To Know the Dependencies Is to Understand the Architecture

Neeraj Sangal

President
Lattix, Inc.
http://www.lattix.com

TS-6037

Dependencies Are Key to Managing Software Architecture

Prevent software architecture from eroding

Learn a new approach to representing and managing software architecture by utilizing inter-module dependencies. This session includes an actual demo and several real life examples.

Agenda

New Representation for Software Architecture Using Dependency Structure Matrix

Architectural Patterns

Design Rules

Architectural Evolution (Demo)

Examples of Real Projects

Q&A

Agenda

New Representation for Software Architecture Using Dependency Structure Matrix

Architectural Patterns

Design Rules

Architectural Evolution (Demo)

Examples of Real Projects

Q&A

The Dependency Model Approach

- Provides a precise big picture view of the architecture
- Enables explicit management of architectural evolution
- Lightweight approach that does not disrupt development
- Has been applied to dozens of applications written in the Java™ language ranging in size from 100 classes to 20,000 classes

What Is a DSM?

		1	2	3	4
Module A	1	▪		X	X
Module B	2		▪	X	
Module C	3	X		▪	X
Module D	4				▪

Fig 1: A Simple DSM

		1	2	3
Module D	1	▪		
Module A-C	2	X	▪	
Module B	3		X	▪

Fig 3: Lower Triangular

		1	2	3	4
Module D	1	▪			
Module A	2	X	▪	X	
Module C	3	X	X	▪	
Module B	4			X	▪

Fig 2: Block Triangular After Partitioning

			1	2	3	4
Module D		1	▪			
A-C	Module A	2	X	▪	X	
	Module C	3	X	X	▪	
Module B		4			X	▪

Fig 4: Hierarchical

What's a Dependency?

- Module A depends on a module B if there are explicit references in A to syntactic elements of B
- Simple but effective notion of dependency that works well for understanding design dependencies, in which modifications to one module might affect another
- Extraction of dependencies can be decoupled from their analysis

Agenda

New Representation for Software Architecture using Dependency Structure Matrix

Architectural Patterns

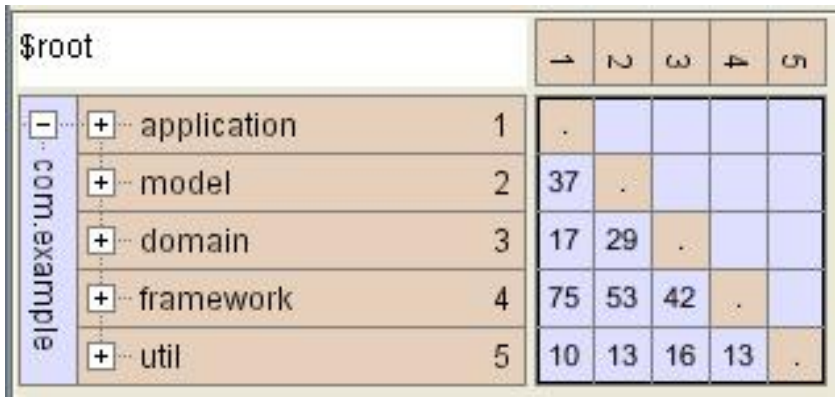
Design Rules

Architectural Evolution (Demo)

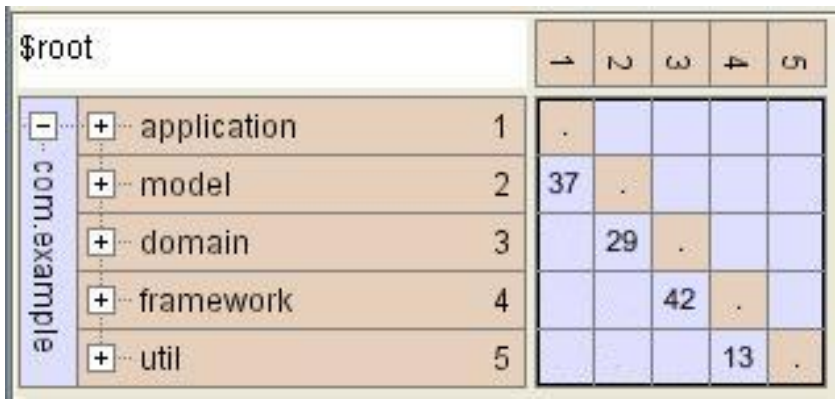
Examples of Real Projects

Q&A

Architectural Patterns—I

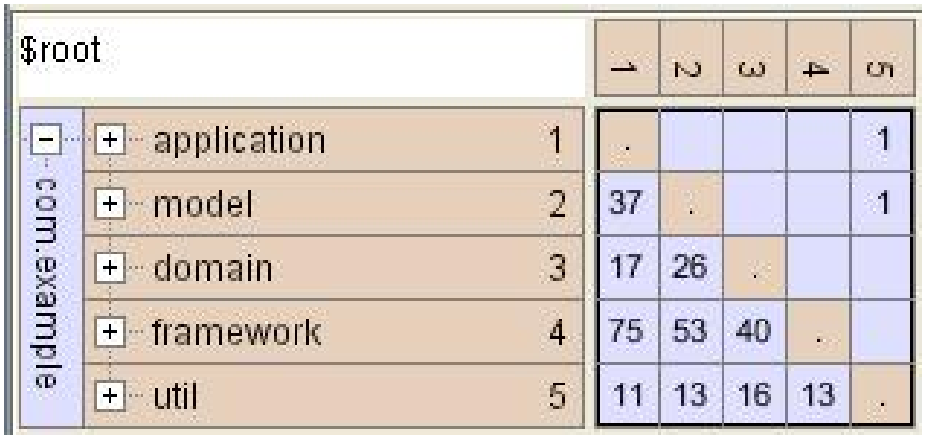


Layered System

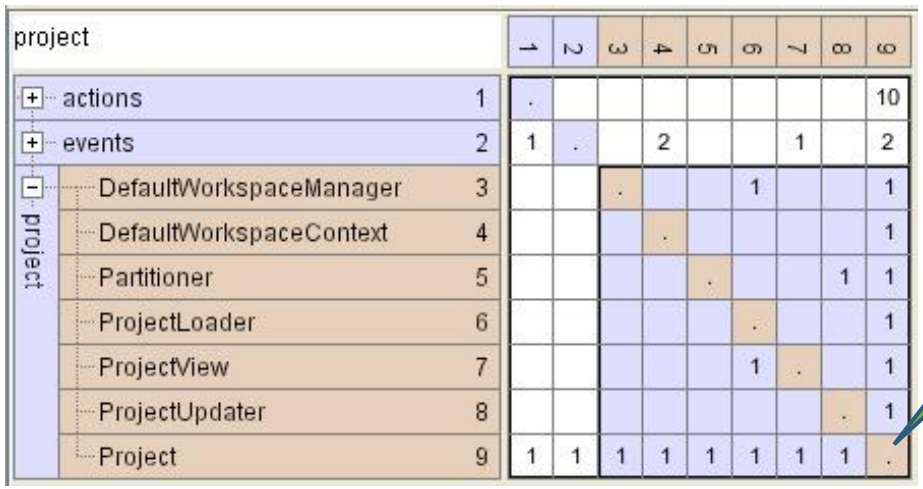


Strictly Layered System

Architectural Patterns—II



$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	.

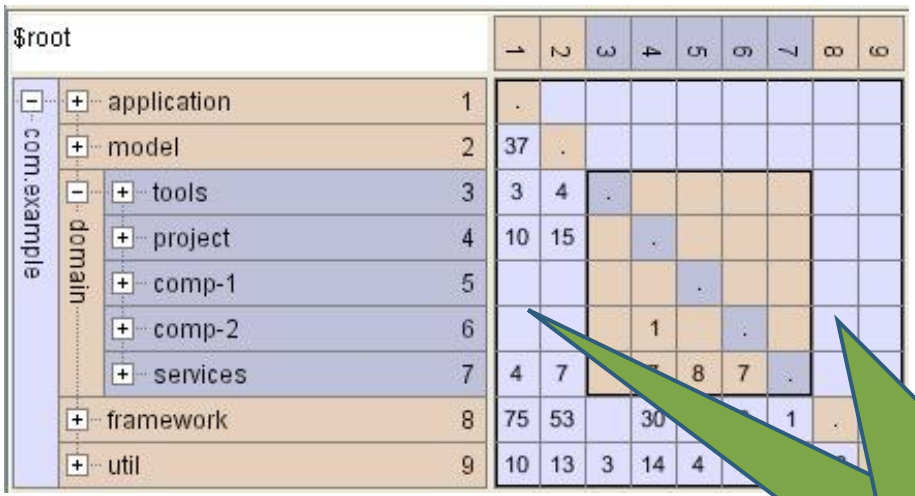


project		1	2	3	4	5	6	7	8	9
	actions 1	.								10
	events 2	1	.		2			1		2
project	DefaultWorkspaceManager 3			.			1			1
	DefaultWorkspaceContext 4				.					1
	Partitioner 5					.			1	1
	ProjectLoader 6						.			1
	ProjectView 7						1	.		1
	ProjectUpdater 8								.	1
	Project 9	1	1	1	1	1	1	1	1	.

Imperfectly Layered System

Change Propagator

Architectural Patterns—III



Private Components

Not Visible
Outside “Domain”

Agenda

New Representation for Software Architecture using Dependency Structure Matrix

Architectural Patterns

Design Rules

Architectural Evolution (Demo)

Examples of Real Projects

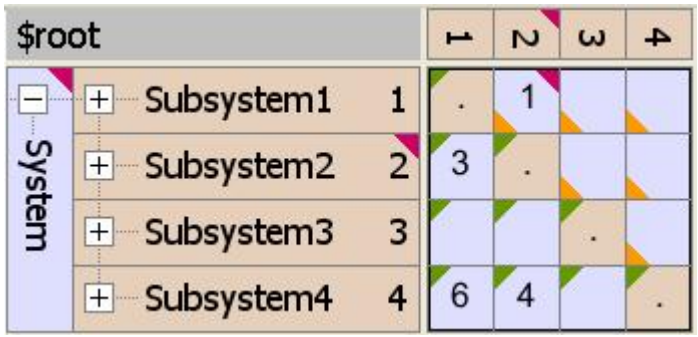
Q&A

Design Rules

- Succinct specification of acceptable and unacceptable dependencies between subsystems
- Each cell of the DSM represents design intent
- DSM offers a powerful way to visualize and specify design rules

Dependency Model = DSM + Design Rules

Design Rules



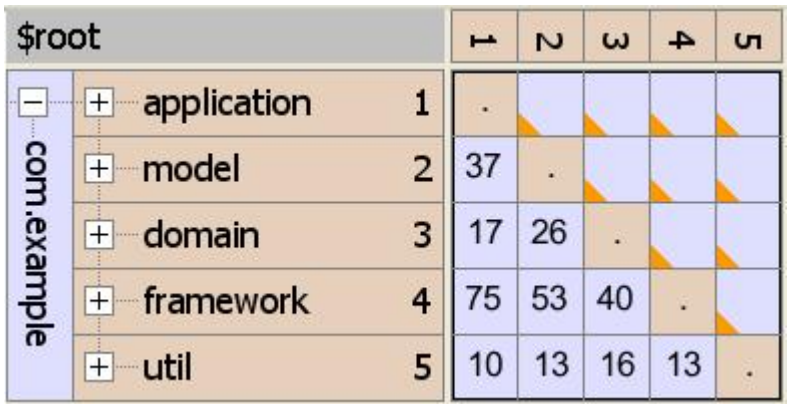
$root		1	2	3	4
System	Subsystem1 1	.	1		
	Subsystem2 2	3	.		
	Subsystem3 3			.	
	Subsystem4 4	6	4		.

DSM with Rules View

Green Triangle—Dependency Acceptable

Yellow Triangle—Dependency Unacceptable

Red Triangle—Rule Violation Discovered

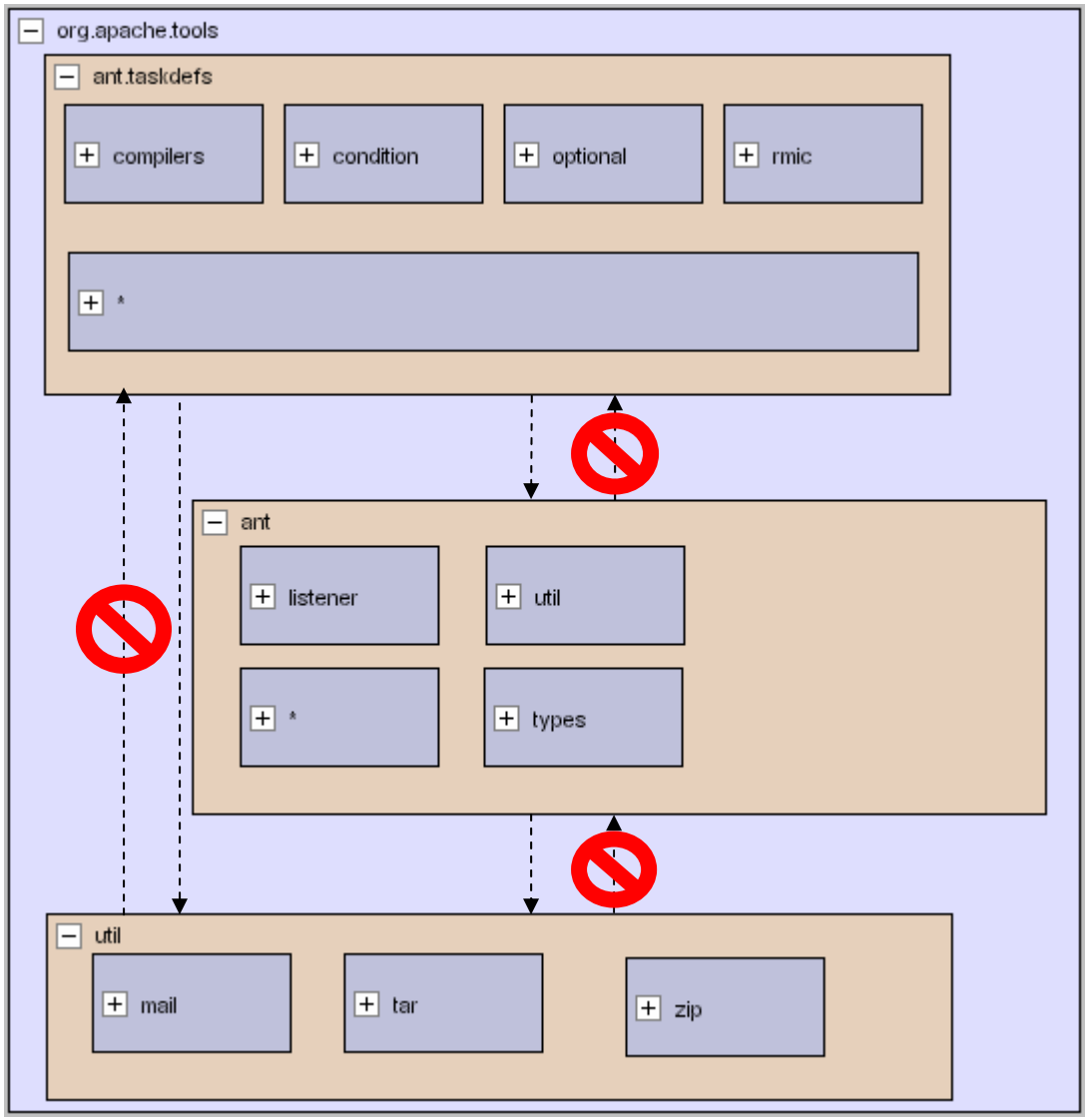


$root		1	2	3	4	5
com.example	application 1	.				
	model 2	37	.			
	domain 3	17	26	.		
	framework 4	75	53	40	.	
	util 5	10	13	16	13	.

Rules for Layering

1. $root can-use $root
2. model cannot-use application
3. domain cannot-use application, model
4. framework cannot use application, model, domain
5. util cannot-use application, model, domain, framework

ANT Conceptual Architecture



Layered Architecture with Three Subsystems

Tasks Use Common Infrastructure

Key Goal: Allow Independent Development of Tasks

DEMO

The Evolution of ANT

Agenda

New Representation for Software Architecture using Dependency Structure Matrix

Architectural Patterns

Design Rules

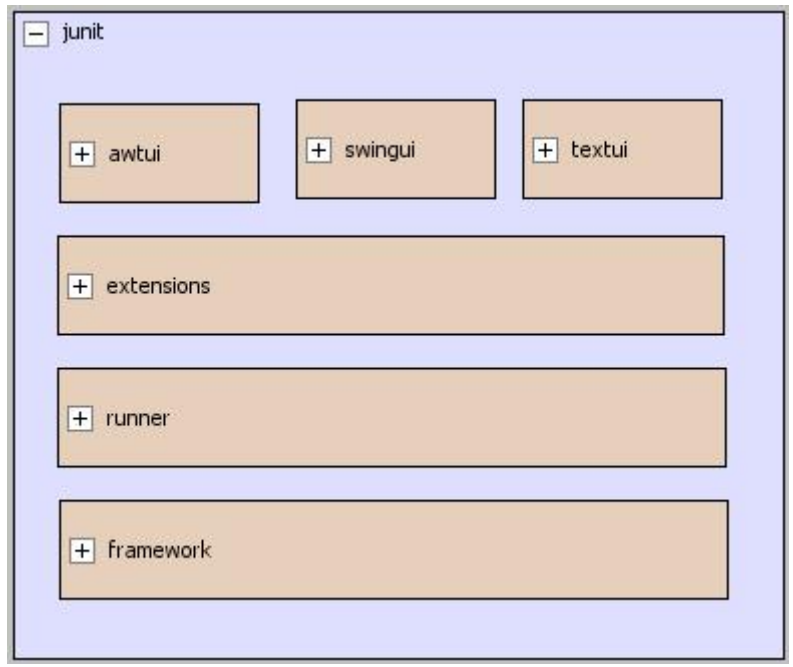
Architectural Evolution (Demo)

Examples of Real Projects

Q&A

Example: JUNIT

$root		1	2	3	4	5	6
junit	awtui 1	.					
	swingui 2		.				
	textui 3			.			
	extensions 4		1		.		
	runner 5	3	13	5		.	
	framework 6	4	19	8	13	6	.



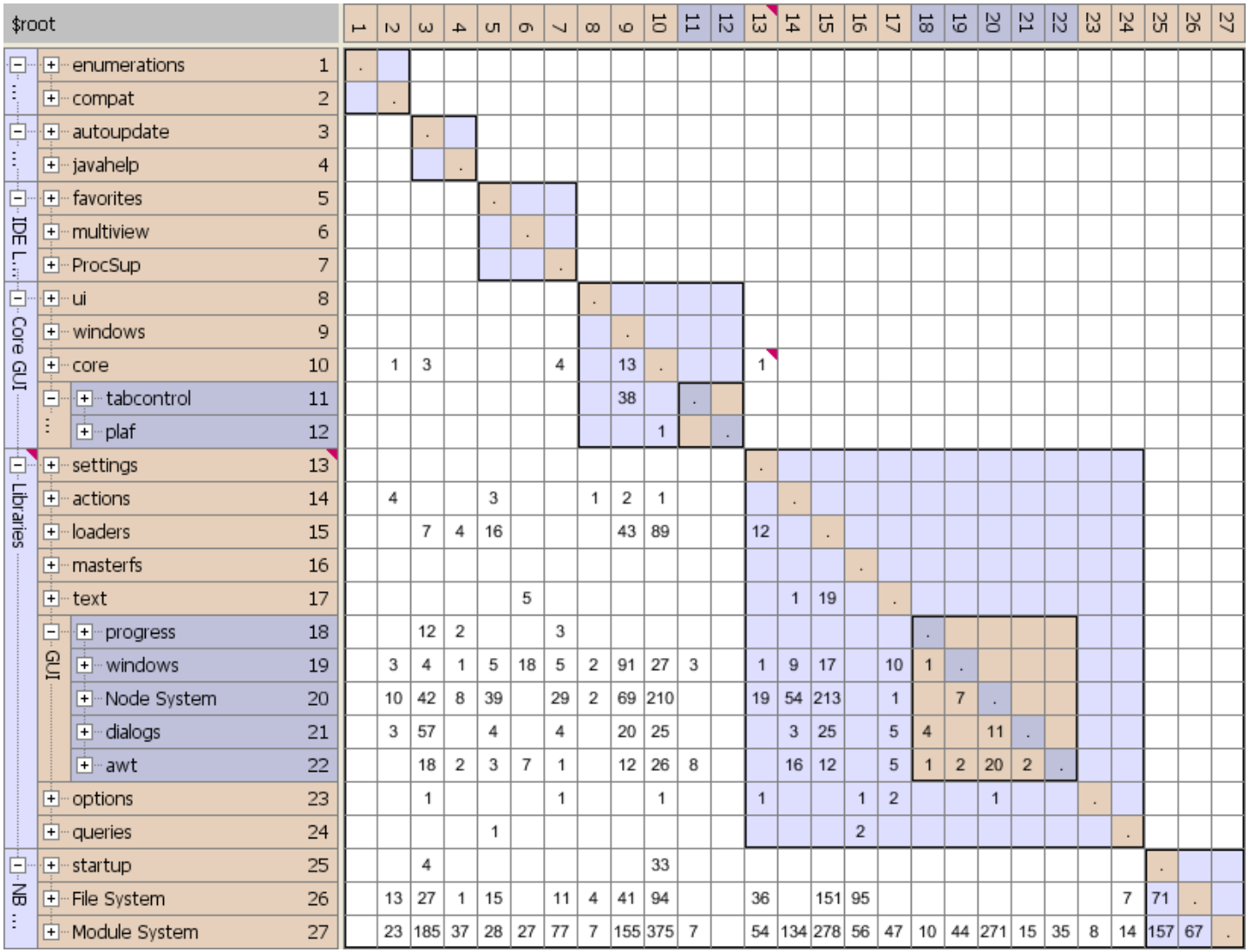
A Layered System with Independent User Interfaces

Example: jEdit

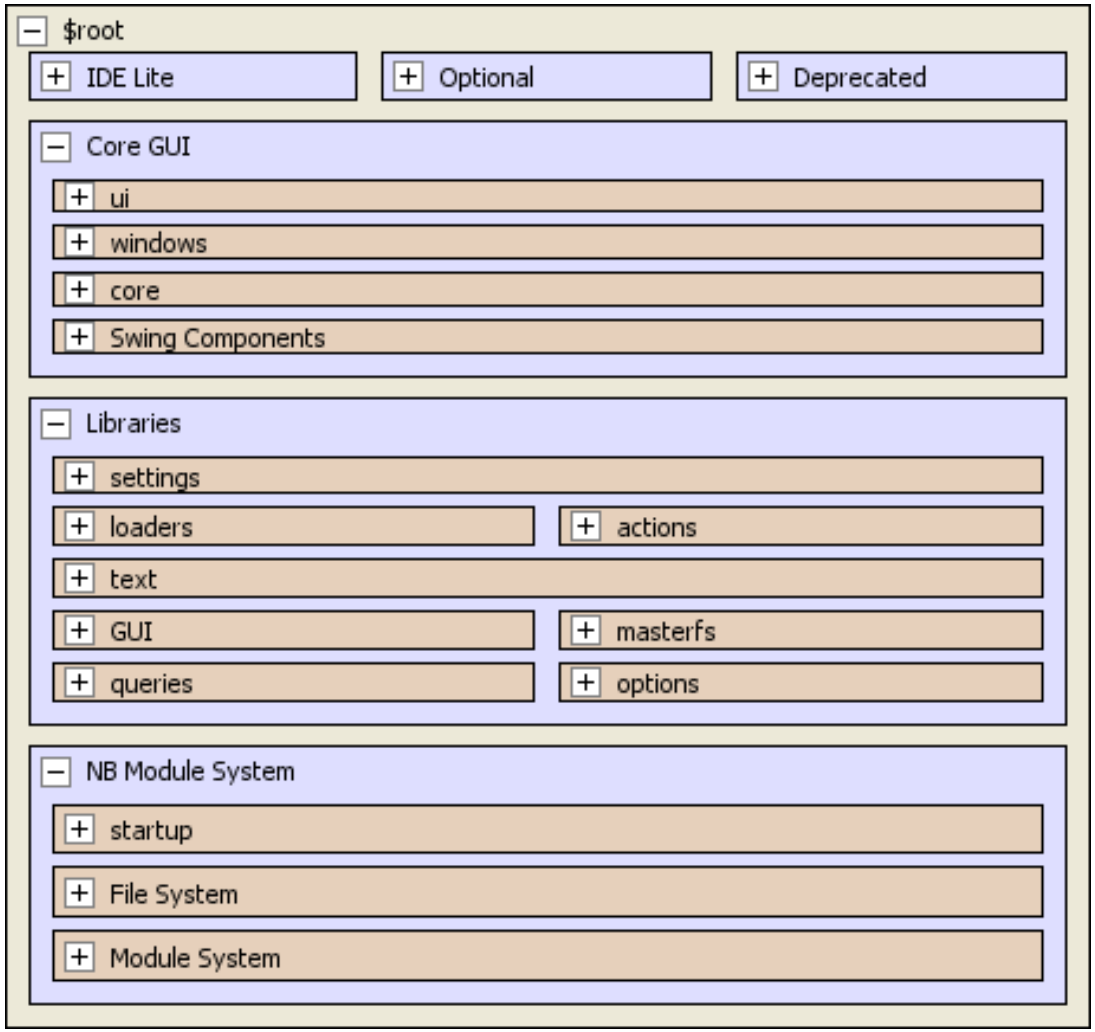
org.gjt.sp		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
jedit	print 1	.														
	proto.jeditresource 2		.													
	help 3			.						1				1		
	options 4				.					2				1		
	menu 5					.								4		
	browser 6			1		7	.							3		
	search 7						3	.						9		
	gui 8			2	23	5	7	12	.		6		2	42		1
	pluginmgr 9				2				1	.				1		
	textarea 10					1		13	11		.	1		21		
	buffer 11				1				1		4	.	1	13		
	io 12			4	1	4	29	9	3	2		3	.	16		
	* 13	11	4	17	103	59	41	51	138	24	31	19	25	.	2	22
	syntax 14	3			4				1		6	1		16	.	
	msg 15			1		2	4	3	4	2			3	25		.

Monolithic System ?

Example: NetBeans™ Platform



Conceptual Architecture for NetBeans™



Precise Big Picture View Derived From The Dependencies

View Shows Accurate Layering And Vertical Splitting

Summary: Big Picture View That Scales

- Highly scalable—Represent massive systems to give you a precise big picture view
- Formalize design intent and prevent architectural erosion
- Easy to adopt—Use it at any stage of the lifecycle
- Critical visibility of the architecture is achieved very quickly—try it out on your own software!

For More Information

- Information on DSMs: http://www.dsmweb.org
- Neeraj Sangal, Ev Jordan, Vineet Sinha, Daniel Jackson, “Using Dependency Models to Manage Complex Software Architecture”, OOPSLA 2005 http://sdg.lcs.mit.edu/pubs/2005/oopsla05-dsm.pdf
- Steven D. Eppinger, “Innovation at the Speed of Information”, Harvard Business Review, January 2001
- Baldwin, C.Y. and Clark K.B., The Power of Modularity Volume 1, MIT Press, Cambridge, MA, 2000
- Lattix: http://www.lattix.com

Q&A

Neeraj Sangal

To Know the Dependencies Is to Understand the Architecture

Neeraj Sangal

President
Lattix, Inc.
http://www.lattix.com

TS-6037