10TH INTERNATIONAL DESIGN STRUCTURE MATRIX CONFERENCE, DSM'08
11 – 12 NOVEMBER 2008, STOCKHOLM, SWEDEN

UNITING SYSTEMS MODELING APPROACHES WITH DSMS

Frank Waldman and Neeraj Sangal
Lattix Inc.

Keywords: DSM, UML, SysML, Architecture, Validation

1 INTRODUCTION

The application of DSM in software development has been focused on visualization and analysis of code bases associated with complex software applications [1,2,3]. 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 is then 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.

We undertook a similar approach to apply DSMs to models that are often created prior to the actual implementation, or generated iteratively during implementation. The benefit of this approach is that it helps identify architectural problems at an early stage. Unexpectedly, it also provides a number of other even more significant benefits – the inter-relationship between various diagrams can now be seen in a single DSM; and the approach was easily extended to system models.

The de-facto standard for modeling software utilizes the Unified Modeling Language (UML). Associated with the UML standard is the XML Metadata Interchange (XMI) standard that is used to storing and exchanging UML models between various tools. Most UML tools support the export of model information into XMI. By creating a module that understands and is able to load in UML models that were exported to XMI, we were able to generate DSMs from these models. As a result, we were able to create DSMs that reflected the inter-relationships between various model diagrams.

Software is often just part of a large complex system which includes hardware, information, personnel, processes and facilities. In order to model these complex systems, architects have utilized a variety of systems modeling approaches. SysML is a recently standardized approach that is a derivative of UML which contains many of the same types of model diagrams. In this paper, we will show how those diagrams can represented in DSMs, which we believe is the first time that DSMs have been applied to UML and SysML models.

2 APPLYING DSM TO SYSTEMS MODELING

The OMG systems modeling language (OMG SysML™) is a general-purpose graphical modeling language for specifying, analyzing, designing, and verifying complex systems. In particular, the language provides graphical representations with a semantic foundation for modeling system requirements, behaviour, structure, and parametrics, which is used to integrate with other engineering analysis models. SysML represents a subset of UML 2 with extensions needed to satisfy the requirements of the UML™ for Systems Engineering RFP.

SysML contains many different types of model diagrams, including use cases, sequence diagrams and package diagrams. In order to gain the full understanding of the overall model, it is necessary to visualize each diagram and combine the different perspectives. For example, in Figure 1 is a sample model provided by Telelogic for their Rhapsody System Designer tool. On the left is the entire model view, which includes sequence diagrams, use cases, and packages for each of the system parts, each of which contains object model diagrams, sequence diagrams, and use cases. It is very difficult to get the big picture view of the system model and understand how these diagrams interrelate.

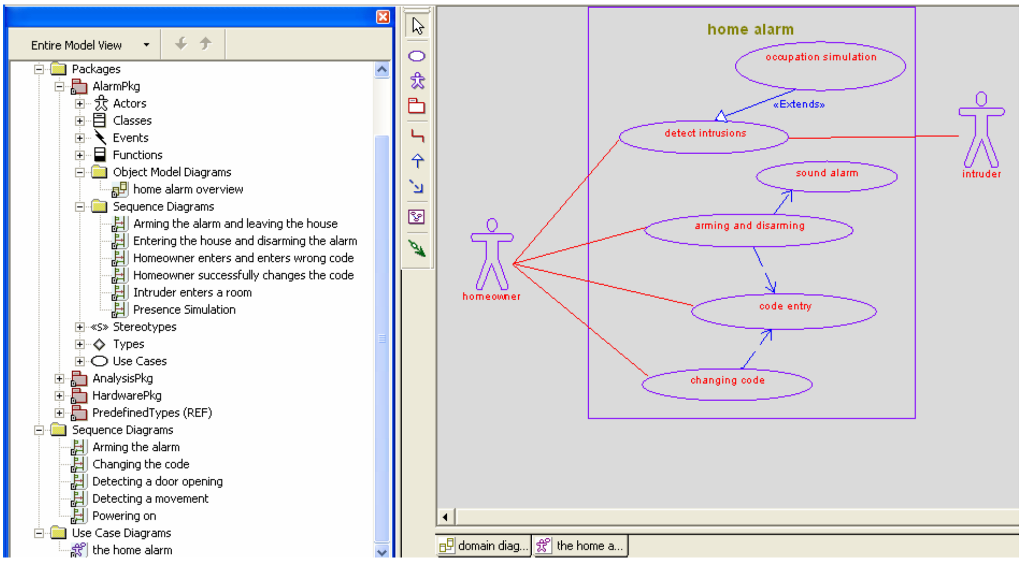
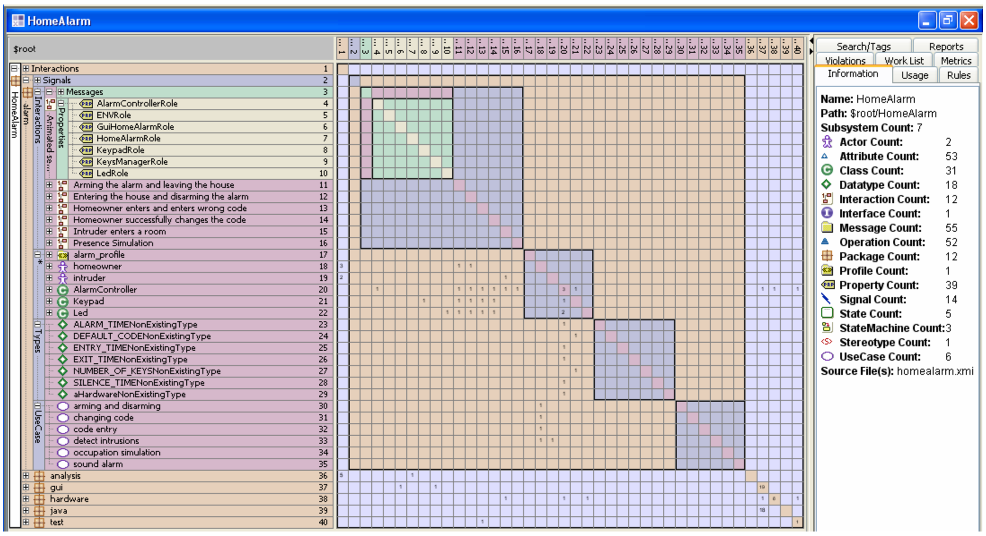


Figure 1. Example of a SysML model and a use case (Home Alarm)

Utilizing the XMI data interchange capability, it is possible to create a DSM model in which the various kinds of model elements of SysML are represented by elements in the DSM. In Figure 2, the DSM includes all of the various diagrams from the Home Alarm SysML model of Figure 1.



Interactions	1
Signals	2
Messages	3
AlarmControllerRole	4
ENVRole	5
GuiHomeAlarmRole	6
HomeAlarmRole	7
KeypadRole	8
KeysManagerRole	9
LedRole	10
Arming the alarm and leaving the house	11
Entering the house and disarming the alarm	12
Homeowner enters and enters wrong code	13
Homeowner successfully changes the code	14
Intruder enters a room	15
Presence Simulation	16
alarm_profile	17
homeowner	18
intruder	19
AlarmController	20
Keypad	21
Led	22
ALARM_TIMENonExistingType	23
DEFAULT_CODENonExistingType	24
ENTRY_TIMENonExistingType	25
EXIT_TIMENonExistingType	26
NUMBER_OF_KEYSNonExistingType	27
SILENCE_TIMENonExistingType	28
aHardwareNonExistingType	29
arming and disarming	30
changing code	31
code entry	32
detect intrusions	33
occupation simulation	34
sound alarm	35
analysis	36
gui	37
hardware	38
java	39
test	40

Name: HomeAlarm
Path: $root/HomeAlarm
Subsystem Count: 7

Actor Count:	2
Attribute Count:	53
Class Count:	31
Datatype Count:	18
Interaction Count:	12
Interface Count:	1
Message Count:	55
Operation Count:	52
Package Count:	12
Profile Count:	1
Property Count:	39
Signal Count:	14
State Count:	5
StateMachine Count:	3
Stereotype Count:	1
UseCase Count:	6

Source File(s): homealarm.xmi

Figure 2. DSM of the Home Alarm SysML model

In this DSM, the static view of the system (such as package and class diagrams) and the behavioral views (such as state and interaction diagrams) have been combined together in order to understand the overall picture. Because these diagrams contain requirements, use case scenarios and design information, the DSM can now be used to unite these different perspectives into a single view so that the impact of change to a requirement can be more readily understood.

3 CASE STUDY

The first case of applying DSM to a SysML model was in a US Government initiative to develop a next generation satellite communications system (HBHT) as an open modular system to be shared by all military services and supplied by multiple vendors (plug and play). A Government Reference Architecture (GRA) is now under development with the participation of industry. Together they plan to develop the GRA Functional Decomposition, validate the GRA through a prototype, and manage to that architecture and standard in development. Their goal is that full documentation shall be developed for all open hardware and software interfaces, sufficient to develop applications with minimum interface or interaction with the module developer.

The SysML model of the GRA has been created as result of the specification by the working groups. After exporting the model in XMI format, a DSM has been created which shows the layered view from the Systems of Systems Architecture and the interrelationships between the hardware and software modules. The DSM is being used by the government contractors and vendors to validate the model and resolve dependencies which violate the desired modularity. The dependency ruleset from the GRA model will then be used to validate each version of software as it is developed for the GRA prototype.

4 CONCLUSION

Through the use of XMI, it is now possible to easily construct a DSM that unites the various perspectives of UML and SysML systems modeling approaches into a scalable big picture view. During the earliest stages of design, it is possible to identify undesirable interdependencies and expose key design decisions for the entire team. Rules can be established in the DSM model for compliance to guide the system developers during development and testing for validation. Over time, the DSM will provide a blue print of the system architecture and how it evolves to meet changing needs.

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] Waldman F and Sangal N. Applying DSM to Enterprise Architectures. 9th International DSM Conference, Munich Germany, October 2007.

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

10TH INTERNATIONAL DSM CONFERENCE

Uniting Systems Modeling Approaches with DSMs

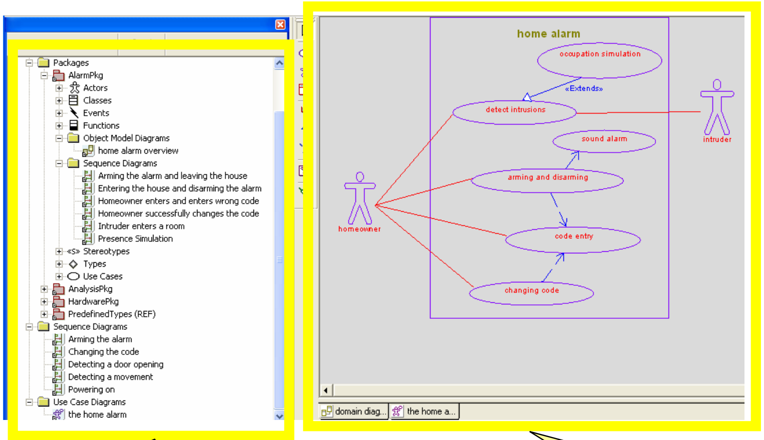
Frank Waldman and Neeraj Sangal

Lattix Inc.
USA

Index

- Introduction
- Systems Modeling with UML and SysML
- Applying DSM to Systems Modeling
- Case Study: SatCom Government Reference Architecture (GRA)
- Summary

Systems Modeling with UML/SysML

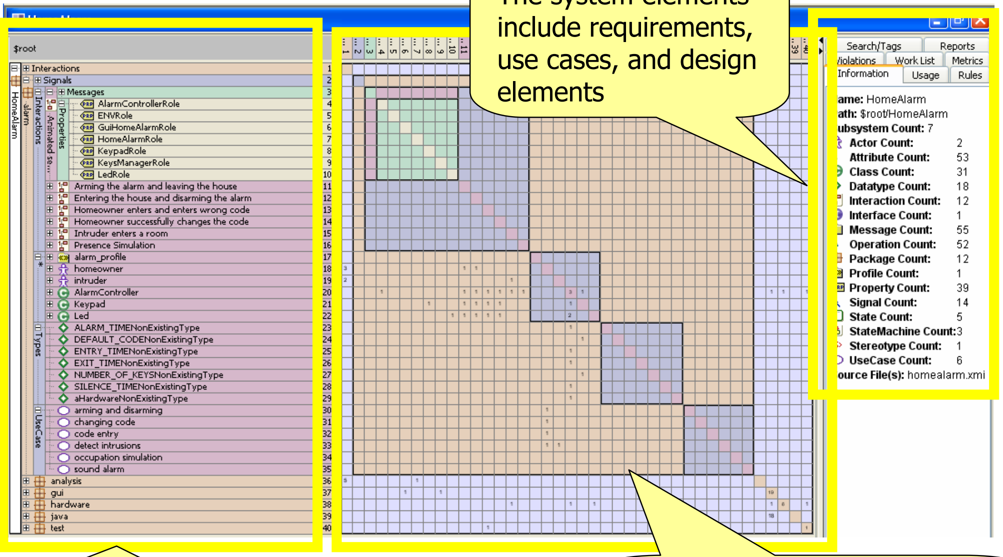


- Packages
  - AlarmPkg
    - Actors
    - Classes
    - Events
    - Functions
    - Object Model Diagrams
      - home alarm overview
    - Sequence Diagrams
      - Arming the alarm and leaving the house
      - Entering the house and disarming the alarm
      - Homeowner enters and enters wrong code
      - Homeowner successfully changes the code
      - Intruder enters a room
      - Presence Simulation
    - «S» Stereotypes
    - Types
    - Use Cases
  - AnalysisPkg
  - HardwarePkg
  - PredefinedTypes (REF)
- Sequence Diagrams
  - Arming the alarm
  - Changing the code
  - Detecting a door opening
  - Detecting a movement
  - Powering on
- Use Case Diagrams
  - the home alarm

The system model includes behavioral (state,interaction,use) and static (package,class) diagrams

Each diagram is view separately but is linked together by dependencies

Applying DSM to Systems Models through XMI



$root

- HomeAlarm
  - Interactions 1
  - alarm
    - Signals 2
    - Interactions
      - Animated se...
        - Messages 3
        - Properties
          - AlarmControllerRole 4
          - ENVRole 5
          - GuiHomeAlarmRole 6
          - HomeAlarmRole 7
          - KeypadRole 8
          - KeysManagerRole 9
          - LedRole 10
      - Arming the alarm and leaving the house 11
      - Entering the house and disarming the alarm 12
      - Homeowner enters and enters wrong code 13
      - Homeowner successfully changes the code 14
      - Intruder enters a room 15
      - Presence Simulation 16
    - *
      - alarm_profile 17
      - homeowner 18
      - intruder 19
      - AlarmController 20
      - Keypad 21
      - Led 22
    - Types
      - ALARM_TIMENonExistingType 23
      - DEFAULT_CODENonExistingType 24
      - ENTRY_TIMENonExistingType 25
      - EXIT_TIMENonExistingType 26
      - NUMBER_OF_KEYSNonExistingType 27
      - SILENCE_TIMENonExistingType 28
      - aHardwareNonExistingType 29
    - UseCase
      - arming and disarming 30
      - changing code 31
      - code entry 32
      - detect intrusions 33
      - occupation simulation 34
      - sound alarm 35
  - analysis 36
  - gui 37
  - hardware 38
  - java 39
  - test 40

ame: HomeAlarm
ath: $root/HomeAlarm
ubsystem Count: 7

Actor Count:	2
Attribute Count:	53
Class Count:	31
Datatype Count:	18
Interaction Count:	12
Interface Count:	1
Message Count:	55
Operation Count:	52
Package Count:	12
Profile Count:	1
Property Count:	39
Signal Count:	14
State Count:	5
StateMachine Count:	3
Stereotype Count:	1
UseCase Count:	6

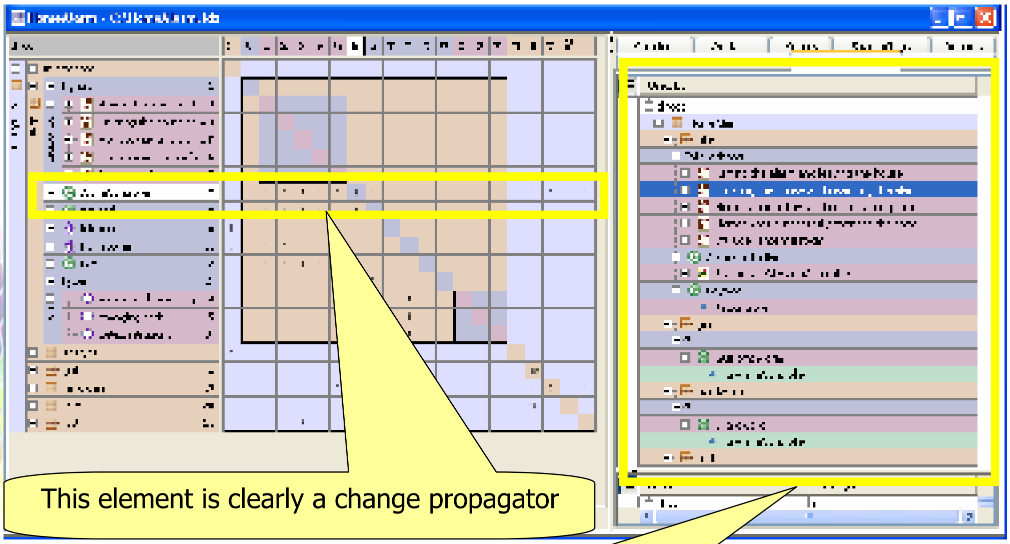
ource File(s): homealarm.xmi

The system elements include requirements, use cases, and design elements

The system elements from each diagram and their dependencies are used to create a hierarchical DSM

DSM Partitioning provides a layered system view and reveals couplings between elements

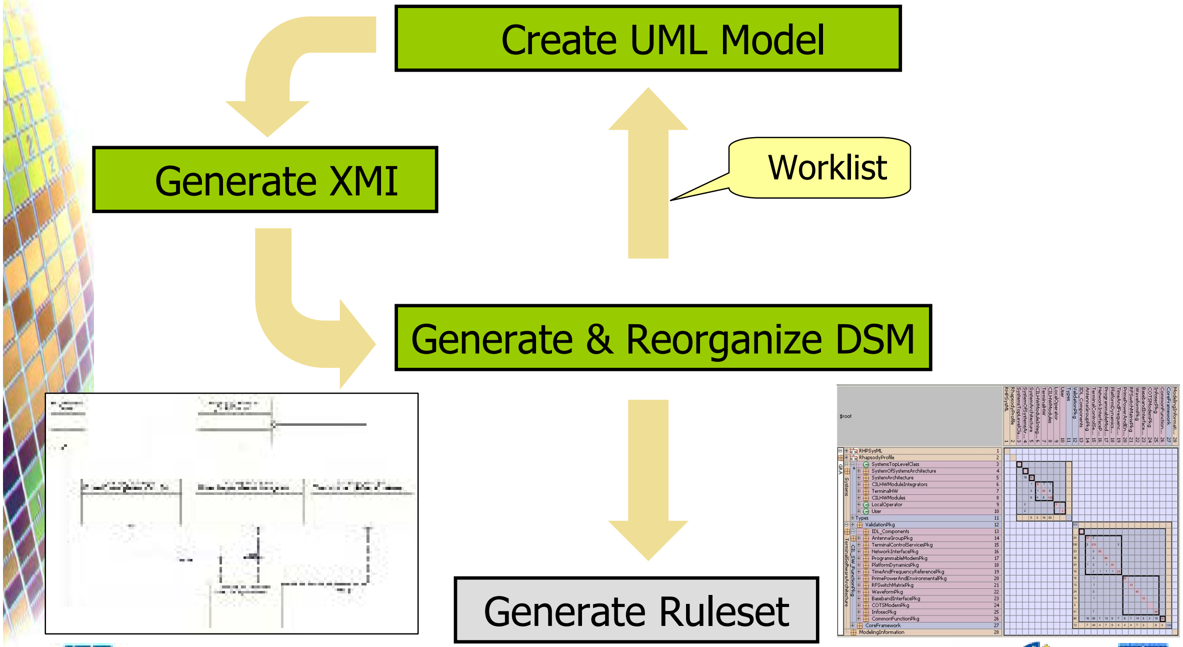
Impact of Change Across the Model



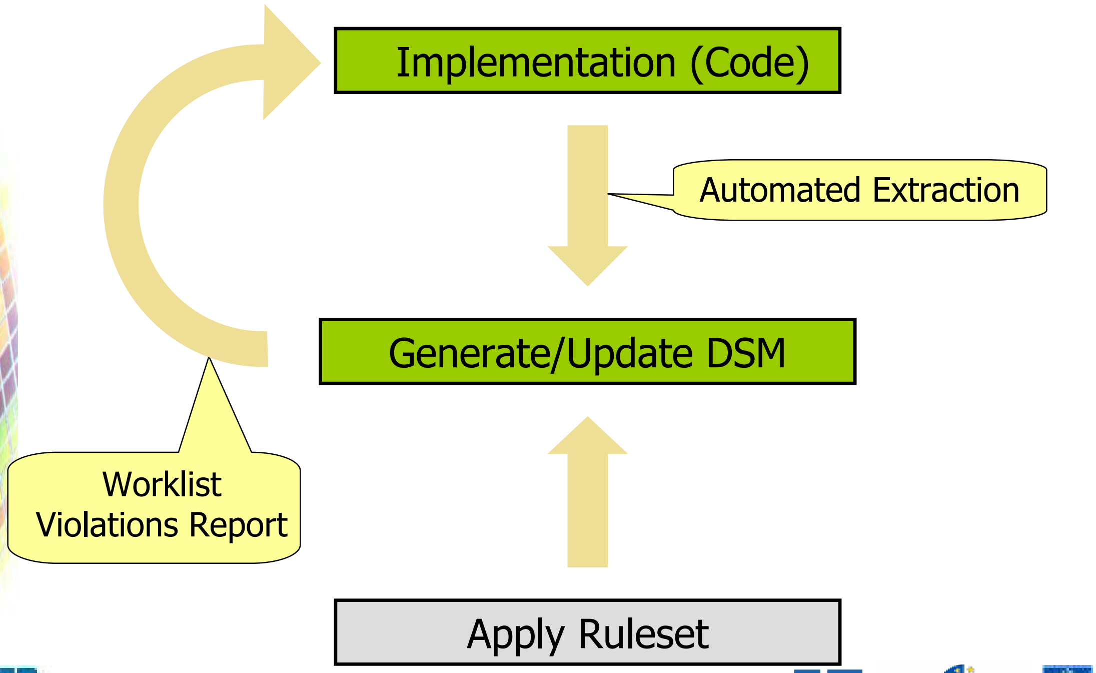
This element is clearly a change propagator

as it factors heavily in the Interactions and is used by the other modules

How It Works



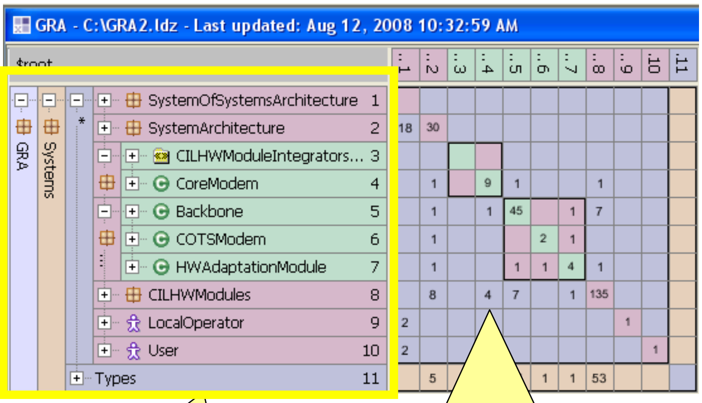
How It Works



Case Study: SatCom GRA

1. US Government wants to develop the next generation satellite communications system (HBHT) as an open modular system to be shared by all services and supplied by multiple vendors (plug and play)
2. Develop a Government Reference Architecture (GRA) based on existing systems, engaging industry by:
   - GRA Functional Decomposition
   - validating the GRA through a prototype
   - managing to that architecture and standard in development
3. “Full documentation shall be developed for all open hardware and software interfaces, sufficient to develop applications with minimum interface or interaction with the module developer”

Case Study: SatCom GRA



GRA - C:\GRA2.ldz - Last updated: Aug 12, 2008 10:32:59 AM

$root

GRA

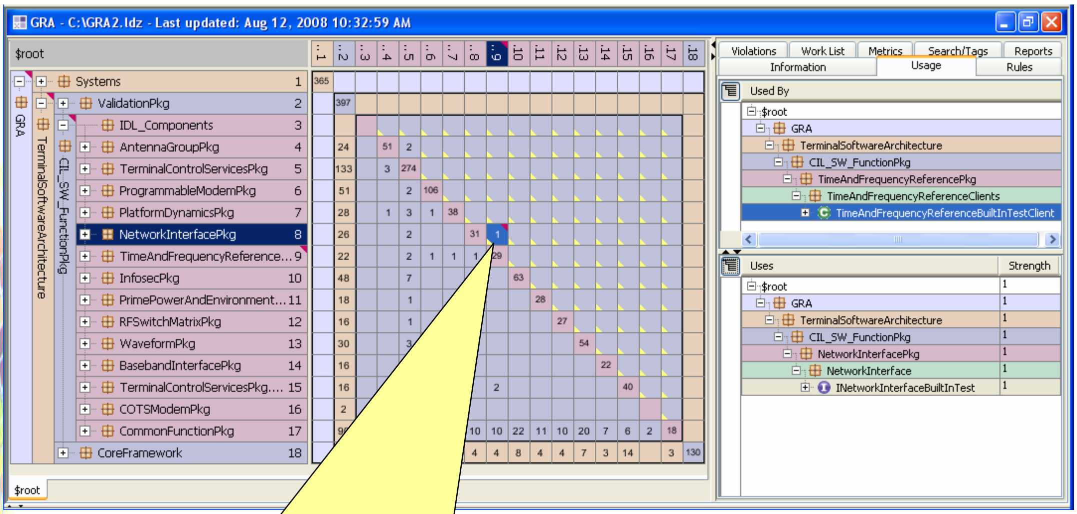
Systems

		.1	.2	.3	.4	.5	.6	.7	.8	.9	.10	.11
SystemOfSystemsArchitecture	1											
SystemArchitecture	2	18	30									
CILHWModuleIntegrators…	3											
CoreModem	4		1		9	1			1			
Backbone	5		1		1	45		1	7			
COTSModem	6		1				2	1				
HWAdaptationModule	7		1			1	1	4	1			
CILHWModules	8		8		4	7		1	135			
LocalOperator	9	2								1		
User	10	2									1	
Types	11		5	[illegible]	[illegible]	[illegible]	1	1	53			

The system model is layered from the S-of-S view down to the modules

The hardware dependencies are mostly symmetrical, but not always

Case Study: SatCom GRA



GRA - C:\GRA2.ldz - Last updated: Aug 12, 2008 10:32:59 AM

$root

GRA

TerminalSoftwareArchitecture

CIL_SW_FunctionPkg

		.1	.2	.3	.4	.5	.6	.7	.8	.9	.10	.11	.12	.13	.14	.15	.16	.17	.18
Systems	1	365																	
ValidationPkg	2		397																
IDL_Components	3																		
AntennaGroupPkg	4		24		51	2													
TerminalControlServicesPkg	5		133		3	274													
ProgrammableModemPkg	6		51			2	106												
PlatformDynamicsPkg	7		28		1	3	1	38											
NetworkInterfacePkg	8		26			2			31	1									
TimeAndFrequencyReference…	9		22			2	1	1	1	29									
InfosecPkg	10		48			7					63								
PrimePowerAndEnvironment…	11		18			1						28							
RFSwitchMatrixPkg	12		16			1							27						
WaveformPkg	13		30			3								54					
BasebandInterfacePkg	14		16												22				
TerminalControlServicesPkg.…	15		16							2						40			
COTSModemPkg	16		2																
CommonFunctionPkg	17		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	10	10	22	11	10	20	7	6	2	18	
CoreFramework	18	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	4	4	8	4	4	7	3	14		3	130

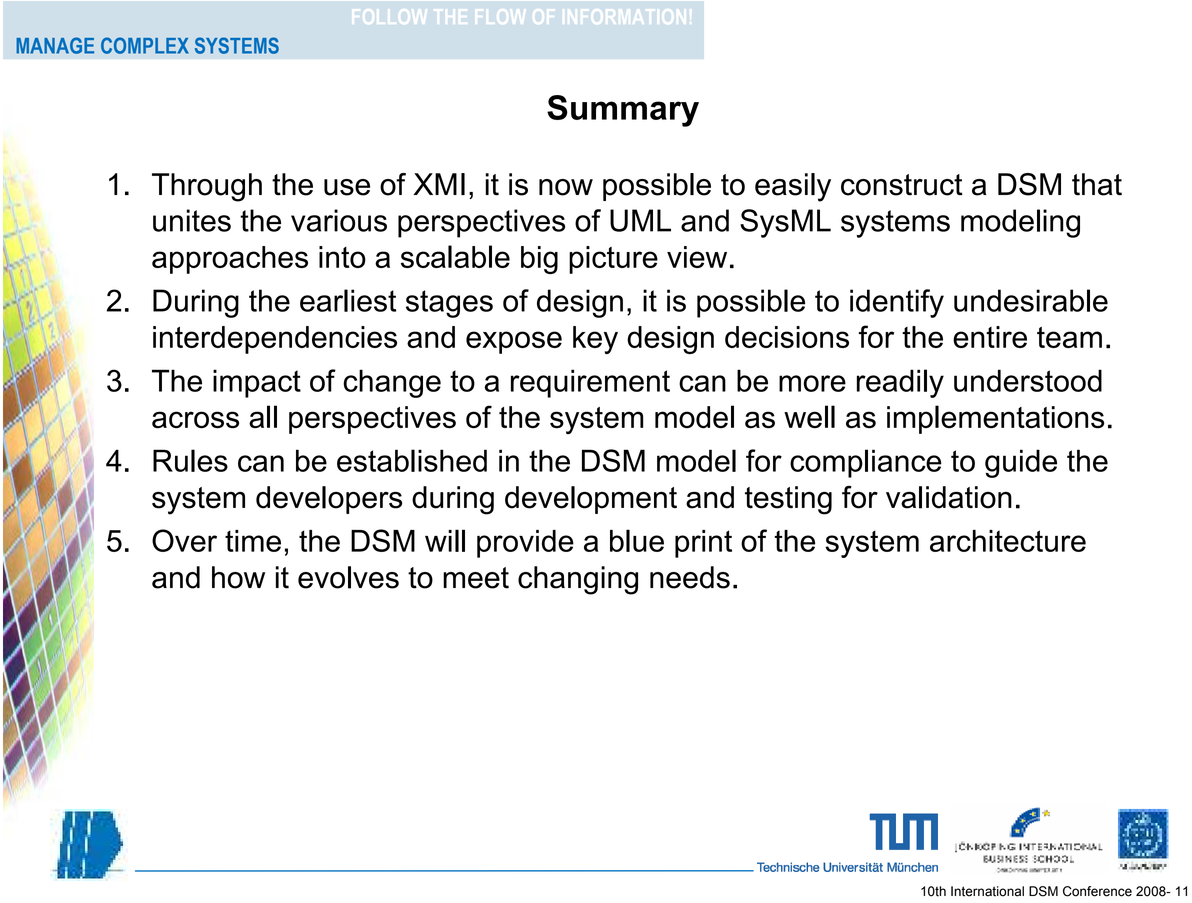
$root

Used By

- $root
  - GRA
    - TerminalSoftwareArchitecture
      - CIL_SW_FunctionPkg
        - TimeAndFrequencyReferencePkg
          - TimeAndFrequencyReferenceClients
            - TimeAndFrequencyReferenceBuiltInTestClient

Uses	Strength
$root	1
GRA	1
TerminalSoftwareArchitecture	1
CIL_SW_FunctionPkg	1
NetworkInterfacePkg	1
NetworkInterface	1
INetworkInterfaceBuiltInTest	1

Validation is achieved by testing each new version of the GRA model against the rules



Summary

1. Through the use of XMI, it is now possible to easily construct a DSM that unites the various perspectives of UML and SysML systems modeling approaches into a scalable big picture view.
2. During the earliest stages of design, it is possible to identify undesirable interdependencies and expose key design decisions for the entire team.
3. The impact of change to a requirement can be more readily understood across all perspectives of the system model as well as implementations.
4. Rules can be established in the DSM model for compliance to guide the system developers during development and testing for validation.
5. Over time, the DSM will provide a blue print of the system architecture and how it evolves to meet changing needs.