OMG Systems Modeling Language
(OMG SysML™)
Tutorial

19 June 2008

Sanford Friedenthal
Alan Moore
Rick Steiner
(emails included in references at end)

Copyright © 2006-2008 by Object Management Group.
Published and used by INCOSE and affiliated societies with permission.
OMG SysML™ Specification

• Specification status
  – Adopted by OMG in May ’06
  – Available Specification v1.0 in Sept ‘07
  – Revision task force for v1.1 in July ‘07

• This tutorial is based on the OMG SysML available specification (formal/2007-09-01)

• This tutorial, the specifications, papers, and vendor info can be found on the OMG SysML Website at http://www.omgsysml.org/
Objectives & Intended Audience

At the end of this tutorial, you should have an awareness of:

• Motivation of model-based systems engineering approach
• SysML diagrams and language concepts
• How to apply SysML as part of a model based SE process
• Basic considerations for transitioning to SysML

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
• Software Engineers who want to better understand how to integrate software and system models
• Familiarity with UML is not required, but it helps
Topics

- Motivation & Background
- Diagram Overview and Language Concepts
- SysML Modeling as Part of SE Process
  - Structured Analysis – Distiller Example
  - OOSEM – Enhanced Security System Example
- SysML in a Standards Framework
- Transitioning to SysML
- Summary
Motivation & Background
SE Practices for Describing Systems

**Past**

- Specifications
- Interface requirements
- System design
- Analysis & Trade-off
- Test plans

**Future**

Moving from Document centric to Model centric
System Modeling

Requirements

Functional/Behavioral Model
- Start
- Shift
- Accelerate
- Brake

Performance Model
- Control Input
- Power Equations
- Vehicle Dynamics

System Model

Structural/Component Model
- Engine
- Transmission
- Transaxle

Other Engineering Analysis Models
- Mass Properties
  - Structural
  - Safety
  - Cost Model

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

• Shared understanding of system requirements and design
  – Validation of requirements
  – Common basis for analysis and design
  – Facilitates identification of risks
• Assists in managing complex system development
  – Separation of concerns via multiple views of integrated model
  – Supports traceability through hierarchical system models
  – Facilitates impact analysis of requirements and design changes
  – Supports incremental development & evolutionary acquisition
• Improved design quality
  – Reduced errors and ambiguity
  – More complete representation
• Supports early and on-going verification & validation to reduce risk
• Provides value through life cycle (e.g., training)
• Enhances knowledge capture
System-of-Systems

Modeling Needed to Manage System Complexity
Stakeholders Involved in System Acquisition

Customers

Developers/Integrators

Project Managers

Regulators

Testers

Vendors

Modeling Needed to Improve Communications
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 XML Metadata Interchange (XMI®) and the evolving AP233 standard (in-process)

SysML is Critical Enabler for Model Driven SE
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
  – Updated version of UML that offers significant capability for systems engineering over previous versions
  – Issued in 2005 with on-going minor revisions

• 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
  – Adopted by OMG in May ’06
Diagram Overview & Language Concepts
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 Extensions
- Blocks
- Item flows
- Value properties
- Allocations
- Requirements
- Parametrics
- Continuous flows
- …
SysML Diagram Taxonomy

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

4 Pillars of SysML – ABS Example

1. Structure

- **bdd** [Package] Structure [ABS Structure Hierarchy]
- **ibd** [Block] Anti-Lock Controller [Basic]
- **<block>>** Library: Electronic Processor
- **<block>>** Library: Anti-Lock Control
- **<block>>** Traction Detector
- **d1**: Traction Detector
- **m1**: Brake Modulator

2. Behavior

- Interaction
- State Machine
- Activity/Function

3. Requirements

- **Vehicle System Specification**
  - **<<requirement>>** Stopping Distance
    - **Id**: "10.2"
    - **Text**: "The vehicle shall stop from 60 miles per hour within 150 ft on a clean dry surface."
- **Braking Subsystem Specification**
  - **<<requirement>>** Anti-Lock Performance
    - **Id**: "33.7"
    - **Text**: "The braking system shall prevent wheel lockup under all braking conditions."

4. Parametrics

- **e1**: Braking Force Equation
  \( f = (f_b + f_f)(1:0) \)
- **e2**: Acceleration Equation
  \( a = m/s^2 \) (\( = m \cdot a \))
- **e3**: Velocity Equation
  \( a = m/s^2 \) (\( = dv/dt \))
- **e4**: Distance Equation
  \( v = dx/dt \)
SysML Diagram Frames

- Each SysML diagram represents a model element
- Each SysML Diagram must have a Diagram Frame
- Diagram context is indicated in the header:
  - Diagram kind (act, bdd, ibd, sd, etc.)
  - Model element type (package, block, activity, etc.)
  - Model element name
  - User defined diagram name or view name
- A separate diagram description block is used to indicate if the diagram is complete, or has elements elided

```
«diagram usage»

Diagram Kind [modelElementType] modelElementName [diagramName]
```

Structural Diagrams

- **SysML Diagram**
  - **Behavior Diagram**
  - **Requirement Diagram**
    - **Activity Diagram**
    - **Sequence Diagram**
    - **State Machine Diagram**
    - **Use Case 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
  – Supports model configuration management (check-in/out)

• Model can be organized in multiple ways
  – By System hierarchy (e.g., enterprise, system, component)
  – By diagram kind (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

**pkg SampleModel [by diagram type]**

- Use Cases
- Requirements
- Behavior
- Structure
- EngrAnalysis

**pkg SampleModel [by level]**

- Enterprise
- System
- Logical Design
- Physical Design
- Verification

**pkg SampleModel [by IPT]**

- Architecture Team
- Requirements Team
- IPT A
- IPT B
- IPT C
Package Diagram - Views

- Viewpoint represents the stakeholder perspective
- View conforms to a particular viewpoint
  - Imports model elements from multiple packages
  - Can represent a model query based on query criteria
- View and Viewpoint consistent with IEEE 1471 definitions
Blocks are Basic Structural Elements

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

<table>
<thead>
<tr>
<th>«block»</th>
</tr>
</thead>
<tbody>
<tr>
<td>BrakeModulator</td>
</tr>
</tbody>
</table>

| allocatedFrom             |
| «activity»Modulate        |
| BrakingForce              |

| values                    |
| DutyCycle: Percentage     |

- Multiple standard compartments can describe the block characteristics
  - Properties (parts, references, values, ports)
  - Operations
  - Constraints
  - Allocations from/to other model elements (e.g. activities)
  - Requirements the block satisfies
  - User defined compartments
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 (composite) block
    - Example - right-front:wheel
  - **Reference property** (typed by a block)
    - A part that is not owned by the enclosing block (not composition)
    - Example – aggregation of components into logical subsystem
  - **Value property** (typed by value type)
    - A quantifiable property 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
  – Supports unique features (e.g., flow ports, value properties)
• Block definition diagram describes the relationship among blocks (e.g., composition, association, specialization)
• Internal block diagram describes the internal structure of a block in terms of its properties and connectors
• Behavior can be allocated to blocks
Block Definition vs. Usage

**Block Definition Diagram**

**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

Enclosing Block
Connector
Item Flow

Port
Part

Internal Block Diagram Specifies Interconnection of Parts
Reference Property Explained

- S1 is a reference part*
- Shown in dashed outline box

*Actual name is reference property
SysML Ports

• Specifies interaction points on blocks and parts
  – Integrates behavior with structure
  – portName:TypeName

• Kinds of ports
  – Standard (UML) Port
    • Specifies a set of required or provided operations and/or signals
    • Typed by a UML interface
  – Flow Port
    • Specifies what can flow in or out of block/part
    • Typed by a block, value type, or flow specification
    • Atomic, non-atomic, and conjugate variations
Port Notation

**Standard Port**

provided interface
(provides the operations)

required interface
(calls the operations)

**Flow Port**

*item flow*

part1:

part2:

part1:

part2:
Delegation Through Ports

• Delegation can be used to preserve encapsulation of block (black box vs white box)
• Interactions at outer ports of Block1 are delegated to ports of child parts
• Ports must match (same kind, type, direction, etc.)
• Connectors can cross boundary without requiring ports at each level of nested hierarchy
Parametrics

• Used to express constraints (equations) between value properties
  – Provides support for engineering analysis
    (e.g., performance, reliability)
  – Facilitates identification of critical performance properties

• Constraint block captures equations
  – Expression language can be formal (e.g., MathML, OCL) or informal
  – Computational engine is provided by applicable analysis tool and not by SysML

• Parametric diagram represents the usage of the constraints in an analysis context
  – Binding of constraint parameters to value properties of blocks (e.g., vehicle mass bound to parameter ‘m’ in \( F = m \times a \))
Defining Vehicle Dynamics

Defining Reusable Equations for Parametrics

Vehicle Dynamics Analysis

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

• Activity specifies transformation of inputs to outputs through a controlled sequence of actions
• Secondary constructs show responsibilities for the activities using activity partitions (i.e., swim lanes)
• SysML extensions to Activities
  – Support for continuous flow modeling
  – Alignment of activities with Enhanced Functional Flow Block Diagram (EFFBD)
Activity Diagram

Activity Diagram Specifies Controlled Sequence of Actions

Input

Output
Routing Flows

**Initial Node** – On execution of parent control token placed on outgoing control flows

**Activity Final Node** – Receipt of a control token terminates parent

**Flow Final Node** – Sink for control tokens

**Fork Node** – Duplicates input (control or object) tokens from its input flow onto all outgoing flows

**Join Node** – Waits for an input (control or object) token on all input flows and then places them all on the outgoing flow

**Decision Node** – Waits for an input (control or object) token on its input flow and places it on one outgoing flow based on guards

**Merge Node** – Waits for an input (control or object) token on any input flows and then places it on the outgoing flow

*Guard expressions can be applied on all flows*
Actions Process Flow of Control and Data

- Two types of flow
  - Object / Data
  - Control
- Unit of flow is called a “token” (consumed & produced by actions)

Actions Execution Begins When Tokens Are Available on “all” Control Inputs and Required Inputs
An Action Can Invoke Another Activity

Activity is Invoked When an Action Begins to Execute
Semantics for Activity Invocation

A call behavior action can have
- 0..* control inputs & outputs
- 0 ..* optional item inputs & outputs
- 0..* required item inputs & outputs
- 0..* streaming (and continuous) item inputs & outputs

Starting an action:
- An action starts when a token is placed on all of its control inputs and all of its required inputs (must meet minimum multiplicity of its input pins) and the previous invoked activity has completed
- An action invokes an activity when it starts, and passes the tokens from its input pins to the input parameter nodes of the invoked activity

During an execution:
- An action continues to accept streaming inputs and produce streaming outputs

Terminating an action:
- An action terminates when its invoked activity reaches an activity final, or when the action receives a control disable, or as a side affect of other behaviors of the parent activity
- The tokens on the output parameter nodes of the activity are placed on the output pins of the action and a control token is placed on each of the control outputs of the action

Following action termination:
- The tokens on the output pins and control outputs of the action are moved to the input pins of the next actions when they are ready to start per above
- The action can restart and invoke the activity again when the starting conditions are satisfied per above

Note: The summary is an approximation of the semantics. The detailed semantics are specified in the UML and SysML specification.
Common Actions

Call Operation Action
(can call leaf level function)

Call Behavior Action

Accept Event Action
(Event Data Pin often elided)

Send Signal Action
(Pins often elided)
Activity Diagram Example
With Streaming Inputs and Outputs

Streaming Inputs and Outputs Continue to Be Consumed and Produced While the Action is Executing
Distill Water Activity Diagram (Continuous Flow Modeling)

Continuous flow means ΔTime between tokens approaches zero.

Actions are enabled by default when activity is enabled.

Continuous Flow

Interruptible Region

Accept Event Action Will Terminate Execution

Continuous Flow Is Representative of Many Physical Processes
Example – Operate Car

**act Operate Car**

- **Turn Key to On**
  - **:Driving**
    - Brake Pressure
      - «continuous»
    - Brake Pressure
      - «continuous»
  - **:Braking**
    - Modulation Frequency
      - «continuous»
      - «optional»
    - Modulation Frequency
      - «continuous»
  - **:Monitor Traction**
    - «optional»

- **Turn Key to Off**
  - **:Braking**
    - «continuous»
    - Braking Pressure
  - **:Driving**
    - «continuous»
    - Brake Pressure

**Enabling and Disabling Actions With Control Operators**
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):**

```
act [Activity] Prevent Lockup [ Actions ]

a1 : Detect Loss of Traction
p1 : TractLoss
a2 : Modulate Braking Force
p2 : TractLoss
of1
```

**Activity Diagram (with Swimlanes):**

```
<<allocate>>
d1 : Traction Detector
m1 : Brake Modulator
allocatedTo
<<connector>> c2 :
```
Activity Decomposition

**Definition**

**Use**
EffBD - Enhanced Functional Flow Block Diagram

2.1 Serial Function

2.2 Multi-exit Function

2.3 Function in Concurrency

2.4 Function in Multi-exit Construct

2.5 Function in an Iterate

2.6 Output Function

Item 1

Item 2

Item 3

Item 4

External Input

External Output

«effbd» act

{cc#1}

{cc#2}

[ before third time ]

[ after third time ]

«optional»

«optional»

«optional»

«optional»

«optional»
Interactions

• Sequence diagrams provide representations of message based behavior
  – represent flow of control
  – describe interactions between parts
• 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
White Box Sequence (StartVehicle)

Decomposition of Black Box Into White Box Interaction
Primary Interaction Operators

- **ref** name
  - reference to a sequence diagram fragment defined elsewhere
- **opt** [condition]
  - has 1 part that may be executed based on a condition/state value
- **alt**
  - has 2 or more parts, but only one executes based on a condition/state
  - an operand fragment labeled [else] is executed if no other condition is true
- **par**
  - has 2 or more parts that execute concurrently
    - Concurrence indicates does not require simultaneous, just that the order is undetermined. If there is only one processor the behavior could be (A then B), (B then A), or (A and B interleaving) …
- **loop** min..max [escape]
  - Has a minimum # of executions, and optional maximum # of executions, and optional escape condition
- **break** [condition]
  - Has an optional guard. If true, the contents (if any) are executed, and the remainder of the enclosing operator is not executed

Provided by Michael Chonoles
Other Interaction Operators

• **critical**
  – The sequence diagram fragment is a critical region. It is treated as atomic – no interleaving with parallel regions

• **neg**
  – The sequence diagram fragment is forbidden. Either it is impossible to occur, or it is the intent of the requirements to prevent it from occurring

• **assert**
  – The sequence diagram fragment is the only one possible (or legal)

• **seq** (weak, the default)
  **strict**
  – Strict: The message exchange occurs in the order described
  – Weak: Each lifeline may see different orders for the exchange (subject to causality)

• **consider** (list of messages)
  **ignore** (list of messages)
  – Consider: List the messages that are relevant in this sequence fragment
  – Ignored: List the messages that may arrive, but are not interesting here

Provided by Michael Chonoles
Trial Result of Vehicle Dynamics

Lifeline are value properties

Timing Diagram Not Part of SysML
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.

- Event types
  - Change event
  - Time event
  - Signal event
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
  - Use is methodology dependent
  - Often accompanied by use case descriptions
- Common functionality can be factored out via «include» and «extend» relationships
- Elaborated via other behavioral representations to describe detailed scenarios
- No change to UML
Operational Use Cases

uc HSUV_UseCases [Operational Use Cases]

HybridSUV

- Flat_Tire
- Accelerate
- Steer
- Park
- Brake

Driver

«include»
«extend»
«include»
«include»
«include»
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 Action to Part Property

Compartment Notation

Callout Notation

Read as follows
“part name has constraints that are allocated to/from an <<element type>> Element Name” OR
“part name has an <<element type>> allocated from Element Name”
“Element Name is allocated to a part called part name”
SysML Allocation of SW to HW

- In UML, the deployment diagram is used to deploy artifacts to nodes.
- In SysML, «allocation» on an ibd and bdd is used to deploy software/data to hardware.

```
ibd [node] SF Residence

«hardware» : Optical Sensor

allocatedFrom
«software» Device Mgr
«software» Event Mgr
«software» Site Config Mgr
«software» Site RDBMS
«software» Site Status Mgr
«software» User I/F
«software» User Valid Mgr

«hardware» : Site Processor

allocatedFrom
«software» SF Comm I/F

«hardware» : NW Hub

allocatedFrom
«software» Site Config Mgr

«hardware» : Site Hard Disk

allocatedFrom
«data» Site Database

«hardware» : User Console

«hardware» : User I/F

«hardware» : User Valid Mgr

allocatedFrom
«data» Video File

«hardware» : DSL Modem

allocatedFrom
«data» Video File

«hardware» : DVD-ROM Drive

allocatedFrom
«data» Site Database

«hardware» : Alarm

allocatedFrom
«data» NW Hub

«hardware» : DVD-ROM Drive

allocatedFrom
«data» Site Database

«hardware» : Alarm
```

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 DeriveReqt, Satisfy, Verify, Refine, Trace, Copy
Requirements Breakdown

**Requirement Relationships Model the Content of a Specification**

```
req [package] HSUVRequirements [HSUV Specification]
```

```
HSUVSpecification

«requirement» Eco-Friendliness
«requirement» Performance
«requirement» Braking
«requirement» FuelEconomy
«requirement» Acceleration

ReefinedBy
«useCase» HSUVUsecases:Accelerate

RefinedBy
«useCase» HSUVUseCases:Accelerate

«requirement» Power

«deriveReqt»

«requirement» Emissions
Id = "R1.2.1"
text = "The vehicle shall meet UltraLow Emissions Vehicle standards."

VerifiedBy
«testCase» MaxAcceleration

SatisfiedBy
«block» PowerSubsystem
```
Example of Derive/Satisfy Requirement Dependencies

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

Problem and Rationale can be attached to any Model Element to Capture Issues and Decisions

```
bd: Master Cylinder requirements

«requirement» Loss of Fluid

«requirement» Reservoir

«block» Brake System

m: MasterCylinder

«satisfy»

«satisfy»

«rationale»
The best-practice solution consists in assigning one reservoir per brakeline. See "automotive_d32_hdb.doc"

«problem» The master cylinder in previous version leaked.
```
Stereotypes & Model Libraries

- Mechanisms for further customizing SysML
- Profiles represent extensions to the language
  - Stereotypes extend meta-classes with properties and constraints
    - Stereotype properties capture metadata about the model element
  - Profile is applied to user model
  - Profile can also restrict the subset of the meta-model used when the profile is applied
- Model Libraries represent reusable libraries of model elements
Stereotypes

Defining the Stereotype

Applying the Stereotype
Applying a Profile and Importing a Model Library

pkg ModelingDomain [Establishing HSUV Model]

«profile»
SysML

«apply» {strict}

«modelLibrary»
SI Definitions

«apply» {strict}

«import»

HSUVModel
SysML Modeling as Part of the SE Process
Distiller Sample Problem
Distiller Problem Statement

The following problem was posed to the SysML team in Dec ’05 by D. Oliver:

Describe a system for purifying dirty water.

- Heat dirty water and condense steam are performed by a Counter Flow Heat Exchanger
- Boil dirty water is performed by a Boiler
- Drain residue is performed by a Drain
- The water has properties: vol = 1 liter, density 1 gm/cm³, temp 20 deg C, specific heat 1 cal/gm deg C, heat of vaporization 540 cal/gm.

A crude behavior diagram is shown.

What are the real requirements?
How do we design the system?
Distiller Types

Batch Distiller

Continuous Distiller

Note: Not all aspects of the distiller are modeled in the example
Distiller Problem – Process Used

• Organize the model, identify libraries needed
• List requirements and assumptions
• Model behavior
  – In similar form to problem statement
  – Elaborate as necessary
• Model structure
  – Capture implied inputs and outputs
    • segregate I/O from behavioral flows
  – Allocate behavior onto structure, flow onto I/O
• Capture and evaluate parametric constraints
  – Heat balance equation
• Modify design as required to meet constraints
• Model the user interaction
• Modify design to reflect user interaction
Distiller Problem – Package Diagram: Model Structure and Libraries

pkg Distiller { model organization }

- Distiller Requirements
- Distiller Use Cases
- Distiller Behavior
- Distiller Structure
- Item Types
- Value Types
- Engineering Analysis
- SI Definitions

package Value Types { value types for distiller }

<<ValueType>>
- Real
- °C
- N/m²
- gm/sec
- cal/sec
- cal/(gm°C)
- cal/gm
- cal/(gm°C)
- cal/gm
- °C
- N/m²
- gm/sec
- cal/sec
- cal/(gm°C)
- cal/gm

dimension = temperature
unit = degrees celcius
dimension = pressure
unit = newtons per square meter
dimension = mass flow rate
unit = grams per second
dimension = heat flow rate
unit = calories per second
dimension = specific heat
unit = calories per gram
dimension = latent heat
unit = calories per gram

dimension = specific heat
unit = calories per gram
dimension = latent heat
unit = calories per gram
Distiller Example
Requirements Diagram

pkg Distiller Requirements

**Source Requirements**

**Distiller Distiller Requirements**

Original Statement

Id = "S0.0"
Text = "Describe a system for purifying dirty water.
- Heat dirty water and condense steam are performed by a Counter Flow Heat Exchanger
- Boil dirty water is performed by a Boiler. Drain residue is performed by a Drain.
The water has properties: vol = 1 liter, density 1 g/m^3, temp 20 deg C, specific heat 1 cal/gm deg C, heat of vaporization 540 cal/gm."

- **<<requirement>> Purify Water**
  Id = "S1.0"
  Text = "The system shall purify dirty water."

- **<<requirement>> Heat Exchanger**
  Id = "S2.0"
  Text = "Heat dirty water and condense steam are performed by a Counter Flow Heat Exchanger."

- **<<requirement>> Boiler**
  Id = "S3.0"
  Text = "Boil dirty water is performed by a Boiler."

- **<<requirement>> Drain**
  Id = "S4.0"
  Text = "Drain residue is performed by a Drain."

- **<<requirement>> Water Properties**
  Id = "S5.0"
  Text = "Water has properties: density 1 g/m^3, temp 20 deg C, specific heat 1 cal/gm deg C, heat of vaporization 540 cal/gm."

- **<<requirement>> Water Initial Temp**
  Id = "S5.1"
  Text = "Water has an initial temp 20 deg C."

**<<requirement>> Distill Water**

(Distiller.Distiller Requirements.Derived.Requirements)

Id = "D1.0"
Text = "The system shall purify water by boiling it."
## Distiller Example: Requirements Tables

### Table: Original Statement

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>text</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0.0</td>
<td>Original Statement</td>
<td>Describe a system for purifying dirty water.</td>
</tr>
<tr>
<td>S1.0</td>
<td>PurifyWater</td>
<td>The system shall purify dirty water.</td>
</tr>
<tr>
<td>S2.0</td>
<td>HeatExchanger</td>
<td>Heat dirty water and condense steam are performed by a ...</td>
</tr>
<tr>
<td>S3.0</td>
<td>Boiler</td>
<td>Boil dirty water is performed by a Boiler.</td>
</tr>
<tr>
<td>S4.0</td>
<td>Drain</td>
<td>Drain residue is performed by a Drain.</td>
</tr>
<tr>
<td>S5.0</td>
<td>WaterProperties</td>
<td>water has properties: density 1 gm/cm3, temp 20 deg C, ...</td>
</tr>
<tr>
<td>S5.1</td>
<td>WaterInitialTemp</td>
<td>water has an initial temp 20 deg C</td>
</tr>
</tbody>
</table>

### Table: Requirements Tree

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>relation</th>
<th>id</th>
<th>name</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1.0</td>
<td>PurifyWater</td>
<td>deriveReq</td>
<td>D1.0</td>
<td>DistillWater</td>
<td>The requirement for a boiling function and a boiler implies that the water must be purified by distillation</td>
</tr>
</tbody>
</table>
Distiller Example – Activity Diagram: Initial Diagram for DistillWater

- This activity diagram applies the SysML EFFBD profile, and formalizes the diagram in the problem statement.
Distiller Example – Activity Diagram:
Control-Driven: Serial Behavior

- **coldDirty**: H₂O (liquid)
- **hotDirty**: H₂O (gas)
- **steam**: H₂O (liquid)
- **pure**: H₂O (liquid)
- **recovered**: Heat
- **external**: Heat
- **discharge**: Residue
- **predischarge**: Residue
- **a1**: HeatWater
- **a2**: BoilWater
- **a3**: CondenseSteam
- **a4**: DrainResidue

Continuous Distiller Here
Distiller Example – Block Definition
Diagram: DistillerBehavior

pkg 1. Initial Behavior [behavior breakdown with ObjectFlows]

Activities (Functions)

Heat Water

- a1
- dQ/dt: calories per second
- values
- water temp: °C (unit = degrees celsius, dimension = temperature)
- specific heat: cal/gm (nonunique unit = calories per gram, dimension = latent heat)
- latent heat: cal/(gm°C) (nonunique unit = calories per gram degree celsius, dimension = specific heat)
- water press: N/m² (unit = newtons per square meter, dimension = pressure)
- mass flow rate: gm/sec (unit = grams per second, dimension = mass flow rate)

Boil Water

- a2

Condense Steam

- a3

Drain Residue

- a4

Distill Water

- Control (not shown on BDD)

Things that flow (ObjectNodes)

Need to consider phases of H₂O
Distiller Example – State Machine
Diagram: States of H2O

States

Transitions

when (water temp == 100 & latent heat of vaporization added)

when (water temp == 100 & latent heat of vaporization removed)

when (water temp == 0 & latent heat of fusion added)

when (water temp == 0 & latent heat of fusion removed)
Distiller Example – Activity Diagram:
I/O Driven: Continuous Parallel Behavior
Distiller Example – Activity Diagram: No Control Flow, ActionPin Notation, Simultaneous Behavior


- a1 : Heat Water (cold dirty : H2O (stream), external : Heat (stream))
- a2 : Boil Water (hot dirty : H2O (stream))
- a3 : Condense Steam (pure : H2O (stream))
- a4 : Drain Residue (discharge : Residue (stream))

Flow arrows:
- of1:冷洗水到热洗水
- of2:热洗水到沸水
- of3:沸水到蒸气
- of4:蒸气到纯水
- of5:预排残渣到排残渣
- of6:排残渣到残渣
- of7:冷洗水到热洗水

Shutdown state:
- shutdown state
Distiller Example – Activity Diagram
(with Swimlanes): DistillWater

Distiller Example – Block Definition
Diagram: DistillerStructure

```
<table>
<thead>
<tr>
<th>pkg</th>
<th>Initial Distiller Structure [ distiller breakdown ]</th>
</tr>
</thead>
</table>

```

```
<<block>>
Distiller

```

```
condenser
<<block>>
Heat Exchanger
Satisfies = ☐ Heat Exchanger

```

```
evaporator
<<block>>
Boiler
Satisfies = ☐ Boiler

```

```
drain
<<block>>
Valve
Satisfies = ☐ Drain

```
Distiller Example – Block Definition Diagram: Heat Exchanger Flow Ports

pkg Initial Distiller Structure [distiller breakdown (ports)]

Heat Exchanger

- c in : Fluid
- c out : Fluid
- h in : Fluid
- h out : Fluid
- constraints: {h out.temp <= 120, c in.temp <= 60, h in.temp <= 120, c out.temp <= 90}

Boiler

- bottom : Heat
- middle : Fluid
- top : Fluid
- bottom : Fluid

Distiller

- condenser
- evaporator
- drain

Valve

- in : Fluid
- out : Fluid

Constraints (on Ports)

Flow Ports (typed by things that flow)
Distiller Example – Internal Block Diagram: Distiller Initial Design
Distiller Example – Table: Functional Allocation

Swimlane Diagram

Exercise for student:
Is allocation complete?
Where is “«objectFlow»of8”?
Parametric Diagram: Heat Balance

class Distiller Isobaric Heat Balance [ composition of equations ]

<<block>>
    : Distiller

main1 : H2O
    <<ValueType>>
    water temp : °C
    mass flow rate : gm/sec

main2 : H2O
    <<ValueType>>
    water temp : °C
    mass flow rate : gm/sec

main3 : H2O
    <<ValueType>>
    mass flow rate : gm/sec

main4 : H2O
    <<ValueType>>
    mass flow rate : gm/sec

q1 : Heat
    <<list>>
    dQ/dt : calories per second

<<constraint>>
    s : Heat Xfer Equation
    (q rate = heat/m rate*heat)

<<constraint>>
    s1 : Single Phase Heat Xfer Equation
    (q rate = th-to/m rate*heat)

<<constraint>>
    condensing : Phase Change Heat Xfer Equation
    (q rate = m rate*latent heat)

<<constraint>>
    boiling : Phase Change Heat Xfer Equation
    (q rate = m rate*heat)
### Distiller Example – Heat Balance Results

#### Table: Isobaric Heat Balance

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specific heat cal/gm-°C</td>
<td>1</td>
</tr>
<tr>
<td>Latent heat cal/cm</td>
<td>540</td>
</tr>
<tr>
<td>Mass flow rate gm/sec</td>
<td>6.8</td>
</tr>
<tr>
<td>Temp °C</td>
<td>100</td>
</tr>
<tr>
<td>dQ/dt cooling water cal/sec</td>
<td>540</td>
</tr>
<tr>
<td>dQ/dt steam-condensate cal/sec</td>
<td>540</td>
</tr>
<tr>
<td>Condenser efficiency</td>
<td>1</td>
</tr>
<tr>
<td>Heat deficit</td>
<td>0</td>
</tr>
<tr>
<td>dQ/dt condensate-steam cal/sec</td>
<td>540</td>
</tr>
<tr>
<td>Boiler efficiency</td>
<td>1</td>
</tr>
<tr>
<td>dQ/dt in boiler cal/sec</td>
<td>540</td>
</tr>
</tbody>
</table>

Note: Cooling water needs to have 6.75x flow of steam! Need bypass between hx_water_out and bx_water_in!

Satisfies «requirement» WaterSpecificHeat
Satisfies «requirement» WaterHeatOfVaporization
Satisfies «requirement» WaterInitialTemp
Distiller Example – Use Case and Sequence Diagrams

Distiller Use Cases: Operate Distiller

Operate Distiller

Operator

Distiller

Turn On

Power Lamp On

High Level Lamp On

Low Level Lamp On

Draining Lamp On

Operating Lamp On

1: Turn On

2: Power Lamp On

3: Operating Lamp On

4: High Level Lamp On

5: Low Level Lamp On

6: Draining Lamp On

7: Turn Off

8: Power Lamp Off

while state=Operating

level=high

level=low

state=draining residue

loop

alt

<<block>>
Distiller Example – Internal Block
Diagram: Distiller Controller

```
class Distiller [block diagram revised & elaborated]

  diverter assembly
    splitter : Tee Fitting
      m2.1 : H2O
      v : V Ctrl
      m2.2 : H2O
    b : Boiler Signals
    v1 : V Ctrl
    v2 : V Ctrl
    feed : Valve
    pwr : Elec Power
    b : Boiler Signals
    bp : Elec Power
    c : Boiler Signals
    main1 : H2O
    main2 : H2O
    v : V Ctrl
    main3 : H2O
    feed ctrl : V Ctrl
    main4 : H2O
    blr status : Blr Sig
    htr pwr : Elec Power
    v : V Ctrl
    htr pwr : Elec Power
    v : V Ctrl
    drain : Valve
    v : V Ctrl
    drain ctrl : V Ctrl
    blr ctrl : Blr Sig
    sludge1 : Residue
    sludge2 : Residue
    condenser : Heat Exchanger
    evaporator : Boiler
    drain : Valve
    user : Control Panel
    heat & valve : Controller
    splitter : Tee Fitting
    main1 : H2O
    main2 : H2O
    v : V Ctrl
    main3 : H2O
    feed ctrl : V Ctrl
    main4 : H2O
    blr status : Blr Sig
    htr pwr : Elec Power
    v : V Ctrl
    htr pwr : Elec Power
    v : V Ctrl
    drain : Valve
    v : V Ctrl
    drain ctrl : V Ctrl
    blr ctrl : Blr Sig
    sludge1 : Residue
    sludge2 : Residue
    condenser : Heat Exchanger
    evaporator : Boiler
    drain : Valve
    user : Control Panel
    heat & valve : Controller
```

4/15/2008 Copyright © 2006-2008 by Object Management Group. 98
Distiller Example – State Machine
Diagram: Distiller Controller

```
stm Controller State Machine

Off
  do Power Light Off

[power = on]

[bx level low]

Filling
  do open feed : Valve

[NOT bx level low]

Warming Up
  do bx1 heater on

[bx1 temp = 100]

Operating

Level Low
  do open feed : Valve

[NOT bx1 level low]

Level OK
  do shut all Valves

[NOT bx1 level high]

Level High
  do open drain : Valve

[NOT bx1 level high]

Building Up Residue
  do close drain : Valve

[residue timer]

Purging Residue
  do open drain : Valve

[drain timer]

Cooling Off
  entry /bx1 heater OFF
  do open feed : Valve, open drain : Valve

[shutdown command]

Draining
  do open drain : Valve

[bx1 temp = 30]

```
OOSEM – ESS Example
Integrated Product Development (IPD) is essential to improve communications.

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

System Modeling Activities – OOSEM
Integrating MBSE into the SE Process

Requirements Traceability is Managed
Through the Entire MBSE Process

- Mission use cases/scenarios
- Enterprise model

Define System Requirements

- System use cases/scenarios
- Elaborated context
- Req’ts diagram

Define Logical Architecture

- Logical architecture

Optimize & Evaluate Alternatives

- Engr Analysis Models
- Trade studies

Validate & Verify System

- Test cases/procedures

Synthesize Physical Architecture

- Node diagram
- HW, SW, Data architecture

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
ESS Requirements Flowdown

Market Needs

ESS System Specification

IntruderDetection

ESS Logical Requirements

ESS Allocated Requirements

IntruderDetection shall detect intruder entry and exit ...

System shall detect intruder entry and exit ...

satisfiedBy Entry/Exit Subsystem

verifiedBy Entry/Exit Detection Test
Operational View Depiction

bdd [package] Enterprise (As Is)

Central Monitoring Station As-Is

Comm Network

Residence

Dispatcher

Police

Intruder
ESS Operational Enterprise To-Be Model

```
<table>
<thead>
<tr>
<th>Domain To-Be</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESS Operational Enterprise</td>
</tr>
<tr>
<td>«moe» OperationalAvailability = {&gt;.99}</td>
</tr>
<tr>
<td>«moe» MissionResponseTime = {&lt;5 min}</td>
</tr>
<tr>
<td>«moe» OperationalCost = {TBD}</td>
</tr>
<tr>
<td>«moe» CostEffectiveness</td>
</tr>
<tr>
<td>MonitorSite ()</td>
</tr>
<tr>
<td>DispatchEmergencyServices ()</td>
</tr>
<tr>
<td>ProvideEmergencyResponse ()</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Physical Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protected Site</td>
</tr>
<tr>
<td>Customer</td>
</tr>
<tr>
<td>Intruder</td>
</tr>
</tbody>
</table>

| System |
| ESS |

| External |
| Emergency Services |
| Dispatcher |
| Responder |
| Police |
| Fire |
| Paramedic |

| External |
| Property |
| Single-family Residence |
| Multi-family Residence |
| Business |

| External |
| Comm Network |

| External |
| Customer |

| External |
| Intruder |
```

System Use Cases - Operate

- Activate/Deactivate
- Monitor Site
- Respond
  - Respond to Break-In
  - Respond to Fire
  - Respond to Medical

«include»
«extend»
System Scenario: Activity Diagram
Monitor Site (Break-In)
ESS Elaborated Context Diagram

ibd [domain] Domain-To-Be

«external» : Emergency Services
: EmergencyServicesIn
: EmergencyServicesOut

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

«external» : Physical Environment
: Environmental_In

«external» : Customer
: CustomerIn
: CustomerOut

«external» : Intruder
: IntruderSignal

«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 ()

: Environmetal_In

Detect Entry Subsystem Scenario

act detectEntry

«logical»
Entry Sensor

«continuous»
Door Input

«continuous»
Window Input

«logical»
Entry/Exit Monitor

Di : Door Input
Wi : Window Input
Ee : SensedEntry

Detect Event

Sensor Output

[State=BreakInResponse]

[Else]

[Status=Event]

Record Event

Log : Event

Alert Status

Event Log

«store»
«logical»
«subsystem»
entry/exit subsystem
Elaborating Logical Component

- Added operations from Detect Entry / Detect Exit logical scenario

- These operations support entry/exit subsystem
ESS Logical Design (Partial)
## Logical Components

<table>
<thead>
<tr>
<th>Entry Sensor</th>
<th>Exit Sensor</th>
<th>Perimeter Sensor</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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<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>X</td>
</tr>
<tr>
<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>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>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>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>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>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></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>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>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>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></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>Optical Sensor</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<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>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>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>
<tr>
<td>Alarm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

### Physical Components

- Allocating Logical Components to HW, SW, Data, and Procedures components
ESS Parametric Diagram
To Support Trade-off Analysis

par [block] EnterpriseEffectivenessModel

«moe»
MissionResponseTime

«moe»
OperationalAvailability

«moe»
OperationalCost

of1 : ObjectiveFunction

\{CE = \text{Sum}(w1 \cdot u(OA) + w2 \cdot u(MRT) + w3 \cdot u(OC))\}

MRT

OA

OC

CE

«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
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
```
OOSEM Browser View
Artisan Studio™ Example
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
  - MARTE
  - HLA
  - MathML
  - System Modeling
  - Simulation & Analysis

- **Interchange & Metamodeling Standards**
  - MOF
  - XMI
  - STEP/AP233
Standards-based Tool Integration with SysML

Systems Modeling Tool

Other Engineering Tools

Model/ Data Interchange

AP233/ XMI

AP233/ XMI
Participating SysML Tool Vendors

- Artisan (Studio)
- EmbeddedPlus (SysML Toolkit)
  - 3rd party IBM vendor
- No Magic (Magic Draw)
- Sparx Systems (Enterprise Architect)
- IBM / Telelogic (Tau and Rhapsody)
- TopCased
- Visio SysML template
Transitioning to SysML
Using Process Improvement To Transition to SysML

Continuous Improvement Cycle

Plan Improvement

Define Improvement

Assess & Measure Improvement

Deploy Improvement

Pilot Improvement
MBSE Transition Plan

• MBSE Scope
• MBSE Responsibilities/Staffing
• Process guidance
  – High level process flow (capture in SEMP)
  – Model artifact checklist
  – Tool specific guidance
• Tool support
  – Modeling tool
  – Requirements management
  – CM
• Training
• Schedule
Typical Integrated Tool Environment

- Project Management
  - SoS/ DoDAF / Business Process Modeling
  - System Modeling
    - SysML
    - Software Modeling
      - UML 2.0
    - Hardware Modeling
      - VHDL, CAD, ..
  - Simulation & Visualization
  - Engineering Analysis

CM/DM
- Product Data Management
- Requirements Management
- Verification & Validation

INCOSE

OMG SYSTEMS MODELING LANGUAGE

SysML
Summary and Wrap up
Summary

- SysML sponsored by INCOSE/OMG with broad industry and vendor participation and adopted in 2006
- 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
- Multiple vendor implementations available
- Standards based modeling approach for SE expected to improve communications, tool interoperability, and design quality
- Plan SysML transition as part of overall MBSE approach
- Continue to evolve SysML based on user/vendor/researcher feedback and lessons learned
References

- **OMG SysML website**
  - [http://www.omgsysml.org](http://www.omgsysml.org)
  - Refer to current version of SysML specification, vendor links, tutorial, and papers
- **UML for Systems Engineering RFP**
  - OMG doc# ad/03-03-41
- **UML 2 Superstructure v2.1.2**
  - OMG doc# formal/2007-11-02
- **UML 2 Infrastructure v2.1.2**
  - OMG doc# formal/2007-11-04

**PAPERS**
- Integrating Models and Simulations of Continuous Dynamics into SysML
  - Thomas Johnson, Christiaan Paredis, Roger Burkhart, Jan ’2008
- Simulation-Based Design Using SysML - Part 1: A Parametrics Primer
  - RS Peak, RM Burkhart, SA Friedenthal, MW Wilson, M Bajaj, I Kim
- Simulation-Based Design Using SysML - Part 2: Celebrating Diversity by Example
  - RS Peak, RM Burkhart, SA Friedenthal, MW Wilson, M Bajaj, I Kim
- SysML and UML 2.0 Support for Activity Modeling,
- The Systems Modeling Language,
  - Matthew Hause, Alan Moore, June ’2006.
- An Overview of the Systems Modelling Language for Products and Systems Development,
- Model-driven systems development,

**TUTORIAL AUTHORS**
- Sanford Friedenthal (sanford.friedenthal@lmco.com)
- Alan Moore (alan.moore@mathworks.co.uk)
- Rick Steiner (fsteiner@raytheon.com)