MEASURING, TRACKING, AND COMMUNICATING CHANGE IN ENTERPRISE SYSTEMS WITH A WEB-BASED REPOSITORY

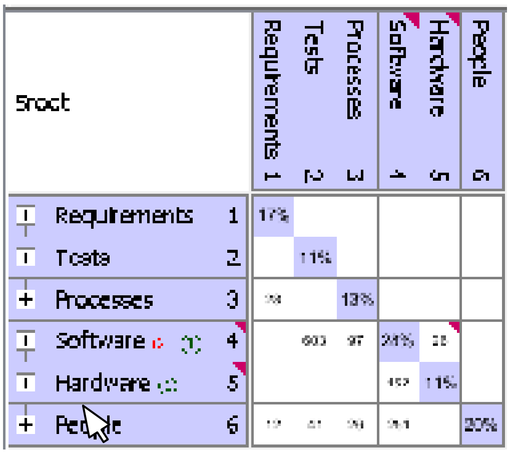
Frank Waldman and Neeraj Sangal
Lattix Inc., USA

Keywords: DSM, MDM, Repository, Track, Trend

1 INTRODUCTION

The application of DSM in software development has been shown to be very effective for visualizing, analyzing and enforcing the architecture of a software system [1,2]. By extracting dependencies automatically from a codebase of a software application, it has been possible to quickly build an initial DSM based upon the actual implementation utilizing its existing structure. The DSM is then transformed to logically reflect the intended architecture of the system, which can be accomplished through both manual manipulation of its hierarchy and the use of special partitioning algorithms.

However, most complex software systems are a combination of technologies, with dependencies that span across different software languages, databases, and configurations [3]. Moreover, expressing the entire system architecture involves mapping of requirements, design parameters and rules, processes or use cases, infrastructure, and even organizations in addition to the software [4]. As illustrated in Figure 1 below, an MDM which captures the mappings between the domains provides a powerful visualization and means for conducting change impact analysis across the entire system.



$root	Requirements 1	Tests 2	Processes 3	Software 4	Hardware 5	People 6
Requirements 1	17%					
Tests 2		11%				
Processes 3	38		13%			
Software 4		603	97	24%	28	
Hardware 5				457	11%	
People 6	12	41	30	351		20%

Figure 1. MDM of an enterprise system

One key to creating and managing the MDM of such extensive systems is the automated extraction of the dependency information. In software systems it is possible to use static code analysis to parse and extract the information from code or SQL, but for other data sources it is necessary to extract from other repositories, tools, or files in many formats, such as text, XML, xls (e.g. Excel), and XMI (e.g UML/SysML). Another key is to be able to merge the data from the various sources into a single DSM project.

The final key is to be able to update the DSM project and produce reports which measure and track changes over time. In this paper, we will describe a new approach which utilizes a web-based repository to communicate changes in enterprise systems.

2 CREATING AN ENTERPRISE SYSTEM DSM

The biggest challenge in creating a DSM of any complex system is capturing and expressing the system and dependency data, especially when the data sources are different for each of the constituent domains. Each domain requires a data model which consists of both the different kinds of system elements and the various kinds of dependencies which define the nature of the relationships between the elements. Many modules have been developed which utilize parsers to recognize the data in the code, SQL, XML, and other standard formats. An Excel convertor was developed to map data from columns in a spreadsheet or DSM macros into a specified data model.

Various scripts have been written not only to extract or import data from non-standard sources, but also create mappings from one domain to another. For example, the dependencies of code to a database can be mapped by looking for patterns in the SQL statements embedded in the code. These patterns will vary depending upon the different conventions employed by the system developers.

Once the data has been captured in the DSM project file repository, it establishes a baseline based on the original data. It is then possible to manually add new elements and dependencies, map dependencies across domains, and perform merges. The existing hierarchy can be manipulated manually or through DSM partitioning algorithms to establish the desired structure or sequence of the system at any level. Dependency rules can then be created to communicate the allowable dependencies between the elements, such that violations will be visible in the DSM.

As the DSM is transformed, the project file repository can be updated manually so that different file versions are created. It can also be updated at any point that there are changes in any of the data sources, either manually or automatically through the use of a command-line utility that is executed periodically at each data source. Reports and exports can also be generated manually or through the command line utilities, but most users found it too difficult to incorporate these into their development environments and so were not communicating the project results effectively.

3 WEB-BASED REPOSITORY APPROACH

A web-based repository is simply an application which can provide viewing of data via a web browser. Each repository is hosted on a networked machine which has a web server that can serve data to the browser. Each repository can host a number of Projects, each of which can have a succession of Project Snapshots in a Project Track. The Project Track can then be used to produce a trend of key metrics and show comparisons between selected Project Snapshots.

The desktop tool, LDM, includes access to the repository so that a user can at any time publish a Project Snapshot. The command line tool, LDC, can also be used to automatically publish a snaphot whenever a data source has been updated. Figure 2 illustrates each piece of the entire system.

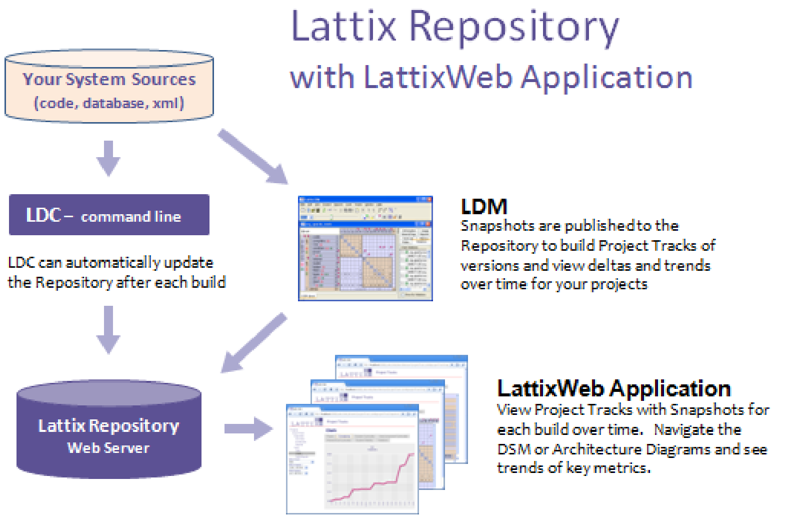

Lattix Repository
with LattixWeb Application
Your System Sources
(code, database, xml)
LDC – command line
LDC can automatically update
the Repository after each build
LDM
Snapshots are published to the
Repository to build Project Tracks of
versions and view deltas and trends
over time for your projects
Lattix Repository
Web Server
LattixWeb Application
View Project Tracks with Snapshots for
each build over time. Navigate the
DSM or Architecture Diagrams and see
trends of key metrics.

Figure 2. Repository system

In order to illustrate the utility of the Repository, consider the following example using an open source system. The initial DSM created with LDM and its Java module has been published as the baseline snapshot, with the view of the DSM and the key architecture metrics shown in Figure 3.

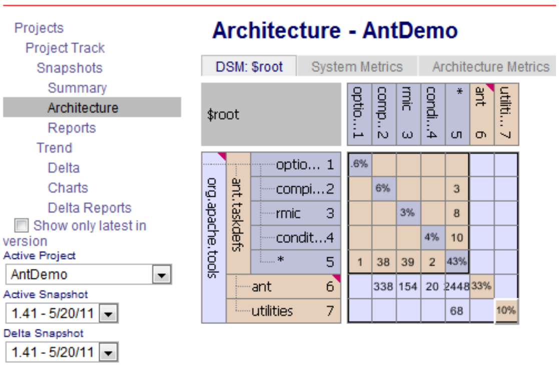
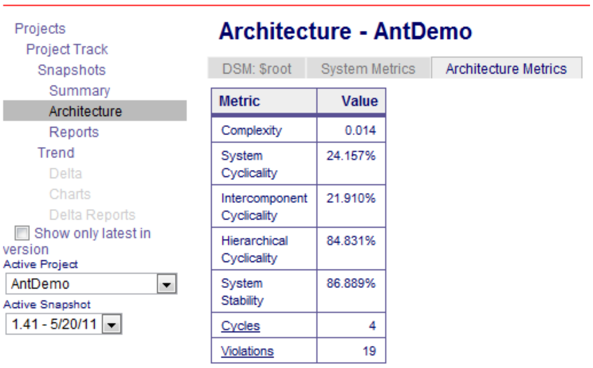


Figure 3a. Architecture baseline



Metric	Value
Complexity	0.014
System Cyclicality	24.157%
Intercomponent Cyclicality	21.910%
Hierarchical Cyclicality	84.831%
System Stability	86.889%
Cycles	4
Violations	19

Figure3b. Architecture metrics baseline

Note that the **ant.taskdef** subsystem is highly coupled. After analysis of the coupling, some changes were made to the subsystem by moving some system elements and hiding (tearing) some dependencies which would require refactoring of the code. As this is a "what-if" scenario, we want to assess the benefit in terms of the architecture metrics. By uploading a snapshot of the revised DSM, we can quickly compare the DSM against the baseline and see the delta which shows the three dependencies that we hid at the element level as shown in Figure 4.

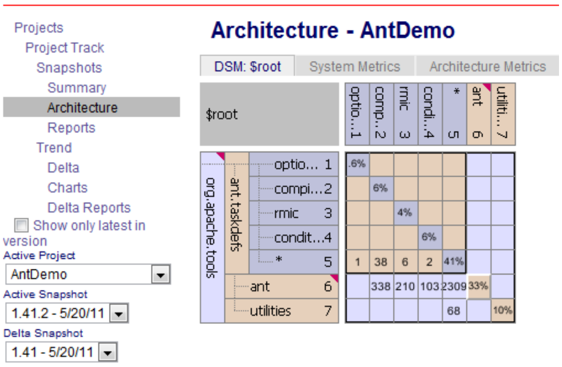


Figure 4a. Architecture of revised DSM

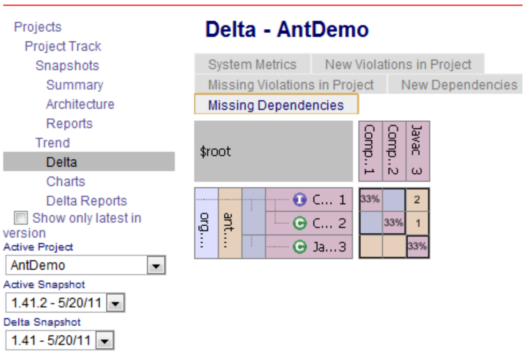
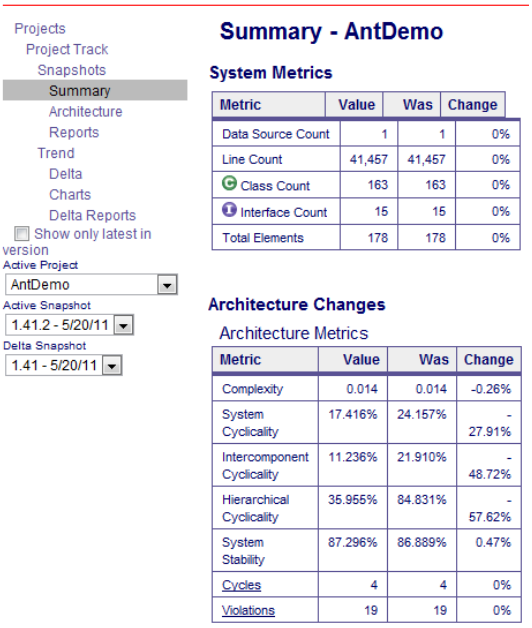


Figure4b. Missing dependencies in Delta

The Summary automatically calculates not only the metrics for the newest snapshot, but also displays a comparison to the previous snapshot in percentage terms. In Figure 5, we can see that our proposed changes would have a significant impact on the cyclicality metrics by decoupling a key subsystem. As these metrics have been shown to have a strong correlation to the rate of defects in the codebase [5], we expect a significant benefit by implementing these changes in the actual code as well as improving the architectural understanding of those who are working on the project.

When the project sources, in this case the Java codebase, are actually revised with each new version release, then the repository provides valuable insight into the trends of the system. We can see in Figure 6 that significant increases in cyclicality occurred between early major releases but lately there have been efforts to control it. Although the level of architectural violations significantly increased with the 1.6 version major release as shown in Figure 7, we can observe by the subsequent drop in later releases that the team was actively working on improving the architectural conformance. We can also see in Figure 8 that the size of the system has steadily increased and is a likely factor in the driving up the complexity while the team makes progress in managing the architecture.



Summary - AntDemo

System Metrics

Metric	Value	Was	Change
Data Source Count	1	1	0%
Line Count	41,457	41,457	0%
Class Count	163	163	0%
Interface Count	15	15	0%
Total Elements	178	178	0%

Architecture Changes

Architecture Metrics

Metric	Value	Was	Change
Complexity	0.014	0.014	-0.26%
System Cyclicality	17.416%	24.157%	-27.91%
Intercomponent Cyclicality	11.236%	21.910%	-48.72%
Hierarchical Cyclicality	35.955%	84.831%	-57.62%
System Stability	87.296%	86.889%	0.47%
Cycles	4	4	0%
Violations	19	19	0%

Figure 5. Architectural changes

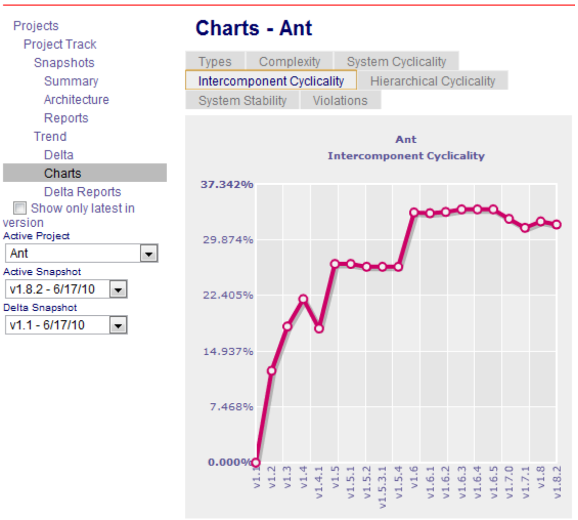


Figure 6. Chart of intercomponent cyclicality

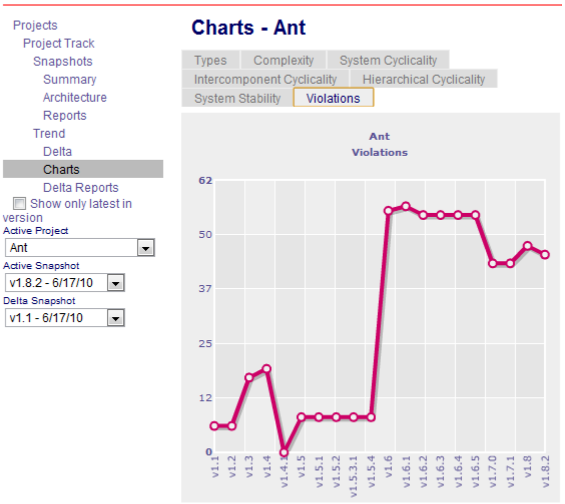


Figure 7. Chart of architectural violations

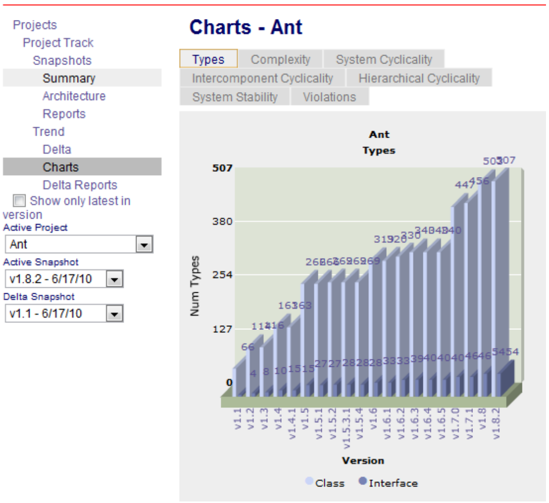


Figure 8. Chart of system elements

4 CONCLUSION

A new web-based DSM repository approach has been introduced to improve the capability of users to measure, track, and communicate the status of complex systems. A web application can enable access to automated reports of metrics and trends, as well as providing the ability to generate comparisons of selected versions of the project.

REFERENCES

[1] Waldman F and Jordan E. (2004). Using DSMs to Manage the Architecture of Software Systems. In *Proceedings of 6th International DSM Conference*, Cambridge UK, September 2004.

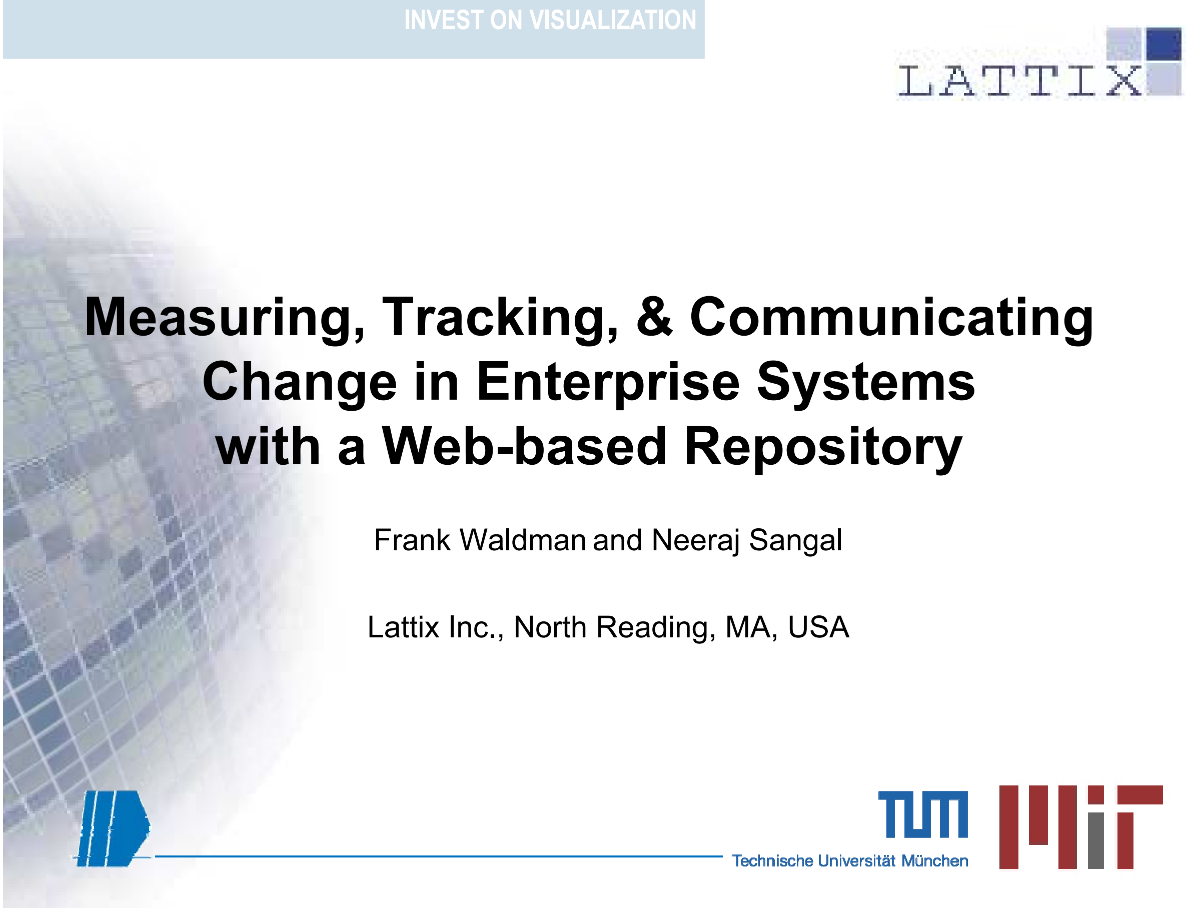
[2] Sangal, N., Jordan, E., Sinha, V. and Jackson, D. (2005). Using Dependency Models to Manage Complex Software Architecture. In *Proceedings of the 20th annual ACM SIGPLAN Conference on Object-Oriented Programming Systems Languages and Applications*, pp 167-176, San Diego, California, October 2005.

[3] Waldman, F. and Sangal, N. (2009). Applying DSM to Enterprise Architectures. In *Proceedings of 11th International DSM Conference*, Clemson, October 2009.

[4] Bhaskara, S. (2010). DSM Based Approach for Managing Requirements, Rules and Design Parameters in Knowledge Based Design Process. In *Proceedings of 12th International DSM Conference*, Cambridge UK, July 2010.

[5] Sosa M.E., Browning T. and Mihm J. (2008). A Dynamic, DSM-based View of Software Architectures and Their Impact on Quality and Innovation. In *Proceedings of 10th International DSM Conference*, Stockholm, Sweden, November 2008.

Contact: Frank Waldman
Lattix Inc.
8 Harper Circle
Andover, MA 01810
USA
Tel. : +1.978.474.5022
Fax : +1.978.222.8468
e-mail: frank.waldman@lattix.com
www.lattix.com

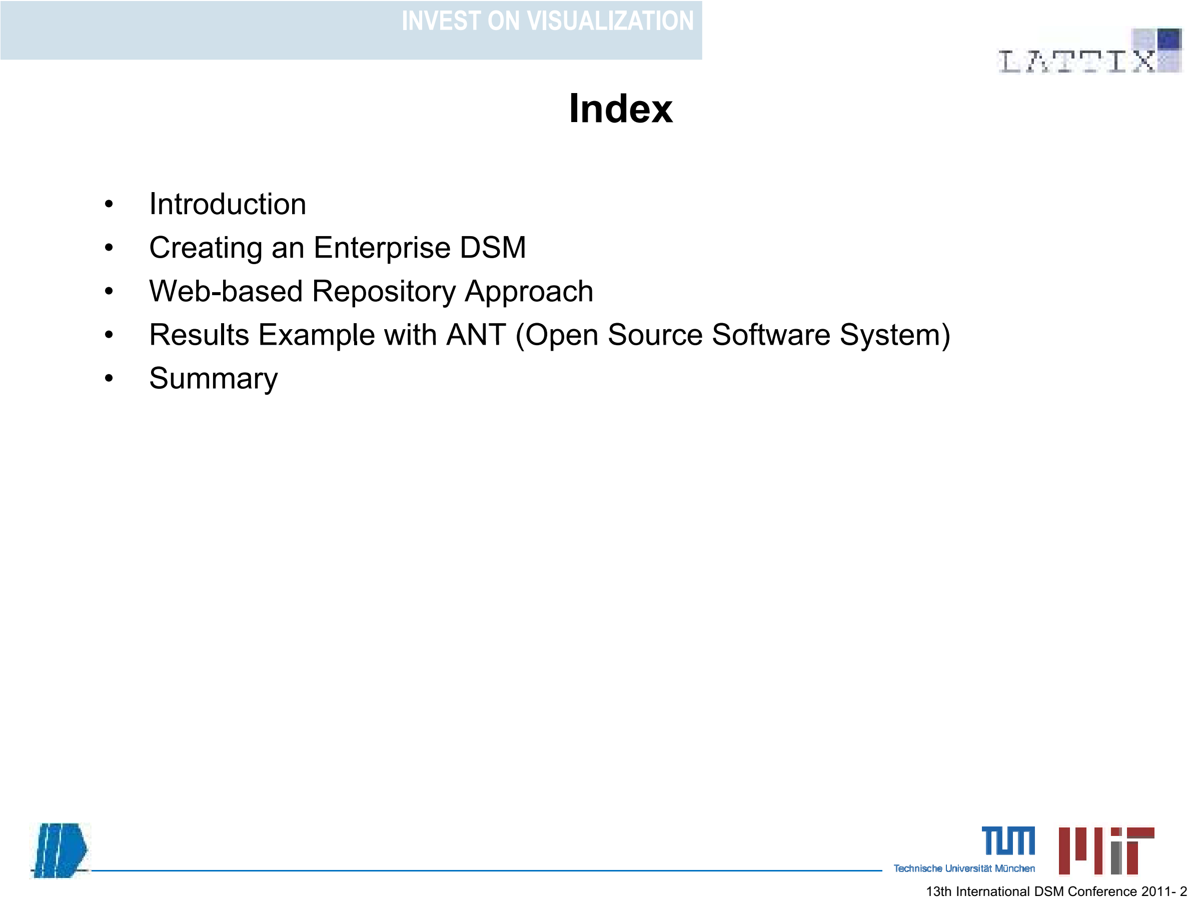


INVEST ON VISUALIZATION

Measuring, Tracking, & Communicating Change in Enterprise Systems with a Web-based Repository

Frank Waldman and Neeraj Sangal

Lattix Inc., North Reading, MA, USA



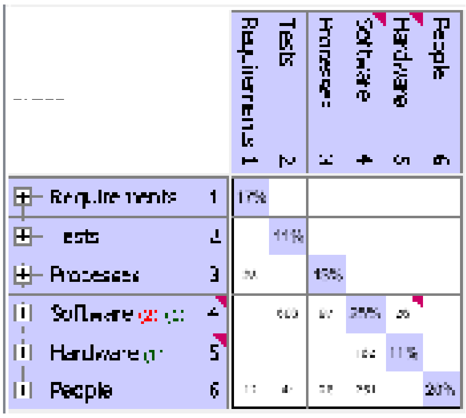
INVEST ON VISUALIZATION

Index

- Introduction
- Creating an Enterprise DSM
- Web-based Repository Approach
- Results Example with ANT (Open Source Software System)
- Summary

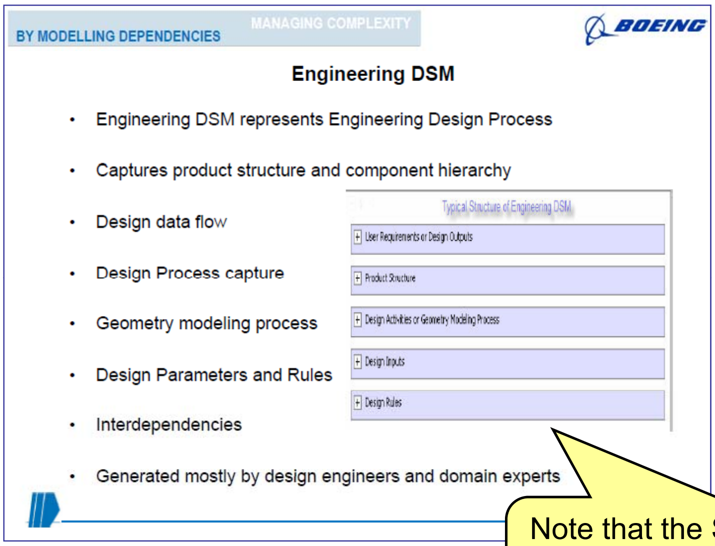
Introduction

- Previous presentations about DSM for software architectures
- Scope has expanded to enterprise systems with mapping of dependencies across many domains
- MDM provides powerful visualization and change impact analysis across the domains
- Key to creating and managing enterprise MDMs is data extraction and integration
- Key to adoption is the capability to automatically measure, track, & report to communicate results
- A new web-based repository approach has been developed



		1 Requirements	2 Tests	3 Processes	4 Software	5 Hardware	6 People
⊞ Requirements	1	17%					
⊞ Tests	2		11%				
⊞ Processes	3	[illegible]		15%			
Software	4		600	97	25%	26	
Hardware	5				164	11%	
People	6	11	4	28	251		20%

MDM Example



BY MODELLING DEPENDENCIES MANAGING COMPLEXITY

BOEING

Engineering DSM

- Engineering DSM represents Engineering Design Process
- Captures product structure and component hierarchy
- Design data flow
- Design Process capture
- Geometry modeling process
- Design Parameters and Rules
- Interdependencies
- Generated mostly by design engineers and domain experts

Typical Structure of Engineering DSM

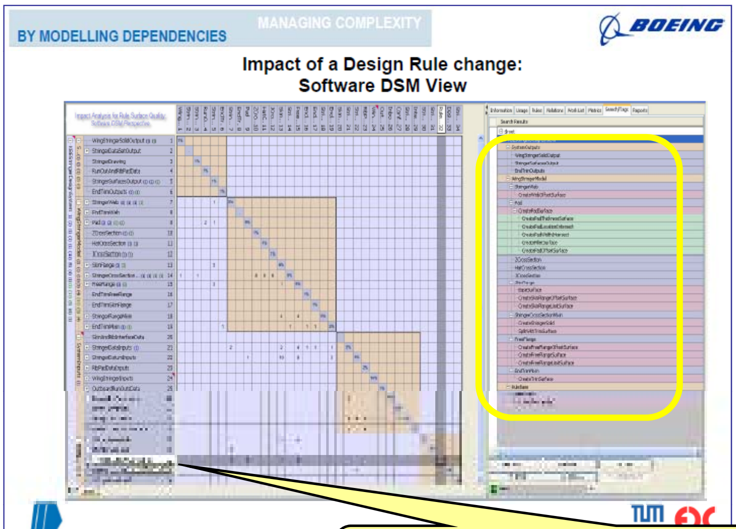
- ⊞ User Requirements or Design Outputs
- ⊞ Product Structure
- ⊞ Design Activities or Geometry Modeling Process
- ⊞ Design Inputs
- ⊞ Design Rules

Note that the Structure is an MDM with Requirements, Product Structure & Design Rules

INVEST ON VISUALIZATION

LATTIX

MDM Change Impact Analysis



BY MODELLING DEPENDENCIES

MANAGING COMPLEXITY

BOEING

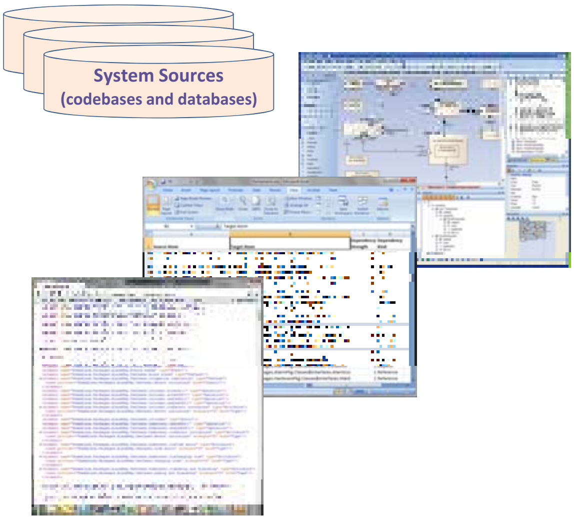
Impact of a Design Rule change:
Software DSM View

Shows how a Design Rule change can impact User Requirements & System Specifications

INVEST ON VISUALIZATION

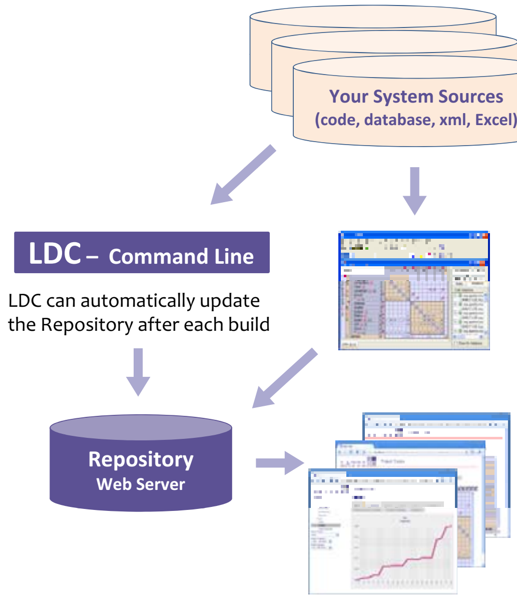
LATTIX

Creating an Enterprise MDM



- Extract data from a wide range of sources including software, databases, models, documents and spreadsheets
- Utilize parsers for languages, SQL, XML, and other standard formats
- An Excel converter can produce the XML data model from either matrix or columnar formats
- Scripts have been written not only for extracting and importing data, but also to create mappings from one domain to another

Repository Overview



Your System Sources
(code, database, xml, Excel)

LDC – Command Line

LDC can automatically update the Repository after each build

Repository
Web Server

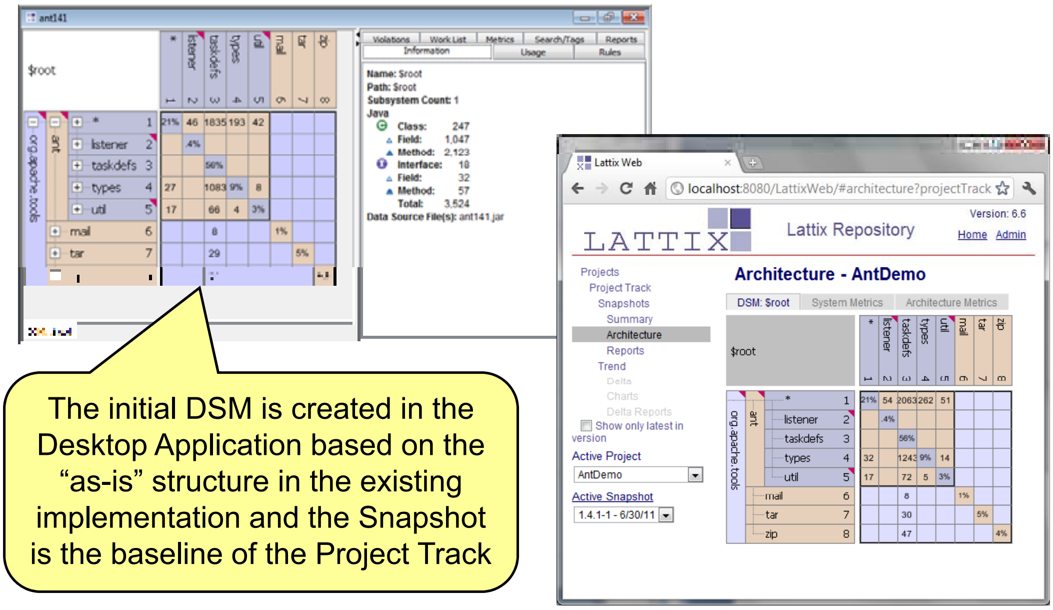
Desktop Application

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

Web Application

View Project Tracks with Snapshots for each build over time. Navigate the DSM and Architecture Diagrams and see trends of key metrics

Project Baseline



ant141

$root

		*	listener	taskdefs	types	util	mail	tar	zip
		1	2	3	4	5	6	7	8
*	1	21%	46	1835	193	42			
listener	2		.4%						
taskdefs	3			56%					
types	4	27		1083	9%	8			
util	5	17		66	4	3%			
mail	6			8			1%		
tar	7			29				5%	

Name: $root
Path: $root
Subsystem Count: 1
Java
Class: 247
Field: 1,047
Method: 2,123
Interface: 18
Field: 32
Method: 57
Total: 3,524
Data Source File(s): ant141.jar

Lattix Web

localhost:8080/LattixWeb/#architecture?projectTrack

LATTIX Lattix Repository Version: 6.6 Home Admin

☐ Show only latest in version

Active Project: AntDemo

Active Snapshot: 1.4.1-1 - 6/30/11

Architecture - AntDemo

$root

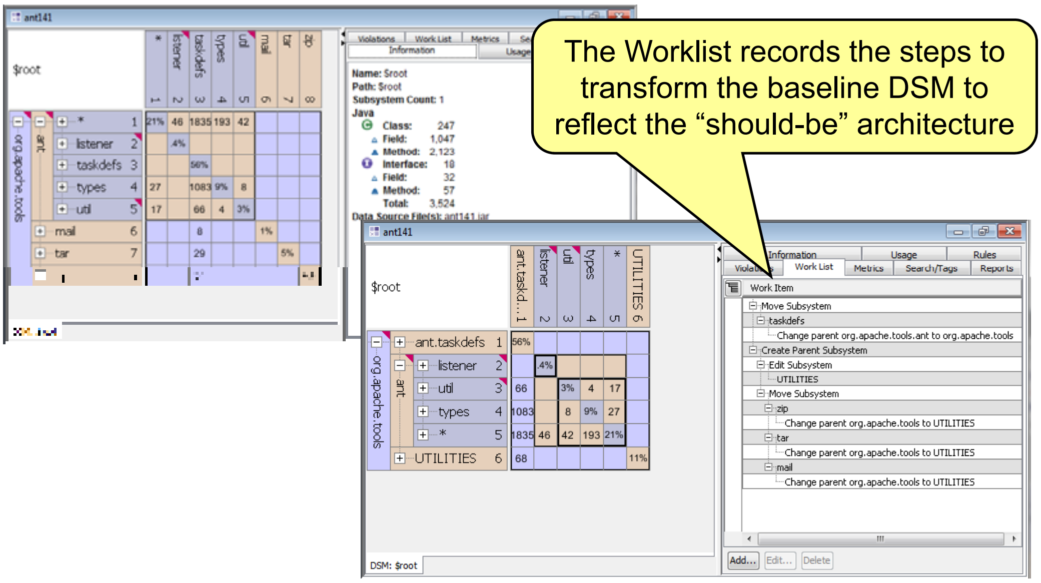
		*	listener	taskdefs	types	util	mail	tar	zip
		1	2	3	4	5	6	7	8
*	1	21%	54	2063	262	51			
listener	2		.4%						
taskdefs	3			56%					
types	4	32		1243	9%	14			
util	5	17		72	5	3%			
mail	6			8			1%		
tar	7			30				5%	
zip	8			47					4%

The initial DSM is created in the Desktop Application based on the "as-is" structure in the existing implementation and the Snapshot is the baseline of the Project Track

INVEST ON VISUALIZATION

LATTIX

DSM Transformation



The Worklist records the steps to transform the baseline DSM to reflect the “should-be” architecture

ant141

$root	* 1	listener 2	taskdefs 3	types 4	util 5	mail 6	tar 7	zip 8
org.apache.tools / ant / * 1	21%	46	1835	193	42			
listener 2		.4%						
taskdefs 3			56%					
types 4	27		1083	9%	8			
util 5	17		66	4	3%			
mail 6			8			1%		
tar 7			29				5%	

Violations | Work List | Metrics | Information | Usage

Name: $root
Path: $root
Subsystem Count: 1
Java

Class:	247
Field:	1,047
Method:	2,123
Interface:	18
Field:	32
Method:	57
Total:	3,524

Data Source File(s): ant141.jar

ant141

$root	ant.taskd... 1	listener 2	util 3	types 4	* 5	UTILITIES 6
org.apache.tools / ant.taskdefs 1	56%					
ant / listener 2		.4%				
util 3	66		3%	4	17	
types 4	1083		8	9%	27	
* 5	1835	46	42	193	21%	
UTILITIES 6	68					11%

DSM: $root

Information | Usage | Rules
Violations | Work List | Metrics | Search/Tags | Reports

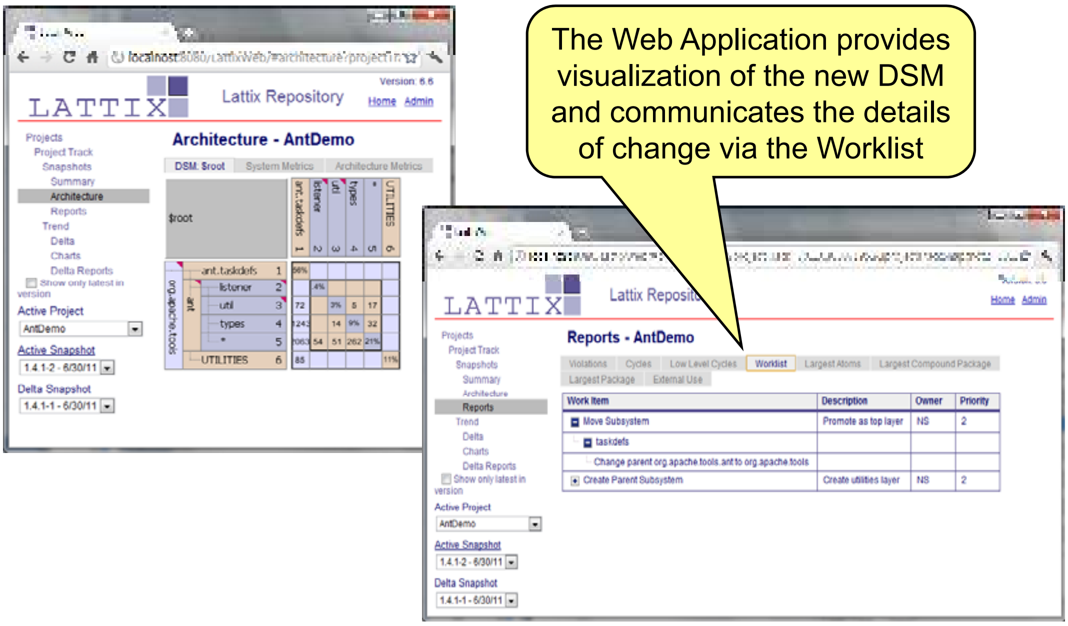
Work Item

- Move Subsystem
  - taskdefs
    - Change parent org.apache.tools.ant to org.apache.tools
- Create Parent Subsystem
  - Edit Subsystem
    - UTILITIES
  - Move Subsystem
    - zip
      - Change parent org.apache.tools to UTILITIES
    - tar
      - Change parent org.apache.tools to UTILITIES
    - mail
      - Change parent org.apache.tools to UTILITIES

INVEST ON VISUALIZATION

LATTIX

Communicating Change



The Web Application provides visualization of the new DSM and communicates the details of change via the Worklist

localhost:8080/LattixWeb/#architecture?projectI...

LATTIX Lattix Repository Version: 6.6 Home Admin

Active Project: AntDemo
Active Snapshot: 1.4.1-2 - 6/30/11
Delta Snapshot: 1.4.1-1 - 6/30/11

Architecture - AntDemo

DSM: $root | System Metrics | Architecture Metrics

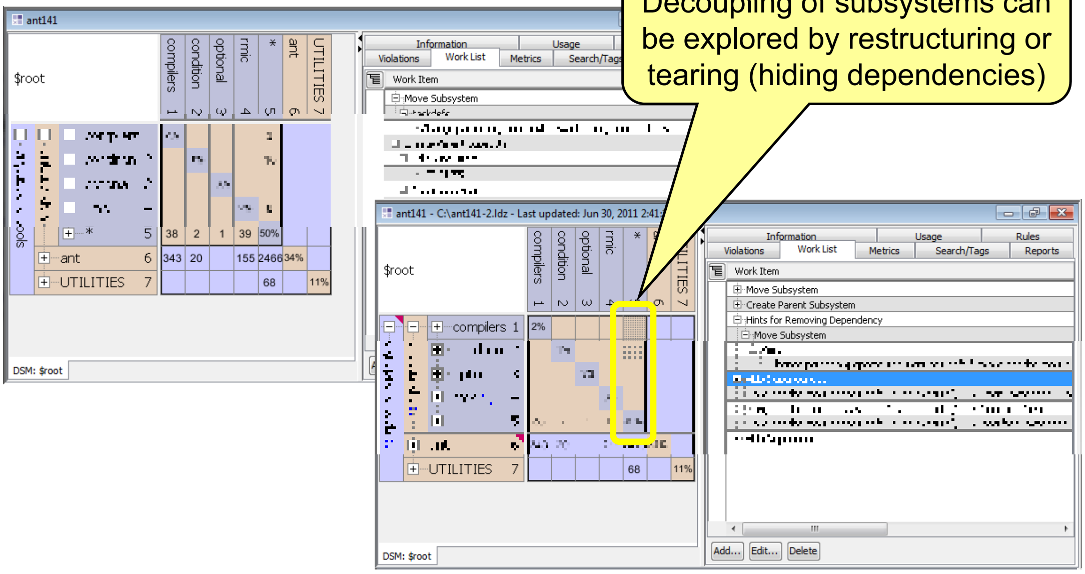
$root	ant.taskdefs 1	listener 2	util 3	types 4	* 5	UTILITIES 6
org.apache.tools / ant.taskdefs 1	56%					
ant / listener 2		.4%				
util 3	72		3%	5	17	
types 4	1243		14	9%	32	
* 5	[illegible]	54	51	262	21%	
UTILITIES 6	85					11%

Reports - AntDemo

Violations | Cycles | Low Level Cycles | Worklist | Largest Atoms | Largest Compound Package | Largest Package | External Use

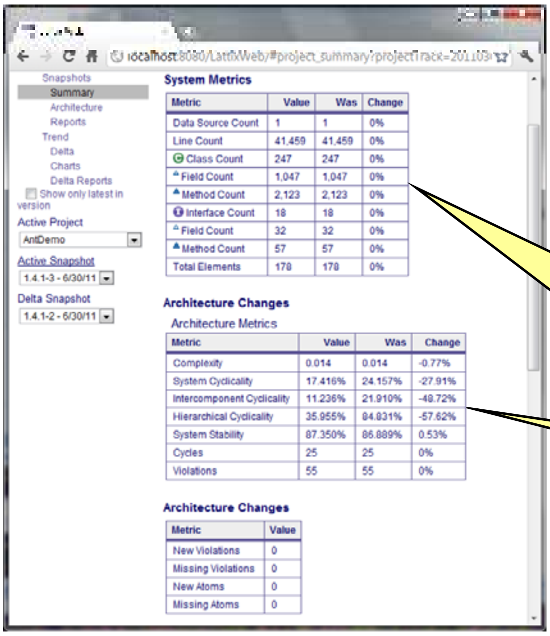
Work Item	Description	Owner	Priority
Move Subsystem	Promote as top layer	NS	2
taskdefs			
Change parent org.apache.tools.ant to org.apache.tools			
Create Parent Subsystem	Create utilities layer	NS	2

Exploring Improvements



Decoupling of subsystems can be explored by restructuring or tearing (hiding dependencies)

Measuring Improvements



System Metrics

Metric	Value	Was	Change
Data Source Count	1	1	0%
Line Count	41,459	41,459	0%
Class Count	247	247	0%
Field Count	1,047	1,047	0%
Method Count	2,123	2,123	0%
Interface Count	18	18	0%
Field Count	32	32	0%
Method Count	57	57	0%
Total Elements	178	178	0%

Architecture Changes

Architecture Metrics

Metric	Value	Was	Change
Complexity	0.014	0.014	-0.77%
System Cyclicality	17.416%	24.157%	-27.91%
Intercomponent Cyclicality	11.236%	21.910%	-48.72%
Hierarchical Cyclicality	35.955%	84.831%	-57.62%
System Stability	87.350%	86.889%	0.53%
Cycles	25	25	0%
Violations	55	55	0%

Architecture Changes

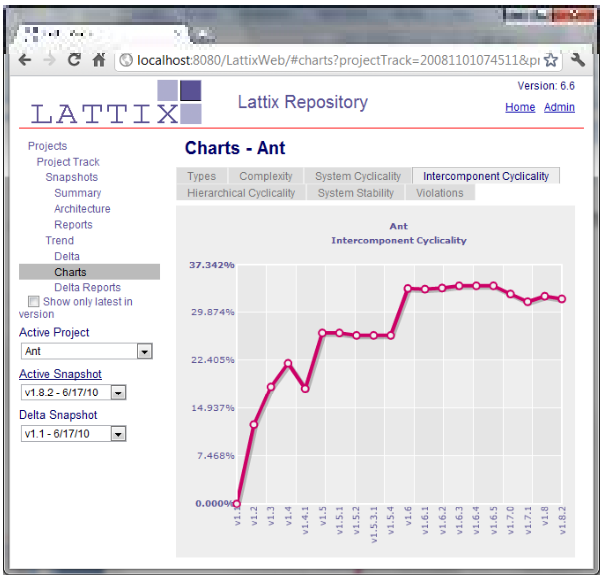
Metric	Value
New Violations	0
Missing Violations	0
New Atoms	0
Missing Atoms	0

The Summary view in the Repository automatically calculates and compares the latest two Snapshots

Since we are only conducting a “what-if” scenario, no changes in the system have yet been made

Since we have removed dependencies and restructured, we can see the significant reduction in cyclicality

Tracking Change



☐ Show only latest in version

Active Project: Ant

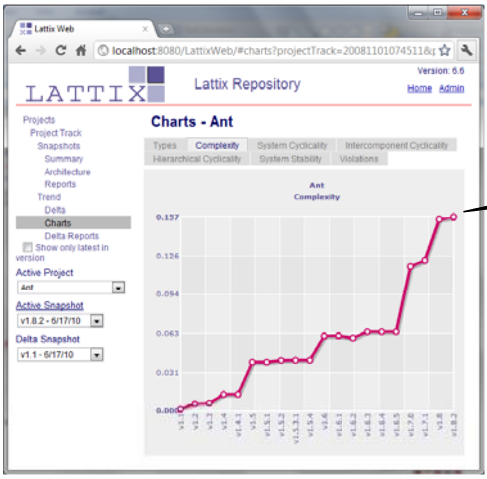
Active Snapshot: v1.8.2 - 6/17/10

Delta Snapshot: v1.1 - 6/17/10

Charts show trends of key metrics over any range of Snapsots, providing a relative measure of changes with each update of the system

In this example of ANT, several years of releases can be examined to see how the system has evolved

Tracking Change



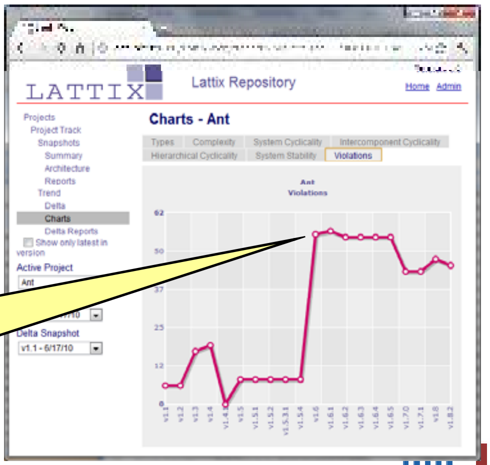
☐ Show only latest in version

Active Project: Ant

Active Snapshot: v1.8.2 - 6/17/10

Delta Snapshot: v1.1 - 6/17/10

With Charts we can see the relative rise in Complexity as the system has grown significantly



☐ Show only latest in version

Active Project: Ant

Delta Snapshot: v1.1 - 6/17/10

but efforts are being made to manage the system architecture and reduce Violations of the dependency rules

Summary

- A new web-based DSM repository approach has been introduced to improve the capability of users to measure, track, and communicate the status of complex systems
- The web application can enable access to automated reports of metrics and trends, as well as providing the ability to generate comparisons of selected versions of the project
- The key enabler for creating and managing enterprise MDMs is data extraction and integration