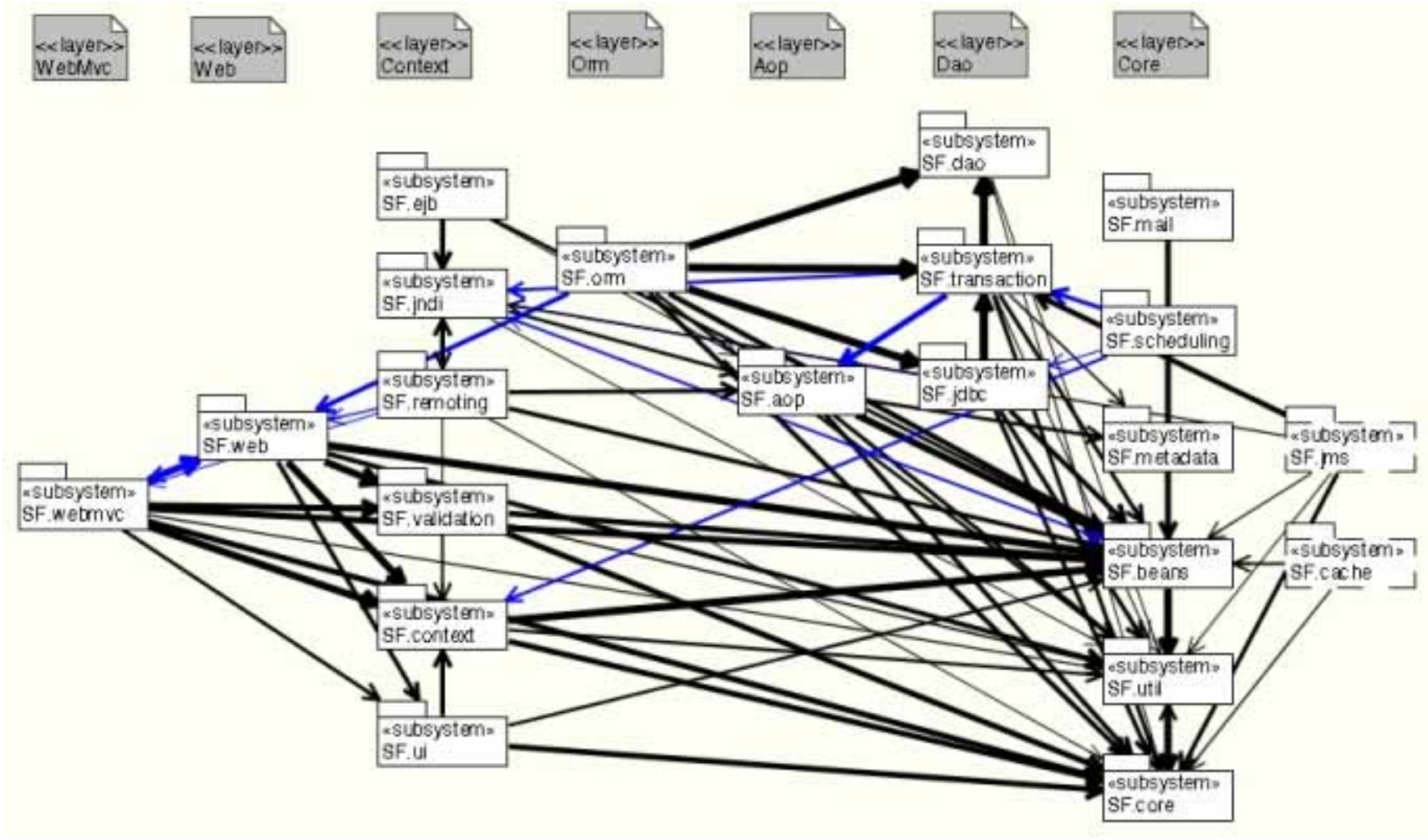
- Depenencies: Informations…





Sotograph: Check Arch. Conformance and Quality

- Arch. violations



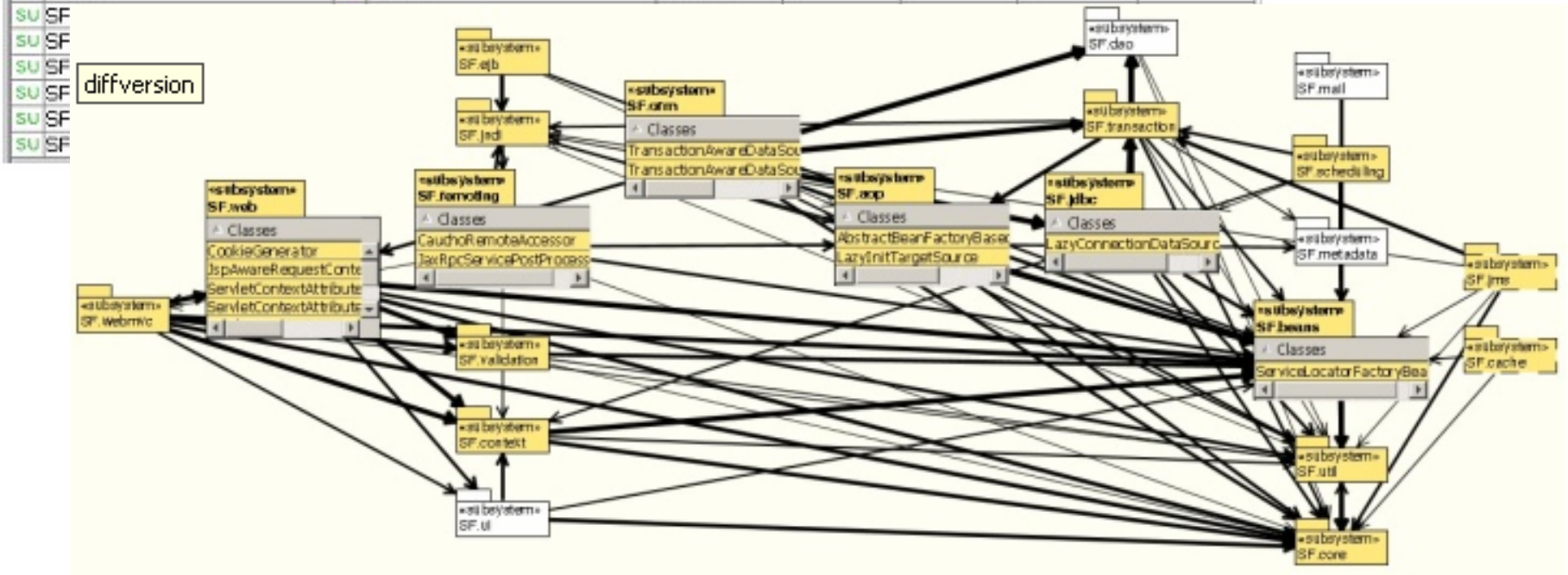
Sotograph: Monitoring Changes

- of Architecture, Quality, Structure

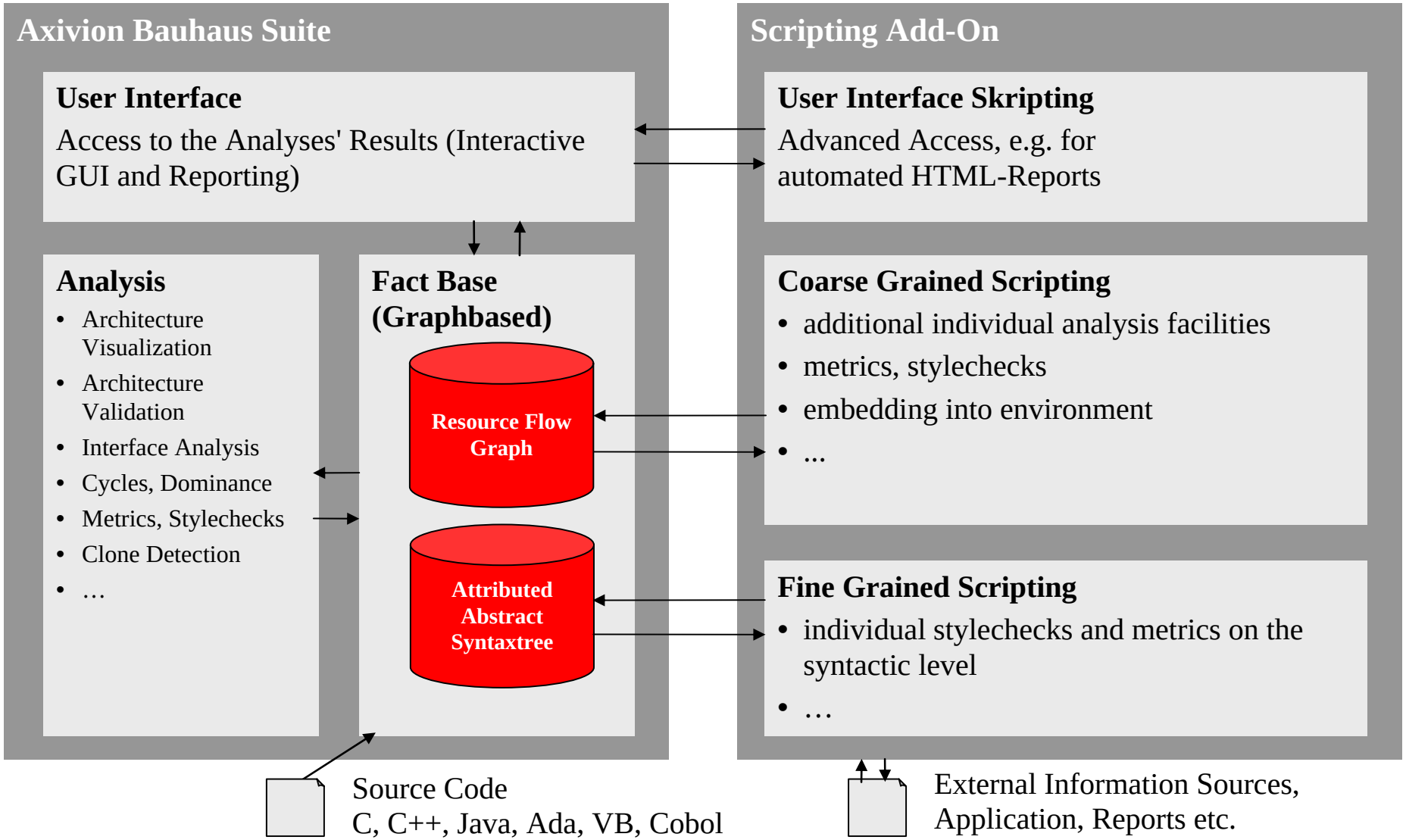
Violations of Architecture Model 'Overview' for v1 = spring20041212_113 and v2 = spring20050131_114

Filter | Remove Filter

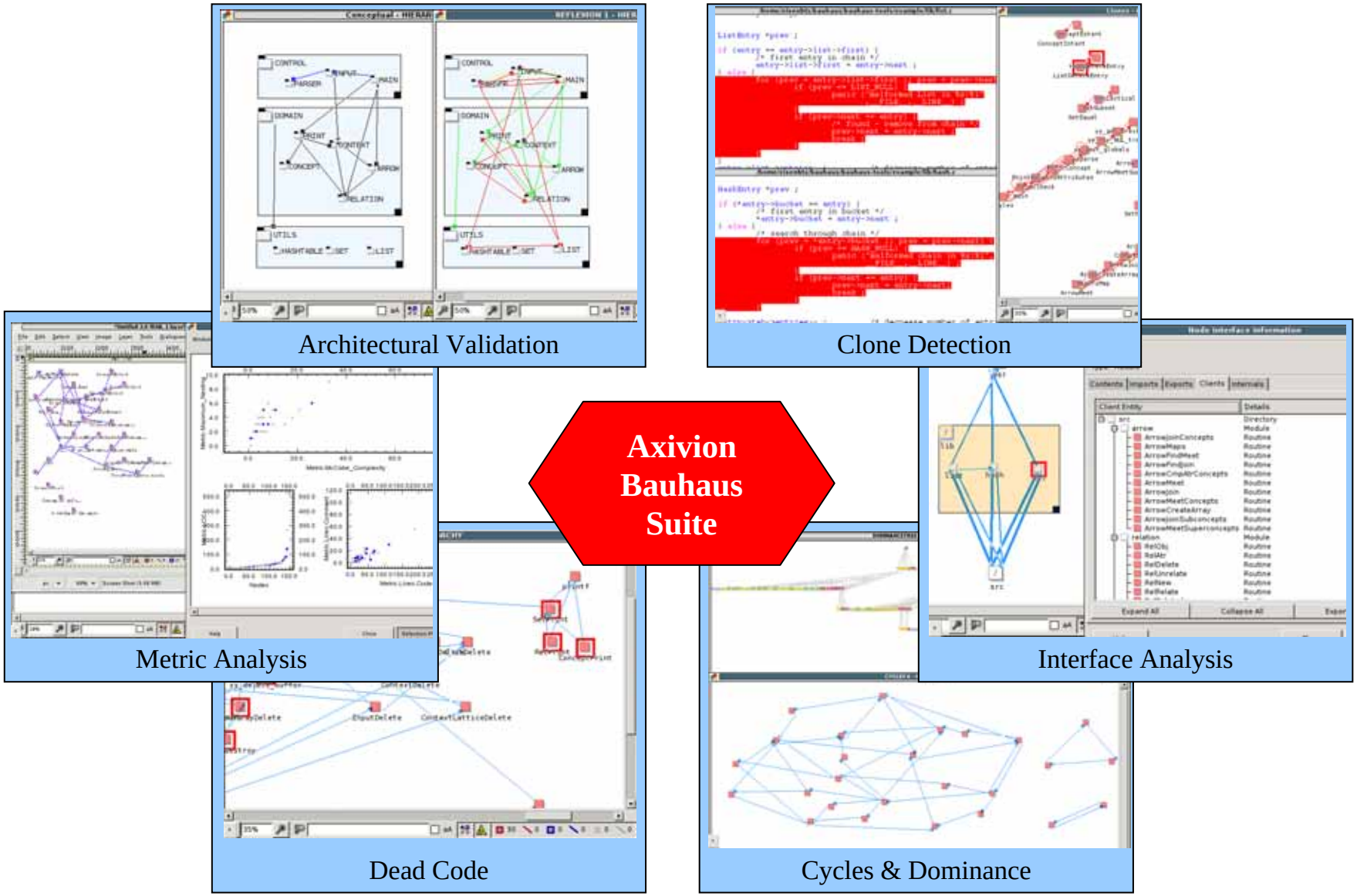
ID	subRefing	ID	subRefed	errorKind	v1	v2	diff	changes
SU	SF.transaction	SU	SF.jndi	UPWARD	10	10	0	
SU	SF.transaction	SU	SF.aop	UPWARD	57	60	3	✘
SU	SF.remoting	SU	SF.webmvc	UPWARD	3	3	0	
SU	SF.remoting	SU	SF.web	UPWARD	4	4	0	
SU	SF.orm	SU	SF.web	UPWARD	25	25	0	
SU	SF							
SU	SF							
SU	SF							
SU	SF							
SU	SF							
SU	SF							



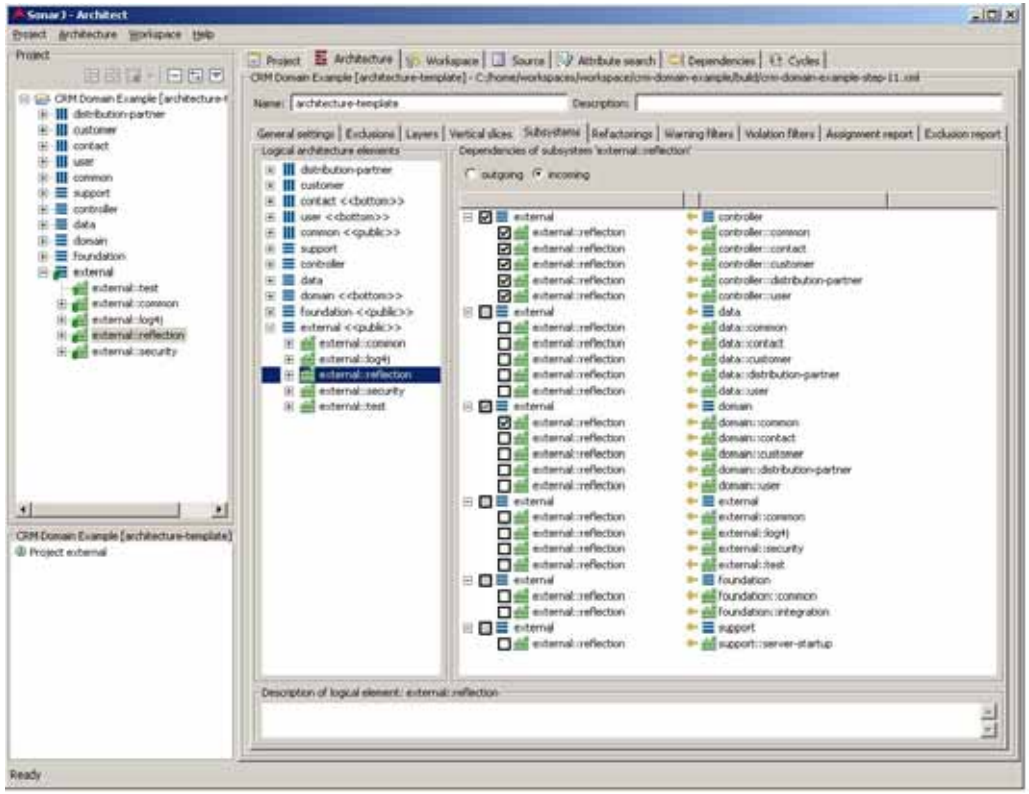
Axivion Bauhaus Suite



Axivion Bauhaus Suite



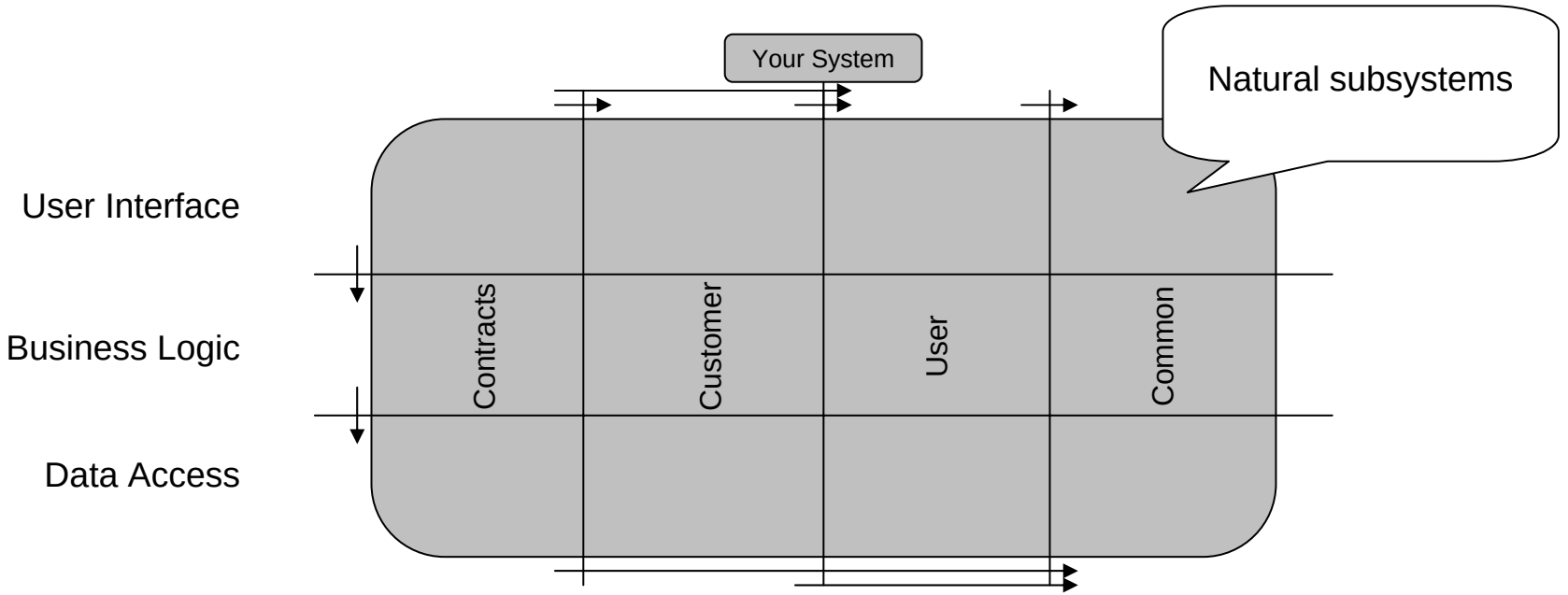
SonarJ: Overview



- Java centric
- Infos via Tables
- No graphs
- Known since 2005
- “In memory DB”
- Good Eclipse-Pluging
- Lightweight approach

SonarJ: Architecture-MetaModel

- Architecture-MetaModel:

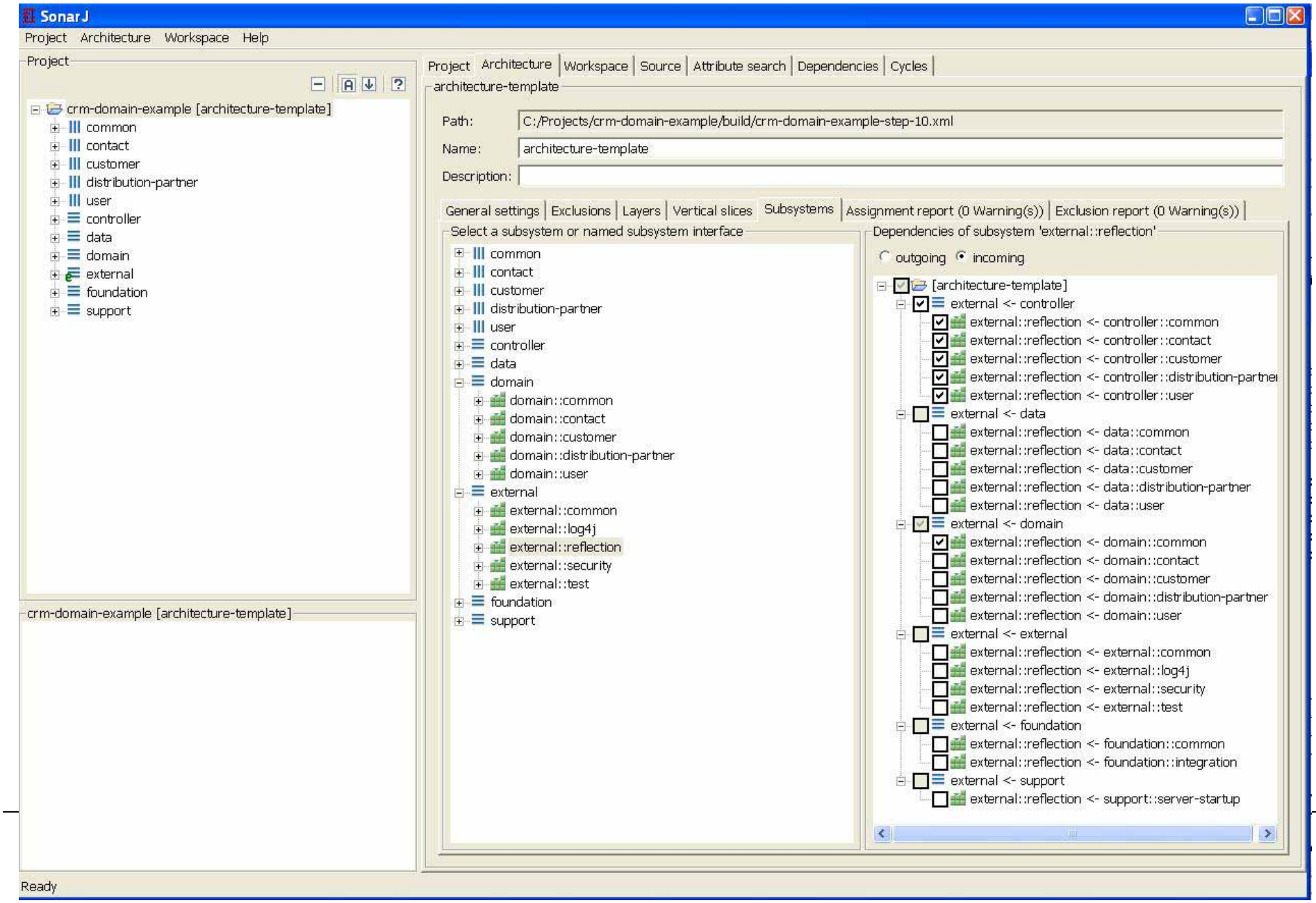


- Step 1: Cut horizontally into Layers
- Step 2: Cut vertically into vertical slices by functional aspects
- Step 3: Defines the rules of engagement

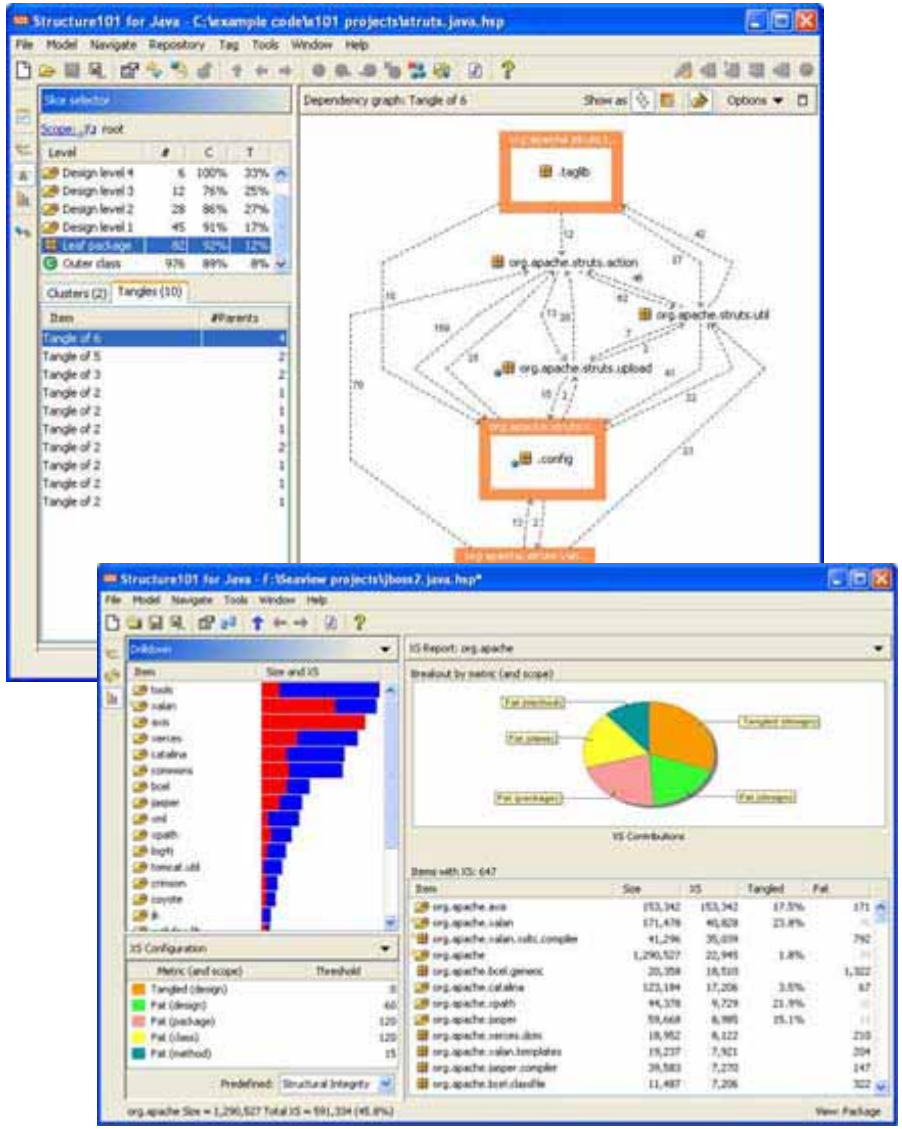
SonarJ: Architecture-MetaModel

- Meta model: layers, vertical slices and subsystems
  - Each subsystem belongs to exactly one layer
  - A subsystem also might belong to a vertical slice
  - The association between vertical slices and subsystems is typically implemented by a naming convention
  - Vertical slices do not have to be present on every layer
  - Technical subsystems typically are not associated with any vertical slice
  - Technical systems often do not have vertical slices at all

SonarJ


SonarJ
Project Architecture Workspace Help
Project
crm-domain-example [architecture-template]
common
contact
customer
distribution-partner
user
controller
data
domain
external
foundation
support
crm-domain-example [architecture-template]
Project Architecture Workspace Source Attribute search Dependencies Cycles
architecture-template
Path: C:/Projects/crm-domain-example/build/crm-domain-example-step-10.xml
Name: architecture-template
Description:
General settings Exclusions Layers Vertical slices Subsystems Assignment report (0 Warning(s)) Exclusion report (0 Warning(s))
Select a subsystem or named subsystem interface
common
contact
customer
distribution-partner
user
controller
data
domain
domain::common
domain::contact
domain::customer
domain::distribution-partner
domain::user
external
external::common
external::log4j
external::reflection
external::security
external::test
foundation
support
Dependencies of subsystem 'external::reflection'
outgoing
incoming
[architecture-template]
external <- controller
external::reflection <- controller::common
external::reflection <- controller::contact
external::reflection <- controller::customer
external::reflection <- controller::distribution-partner
external::reflection <- controller::user
external <- data
external::reflection <- data::common
external::reflection <- data::contact
external::reflection <- data::customer
external::reflection <- data::distribution-partner
external::reflection <- data::user
external <- domain
external::reflection <- domain::common
external::reflection <- domain::contact
external::reflection <- domain::customer
external::reflection <- domain::distribution-partner
external::reflection <- domain::user
external <- external
external::reflection <- external::common
external::reflection <- external::log4j
external::reflection <- external::security
external::reflection <- external::test
external <- foundation
external::reflection <- foundation::common
external::reflection <- foundation::integration
external <- support
external::reflection <- support::server-startup
Ready

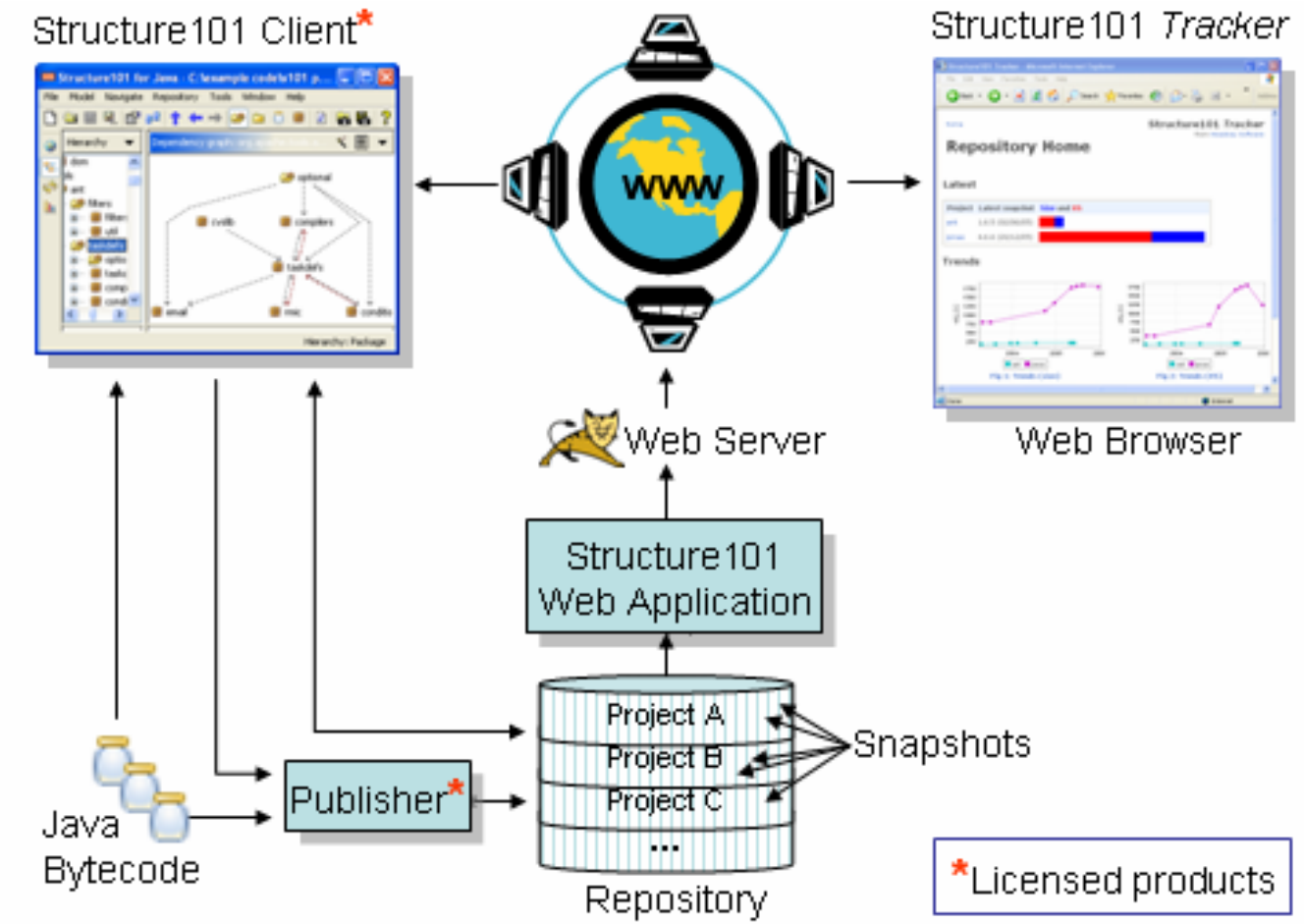
Structure101: Overview



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

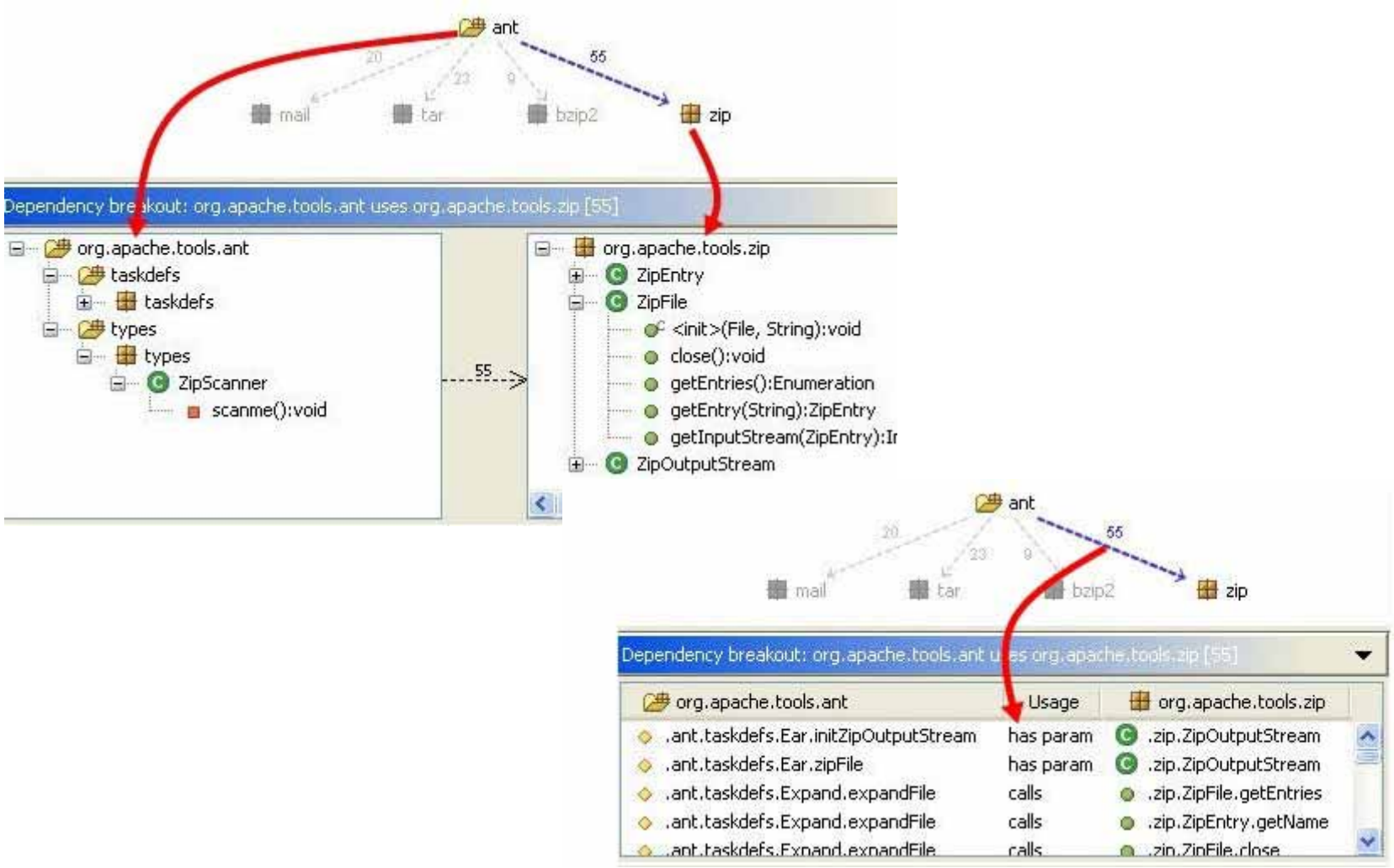
Structure101

- Structure101 Architecture

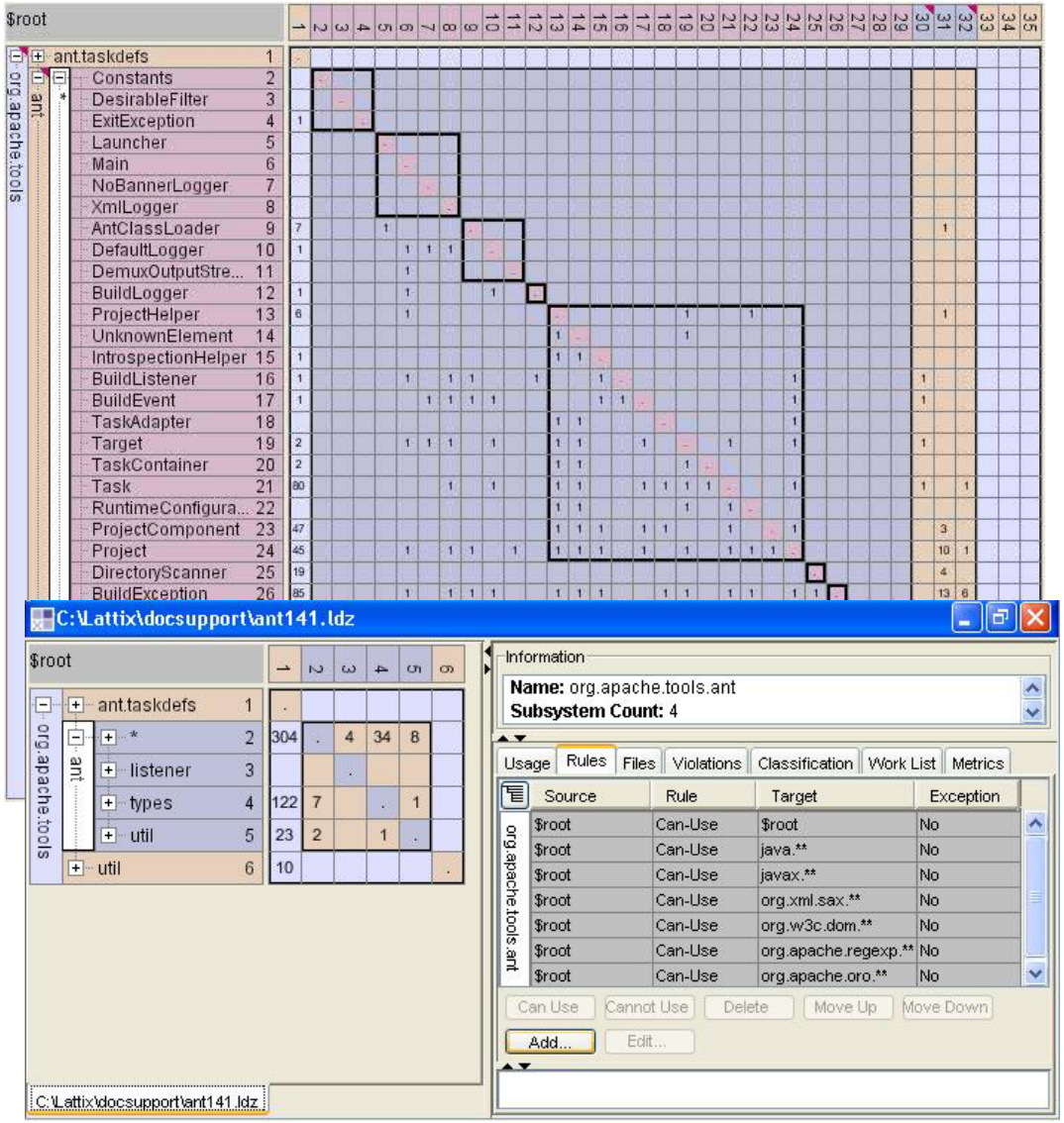


Structure101: Architecture Visualization

- Dependency Analyse



Lattix: Overview



- Java, (C++ via BSC, doxygen)
- Infos via DSM
- No graphics (or weak)
- Known since 2004
- “In memory DB”
- Lightweight approach
- Fat client
- Trend via cmd line
  + own report

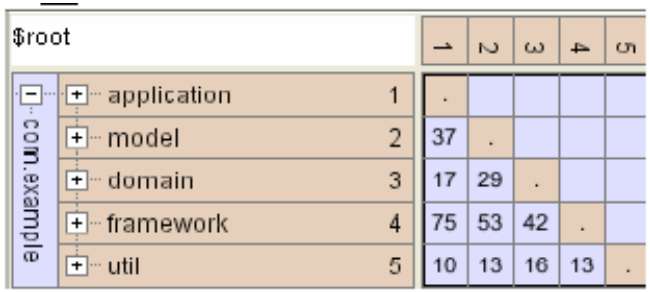
Lattix: DSM Principle

- Artefacts (e.g. Subsystems, Packages, Types, etc.) are displayed in Matrix
  - Colums show “using-” relations
  - Rows show “is used from-” relations
- Artefacts can be
  - Grouped in Subsystems, Layers
  - Arranged hierarchically
- Architecture State can be read via Matrix

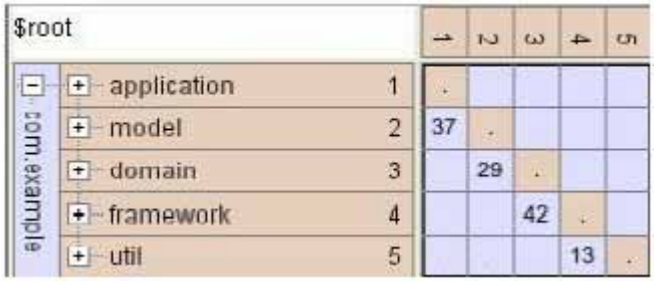
- Partitioning algorithms can identifiy highly coupled artefacts
- Rules for allowed/forbidden Relationsships

Lattix: DSM Examples

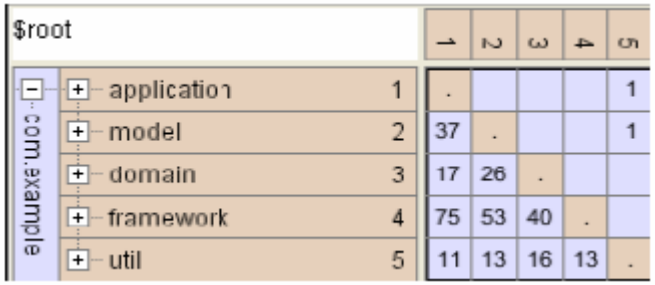
- Example Architectures, for direct reading from Matrix



$root		1	2	3	4	5
com.example	application 1	.				
	model 2	37	.			
	domain 3	17	29	.		
	framework 4	75	53	42	.	
	util 5	10	13	16	13	.

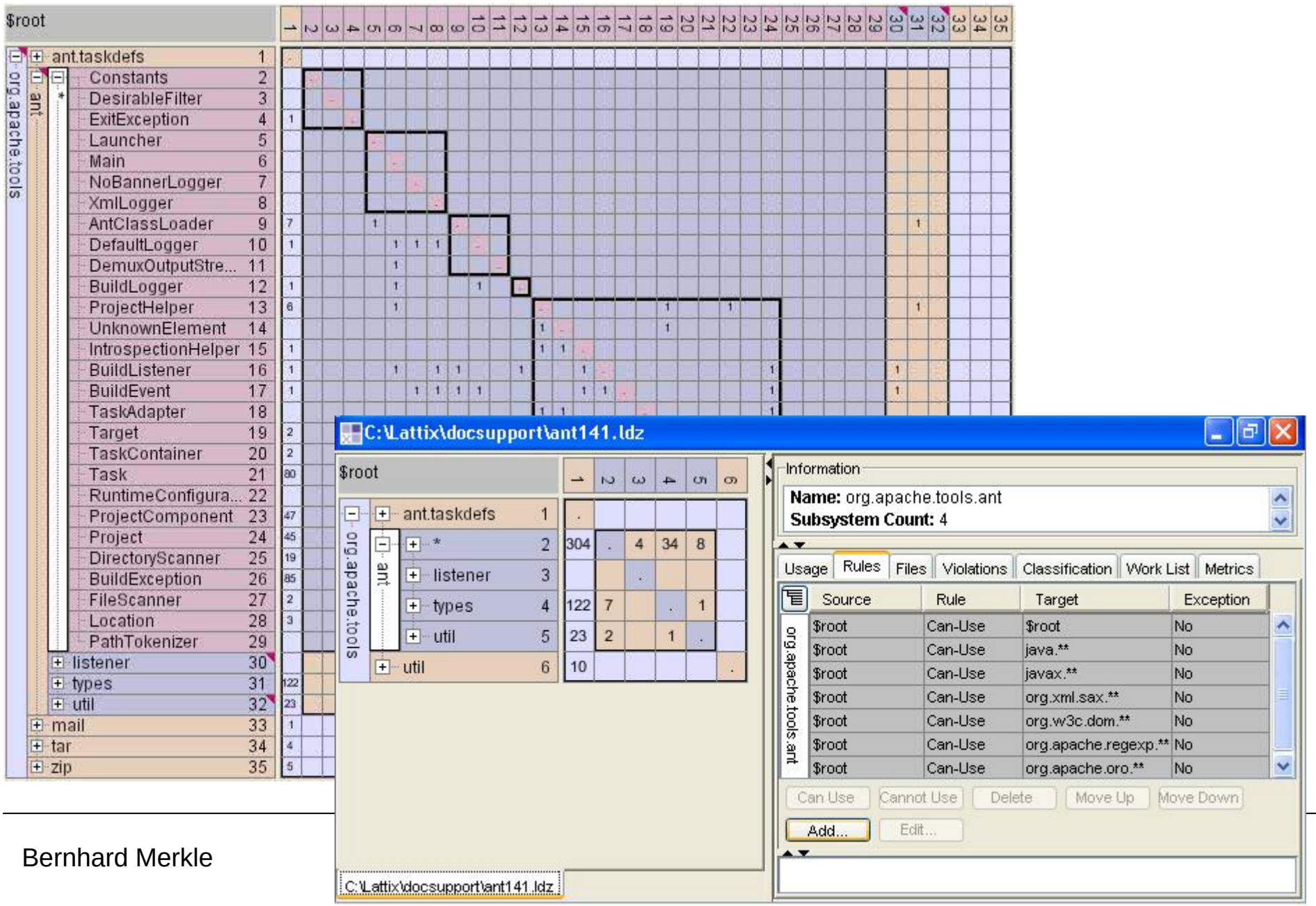


$root		1	2	3	4	5
com.example	application 1	.				
	model 2	37	.			
	domain 3		29	.		
	framework 4			42	.	
	util 5				13	.

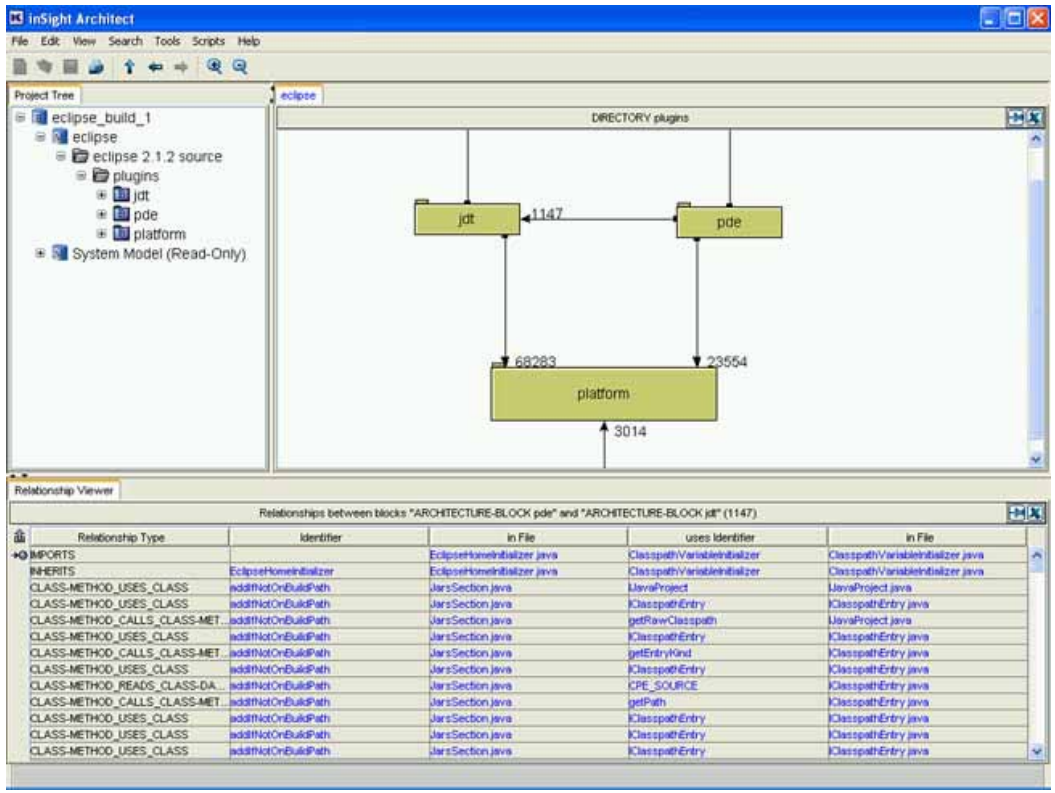


$root		1	2	3	4	5
com.example	application 1	.				1
	model 2	37	.			1
	domain 3	17	26	.		
	framework 4	75	53	40	.	
	util 5	11	13	16	13	.

Lattix: Rules and Partitionierung

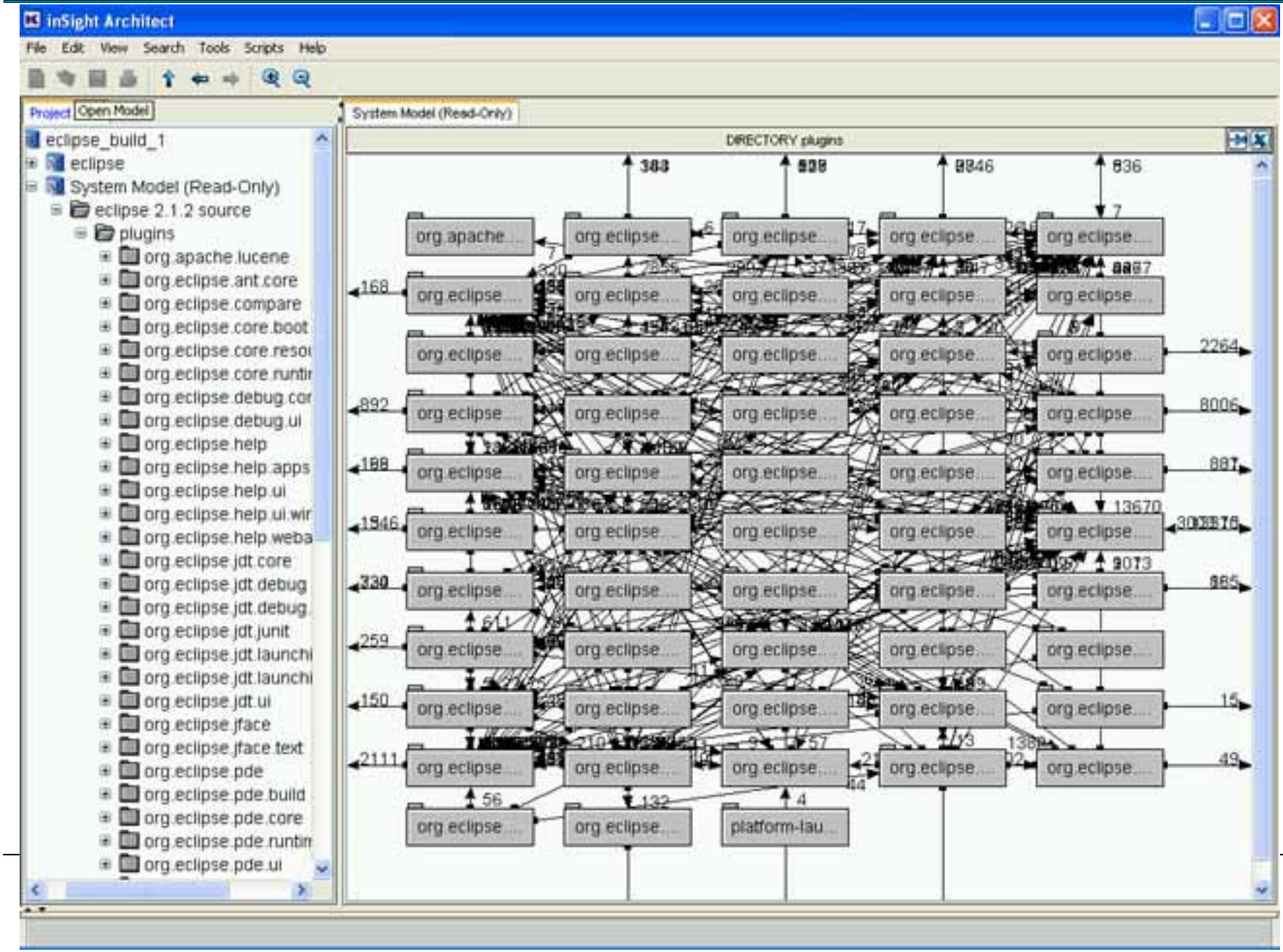


Klocwork: K7 Overview

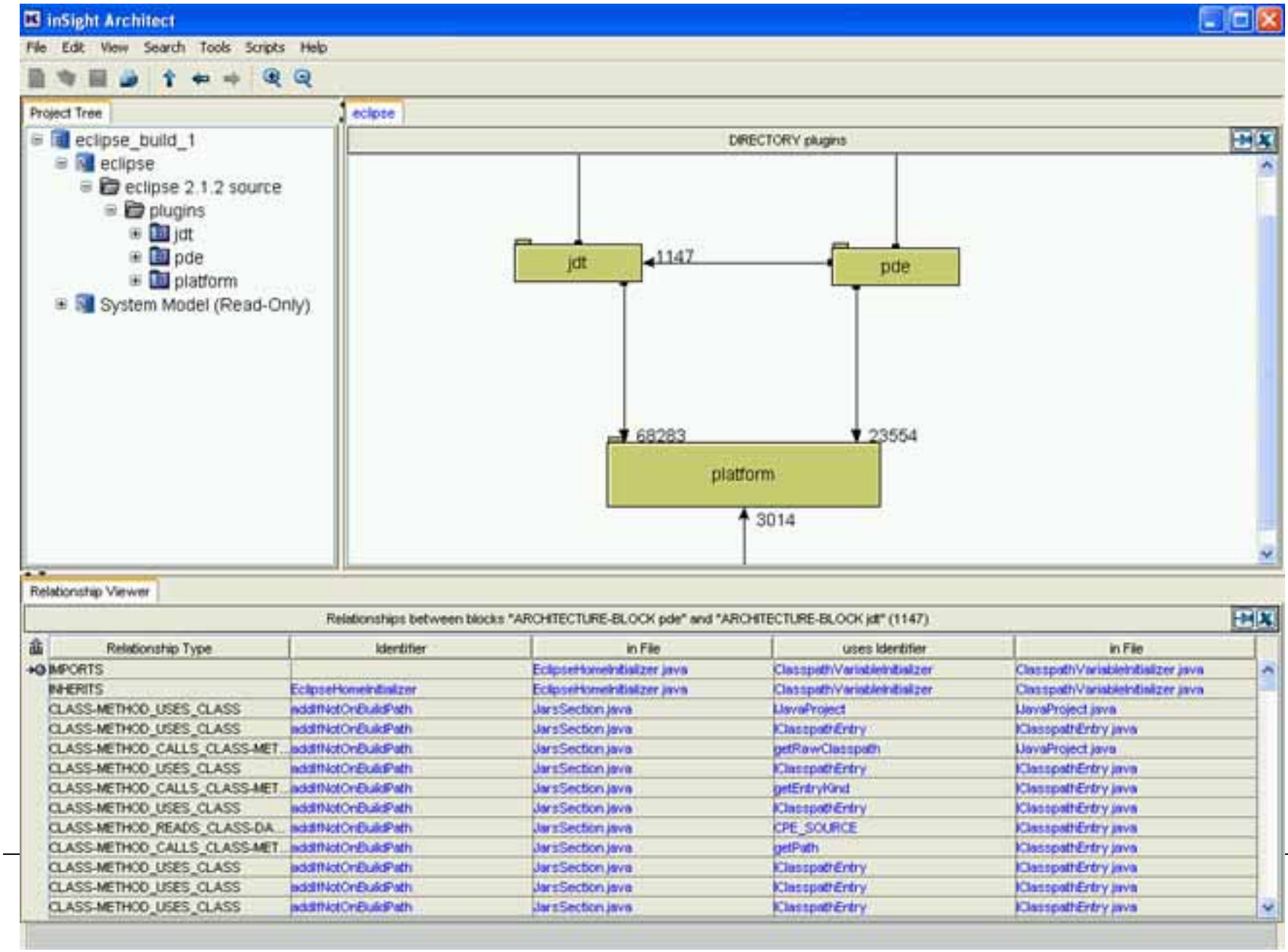


- Static Analysis Tool with Archtecture addon
  - Inforce, Inspect
  - Insight, Project Central
- Infos via Table + Graph, but WEAK layout algorithms !
- NOT Out-of-the box, but can be customized via tcl scripts
- Mysql DB
- Fat GUI Client, Web Report
- Java, C++

Klocwork: K7 insight



Klocwork: K7 insight



Tool Comparison

- Target audience
- Languages
- Handling
- Process
- IDE Integration
- Infrastructure
- Lightweight, Powerfull, Compliacated
- Features (that you (will) need)

Take home

- Today's IDEs / mechanisms are not suited for architectural analysis
  → Use a “lint4Architecture” (no official, my term)
- Tool support is a necessary
  → Architecture monitoring (possible with a small weekly time investment)

- Management…can be convinced if existing problems become visible
  → pays off very fast (e.g. one week jdepend analysis vs. Sotograph refactoring done)

- Rules can/will be violated
  →There is always a “good” reason for that
- Rule can be checked
  → Tool support can automate the process
  → If you have continous build system, start emploing a “lint4Architecture” now !!!

Informations

- Wikipedia
  - http://en.wikipedia.org/wiki/Software_visualization
  - http://en.wikipedia.org/wiki/List_of_tools_for_static_code_analysis
- Books:
  - Refactoring in Large Software Projects
  - Patterns
  - AntiPatterns
  - Metrics
  - Architecture