ACCU 2007

Linting Software Architectures

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

mailto: <u>Bernhard.Merkle@sick.de</u> mailto: <u>Bernhard.Merkle@googlemail.com</u>

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... ;-)



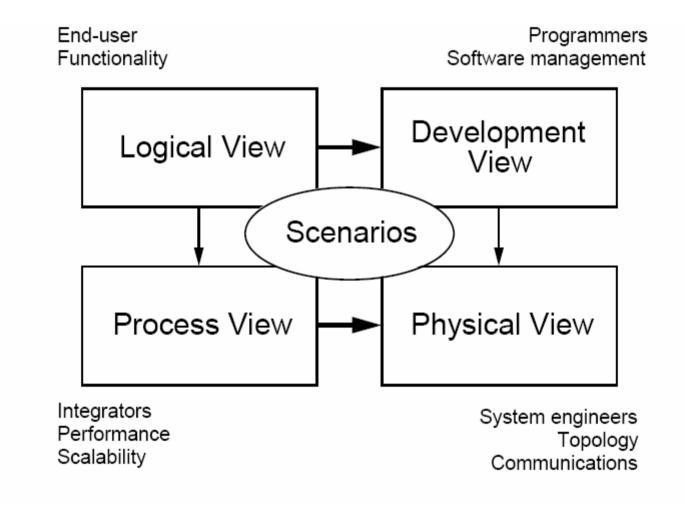
Software-Architecture: Definitions

• 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 ? ;-)

\rightarrow 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 (\rightarrow 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:

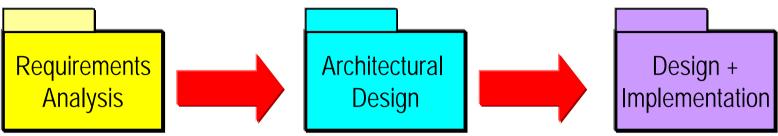
- Goal: (on all Levels)
 - find, avoid Problems, Increase QA (and measure it)
- Micro-Level
 - Code, MIRSA-C
 - E.g: =, ==, {},
- Marco-Level
 - Class-Design, Effective Rules, C++, Java, C#
 - E.g: by reference, String concat, Exception-Handling
- Architecture-Level:
 - Layers, Graphs, Subsystems, Compoments,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

Different kinds of Architecture-Analysis

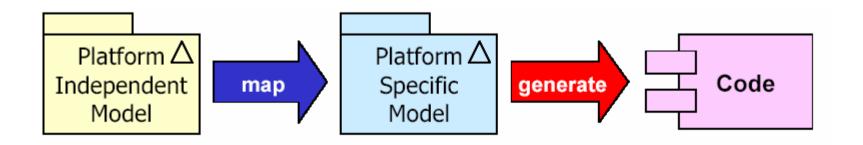
- Consistency-Analysis
- Rating of Architecture
- Discover a Architecture
- Measure real facts (e.g. metrics)
- Monitoring changes, trends (QA)

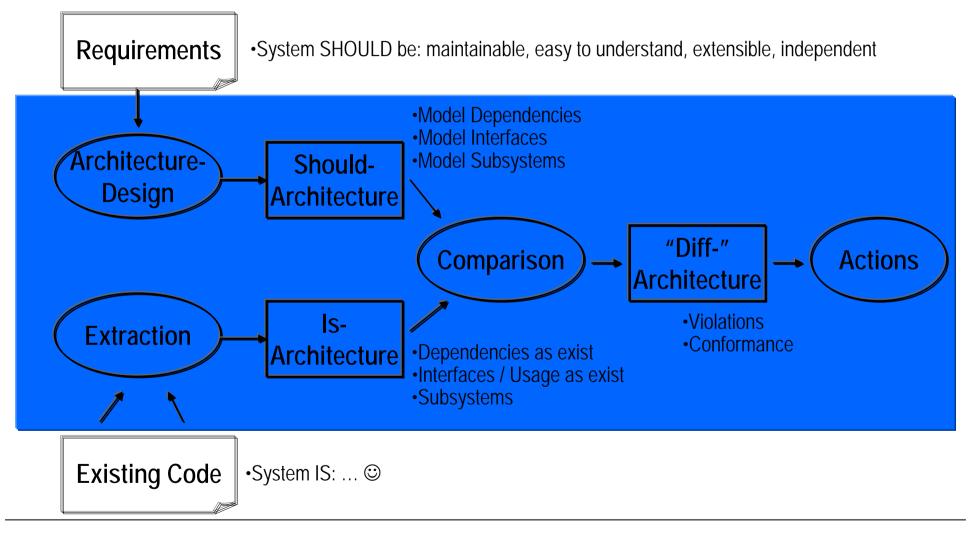
Consistency-Analysis

• Aim: No inconsistency



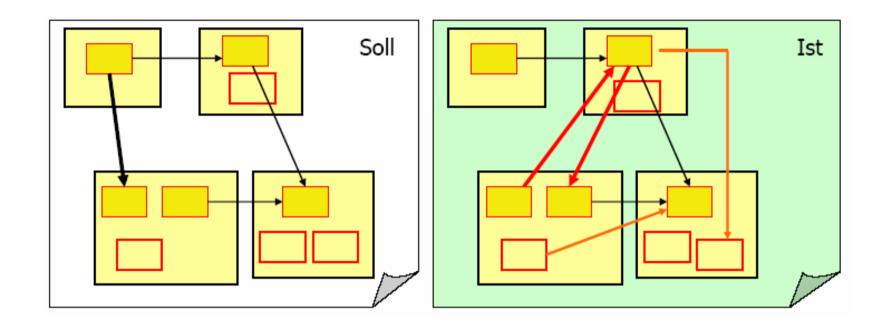
- Dispersion (no toolchain, information loss)
- Declay of Architecture, Rules vilotated, (over project time, various reaons...)
- Deviation Comparison





Consistency-Analysis: Things become VISIBLE

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



How to cope with violations...

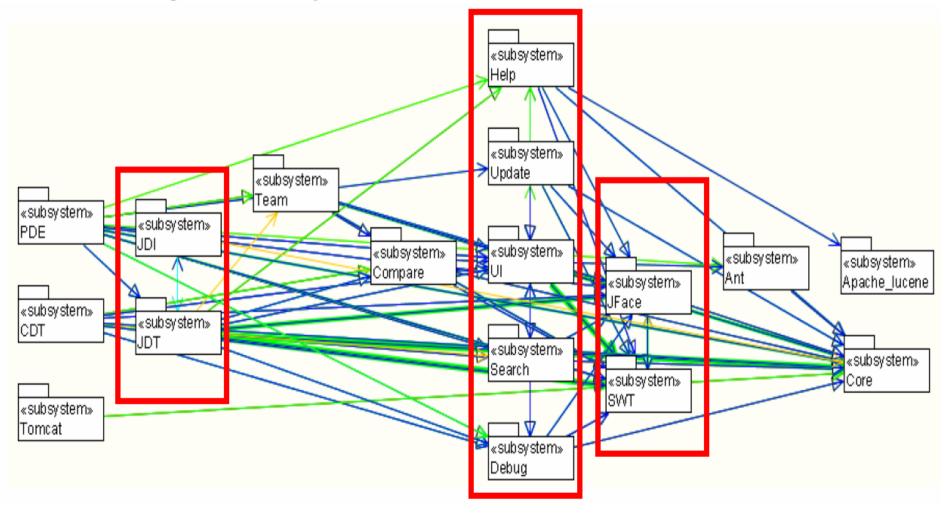
- Identify violations
 - Where
 - Quantity, Quality
 - Heaviness, Impact
- Handling violations
 - Fix possible ? (effort, costs, time)
 - Virtual refactorings, Simulations
 - List with modifications
 - Programmer implements fixes
 - Sometime, "autofix" lint ?... ☺

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 ?

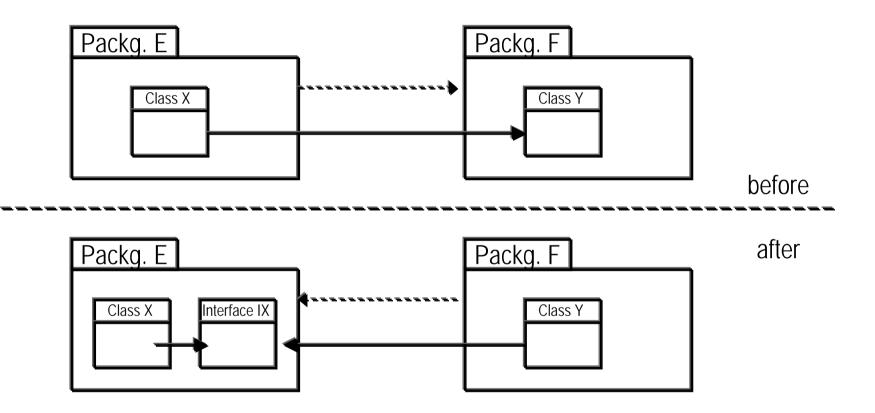
Rating of Architecture: e.g. Cycles

• Handling of Subsystems becomes difficult...



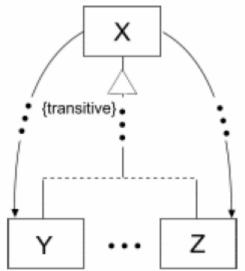
Rating of Architecture: e.g. Coupling

• 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

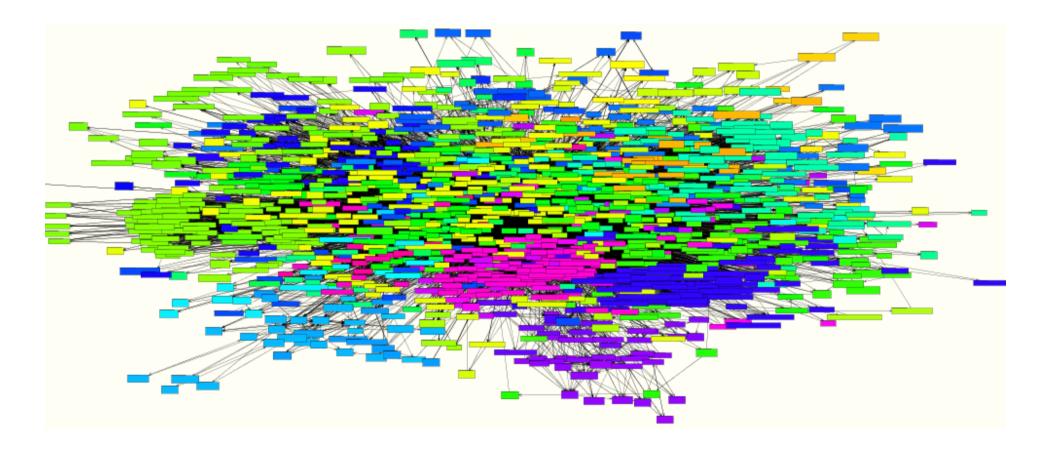
			Outliers 💽 Not Filtered	▼ Displayed 23	Total 305	
Select MetricModel(s)			Class	Package	Value	
Measures	-		JEditTextArea	org.gjt.sp.jedit.textarea	193	
All Metrics	-		Buffer	org.gjt.sp.jedit	117	
A Name	-	CL	JThis	bsh	84	
ClassAttributeNotUsed		CL	Parser	bsh	83	
		CL	jEdit	org.gjt.sp.jedit	79	
ClassAttributes	- 11		TextAreaPainter	org.gjt.sp.jedit.textarea	47	
ClassChars	-	CL	Interpreter	bsh	45	
ClassExcessiveMethodOverloading	-	CL	View	org.gjt.sp.jedit	42	
ClassInboundRefClass	-	CL	NameSpace	bsh	38	
ClassInboundUsageClass	- 11	CL	Gutter	org.gjt.sp.jedit.textarea	37	
ClassMethods	_	CL	GUIUtilities	org.gjt.sp.jedit	35	
ClassMethodsNotUsed		CL	MiscUtilities	org.gjt.sp.jedit	33	
ClassOutboundInhClass		CL	SearchAndReplace	org.gjt.sp.jedit.search	29	
ClassOutboundRefClass		CL	Primitive	bsh	28	
ClassPrivateMethodNotUsed		CL	VFSBrowser	org.gjt.sp.jedit.browser	27	
ClassPublicAttributes		CL	XmlParser	com.microstar.xml	27	
ClassPublicMethodNoGetSet		CL	RE	gnu.regexp	26	
ClassPublicMethods			VES	org.gjt.sp.jedit.io	26	
FileChars		CL	BshClassManager	bsh	22	
FileCommentBlocks			ASCII UCodeESC CharStream	bsh	21	
FileCommentLines			FoldVisibilityManager	org.git.sp.jedit.textarea	21	
FileCommentedOutCodeLines			Macros	org.gjt.sp.jedit	21	
FileHack			OffsetManager	org.git.sp.jedit.buffer	21	
FileInstanceOf		-	onsechanager	orgrgjetsprjodietbarror	21	
FileJavaDocComments						
FileLOC	-					
Filter Management						
Apply to All Metrics						
🦳 Make Filter Persistent		 Met	ric Description			
Add Id Filter		_	ClassPublicMethods: This metric counts for a class the			
Add Name Filter			number of public methods. Declared methods (e.g. in			
Containing Regular Expression:		interfaces (JAVA) or abstract classes (JAVA/C++)) and				
			fined methods are consider nsidered. Non-overridden d			
	_	,00	menters, non overredden d			

Bernhard Merkle

Linting Software-Architectures ACCU 2007 page:19

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)

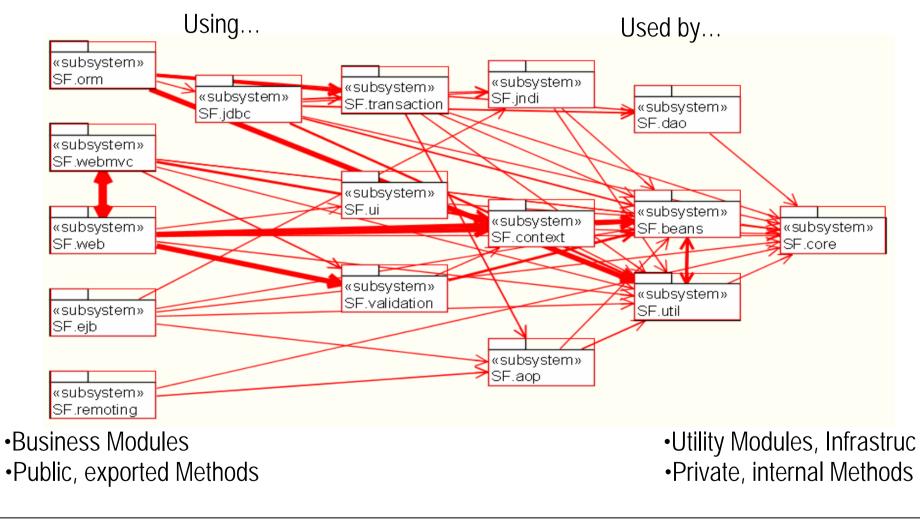
- Visualisation of _existing_ Architecture (Layout !)
 - Architecture often implicit
 - Undocumented
 - new staff in project,
 - Quick Overview of external software
 - Erosion and Analysis
 - Discover central abstractions/key concepts, e.g. Worker-classes
 - Typical Usage of certain artefacts, Patterns
 - 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

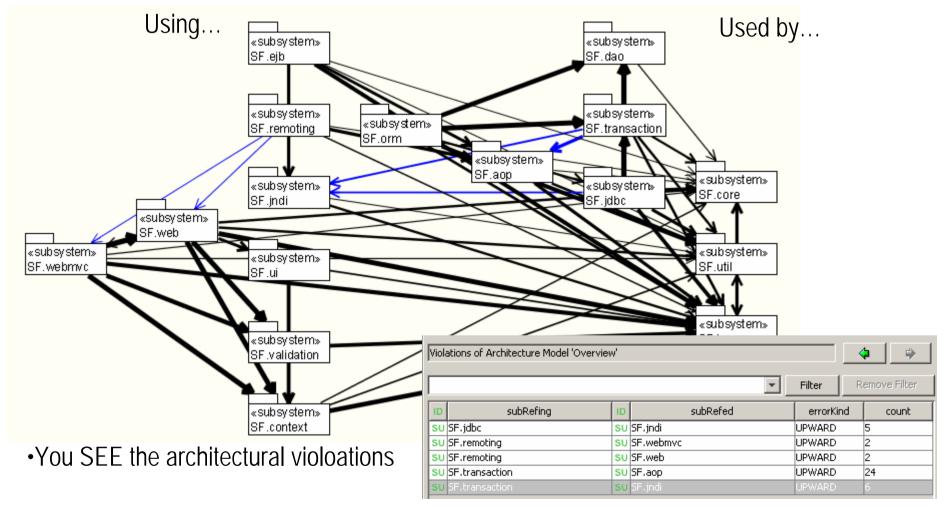
Discover a Architecture: Level of Abstraction

• Topologic sorted layout, only Call-Relationships



Discover vs.Model a Architecture: Variance comparison

• Arch. sorted layout, only Call-Relationships

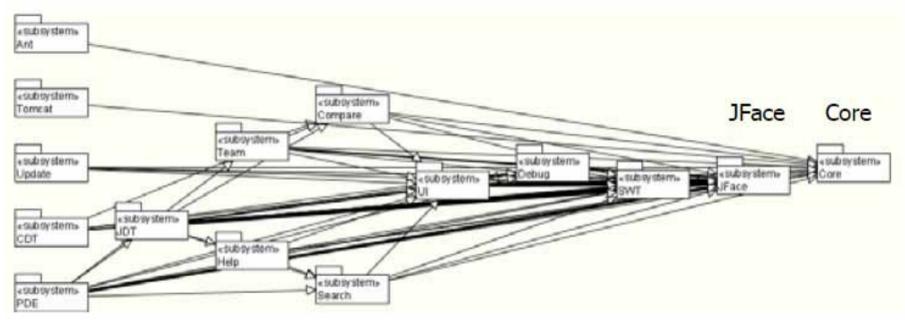


Discover a Architecture: Level of Abstraction

• 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 \rightarrow Actual state of Arch
 - Description of Arch Rules \rightarrow 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:
 - Bauhaus:

<u>www.software-tomography.de</u> <u>www.axivion.com</u>

- SonarJ:
- Structure101:
- Lattix:
- www.hello2morrow.de www.headwaysoftware.com www.lattix.com
- Klocwork K7: <u>www.klocwork.com</u>
- XRadar (opensource): www.xradar.org
- Others: CodeCrawler, SeeSoft, ResourceStandardMetrics

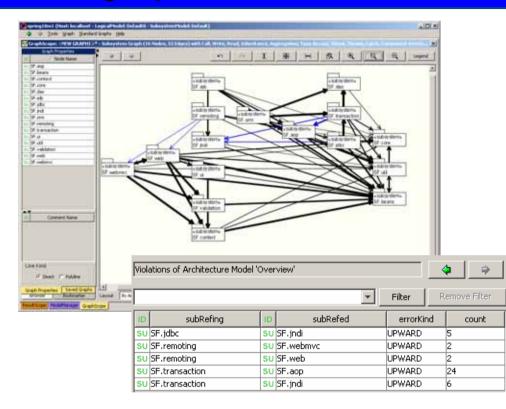
Basic Approaches

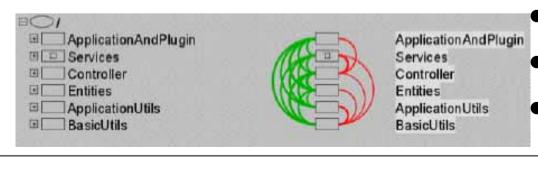
- Basic approaches
 - Your makesystem...
 - makedepend, jdepend
 - RE code into UML model
 - Eclipse (Java Build Path)

🖨 Add Exclusion Pattern	×
Enter a pattern for excluding files f '?' and '**'. Examples: 'java/util/A*	rom the source folder. Allowed wildcards are '*', iava' 'iava/util/' '**/Tect*'
Exclusion pattern (Path relative to	
	Bro <u>w</u> se
?	OK Cancel

E Inclusion and Exclusion Patterns	×
Inclusion and Exclusion Patterns	
Add or remove inclusion and exclusion patterns to source folder	
Inclusion patterns:	
	A <u>d</u> d
	Add <u>M</u> ultiple
	Edit
	<u>R</u> emove
Exclusion patterns:	
	<u>A</u> dd
	Add M <u>u</u> ltiple
	Edi <u>t</u> ,
	Remove
? Einish	Cancel

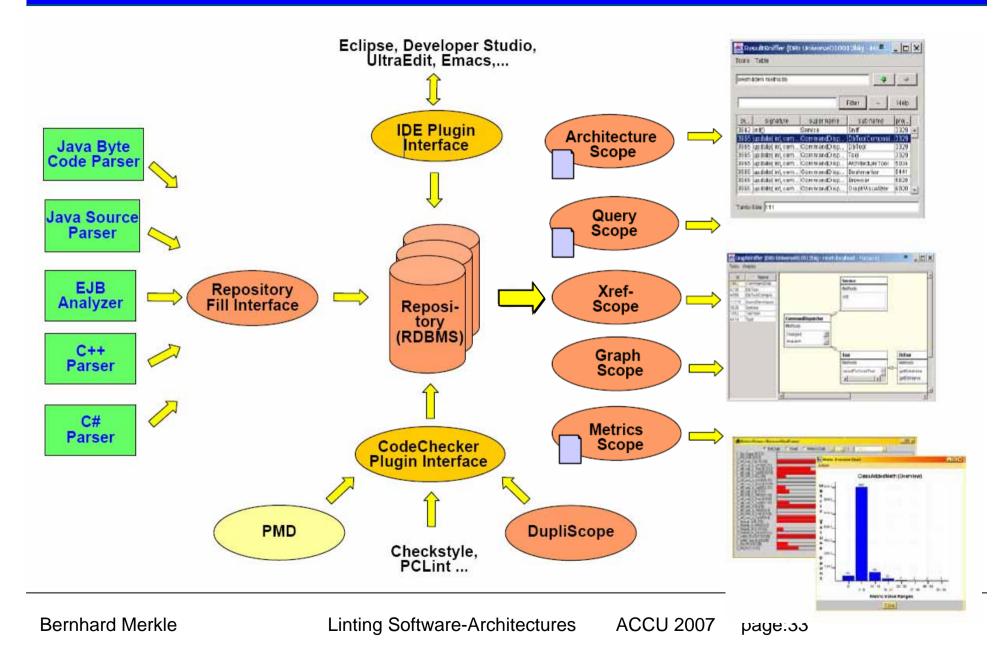
Sotograph: Overview



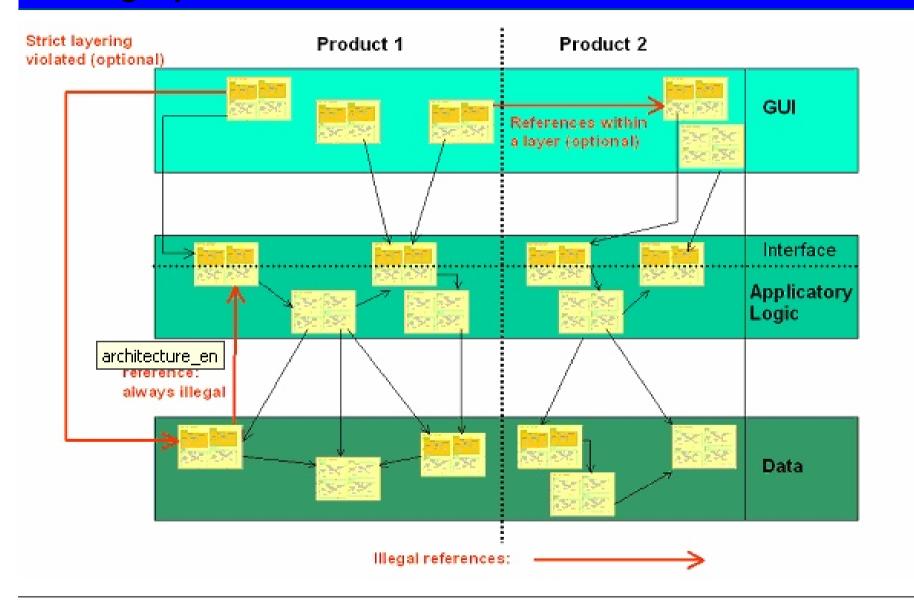


- VERY powerfull
- Infos via Table + Graph
- Cool layout algorithms
- Known since 2003 (NG"SNIFF++")
- Mysql DB, open schema
- Fat GUI Client, Web Report
- About 200+ Metrics
- Arbitrary User queries
- Trend Analysis
- Virtual Refactoring
- Java, C++, C#, source parser
- Lightweight SotoArch 2007

Sotograph



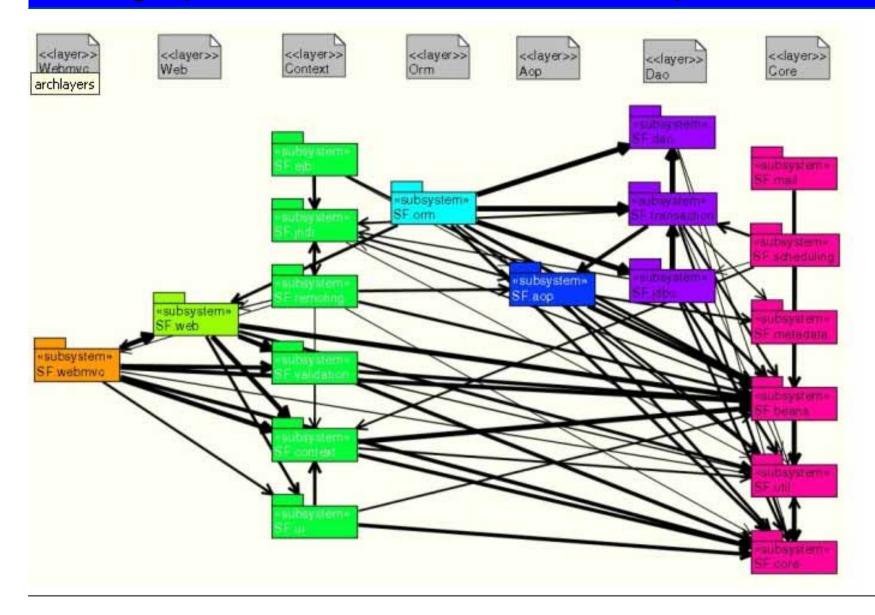
Sotograph: Source and Architecture



Bernhard Merkle

Linting Software-Architectures ACCU 2007 page:34

Sotograph: Structure and Relationships

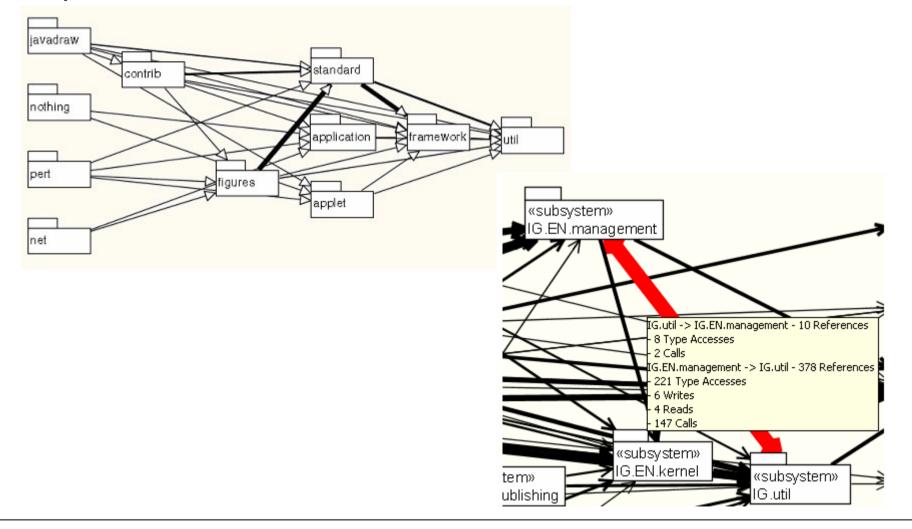


Bernhard Merkle

Linting Software-Architectures ACCU 2007 page:35

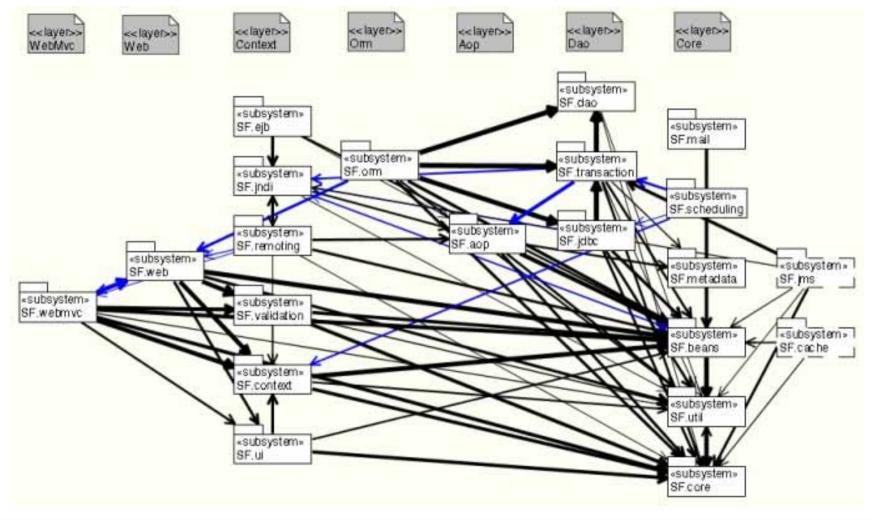
Sotograph: Structure and Relationships

• Depenencies: Informations...



Sotograph: Check Arch. Conformance and Quality

• Arch. violations

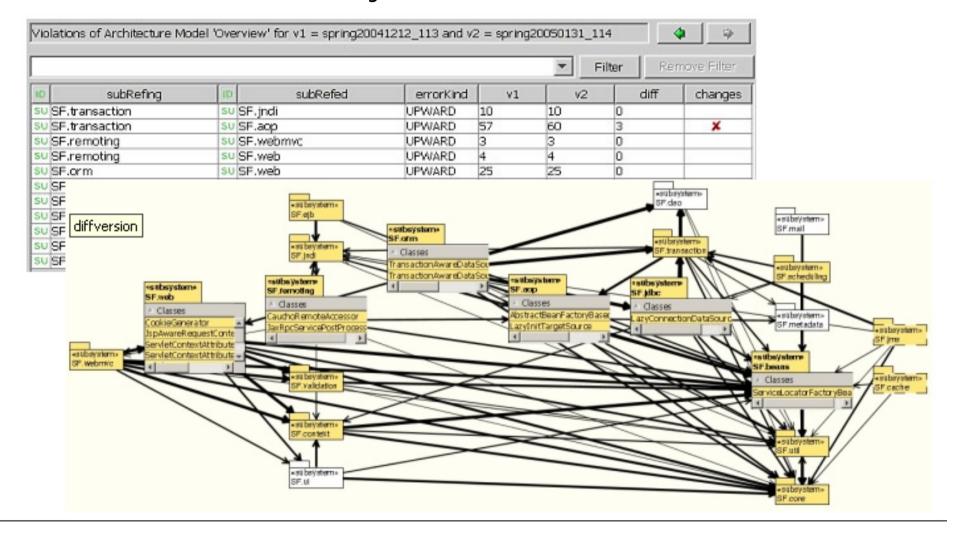


Bernhard Merkle

Linting Software-Architectures ACCU 2007 page:37

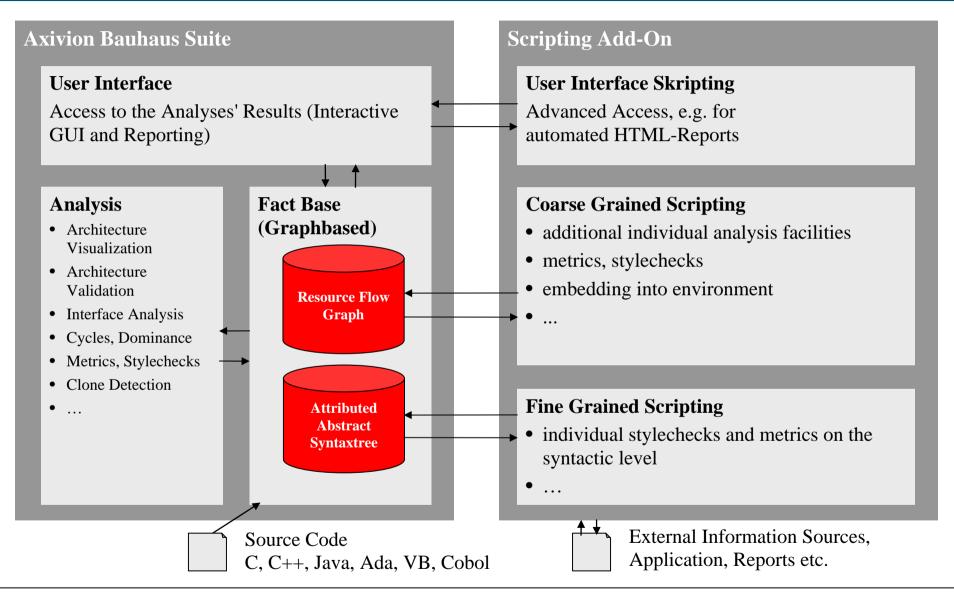
Sotograph: Monitoring Changes

• of Architecture, Quality, Structure



Bernhard Merkle

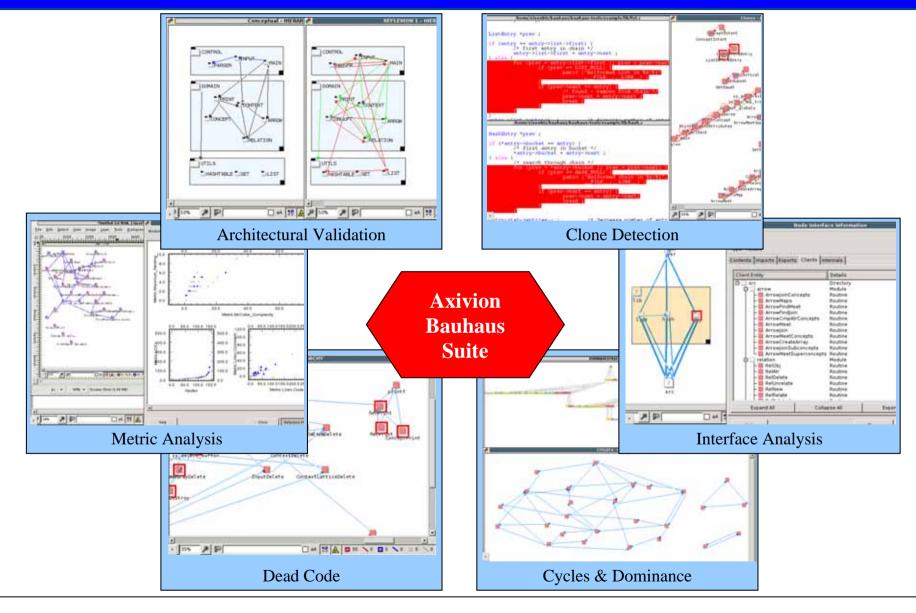
Axivion Bauhaus Suite



Bernhard Merkle

page:39

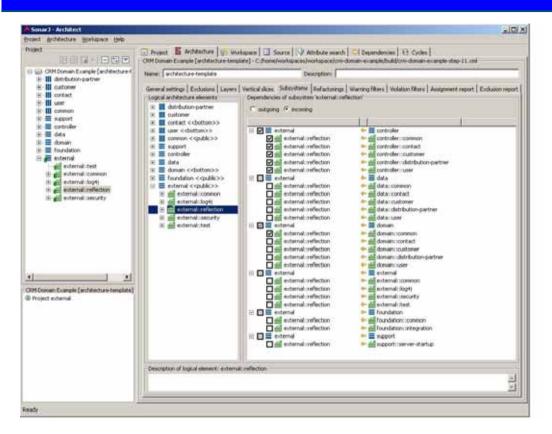
Axivion Bauhaus Suite



Bernhard Merkle

Linting Software-Architectures ACCU 2007 page:40

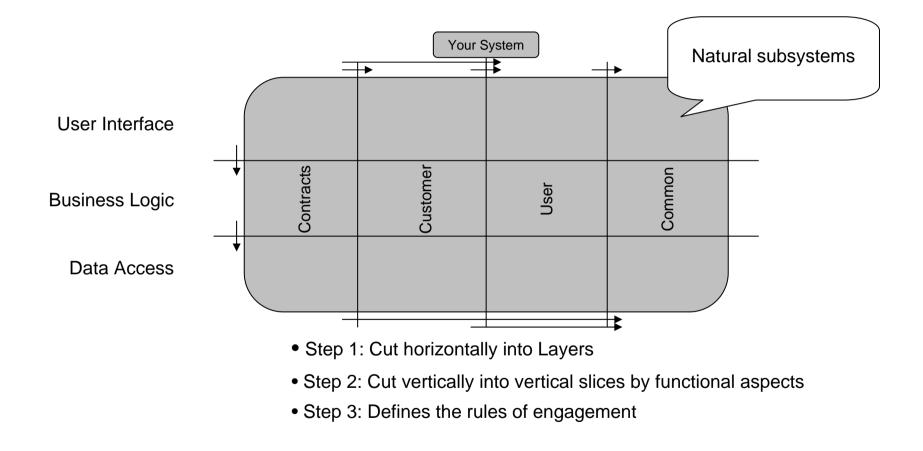
SonarJ: Overview



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

SonarJ: Architecture-MetaModel

• Architecture-MetaModel:



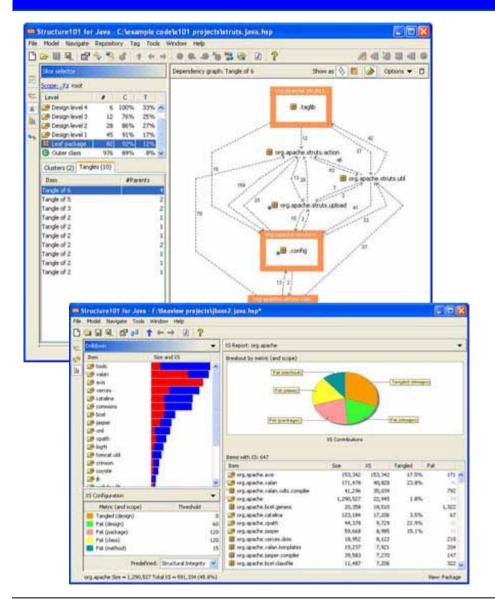
SonarJ: Architecture-MetaModel

- 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 Project Architecture Workspace Source Attribute search Dependencies Cycles - A 4 ? architecture-template 🖃 🗁 crm-domain-example [architecture-template] Path: C:/Projects/crm-domain-example/build/crm-domain-example-step-10.xml + III common + III contact architecture-template Name: Ŧ III customer Description: 🗄 📶 distribution-partner 🕂 📶 user General settings Exclusions Layers Vertical slices Subsystems Assignment report (0 Warning(s)) Exclusion report (0 Warning(s)) + = controller -Select a subsystem or named subsystem interface Dependencies of subsystem 'external::reflection' 🗐 🔳 data 🖭 📶 common 🕂 📃 domain Coutaoing C incoming 🗄 🗾 external + III contact 🖃 🗹 😂 [architecture-template] + III customer 👳 🗮 foundation 🗄 🔽 🚍 external <- controller 🗄 IIII distribution-partner 庄 🧮 support 🗹 i external::reflection <- controller::common 🛨 📶 user external::reflection <- controller::contact + = controller external::reflection <- controller::customer 🕂 🔳 data 🔽 👬 external : : reflection <- controller : : distribution-partner 🖃 🗮 domain external::reflection <- controller::user 🛨 👬 domain::common 🗄 🥅 🗮 external <- data 🗄 🚮 domain::contact external::reflection <- data::common 🕂 👬 domain::customer million <- data::contact 🗄 📑 domain::distribution-partner milli external::reflection <- data::customer 표 🚮 domain::user 🗖 👬 external::reflection <- data::distribution-partner = = external T i external::reflection <- data::user 🗄 📑 external::common 🖻 📝 🚍 external <- domain 🕂 📑 external::log4j 🗹 👬 external::reflection <- domain::common 🗄 🚟 external::reflection external::reflection <- domain::contact 🗄 📑 external::security million <- domain::customer 🕂 👬 external::test 🗖 🚎 external::reflection <- domain::distribution-partner 🛨 🗮 foundation crm-domain-example [architecture-template] milli external::reflection <- domain::user 🗄 🗮 support 🗄 🥅 🧮 external <- external 🔲 🚟 external::reflection <- external::common mile external::reflection <- external::log4j external::reflection <- external::test E ■ ■ external <- foundation</p> million <- foundation::common external::reflection <- foundation::integration 🗄 🥅 🧮 external <- support milder sternal::reflection <- support::server-startup < >

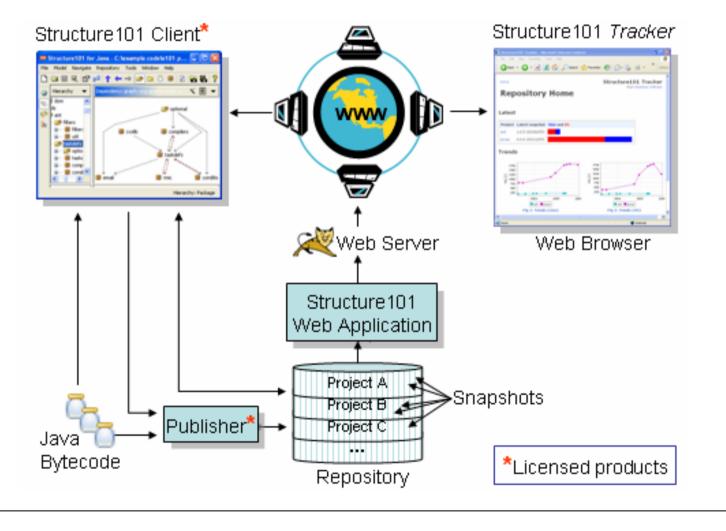
Structure101: Overview



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

Structure101

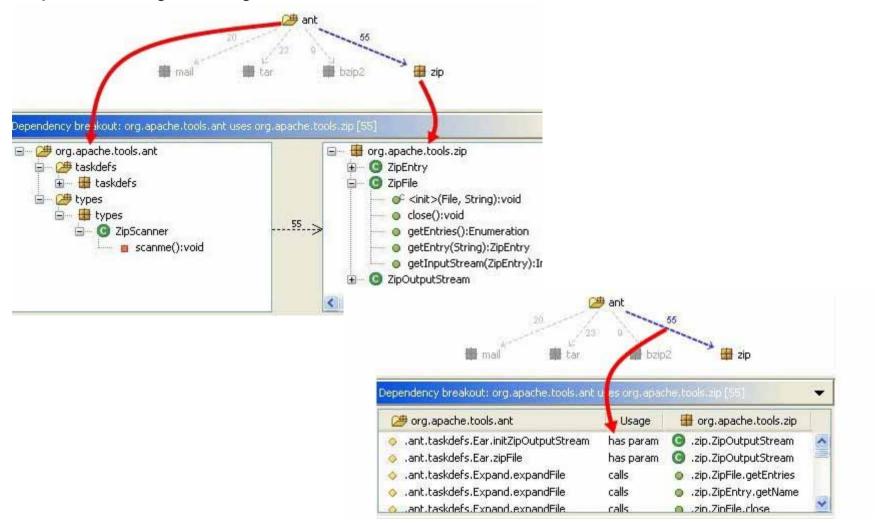
• Structure101 Architecture



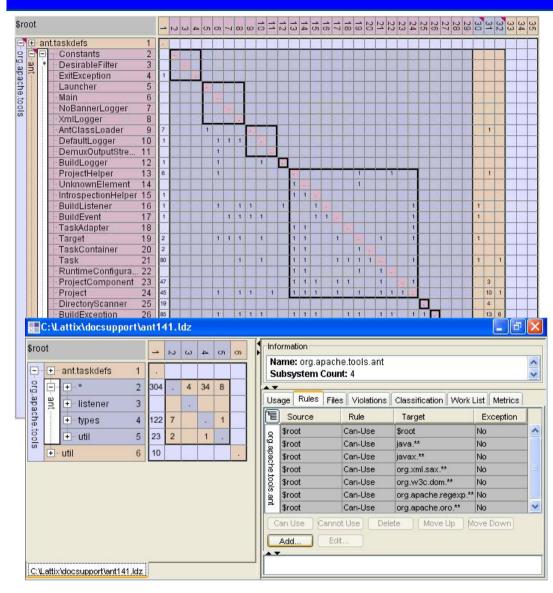
Bernhard Merkle

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

Bernhard Merkle

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 identify highly coupled artefacts
- Rules for allowed/forbidden Relationsships

Lattix: DSM Examples

• Example Architectures, for direct reading from Matrix

\$roc	ot	<u> </u>	2	ω	4	თ
· _ · · ·	·∓ […] application 1					
CON	🕂 model 2	37				
com.example	tomain 3	17	29			
dul	🕂 framework 4	75	53	42		
æ	±util 5	10	13	16	13	

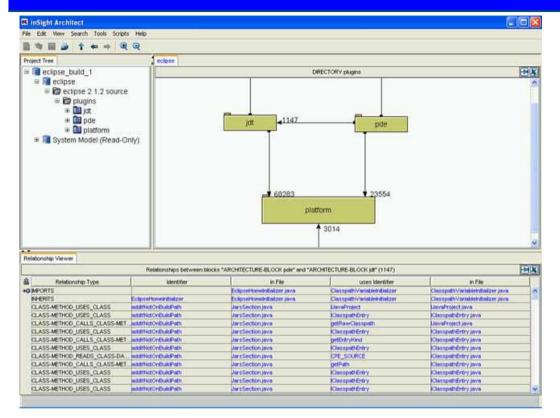
\$ro	ot			2	ω	4	ch
Ð	application	1	14				
con	+ model	2	37				
1.exa	🛨 domain	3		29	1	0	
com.example	+- framework	4			42	4	
τ D		5				13	+

\$roc	t	<u> </u>	2	ω	4	თ
	🕂 application 1					1
com	+-model 2	37				1
com.example	±-domain 3	17	26			
dure	+-framework 4	75	53	40		
0 0	±−util 5	11	13	16	13	

Lattix: Rules and Partitionierung

\$root			-	20	4 0	m	თ	7		010	5 11	12	1 4	15	16	17	10	20	22	23	22	26	28	29	30	32	33	36					
-	ant	taskdefs 1 Constants 2 DesirableFilter 3 ExitException 4 Launcher 5 Main 6 NoBannerLogger 7 XmlLogger 8	1		1	-		-	03	1	1		1	1	101	1	1		11	1	1	11		11	-	1.0	11	1	1				
0 E		Constants 2		1200					U.																			1	1				
ie D	*	DesirableFilter 3		i					T															T					1				
tr		ExitException 4	the second	i	141				<u>n</u>					1															1				
ich		Launcher 5												1						11						13			1				
e.to		Main 6	5			Ē	-		1						171					1						133		1	1				
0		NoBannerLogger 7				Ē		6	1					1												12			1				
S		XmlLogger 8	3																							18			1				
		AntClassLoader 9				1																			1				1				
		DefaultLogger 10					1	1	1	118					.0,	1													1				
		DemuxOutputStre 11					45]				
		BuildLogger 12					1			1		-								1 11									1				
		ProjectHelper 13					1		Ĩ.	1			611	1		Ĭ.	1		1).						1				1				
		UnknownElement 14							Ō'				1 -			Ū.	1			1 1									l I				
		IntrospectionHelper 15											1 1													14			1				
		BuildListener 16					1		1	1		1		1							1				1	1							
		BuildEvent 17						1	1	1 1			_	1	1	3							_		1	13							
		TaskAdapter 18			-								1 1								1								I				
		Target 19			×	C:	L a	itti	c\d	ocs	up	pori	t\ai	nt1	41.	ldz																	
1000		TaskContainer 20				_							Ĩ		1	T					- le		ation	12									
		Task 21			\$ro	ot								-	N	ω	4	On	0			girteria		8									
		RuntimeConfigura 22				Te	1			1101102		-	, i	_	-		-	1		i			e: or	-				ant					^
		ProjectComponent 23				1000	S	ant.t	as	kde	IS	1	1	<u>12</u>					_			Subs	syste	m C	our	nt: 4							~
		Project 24	45		3	Ē		+	*				2	304	1	4	34	8				•		_					-24				
		DirectoryScanner 25			in	a		+	Ret	0.000			3	-				-		1	U	sage	Rul	les	File	s I	/iola	tions	Clas	sification	Work	List Metri	s
		BuildException 26	85		ag	5	1	-	1151	ene			3				_		-		1	_											1
		FileScanner 27			e l			+	typ	es			4	122	7			1					Sourc	e	_	R	ule		la	rget	_	Exceptio	
		Location 28	3		org.apache.tools		1	+	util	-		3	5	23	2		1	1.		1	l g	\$r	oot			Ca	n-Us	e	\$roo	ot		No	^
	-	PathTokenizer 29 istener 30			8			07.2.5	uuu				2201	257.5	2						l Q a	\$r	oot			Ca	n-Us	е	java	a.**		No	
	計畫	istener 30				1 1							6	10							pag	\$r	oot			Ca	n-Us	e	java	ax.**		No	
						1C	L	IITH					~							_	11 🖂					Ca	n-us					- 2010 - 2010	
	+ t	ypes 31	122			12		JTH					0								Πā	_ ⊅r	oot			10000	n-Us n-Us	202	orq.	.xml.sax.**		No	
	± t € t	ypes 31 util 32	122					JTH													le lo	- Dr Sr				Ca	n-Us	е		xml.sax.** w3c.dom	**	10000	
÷	+ t + t ma	ypes 31 util 32 il 33	122																		org.apache.tools.a	sr \$r	oot			Ca Ca	n-Us n-Us	e e	org.	w3c.dom.		No	
÷	+ t + t ma tar	ypes 31 util 32 il 33 34	122 23 1 4																		ie.toois.ant	\$r	oot oot			Ca Ca Ca	n-Us n-Us n-Us	e e e	org.	.w3c.dom. .apache.re	gexp.*	No No	
÷	+ t + t ma	ypes 31 util 32 il 33	122 23 1 4					utii					0								ie.tools.ant	\$r	oot			Ca Ca Ca	n-Us n-Us	e e e	org.	w3c.dom.	gexp.*	No	-
÷	+ t + t ma tar	ypes 31 util 32 il 33 34	122 23 1 4				-	utii													ie.tools.ant	\$r \$r	oot oot] @	anno	Ca Ca Ca Ca	n-Us n-Us n-Us n-Us	e e e	org.	.w3c.dom. .apache.re	gexp.* o.**	No No	
÷	+ t + t ma tar	ypes 31 util 32 il 33 34	122 23 1 4				1	utii													ie.tools.ant	\$r \$r Can	oot oot oot Use]6		Ca Ca Ca Ca	n-Us n-Us n-Us n-Us	e e e	org. org. org.	.w3c.dom. .apache.re .apache.or	gexp.* o.**	No No No	
1	I t I t I t I t ar zip	ypes 31 util 32 il 33 34 35	122 23 1 4					utii													ie.tools.ant	\$r \$r	oot oot oot Use		anno Edi	Ca Ca Ca Ca	n-Us n-Us n-Us n-Us	e e e	org. org. org.	.w3c.dom. .apache.re .apache.or	gexp.* o.**	No No No	
1	I t I t I t I t ar zip	ypes 31 util 32 il 33 34	122 23 1 4				- <u></u>														ie.tools.ant	\$r \$r Can	oot oot oot Use	0		Ca Ca Ca Ca	n-Us n-Us n-Us n-Us	e e e	org. org. org.	.w3c.dom. .apache.re .apache.or	gexp.* o.**	No No No	

Klocwork: K7 Overview

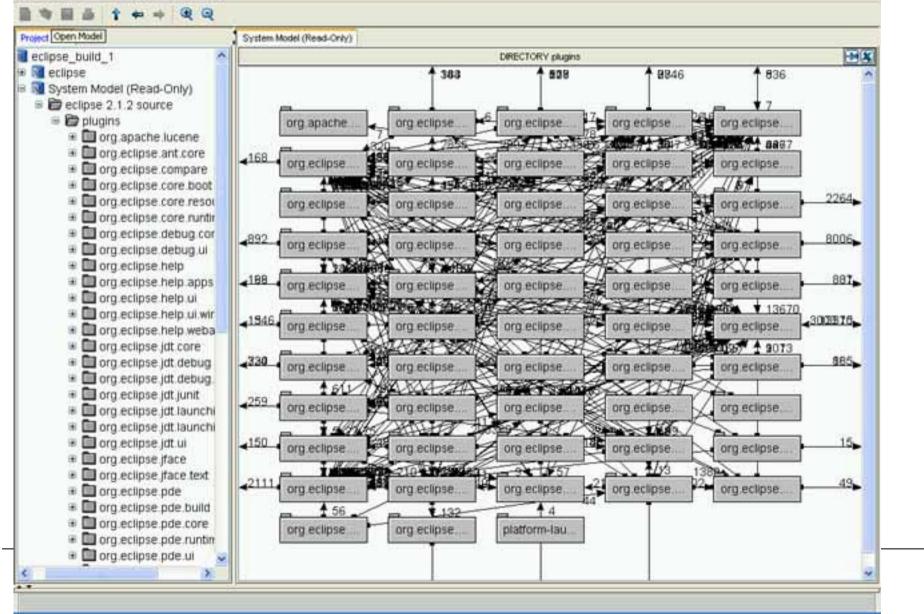


- Static Analysis Tool with Archtecture addon
 - Inforce, Inspect
 - Insight, Project Central
- Infos via Table + Graph, but WEAK layout algorithms !
- NOT Out-of-the box, but can be customized via tcl scripts
- Mysql DB
- Fat GUI Client, Web Report
- Java, C++

Klocwork: K7 insight

K inSight Architect

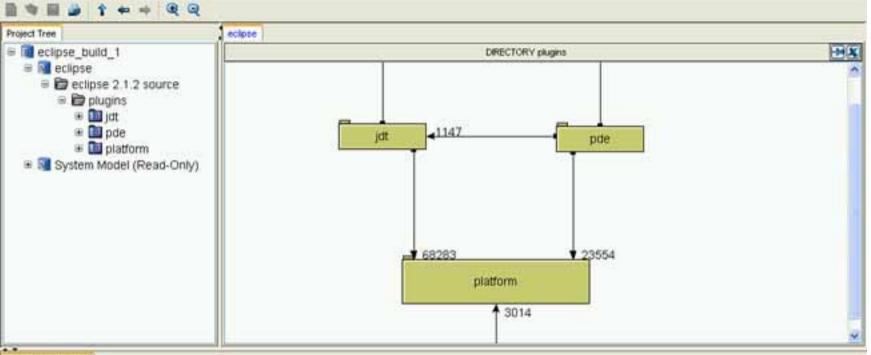
File Edit View Search Tools Scripts Help



Klocwork: K7 insight

K inSight Architect

File Edit View Search Tools Scripts Help



Relationiship Viewer

Relationship Type	Identifier	in File	uses Identifier	in File	
MPORTS		Eclipsettoweinitalizer java	Classpeth/VariableInitializer	Classpath/ariableInitializer.java	
INHERITS	EclpseHomeinEnlzer	Eclpsettomeinitializer Java	Classpoth/variableInitial.zer	ClosspathVariabieIndializer Java	
CLASS-METHOD_USES_CLASS	nddifNotOnEuildPath	JarsSection Java	LlavaProject	UavaProject java	
CLASS-METHOD_USES_CLASS	addtNotOnEukdPath	Jars Section Java	ClasspotrEntry	ClasspathEntry java	
CLASS-METHOD_CALLS_CLASS-M	TaddtNotOnEukdPath	JarsSection Java	getRawClasspath	UsvaProject jays	
CLASS-METHOD_USES_CLASS	additNotOnEu8dPath	JarsSection Java	ClasspothEntry	ClasspathEntry java	
CLASS-METHOD_CALLS_CLASS-ME	T. additNotOnEuklPath	JarsSection Java	getEntryKind	ClasspathEntry java	
CLASS-METHOD_USES_CLASS	additNotOnBuildPath	JarsSection Java	ClasspothEntry	ClasspathEntry Java	
CLASS-METHOD_READS_CLASS-D	addition ConEuldPath	Jars Section Java	CPE_SOURCE	ClasspathEntry Java	
CLASS-METHOD_CALLS_CLASS-M	T. addtNotOnEuktPath	JarsSection Java	getPath	ClasspathEntry Java	
CLASS-METHOD_USES_CLASS	addtNotOnEu8dPath	JarsSection Java	ClasspottEntry	ClasspathEntry java	
CLASS-METHOD_USES_CLASS	additNotOnEuidPath	JarsSection Java	ClasspothEntry	ClasspothEntry Java	
CLASS-METHOD_USES_CLASS	addtNotOnBuikiPath	Jars Section Java	KlasspathEntry	ClassipathEntry Java	

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
 - \rightarrow 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 <u>analysis</u> vs. Sotograph refactoring <u>done</u>)
- Rules can/will be violated
 - \rightarrow There is always a "good" reason for that
- Rule can be checked
 - ightarrow Tool support can automate the process
 - \rightarrow 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

