Managing Dependencies Between Software Modules Using DSM

johan.van.den.muijsenberg@alten.nl

Managing Dependencies

- Why you should care
- Reasons we fail
- How design structure matrix can help
  - Method
  - Tooling
  - VanDerLande Experiences

Business Case

Why you should care about managing dependencies

Business Needs and Obstacles

- Development Speed ⟷ Rigidity
  - Extra effort cascading changes due to chain of dependencies
- Software Platform ⟷ Immobility
  - Can not isolate reusable parts due to excessive dependencies
- Schedule Predictability ⟷ Fragility
  - Frequent unexpected failures in other parts due to complex or implicit dependencies
- Early/continuous integration ⟷ Testability
  - Can not unit test due to excessive dependencies

Dependencies are Essential

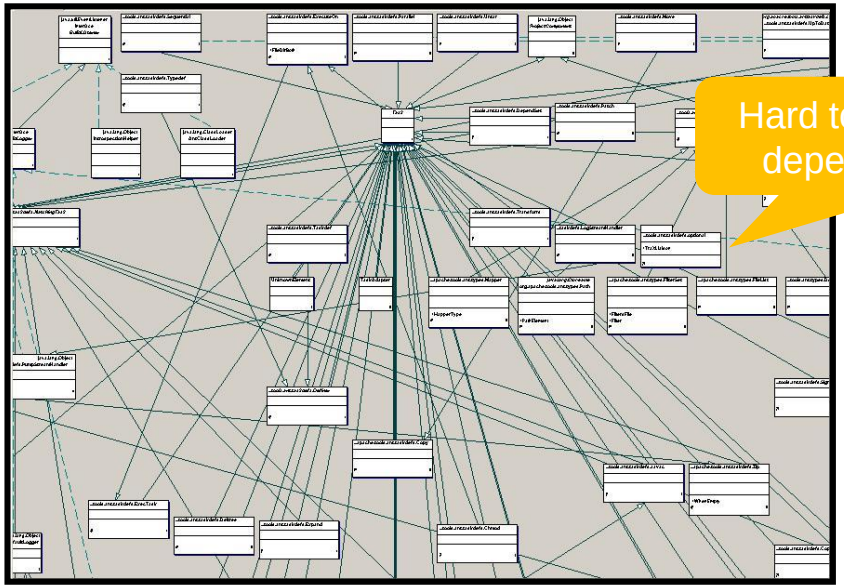
- Modularity in software needed
  - Manage complexity
  - Allow parallel work
  - Provide design flexibility through encapsulation
- Decisions on modularization affect dependencies
  - Well known design principles
    - Coupling and cohesion
    - Single responsibility principle
    - Avoid dependency cycles
    - Depend towards interfaces

Reasons We Fail

- Lack of awareness in some cases
- Lack of precision principles
- Architecture documentation intended for understanding
  - Partial description - Not complete/formal - Gaps
- Developers can easily violate defined architecture
  - At source code level

UML Limitations

- UML not suitable for managing dependencies
  - Easily overwhelmed by dependencies
  - Dependencies in model not in any view



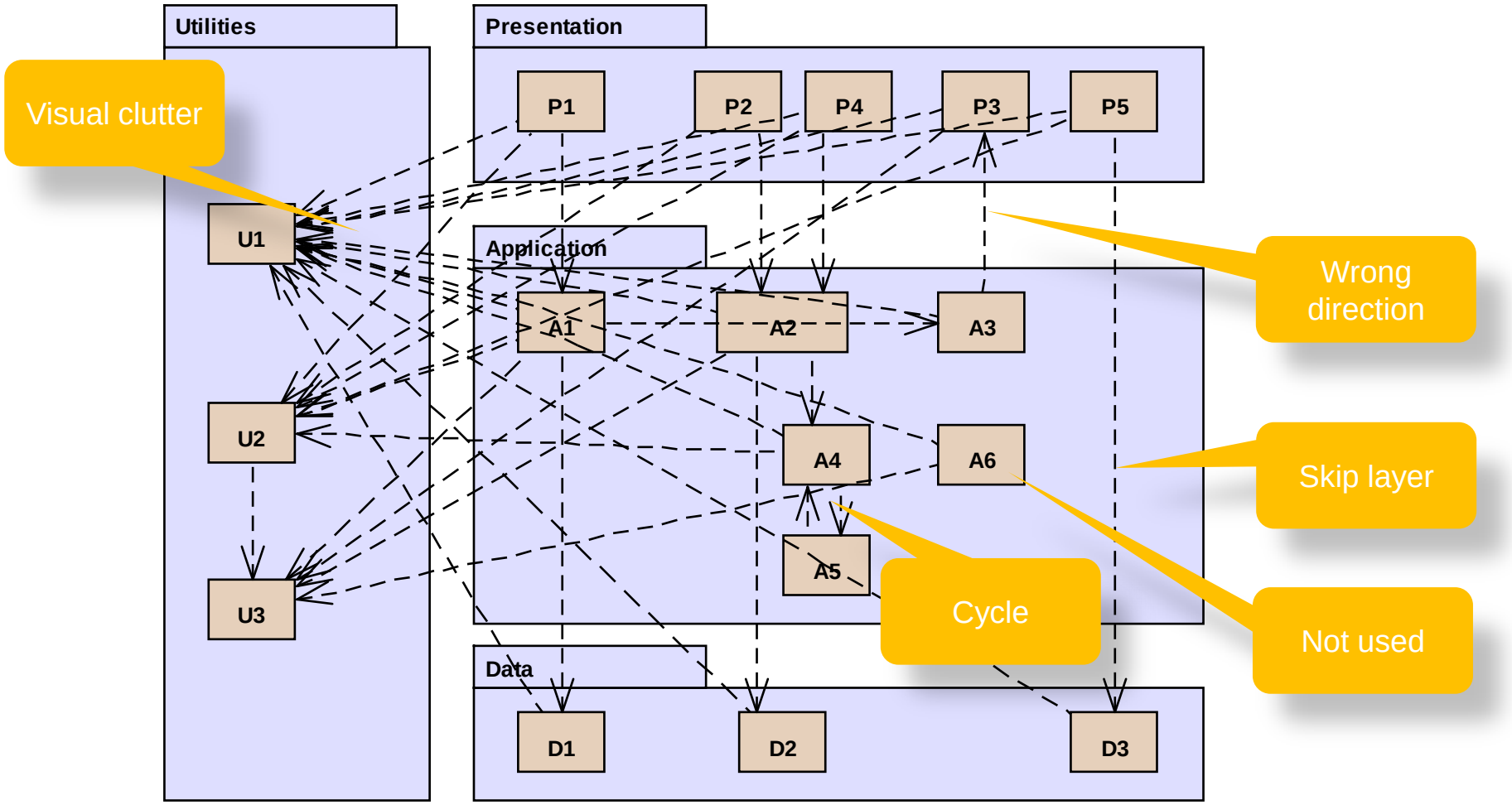
Dependency Structure Matrix

A overview of the method

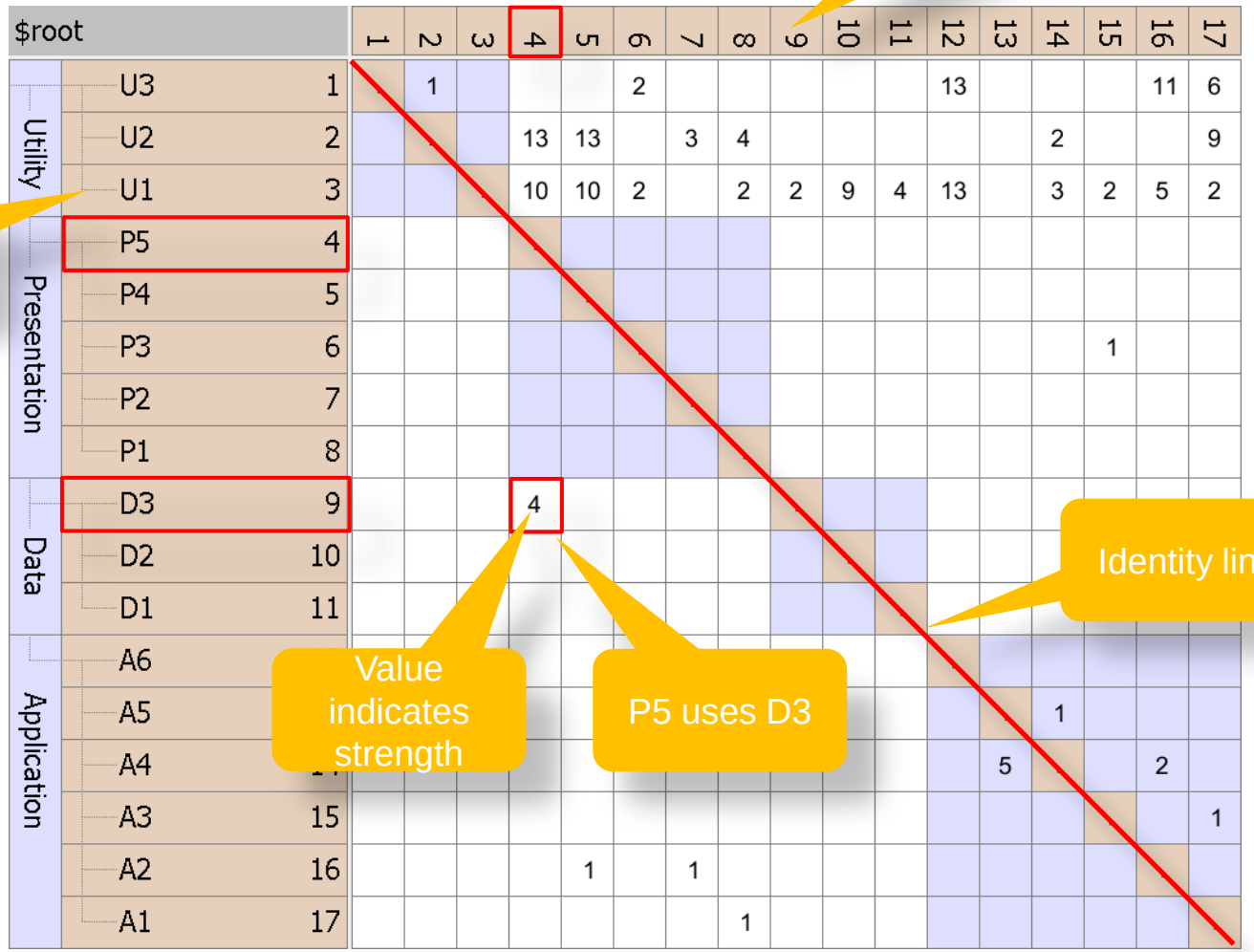
DSM Overview

- Created in 1970s
- Used to manage dependencies in very complex systems

Example


Utilities
Presentation
Application
Data
P1
P2
P4
P3
P5
U1
U2
U3
A1
A2
A3
A4
A6
A5
D1
D2
D3
Visual clutter
Wrong direction
Skip layer
Cycle
Not used

DSM Definition



Initial DSM

Key Strength - Concise

$root			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Utility	U3	1	.	1				2						13				11	6
	U2	2		.		13	13		3	4						2			9
	U1	3			.	10	10	2		2	2	9	4	13		3	2	5	2
Presentation	P5	4				.													
	P4	5					.												
	P3	6						.									1		
	P2	7							.										
	P1	8								.									
Data	D3	9				4					.								
	D2	10										.						7	
	D1	11											.						4
Application	A6	12												.					
	A5	13													.	1			
	A4	14													5	.		2	
	A3	15															.		1
	A2	16					1		1									.	
	A1	17								1									.

Initial DSM

Key Strength - Concise

$root		1	2	3	4	5	6	7	8	9
Utility	1	.	59	15	26		5	2	16	17
Presentation	2		.					1		
Data	3		4	.					7	4
Application / A6	4				.					
Application / A5	5					.	1			
Application / A4	6					5	.		2	
Application / A3	7							.		1
Application / A2	8		2						.	
Application / A1	9		1							.

External dependencies aggregated

Partially collapsed DSM

Key Strength - Concise

$root		1	2	3	4
Utility	1	.	59	15	66
Presentation	2		.		1
Data	3		4	.	11
Application	4		3		.

Fully collapsed DSM

External dependencies further aggregated

Internal dependencies hidden

Can be used to represent very large systems with thousands of elements

Key Strength - Analysis

- Find layering by partitioning
- Cycles easily visible

$root		1	2	3	4
Utility	1	.	59	15	66
Presentation	2		.		1
Data	3		4	.	11
Application	4		3		.

Initial DSM

Key Strength - Analysis

- Find layering by partitioning
- Cycles easily visible

Consumers to top

Cycle

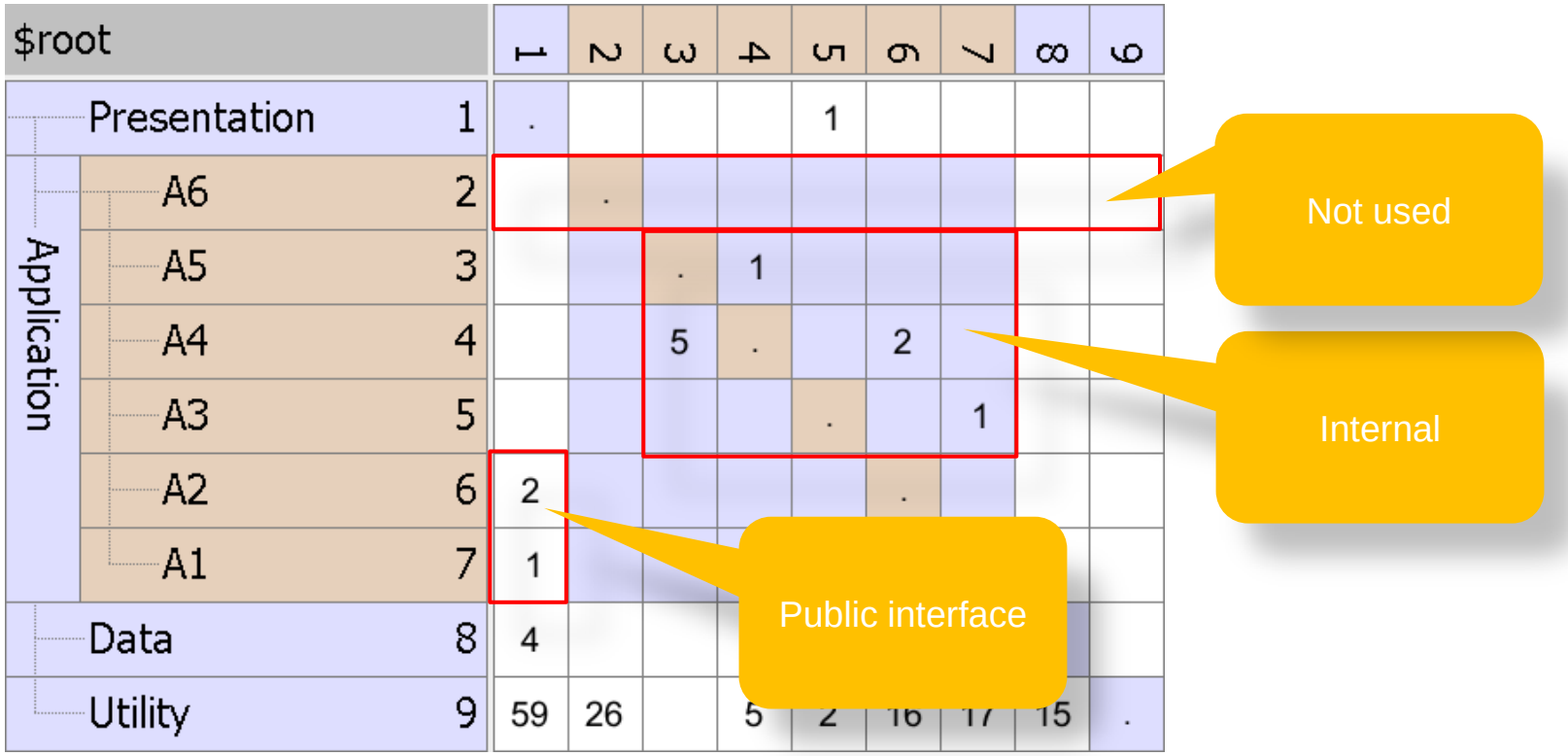
$root		1	2	3	4
Presentation	1	.	1		
Application	2	3	.		
Data	3	4	11	.	
Utility	4	59	66	15	.

Partitioned DSM

Providers to bottom

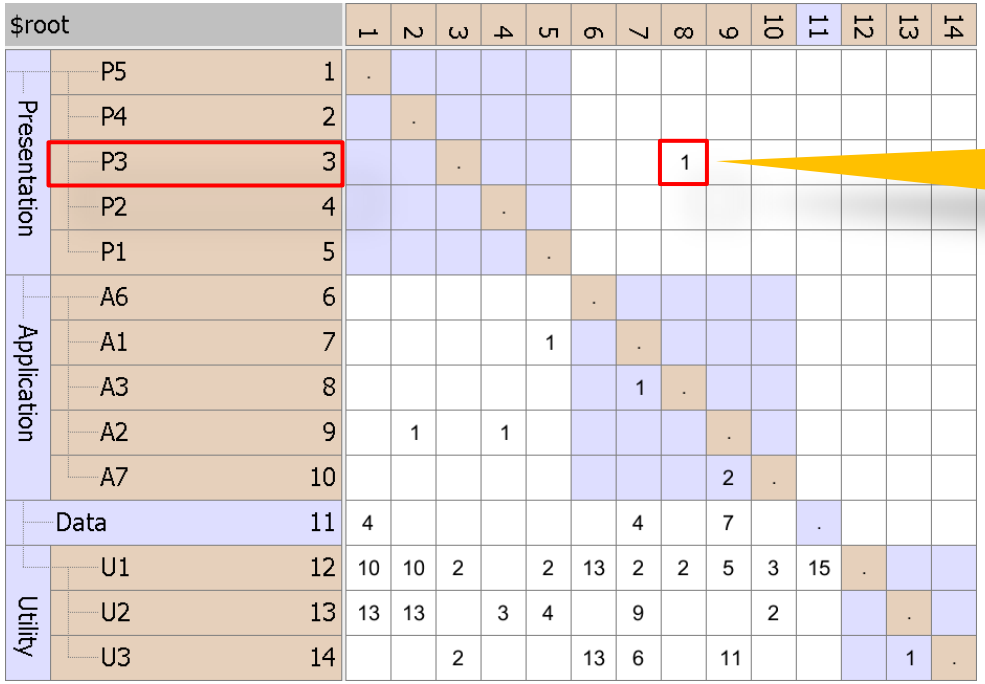
Key Strength - Analysis

- Discover public, internal and unused code



Key Strength - Refactoring

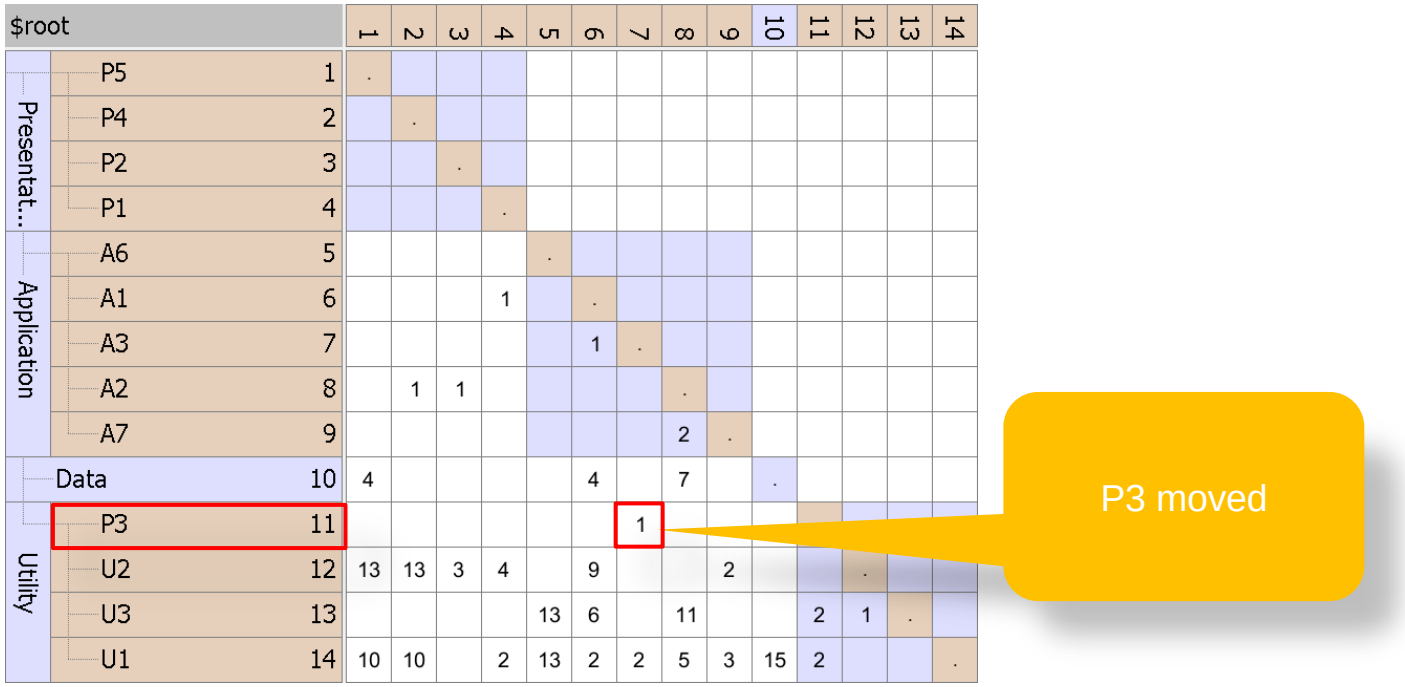
- What if scenarios can be done without changing code
  - Moving elements



Block triangular DSM

Key Strength - Refactoring

- What if scenarios can be done without changing code
  - Moving elements



Lower triangular DSM

Key Strength - Refactoring

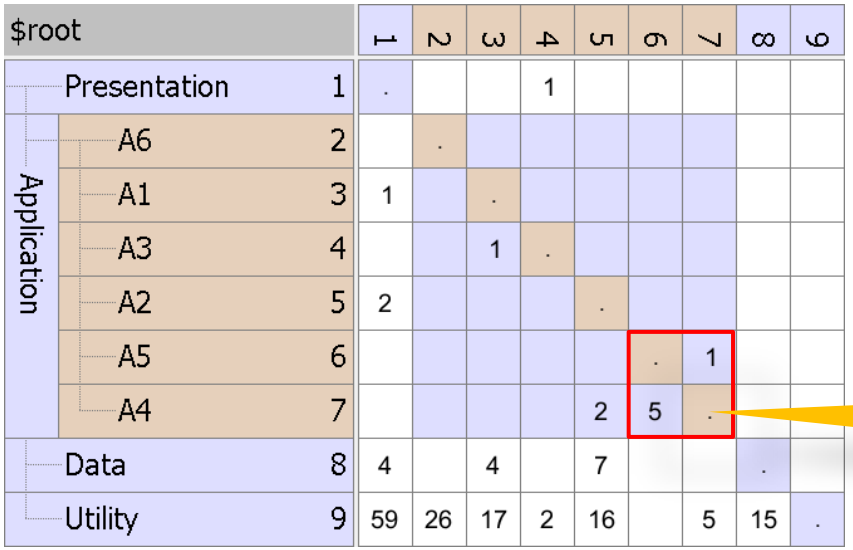
- What if scenarios can be done without changing code
  - Grouping elements

$root		1	2	3	4	5	6	7	8	9
Presentation	1	.				1				
Application: A6	2		.							
Application: A5	3			.	1					
Application: A4	4			5	.		2			
Application: A3	5					.		1		
Application: A2	6	2					.			
Application: A1	7	1						.		
Data	8	4					7	4	.	
Utility	9	59	26		5	2	16	17	15	.

Initial DSM

Key Strength - Refactoring

- What if scenarios can be done without changing code
  - Grouping elements



$root		1	2	3	4	5	6	7	8	9
Presentation	1	.			1					
Application: A6	2		.							
Application: A1	3	1		.						
Application: A3	4			1	.					
Application: A2	5	2				.				
Application: A5	6						.	1		
Application: A4	7					2	5	.		
Data	8	4		4		7			.	
Utility	9	59	26	17	2	16		5	15	.

Block triangular DSM after partitioning

Key Strength - Refactoring

- What if scenarios can be done without changing code
  - Grouping elements

$root		1	2	3	4	5	6	7	8	9
Presentation	1	.			1					
Application / A6	2		.							
Application / A1	3	1		.						
Application / A3	4			1	.					
Application / A2	5	2				.				
Application / A7 / A5	6						.	1		
Application / A7 / A4	7					2	5	.		
Data	8	4		4		7			.	
Utility	9	59	26	17	8	16		5	15	.

Lower triangular DSM

Grouped into A7

Key Strength - Refactoring

- What if scenarios can be done without changing code
  - Grouping elements

$root		1	2	3	4	5	6	7	8
Presentation	1	.			1				
Application A6	2		.						
Application A1	3	1		.					
Application A3	4			1	.				
Application A2	5	2				.			
Application A7	6					2	.		
Data	7	4		4		7		.	
Utility	8	59	2	17	2	16	5	15	.

Lower triangular DSM

Grouped into A7

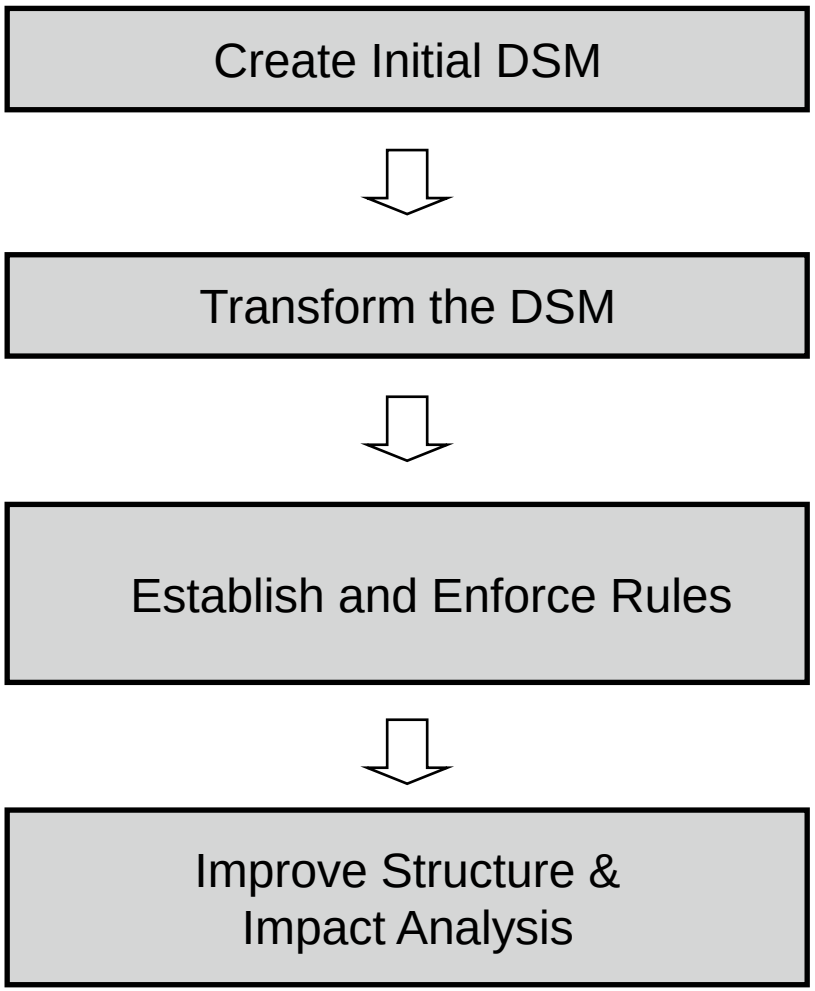
Lattix

Tool to apply DSM technology to software architecture

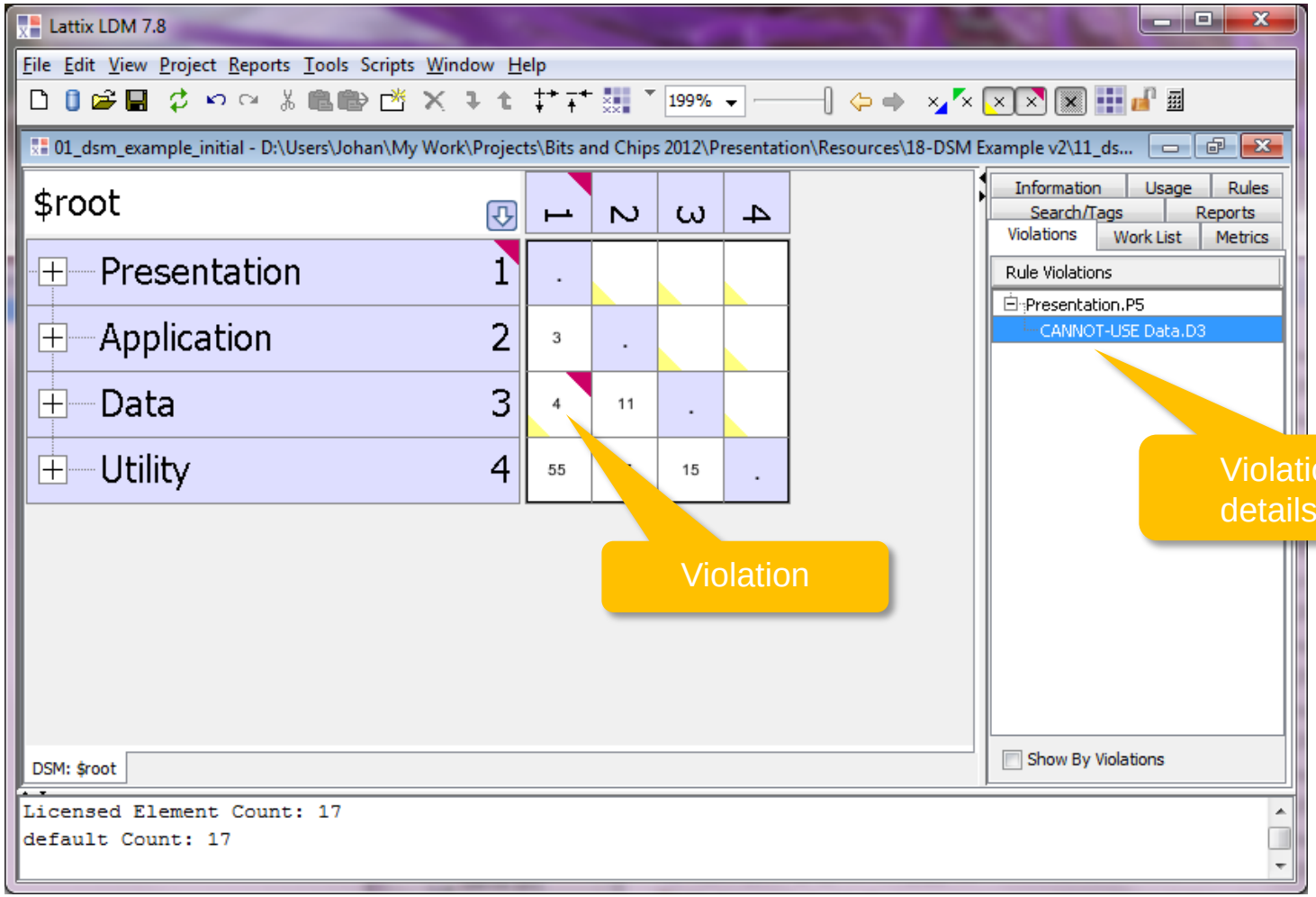
Introducing Lattix

- Discover and Identify Issues with Dependencies
  - Analysis using DSM techniques
- Specify/Enforce Architectures
  - Dependency rules
- Re-engineer and Refactor
  - Impact analysis
- Track, Measure and Report on Changes and Trends
  - Metrics and repository
- Wide range of input sources
  - C++, C, Java, Fortran, Ada, UML, .NET, databases, XML, Excel, LDI, etc…..

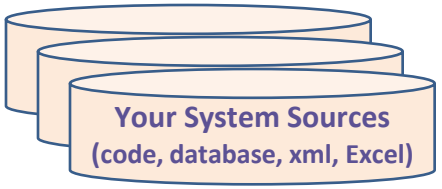
General Approach



Design Rules and Violations

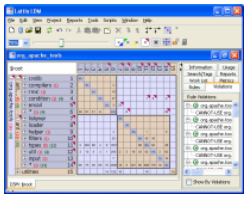


Lattix Toolset



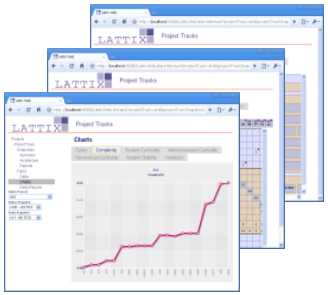
LDC – command line

LDC can automatically update the Repository after each build



LDM

Snapshots are published to the Repository to build Project Tracks of versions and view deltas and trends over time for your projects



LattixWeb

View Project Tracks with Snapshots for each build over time.

- Key metrics
- Violations
- Interactive DSM
- Trends
- Work list

VanDerLande Experiences

Quality Improvement

- Important ISO9126 Attributes
  - Maturity - Frequency of failure by faults
  - Analyzability - Effort for diagnosis of failures or for identification of parts to be modified
  - Testability - Effort needed for validating the modified software
  - Stability - Risk of unexpected effect of modifications
- Related to dependencies
  - Use DSM approach

Pilot

- Problem
  - Software related to volume or weight measurement must be certified by NMI and can not be changed easily
- Goal
  - Minimize amount of certified code
- Initial DSM analysis
  - 12% code base should certified due to indirect dependencies
- After refactoring using DSM
  - Reduced certified to 4% code base

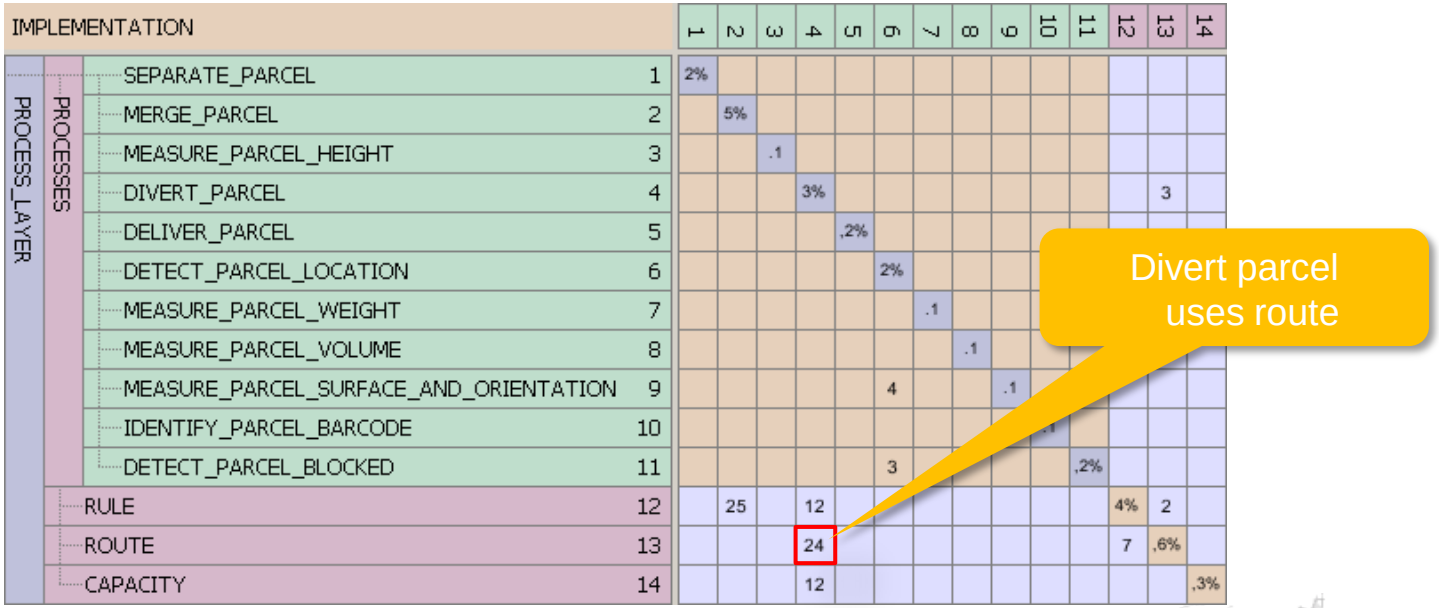
Follow up

- Lattix integrated into build
  - Using Jenkins
- Lattix considered for architecture improvement

Summary

Summary

- Lattix/DSM excellent for analysis/refactoring code base
  - Scales much better than UML
  - Analyze refactoring scenarios without code changes
  - Allows reasoning at higher abstraction level

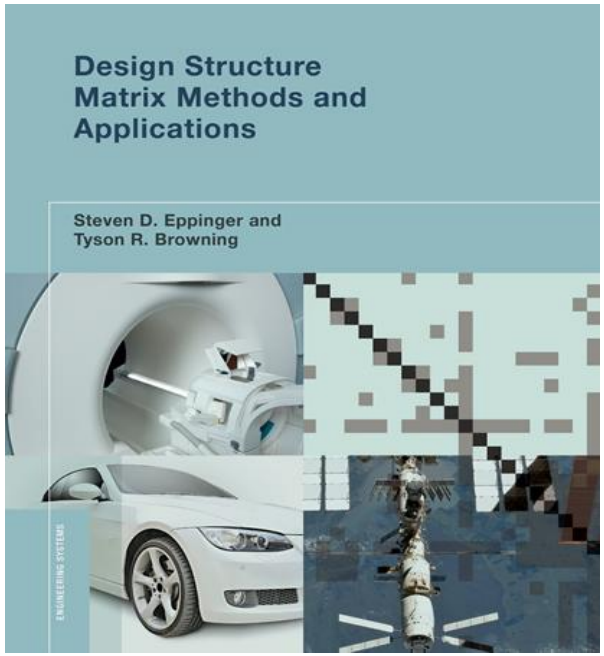


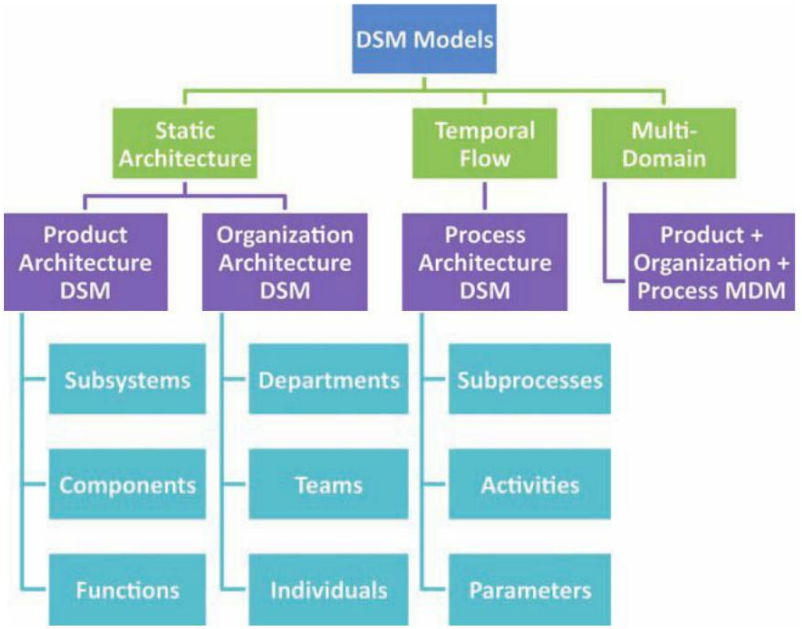
Summary

- Lattix/DSM fits well in agile approach
  - Use dependency rules to document architecture
  - Avoid architecture degradation by integrating Lattix in continuous build
- Lattix/DSM used for product line architecture (Ricoh)
  - Support migration
  - Allow evolution
  - Check conformance code

Summary

- DSMs can be used for many other purposes
  - For details see book on DSM





More Information

- www.dsmweb.org
  - General information on DSMs
- www.lattix.com
  - Overview
  - Demo tour
  - Knowledge base

Thanks

Any Questions ?

Visit our stand for more information