National Defense Industrial Association
9th Annual Systems Engineering Conference
San Diego, CA
October 23, 2006
Caveat

• These materials have been modified slightly from the original Tutorial given at INCOSE 2006
  – Softcopy of Full Tutorial available at:
• This material is based on version 1.0 of the SysML specification (ad-06-03-01)
  – Adopted by OMG in May ’06
  – Going through finalization process
• OMG SysML Website
  – http://www.omg.sysml.org/
Objectives & Intended Audience

At the end of this tutorial, you should understand the:
• Benefits of model driven approaches to systems engineering
• Types of SysML diagrams and their basic constructs
• Cross-cutting principles for relating elements across diagrams
• Relationship between SysML and other Standards
• Introduction to principles of a OO System Engineering Method

This course is **not intended to make you a systems modeler!**
You must use the language.

**Intended Audience:**
• Practicing Systems Engineers interested in system modeling
  – Already familiar with system modeling & tools, or
  – Want to learn about systems modeling
• Software Engineers who want to express systems concepts
• Familiarity with UML is not required, but it will help
Topics

• Motivation & Background
• Diagram Overview
• SysML Modeling as Part of SE Process
• OOSEM – Enhanced Security System Example
• SysML in a Standards Framework
• Transitioning to SysML
• Summary
Background
System Modeling

Integrated System Model Must Address Multiple Aspects of a System
Model Based Systems Engineering

Benefits

• Improved communications
• Assists in managing complex system development
  – Separation of concerns
  – Hierarchical modeling
  – Facilitates impact analysis of requirements and design changes
  – Supports incremental development & evolutionary acquisition
• Improved design quality
  – Reduced errors and ambiguity
  – More complete representation
• Early and on-going verification & validation to reduce risk
• Other life cycle support (e.g., training)
• Enhanced knowledge capture
Modeling at Multiple Levels of the System
What is SysML?

• A graphical modelling language in response to the UML for Systems Engineering RFP developed by the OMG, INCOSE, and AP233
  – a UML Profile that represents a subset of UML 2 with extensions

• Supports the specification, analysis, design, verification, and validation of systems that include hardware, software, data, personnel, procedures, and facilities

• Supports model and data interchange via XMI and the evolving AP233 standard (in-process)
What is SysML (cont.)

• *Is* a visual modeling language that provides
  – Semantics = meaning
  – Notation = representation of meaning

• *Is not* a methodology or a tool
  – SysML is methodology and tool independent
UML/SysML Status

• **UML V2.0**
  – Updated version of UML that offers significant capability for systems engineering over previous versions
  – Finalized in 2005 (formal/05-07-04)

• **UML for Systems Engineering (SE) RFP**
  – Established the requirements for a system modeling language
  – Issued by the OMG in March 2003

• **SysML**
  – Industry Response to the UML for SE RFP
  – Addresses most of the requirements in the RFP
  – Version 1.0 adopted by OMG in May ’06 / In finalization
  – Being implemented by multiple tool vendors
Diagram Overview
Relationship Between SysML and UML

UML 2

SysML

UML reused by SysML (UML4SysML)

SysML extensions to UML (SysML Profile)

UML not required by SysML (UML - UML4SysML)
SysML Diagram Taxonomy

- Behavior Diagram
- Requirement Diagram
- Structure Diagram
  - Activity Diagram
  - Sequence Diagram
  - State Machine Diagram
  - Use Case Diagram
  - Block Definition Diagram
  - Internal Block Diagram
  - Package Diagram
  - Parametric Diagram

- Same as UML 2
- Modified from UML 2
- New diagram type
4 Pillars of SysML – ABS Example

1. Structure

```
<table>
<thead>
<tr>
<th>bdd [package] VehicleStructure [ABS-Block Definition Diagram]</th>
</tr>
</thead>
<tbody>
<tr>
<td>«block» Library:: Electronic Processor</td>
</tr>
<tr>
<td>«block» Anti-Lock Controller</td>
</tr>
<tr>
<td>«block» Traction Detector</td>
</tr>
<tr>
<td>ibd [block] Anti-LockController [Internal Block Diagram]</td>
</tr>
<tr>
<td>d1:Traction Detector</td>
</tr>
<tr>
<td>c1:modulator interface</td>
</tr>
<tr>
<td>m1:Brake Modulator</td>
</tr>
</tbody>
</table>
```

2. Behavior

```
<table>
<thead>
<tr>
<th>sd ABS_ActivationSequence [Sequence Diagram]</th>
</tr>
</thead>
<tbody>
<tr>
<td>stm TireTraction [State Diagram]</td>
</tr>
<tr>
<td>act PreventLockup [Activity Diagram]</td>
</tr>
<tr>
<td>DetectLossOf Traction</td>
</tr>
<tr>
<td>TractionLoss</td>
</tr>
<tr>
<td>Modulate BrakingForce</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>interaction</td>
</tr>
<tr>
<td>state machine</td>
</tr>
<tr>
<td>activity/function</td>
</tr>
</tbody>
</table>
```

3. Requirements

```
<table>
<thead>
<tr>
<th>req [package] VehicleSpecifications [Requirements Diagram - Braking Requirements]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vehicle System Specification</td>
</tr>
<tr>
<td>Braking Subsystem Specification</td>
</tr>
<tr>
<td>«requirement» StoppingDistance</td>
</tr>
<tr>
<td>id=&quot;102&quot; text=&quot;The vehicle shall stop from 60 mph within 150 ft on a clean dry surface.&quot;</td>
</tr>
<tr>
<td>«requirement» Anti-LockPerformance</td>
</tr>
<tr>
<td>id=&quot;337&quot; text=&quot;Braking subsystem shall prevent wheel lockup under all braking conditions.&quot;</td>
</tr>
</tbody>
</table>
```

4. Parametrics

```
<table>
<thead>
<tr>
<th>par [constraintBlock] StraightLineVehicleDynamics [Parametric Diagram]</th>
</tr>
</thead>
<tbody>
<tr>
<td>:Accelleration Equation</td>
</tr>
<tr>
<td>F = ma</td>
</tr>
<tr>
<td>:VelocityEquation</td>
</tr>
<tr>
<td>v = dv/dt</td>
</tr>
<tr>
<td>:DistanceEquation</td>
</tr>
<tr>
<td>v = dx/dt</td>
</tr>
</tbody>
</table>
```

11 July 2006

Copyright © 2006 by Object Management Group.
Cross Connecting Model Elements

1. Structure

- [Internal Block Diagram]
  - `satisfies`
    - `<requirement>` Anti-Lock Performance
  - `allocatedFrom`
    - `<activity>` Traction
  - `values`
    - DutyCycle: Percentage

- `allocatedFrom`
  - `<ObjectNode>` TractionDetector
  - `<interaction>` MinimumStoppingDistance

- `allocatedFrom`
  - `<ObjectNode>` Anti-LockController
  - `<interaction>` Anti-LockPerformance

2. Behavior

- `[Swimlane Diagram]`
  - `act` PreventLockup
  - `<allocate>` TractionDetector
  - `<allocate>` BrakeModulator
  - `<allocate>` BrakeModulator Interface

- `[Parametric Diagram]`
  - `par` StraightLineVehicleDynamics
    - `v.chassis.tire.Friction`
    - `v.brake.abs.m1.DutyCycle`
    - `v.brake.rotor.BrakingForce`
    - `v.Weight`
    - `f = (tf*bf)*(1-tl)`
    - `F = ma`

3. Requirements

- `<requirement>` StoppingDistance
  - `id`="102"
  - `text"The vehicle shall stop from 60 mph within 150 ft on a clean dry surface."

- `<requirement>` Anti-LockPerformance
  - `id`="337"
  - `text"Braking subsystem shall prevent wheel lockup under all braking conditions."

4. Parametrics
Structural Diagrams

- Activity Diagram
- Sequence Diagram
- State Machine Diagram
- Use Case Diagram

SysML Diagram

- Behavior Diagram
- Requirement Diagram
- Structure Diagram

- Block Definition Diagram
- Internal Block Diagram
- Package Diagram

- Parametric Diagram

- Same as UML 2
- Modified from UML 2
- New diagram type
Package Diagram

• Package diagram is used to organize the model
  – Groups model elements into a name space
  – Often represented in tool browser
• Model can be organized in multiple ways
  – By System hierarchy (e.g., enterprise, system, component)
  – By domain (e.g., requirements, use cases, behavior)
  – Use viewpoints to augment model organization
• Import relationship reduces need for fully qualified name (package1::class1)
Package Diagram
Organizing the Model
Package Diagram - Views

- Model is organized in one hierarchy
- Viewpoints can provide insight into the model using another principle
  - E.g., analysis view that spans multiple levels of hierarchy
  - Can specify diagram usages, constraints, and filtering rules
  - Consistent with IEEE 1471 definitions
Blocks are Basic Structural Elements

- Provides a unifying concept to describe the structure of an element or system
  - Hardware
  - Software
  - Data
  - Procedure
  - Facility
  - Person

- Multiple compartments can describe the block characteristics
  - Properties (parts, references, values)
  - Operations
  - Constraints
  - Allocations to the block (e.g. activities)
  - Requirements the block satisfies

```
«block»
BrakeModulator

allocatedFrom
«activity»Modulate
BrakingForce

values
DutyCycle: Percentage
```
Block Property Types

- Property is a structural feature of a block
  - **Part property** aka. part (typed by a block)
    - Usage of a block in the context of the enclosing block
    - Example - right-front:wheel
  - **Reference property** (typed by a block)
    - A part that is not owned by the enclosing block (not composition)
    - Example - logical interface between 2 parts
  - **Value property** (typed by value type)
    - Defines a value with units, dimensions, and probability distribution
    - Example
      - Non-distributed value: tirePressure:psi=30
      - Distributed value: «uniform» {min=28,max=32} tirePressure:psi
Using Blocks

• Based on UML Class from UML Composite Structure
  – Eliminates association classes, etc.
  – Differentiates value properties from part properties, add nested connector ends, etc.
• Block definition diagram describes the relationship among blocks (e.g., composition, association, classification)
• Internal block diagram describes the internal structure of a block in terms of its properties and connectors
• Behavior can be allocated to blocks

Blocks Used to Specify Hierarchies and Interconnection
Block Definition vs. Usage

**Block Definition Diagram**

```
bdd [package] VehicleStructure [ABS-Block Definition Diagram]
```

- «block» Library:: Electronic Processor
- «block» Anti-Lock Controller
- «block» Traction Detector
- «block» Brake Modulator
- «block» Sensor

**Internal Block Diagram**

```
ibd [block] Anti-LockController [Internal Block Diagram]
```

- c2:sensor Interface
- d1:Traction Detector
- m1:Brake Modulator Interface

**Definition**
- Block is a definition/type
- Captures properties, etc.
- Reused in multiple contexts

**Usage**
- Part is the usage in a particular context
- Typed by a block
- Also known as a role
Internal Block Diagram (ibd)
Blocks, Parts, Ports, Connectors & Flows

Internal Block Diagram Specifies Interconnection of Parts
S1 is a reference part in ibd shown in dashed outline box

**ibd** [block] Anti-LockController [Internal Block Diagram]

- c2: sensor Interface
  - s1: Sensor
- c1: modulator Interface
  - d1: Traction Detector
  - m1: Brake Modulator
SysML Port

• Specifies interaction points on blocks and parts
  – Supports integration of behavior and structure

• Port types
  – Standard (UML) Port
    • Specifies a set of operations and/or signals
    • Typed by a UML interface
  – Flow Port
    • Specifies what can flow in or out of block/part
    • Typed by a flow specification

2 Port Types Support Different Interface Concepts
Port Notation

**Standard Port**

- **provided interface**
  - (provides the operations)

**Flow Port**

- **required interface**
  - (calls the operations)

**Item flow**

- part1: 
- part2:
Parametrics

- Used to express constraints (equations) between value properties
  - Provides support for engineering analysis (e.g., performance, reliability)
- Constraint block captures equations
  - Expression language can be formal (e.g., MathML, OCL) or informal
  - Computational engine is defined by applicable analysis tool and not by SysML
- Parametric diagram represents the usage of the constraints in an analysis context
  - Binding of constraint usage to value properties of blocks (e.g., vehicle mass bound to F = m \times a)
Defining Vehicle Dynamics

Defining Reusable Equations for Parametrics
Vehicle Dynamics Analysis

Using the Equations in a Parametric Diagram to Constrain Value Properties
Behavioral Diagrams

SysML Diagram

Behavior Diagram

Requirement Diagram

Structure Diagram

Activity Diagram
Sequence Diagram
State Machine Diagram
Use Case Diagram
Block Definition Diagram
Internal Block Diagram
Package Diagram

Same as UML 2
Modified from UML 2
New diagram type
Activities

- Activity used to specify the flow of inputs/outputs and control, including sequence and conditions for coordinating activities
- Secondary constructs show responsibilities for the activities using swim lanes
- SysML extensions to Activities
  - Support for continuous flow modeling
  - Alignment of activities with Enhanced Functional Flow Block Diagram (EFFBD)
Activity Diagram Notation

- Join and Merge symbols not included
- Activity Parameter Nodes on frame boundary correspond to activity parameters

11 July 2006 Copyright © 2006 by Object Management Group.
Activity Diagrams
Pin vs. Object Node Notation

- Pins are kinds of Object Nodes
  - Used to specify inputs and outputs of actions
  - Typed by a block or value type
  - Object flows connect object nodes
- Object flows between pins have two diagrammatic forms
  - Pins shown with object flow between them
  - Pins elided and object node shown with flow arrows in and out

Pins must have same characteristics (name, type etc.)
Explicit Allocation of Behavior to Structure Using Swimlanes

Activity Diagram (without Swimlanes)

Activity Diagram (with Swimlanes)
SysML EFFBD Profile

EFFBD - Enhanced Functional Flow Block Diagram

Aligning SysML with Classical Systems Engineering Techniques
Distill Water Activity Diagram (Continuous Flow Modeling)

Representing Distiller Example in SysML Using Continuous Flow Modeling

11 July 2006  Copyright © 2006 by Object Management Group.
Activity Decomposition

**Definition**

`act PreventLockup [Activity Diagram]`

- `a1:DetectLossOf Traction`
- `a2:ModulateBrakingForce`

**Use**

`bdd PreventLockup [Activity Breakdown]`

- «activity» `PreventLockup`
  - «activity» `DetectLossOf Traction`
  - «activity» `ModulateBrakingForce`
Interactions

- Sequence diagrams provide representations of message based behavior
  - represent flow of control
  - describe interactions
- Sequence diagrams provide mechanisms for representing complex scenarios
  - reference sequences
  - control logic
  - lifeline decomposition
- SysML does not include timing, interaction overview, and communications diagram
Black Box Interaction (Drive)

UML 2 Sequence Diagram Scales by Supporting Control Logic and Reference Sequences
Black Box Sequence (StartVehicle)

Simple Black Box Interaction

References Lifeline Decomposition For White Box Interaction

driver:Driver
State Machines

- Typically used to represent the life cycle of a block
- Support event-based behavior (generally asynchronous)
  - Transition with trigger, guard, action
  - State with entry, exit, and do-activity
  - Can include nested sequential or concurrent states
  - Can send/receive signals to communicate between blocks during state transitions, etc.
Operational States (Drive)

Transition notation: trigger[guard]/action
Use Cases

- Provide means for describing basic functionality in terms of usages/goals of the system by actors
- Common functionality can be factored out via include and extend relationships
- Generally elaborated via other behavioral representations to describe detailed scenarios
- No change to UML
Operational Use Cases
Cross-cutting Constructs

- Allocations
- Requirements
Allocations

- Represent general relationships that map one model element to another
- Different types of allocation are:
  - Behavioral (i.e., function to component)
  - Structural (i.e., logical to physical)
  - Software to Hardware
  - ...
- Explicit allocation of activities to structure via swim lanes (i.e., activity partitions)
- Both graphical and tabular representations are specified
Different Allocation Representations
(Tabular Representation Not Shown)

Allocate Relationship

Explicit Allocation of Activity to Swim Lane

Compartment Notation

Callout Notation
SysML Allocation of SW to HW

- In UML the deployment diagram is used to deploy artifacts to nodes
- In SysML allocation on ibd and bdd is used to deploy software/data to hardware
Requirements

• The «requirement» stereotype represents a text based requirement
  – Includes id and text properties
  – Can add user defined properties such as verification method
  – Can add user defined requirements categories
    (e.g., functional, interface, performance)

• Requirements hierarchy describes requirements contained in a specification

• Requirements relationships include DeriveReq, Satisfy, Verify, Refine, Trace, Copy
Requirements Breakdown

- **Eco-Friendliness**
- **Performance**
- **Braking**
- **Fuel Economy**
- **Accelleration**

**Id = "R1.2.1"**

**text = "The vehicle shall meet Ultra-Low Emissions Vehicle standards."**

**RefinedBy**

- **«useCase» HSUVUseCases::Accelerate**

**SatisfiedBy**

- **«block» PowerSubsystem**

**VerifiedBy**

- **«testCase» MaxAcceleration**
Example of Derive/Satisfy Requirement Dependencies

Client depends on supplier (i.e., a change in supplier results in a change in client)

Arrow Direction Opposite Typical Requirements Flow-Down
Problem and Rationale can be attached to any Model Element to Capture Issues and Decisions.
SysML Modeling as Part of the SE Process
OOSEM – Enhance Security System (ESS) Example
System Development Process

Integrated Product Development (IPD) is essential to improve communications.

A Recursive V process that can be applied to multiple levels of the system hierarchy.

Stakeholder Reqts

Manage System Development

Plan
Systems Modeling Activities - OOSEM

Major SE Development Activities

- Analyze Needs
- Define System Requirements
- Define Logical Architecture
- Synthesize Physical Architecture
- Optimize & Evaluate Alternatives
- Validate & Verify System

Common Subactivities

- Mission use cases/scenarios
- System use cases/scenarios
- System use cases/scenarios
- System use cases/scenarios
- Test cases/procedures
- Test cases/procedures
- Test cases/procedures
- Test cases/procedures

Enhanced Security System Example

- The Enhanced Security System is the example for the OOSEM material
  - Problem fragments used to demonstrate principles
  - Utilizes Artisan RTS™ Tool for the SysML artifacts
Market Needs

ESS System Specification

IntruderDetection

ESS Logical Requirements

ESS Allocated Requirements

Intruder Detection Test

Entry/Exit Subsystem
Operational View Depiction
ESS Elaborated Context Diagram

**ibd [domain] Domain-To-Be**

- **«external»** : Emergency Services
  - EmergencyServicesIn
  - EmergencyServicesOut

- **«system»** : ESS
  - «perf» Power = {<100 watts}
  - «perf» Reliability
  - «phys» SiteInstallDwg
  - «store» EventLog
  - «store» SystemState
  - DetectEntry()
  - DetectExit()
  - ReportEntry()
  - ReportExit()
  - GenerateAlarm()
  - ValidateEntry()
  - InternalMonitor()
  - DetectFire()
  - DetectMedicalEmergency()
  - RequestUserID()
  - ValidateUserID()
  - SetTimer()
  - ActivateSystem()
  - ProtectPrivacy()
  - Status Update()
  - DetectFault()

- **«external»** : Property
  - Power
  - Door Input
  - Window Input

- **«external»** : Physical Environment
  - Environmental_In

- **«external»** : Emergency Services
  - Customer
    - CustomerOut
    - CustomerIn
  - Intruder
    - AlarmSignal
    - IntruderSignal

- **«external»** : Physical Environment

- **«external»** : Customer

- **«external»** : Intruder
ESS Logical Design – Example Subsystem

Diagram showing the logical design of an Entry/Exit Subsystem with inputs and outputs:

- **Door Input**
- **Window Input**
- **Entry Sensor**
- **Exit Sensor**
- **SensedEntry**: Entry Sensor feedback
- **SensedExit**: Exit Sensor feedback
- **Event Monitor**: Entry/Exit Monitor
- **Entry/Exit Alert Status**
- **Event Log**
- **Alert Status**
### ESS Allocation Table (partial)

- Allocating Logical Components to HW, SW, Data, and Procedures components

#### Logical Components

<table>
<thead>
<tr>
<th>Type</th>
<th>Entry Sensor</th>
<th>Exit Sensor</th>
<th>Perimeter Monitor</th>
<th>Entry/Exit Monitor</th>
<th>Event Monitor</th>
<th>Site Comms I/F</th>
<th>Event Log</th>
<th>Customer I/F</th>
<th>Customer Output Mgr</th>
<th>System Status</th>
<th>Fault Mgr</th>
<th>Alarm Generator</th>
<th>Alarm I/F</th>
</tr>
</thead>
<tbody>
<tr>
<td>«software»</td>
<td>Device Mgr</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SF Comm I/F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>User I/F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Event Mgr</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Site Status Mgr</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Site RDBMS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CMS RDBMS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>«data»</td>
<td>Video File</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CMS Database</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Site Database</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>«hardware»</td>
<td>Optical Sensor</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DSL Modem</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>User Console</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Video Camera</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ESS Parametric Diagram
To Support Trade-off Analysis

par [block] EnterpriseEffectivenessModel

«moe»
MissionResponseTime

«moe»
OperationalAvailability

«moe»
OperationalCost

of1 : ObjectiveFunction

{CE = \sum (w1*u(OA) + w2*u(MRT) + w3*u(OC))}

CE
MRT
OA
OC

«moe»
CostEffectiveness

Entry/Exit Test Case

sd Entry/Exit Detection Test

Description

Intruder enters through front door
Door sensor detects entry
New alert status sent to central system
Intruder leaves through lounge window
Window sensor detects exit
Changed alert status sent to central system

seq

Enter
:SensedEntry

Exit
:SensedExit

Intruder Entry: Alert Status

Intruder Exit: Alert Status
SysML in a Standards Framework
Systems Engineering Standards Framework (Partial List)

- **Process Standards**
  - EIA 632
  - ISO 15288
  - IEEE 1220
  - CMMI

- **Architecture Frameworks**
  - FEAF
  - DoDAF
  - MODAF
  - Zachman FW

- **Modeling Methods**
  - HP
  - OOSE
  - SADT
  - Other

- **Modeling & Simulation Standards**
  - IDEF0
  - SysML
  - UPDM

- **Simulation & Analysis**
  - HLA
  - MathML

- **Interchange & Metamodeling Standards**
  - MOF
  - XMI
  - STEP/AP233

- **Data Repository**

- **Implemented By Tools**
Standards-based Tool Integration with SysML
Participating SysML Tool Vendors

- Artisan
- EmbeddedPlus
  - 3rd party IBM vendor
- Sparx Systems
- Telelogic (includes I-Logix)
- Vitech

Note: Free Visio SysML Template available at OMG SysML site (http://www.omgsysml.org)
UML Profile for DoDAF/MODAF (UPDM) Standardization

• Current initiative underway to develop standard profile for representing DODAF and MODAF products
  – Requirements for profile issued Sept 05
  – Final submissions expected Dec ‘06
• Multiple vendors and users participating
• Should leverage SysML
Transitioning to SysML
Using Process Improvement To Transition to SysML
### Integrated Tool Environment

#### Project Management

<table>
<thead>
<tr>
<th>CM/DM</th>
<th>Requirements Management</th>
<th>Engineering Performance Analysis</th>
<th>System Modeling</th>
<th>Verification &amp; Validation</th>
<th>Specialty Engineering Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product Data Management</td>
<td>SoS / DoDAF / Business Process Modeling</td>
<td>Software Modeling</td>
<td>Hardware Modeling</td>
<td>SysML</td>
<td>VHDL, CAD, ..</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UML 2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Summary and Wrap up
Summary

• SysML sponsored by INCOSE/OMG with broad industry and vendor participation
• SysML provides a general purpose modeling language to support specification, analysis, design and verification of complex systems
  – Subset of UML 2 with extensions
  – 4 Pillars of SysML include modeling of requirements, behavior, structure, and parametrics
• OMG SysML Adopted in May 2006
• Multiple vendor implementations announced
• Standards based modeling approach for SE expected to improve communications, tool interoperability, and design quality
References

- OMG SysML website
  - [http://www.omgsysml.org](http://www.omgsysml.org)
- UML for Systems Engineering RFP
  - OMG doc# ad/03-03-41
- UML 2 Superstructure
  - OMG doc# formal/05-07-04
- UML 2 Infrastructure
  - OMG doc# ptc/04-10-14