9TH INTERNATIONAL DESIGN STRUCTURE MATRIX CONFERENCE, DSM'07
16 – 18 OCTOBER 2007, MUNICH, GERMANY

APPLYING DSM TO ENTERPRISE ARCHITECTURES

Frank Waldman and Neeraj Sangal
Lattix Inc.

Keywords: DSM, Enterprise Architecture, Multi-Domain

1 INTRODUCTION

The application of DSM in software development has been focused on visualization and analysis of code bases associated with complex software applications. By extracting dependencies automatically from the code base of an application, it has been possible to quickly build an initial DSM based upon its code organization. The DSM must then be transformed to reflect the intended architecture of the application, which can be accomplished through both manual manipulation of its hierarchy and the use of special partitioning algorithms. Much value has been achieved by refactoring the code base to eliminate dependencies which violate the intended architecture and enforcing rules for allowable dependencies during subsequent builds of the application.

It is insufficient to consider today's complex software systems only in terms of code written in a specific language. They consist of multiple elements in a variety of languages, application frameworks, web services, databases, and configuration files. It is preferable to treat a complex software system as a system-of-systems than spans multiple domains. Interdependencies exist between these many domains of the system and an understanding of the overall architecture as well as the explicit structure of each domain is required.

Driven by customer requests, we have extended our DSM approach beyond software applications and have developed the capability to map dependencies across the domains of an enterprise architecture.

2 MULTI-DOMAIN DSM APPROACH

Our approach to managing the architecture of software applications, which was presented at previous DSM conferences [1−3], also works well in other domains. While each domain has different kinds of elements and different types of dependencies, the same data model and DSM analysis can be applied. For example, an application code base consists of packages with classes or directories with files, while a database system includes schemas, tables, packages, sequences, etc. Database architectures can have subsystems of schemas which are layered just like applications, with similar rules to prevent unwanted interdependencies.

Mapping dependencies across domains in software systems has been accomplished through a variety of techniques. For example, Hibernate enables the mapping of a database object to the application objects which use it. A system that uses Hibernate has Java code, Hibernate mapping files, and databases. Parsing just the Java code would provide interdependencies between Java classes, while parsing the database code provides interdependencies of its elements. By parsing the Hibernate mapping files, it is possible to extract the dependencies between database elements and Java code. The resulting DSM is now a multi-domain DSM which includes the application, the Hibernate mapping layer, and the database. In addition, it is possible to merge elements from the domains to eliminate the mapping layer and show direct connections in the DSM (see Figure 1).

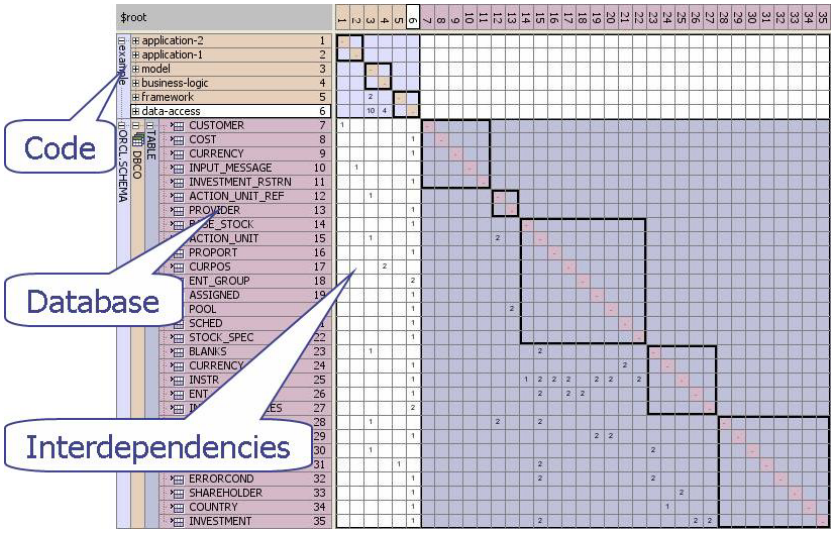


Figure 1. Multi-domain DSM of Application with Database

Spring Framework is a very popular application framework that uses metadata to configure enterprise Java applications. The architecture of the enterprise application is driven by the Spring configuration, which can now be parsed to extract the structure and dependencies between its elements such as Spring beans and Java classes (as illustrated in Figure 2).

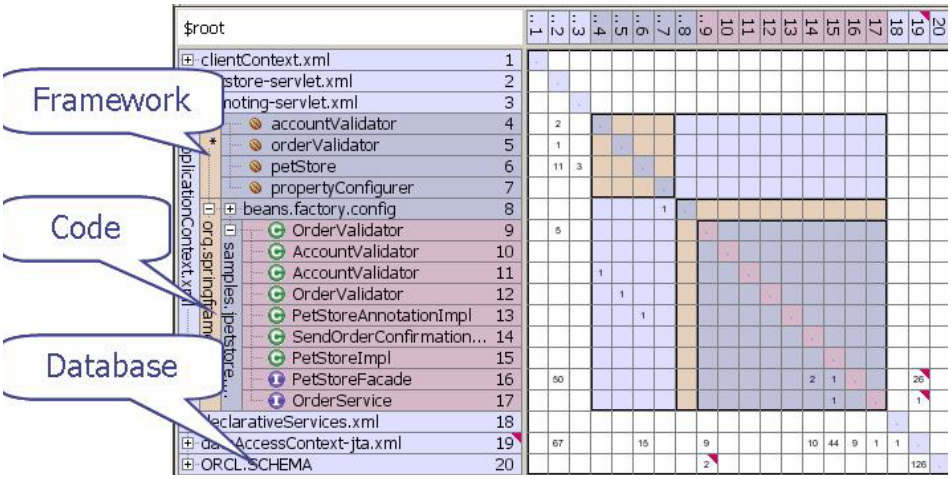


Figure 2. Multi-domain DSM with Application Framework

Finally, it is possible to extend this multi-domain approach to business processes and software services. This, for the first time, allows users to understand the relationships between business processes and the software architecture. Users can now query the system to understand which business processes would be affected by changes to the software architecture or how the architecture must change to accommodate evolving business processes.

3 CONCLUSION

Multi-domain DSM have can used to create the big picture view of the enterprise software architecture, extracted from the actual implementation. The hierarchy in the multi-domain DSM enables the scalability needed to represent the thousands of elements and millions of dependencies in complex enterprise architectures. With the enterprise architecture DSM, it is possible to reduce risk by better understanding the impact of change and how change propagates.

REFERENCES

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

[2] Waldman F and Sangal N. Results of DSM Analysis of Industrial Software Systems. 7th International DSM Conference, Seattle, November 2005.

[3] Sangal N. New Techniques for Leveraging Hierarchy in DSMs. 7th International DSM Conference, Seattle, November 2005.

Contact: Frank Waldman
Lattix Inc.
8 Harper Circle
Andover, MA 01810
USA
+1.978.474.5022
+1.978.222.8468
frank.waldman@lattix.co
http://www.lattix.com

Applying DSM to Enterprise Architectures

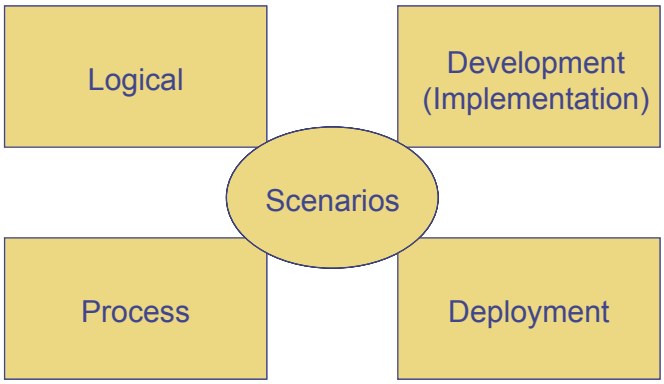
Frank Waldman and Neeraj Sangal

Lattix Inc.
USA

IN COOPERATION WITH BMW GROUP
CAPITALIZE ON COMPLEXITY
LATTIX

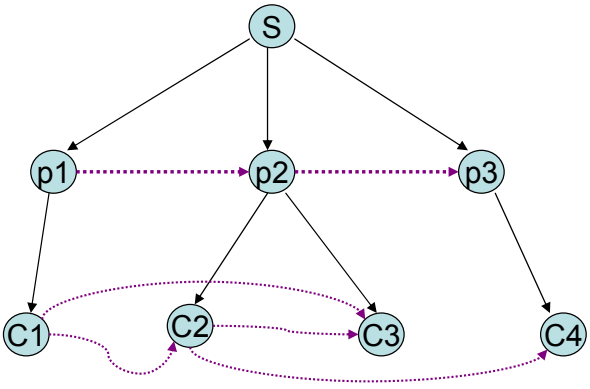
Index

Software Architecture Views



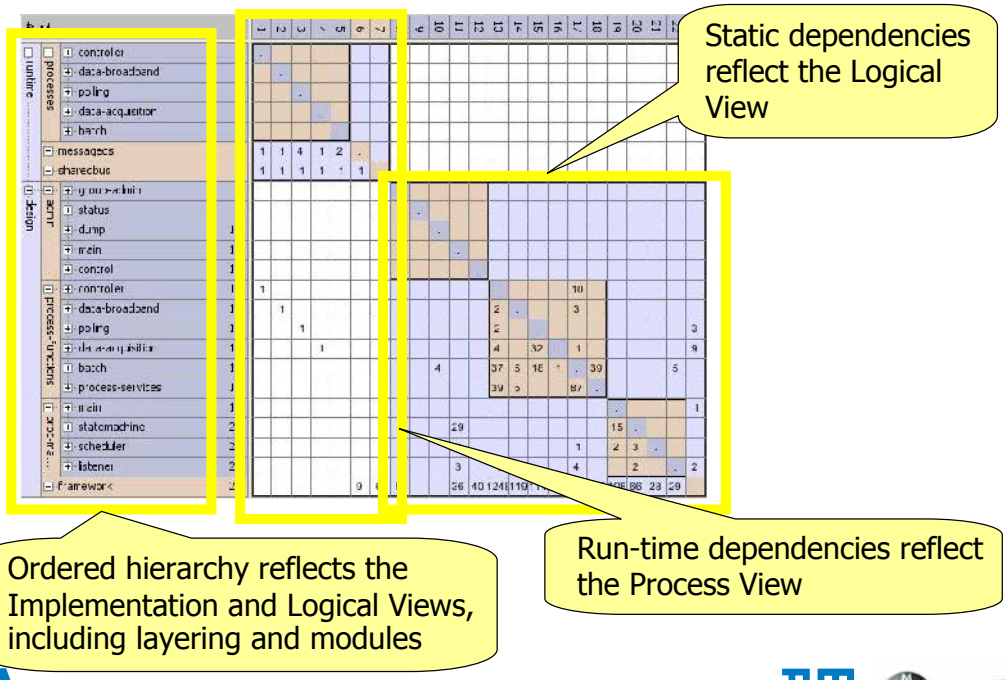
Kruchten, Philippe, "Architectural Blueprints—The "4+1" View Model of Software Architecture," IEEE Software 12(6), Nov 1995

Combining Implementation and Logical Views

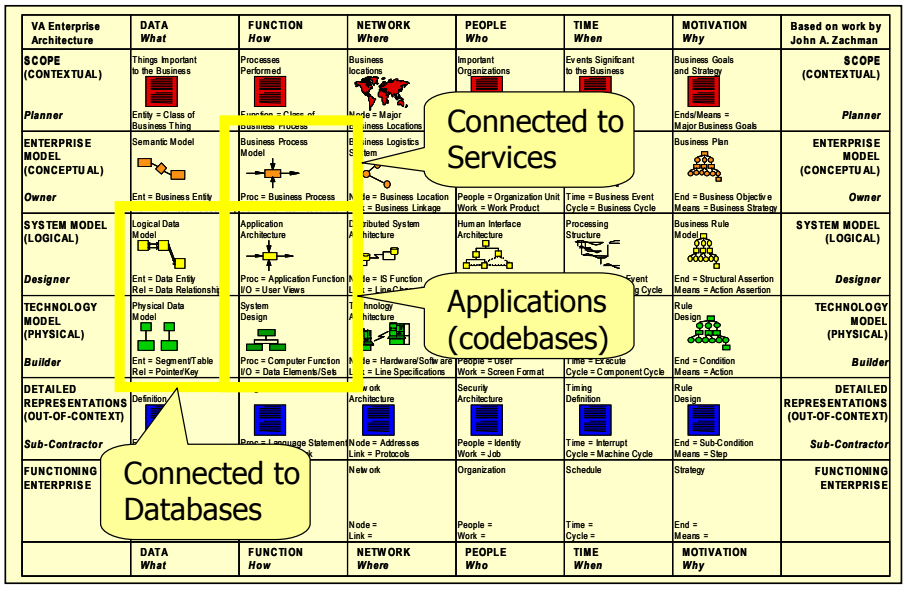


Combines two Views
- Implementation View (Solid Arrows)
- Logical View (Dotted Arrows)

Combining Logical, Implementation, & Process Views



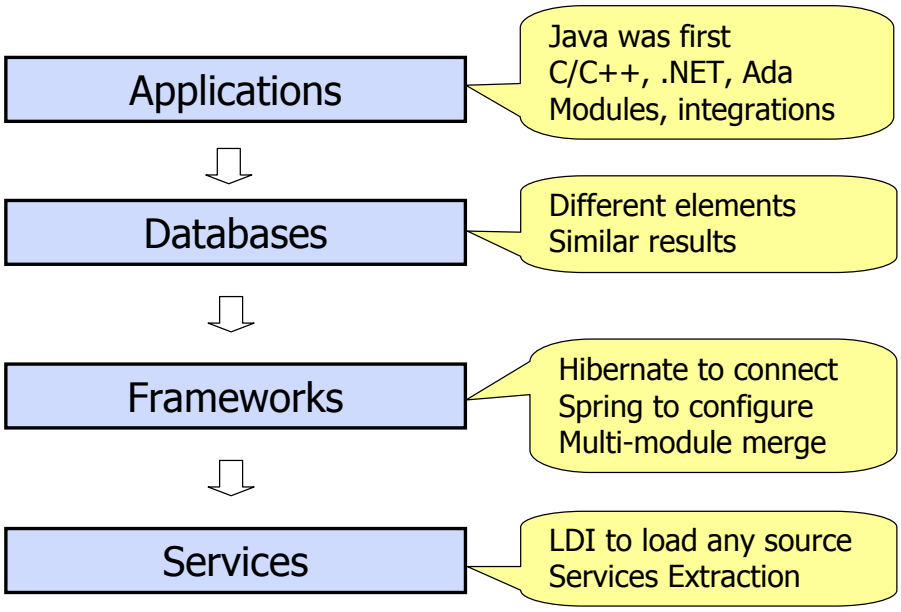
Enterprise Architecture



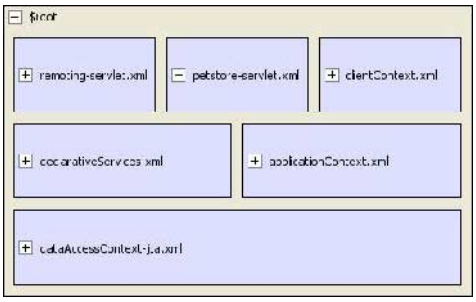
VA Enterprise Architecture	DATA *What*	FUNCTION *How*	NETWORK *Where*	PEOPLE *Who*	TIME *When*	MOTIVATION *Why*	Based on work by John A. Zachman
SCOPE (CONTEXTUAL) *Planner*	Things Important to the Business Entity = Class of Business Thing	Processes Performed	Business locations Node = Major	Important Organizations	Events Significant to the Business	Business Goals and Strategy Ends/Means = Major Business Goals	SCOPE (CONTEXTUAL) *Planner*
ENTERPRISE MODEL (CONCEPTUAL) *Owner*	Semantic Model Ent = Business Entity	Business Process Model Proc = Business Process	Business Logistics = Business Location	People = Organization Unit Work = Work Product	Time = Business Event Cycle = Business Cycle	Business Plan End = Business Objective Means = Business Strategy	ENTERPRISE MODEL (CONCEPTUAL) *Owner*
SYSTEM MODEL (LOGICAL) *Designer*	Logical Data Model Ent = Data Entity Rel = Data Relationship	Application Architecture Proc = Application Function I/O = User Views	Distributed System Architecture = IS Function	Human Interface Architecture	Processing Structure	Business Rule Model End = Structural Assertion Means = Action Assertion	SYSTEM MODEL (LOGICAL) *Designer*
TECHNOLOGY MODEL (PHYSICAL) *Builder*	Physical Data Model Ent = Segment/Table Rel = Pointer/Key	System Design Proc = Computer Function I/O = Data Elements/Sets	Technology Architecture = Hardware/Software = Line Specifications	People = User Work = Screen Format	Time = Execute Cycle = Component Cycle	Rule Design End = Condition Means = Action	TECHNOLOGY MODEL (PHYSICAL) *Builder*
DETAILED REPRESENTATIONS (OUT-OF-CONTEXT) *Sub-Contractor*	Definition	Language Statement	Network Architecture Node = Addresses Link = Protocols	Security Architecture People = Identity Work = Job	Timing Definition Time = Interrupt Cycle = Machine Cycle	Rule Design End = Sub-Condition Means = Step	DETAILED REPRESENTATIONS (OUT-OF-CONTEXT) *Sub-Contractor*
FUNCTIONING ENTERPRISE			Network Node = Link =	Organization People = Work =	Schedule Time = Cycle =	Strategy End = Means =	FUNCTIONING ENTERPRISE
	DATA *What*	FUNCTION *How*	NETWORK *Where*	PEOPLE *Who*	TIME *When*	MOTIVATION *Why*	

Zachman Framework

Evolution of Multi-Domain DSM



Example: PetStore (Enterprise Application)

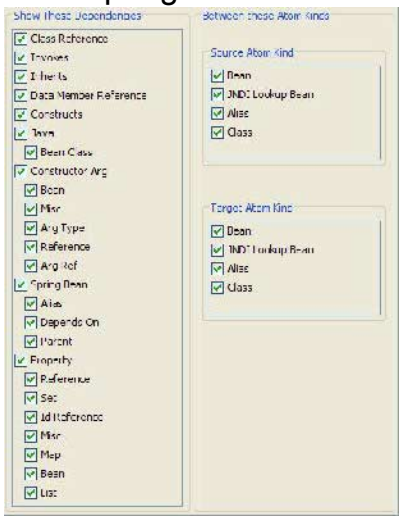


$root	1	2	3	4	5	6
clientContext.xml 1						
petstore-servlet.xml 2						
remoting-servlet.xml 3						
applicationContext.xml 4		14	3			
declarativeServices.xml 5						
dataAccessContext-jta.xml 6				5	1	

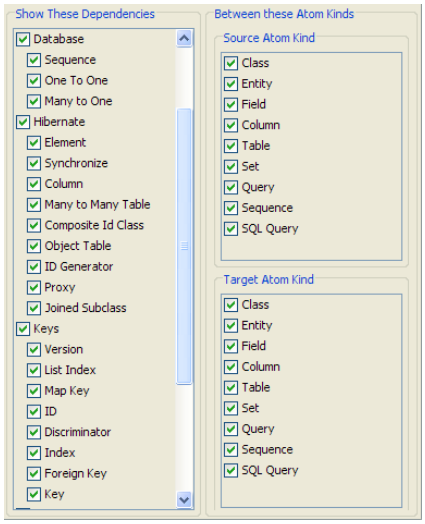
- Uses the Spring Framework to configure the application
- The application logic is written in Java which should conform to the architecture
- The system uses an Oracle database
- The Hibernate Framework is used to map the Java objects to the data objects which they use

Example: PetStore (Enterprise Application)

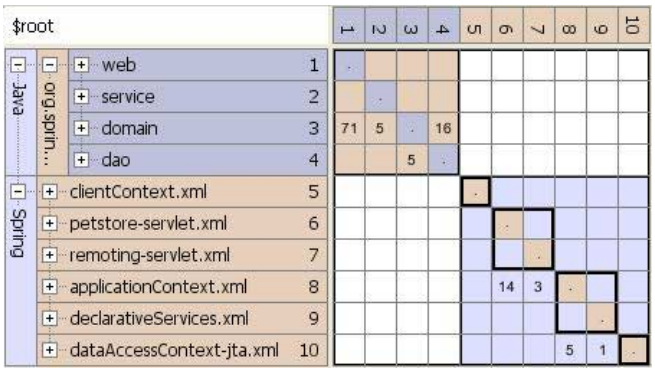
Spring Module



Hibernate Module

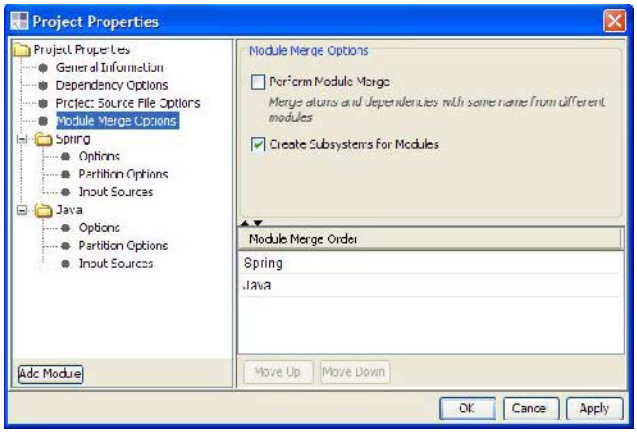


Example: PetStore (Enterprise Application)



- Each module can be initially loaded as its own subsystem
- The application codebase only loosely maps to the framework

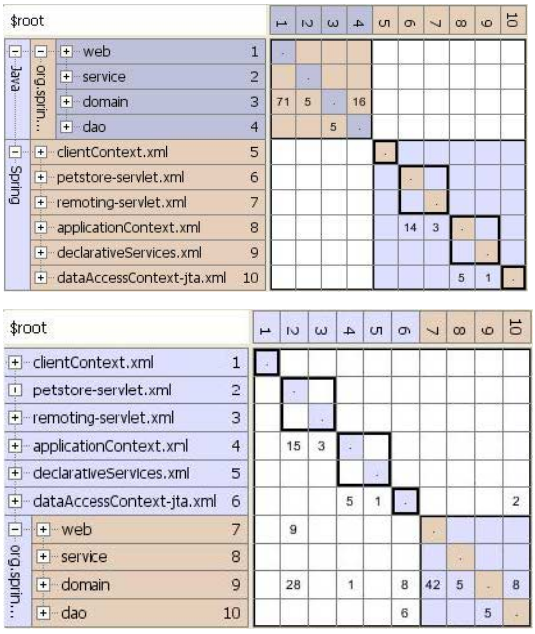
Example: PetStore (Enterprise Application)



- Merging combines elements in both modules
- In this example, the Spring classes are merged with the Java classes

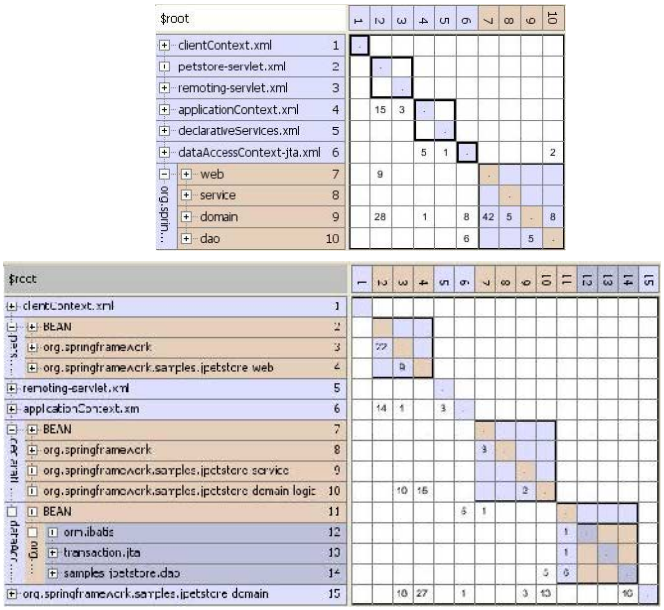
IN COOPERATION WITH BMW GROUP
CAPITALIZE ON COMPLEXITY
LATTIX

Example: PetStore (Enterprise Application)



- Comparing the DSM before and after merge, the interdependencies between the modules are now understood
- The classes which are not in the right place in the Logical structure are quickly identified

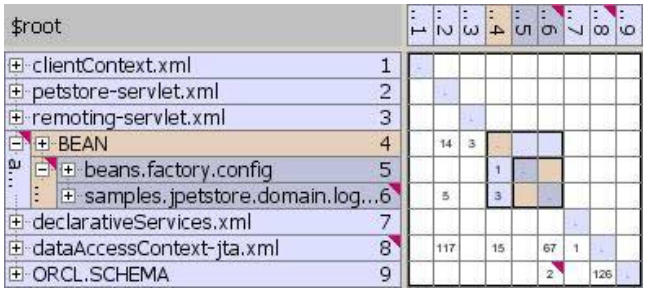
Example: PetStore (Enterprise Application)



- Decouple by moving Java classes from one partition to another
- Use the Spring modularization to move the Java classes to the appropriate subsystem
- The architecture is now expressed across both domains

IN COOPERATION WITH BMW GROUP
CAPITALIZE ON COMPLEXITY
LATTIX

Example: PetStore (Enterprise Application)



- Hibernate module is added to the merge sequence to show the dependencies of the code on the database
- Easy to identify the violation of improper access to the database
- Impact analysis can now be performed across all domains

Summary

1. Enterprise Architectures consist of many subsystems in different domains which are interdependent
2. Dependencies in constituent domains can be extracted from actual implementation and loaded into a DSM using modules for each domain
3. Merging the elements that are common to different modules enables a multi-domain DSM to show the interdependencies between domains
4. Once the Enterprise Architecture is expressed in one DSM, impact analysis can be performed which identifies the extent of change propagation across the constituent domains