OMG Systems Modeling Language
(OMG SysML™)
Tutorial

11 July 2006

Sanford Friedenthal
Alan Moore
Rick Steiner

Copyright © 2006 by Object Management Group.
Published and used by INCOSE and affiliated societies with permission.
Caveat

- 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
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
• High-level process 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
  – 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 (30)
• Diagram Overview (135)
• SysML Modeling as Part of SE Process (120)
  – Structured Analysis – Distiller Example
  – OOSEM – Enhanced Security System Example
• SysML in a Standards Framework (20)
• Transitioning to SysML (10)
• Summary (15)
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

- 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
System-of-Systems

Modeling Needed to Manage System Complexity
Modeling at Multiple Levels of the System
Stakeholders Involved in System Acquisition

- Customers
- Developers/Integrators
- Project Managers
- Vendors
- Regulators
- Testers

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 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 V2.0
  – 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
SysML Team Members

• Industry & Government
  – American Systems, BAE SYSTEMS, Boeing, Deere & Company, EADS-Astrium, Eurostep, Lockheed Martin, Motorola, NIST, Northrop Grumman, oose.de, Raytheon, THALES

• Vendors

• Academia
  – Georgia Institute of Technology

• Liaison Organizations
  – INCOSE, ISO AP233 Working Group
Diagram Overview
Relationship Between SysML and UML

UML 2

SysML

UML not required by SysML (UML - UML4SysML)

UML reused by SysML (UML4SysML)

SysML extensions to UML (SysML Profile)
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**

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

1. Structure

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

«block» Library::Electronic Processor

«block» Anti-Lock Controller

«block» Traction Detector

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

c1:modulator interface

m1:Brake Modulator

2. Behavior

interaction state machine activity/function

sd ABS_ActivationSequence [Sequence Diagram]

stm TireTraction [State Diagram]

act PreventLockup [Activity Diagram]

DetectionLossOfTraction → TractionLoss → ModulateBrakingForce

3. Requirements

Vehicle System Specification

Braking Subsystem Specification

«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."

«deriveReqt»

4. Parametrics

par [constraintBlock] StraightLineVehicleDynamics [Parametric Diagram]

:BrakingForce Equation \[f = (tf*bf)*(1-tl)\]

:f:

:F:

:Accelleration Equation \[F = ma\]

:a:

:v:

:VelocityEquation \[a = dv/dt\]

:x:
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, seq, etc.)
  – Model element type (activity, block, interaction, etc.)
  – Model element name
  – Descriptive diagram name or view name
• A separate diagram description block is used to indicate if the diagram is complete, or has elements elided

Diagram Description

- Version:
- Description:
- Completion status:
- Reference:
  (User-defined fields)
Structural Diagrams
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

pkg SampleModel [by diagram type]

Use Cases
Requirements
Behavior
Structure
EngrAnalysis

pkg SampleModel [by level]

Enterprise
System
Logical Design
Allocated Design
Verification

pkg SampleModel [by IPT]

Architecture Team
Requirements Team
IPT A
IPT B
IPT C

By Diagram Type
By Hierarchy
By IPT
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

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

| 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
  - s1:Sensor

**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.
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**

- **required interface**
  - **calls the operations**

**Flow Port**

- **item flow**

**Objects**

- `part1`
- `part2`
Delegation Through Ports

- Delegation can be used to preserve encapsulation of block.
- Interactions at outer ports of Block1 are delegated to ports of child parts.
- Ports must match (same kind, types, direction etc.).
- (Deep-nested) Connectors can break encapsulation if required (e.g. in physical system modeling).
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 $F= m \times a$)

Parametrics Enable Integration of Engineering Analysis with Design Models
Defining Vehicle Dynamics

Defining Reusable Equations for Parametrics
Vehicle Dynamics Analysis

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

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

[Diagram showing relationships between diagrams:]
- Behavior Diagram
- Requirement Diagram
- Structure Diagram
- Parametric 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

Activity Diagram Notation

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

- `bdd PreventLockup [Activity Breakdown]`
  - `«activity» PreventLockup`
    - a1: `«activity» DetectLossOf Traction`
    - a2: `«activity» ModulateBrakingForce`

**Use**

- `act PreventLockup [Activity Diagram]`
  - a1: `DetectLossOf Traction`
  - a2: `Modulate BrakingForce`

11 July 2006

Copyright © 2006 by Object Management Group.
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
White Box Sequence (StartVehicle)

Decomposition of Black Box Into White Box Interaction
Trial Result of Vehicle Dynamics

Typical Example of a Timing Diagram

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.
Operational States (Drive)

Transition notation: trigger[guard]/action

Nominal states only

stm HSUVOperationalStates

Off

keyOff/

start[in neutral]/start engine

shutOff/stop engine

Operate

Idle

accelerate/

when (speed = 0)

releaseBrake/

Accelerating/Cruising

engageBrake/

Braking

Nominal states only
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
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 DeriveReqt, Satisfy, Verify, Refine, Trace, Copy
Requirements Breakdown

Requirement Relationships Model the Content of a Specification

- **Requirement Relationships**
  - **Eco-Friendliness**
  - **Performance**
  - **Braking**
  - **Fuel Economy**
  - **Acceleration**

- **HSUVSpecification**

- **Emissions**
  - Id = "R1.2.1"
  - text = "The vehicle shall meet Ultra-Low Emissions Vehicle standards."

- **Power**
  - **RefinedBy**
    - «useCase» HSUVUseCases::Accelerate
  - **VerifiedBy**
    - «testCase» MaxAcceleration
  - **SatisfiedBy**
    - «block» PowerSubsystem

- **Reqt derive**
Example of Derive/Satisfy Requirement Dependencies

- «requirement» OffRoadCapability
- «requirement» Accelleration
- «requirement» CargoCapacity

Arrow Direction Opposite Typical Requirements Flow-Down

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
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

«import»
HSUVModel
Cross Connecting Model Elements

1. Structure

Vehicle System Specification

- «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."

Braking Subsystem Specification

- «deriveReqt»

2. Behavior

- «allocate» TractionDetector
- «allocate» BrakeModulator

- «activity» DetectLossOf Traction
- «activity» Modulate BrakingForce

- «connector» c1:modulatorInterface

3. Requirements

- «requirement» Anti-Lock Performance
- «requirement» Minimum Stopping Distance

4. Parametrics

- «parametric» StraightLineVehicleDynamics

Copyright © 2006 by Object Management Group.
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/cm^3, temp 20 deg C, specific heat 1cal/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
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
Distiller Problem – Package Diagram:
Model Structure and Libraries

```
bdd [package] ValueTypes

- «valueType» Real
  - unit = ºC
  - dimension = temperature

- «valueType» press
  - unit = N/m^2
  - dimension = pressure

- «valueType» massFlowRate
  - unit = gm/sec
  - dimension = massFlowRate

- «valueType» dQ/dt
  - unit = cal/sec
  - dimension = heatFlowRate

- «valueType» efficency
  - unit = null
  - dimension = efficency

- «valueType» specificHeat
  - unit = cal/(gmºC)
  - dimension = specificHeat

- «valueType» latentHeat
  - unit = cal/gm
  - dimension = latentHeat

pkg [model] Distiller [Model Overview]

- DistillerRequirements
- DistillerUseCases
- DistillerBehavior
- DistillerStructure
- ValueTypes
- ItemTypes
```
Distiller Example
Requirements Diagram

1. The system shall purify dirty water.
2. The system shall purify water by boiling it.
3. Heat dirty water and condense steam are performed by a Counter Flow Heat Exchanger.
4. Boil dirty water is performed by a Boiler.
5. The water has properties: vol = 1 liter, density 1 gm/cm³, temp 20 deg C, specific heat 1cal/gm deg C, heat of vaporization 540 cal/gm.
6. Drain residue is performed by a Drain.

- 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 1cal/gm deg C, heat of vaporization 540 cal/gm.
Distiller Example: Requirements Tables

**table [requirement] OriginalStatement [Decomposition of OriginalStatement]**

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>text</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0.0</td>
<td>OriginalStatement</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 [requirement] PurifyWater [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>deriveReqt</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
Distiller Example – Block Definition Diagram: DistillerBehavior

- **Activities (Functions)**
  - HeatWater
  - BoilWater
  - CondenseSteam
  - DrainResidue

- **Control**
  (not shown on BDD)

- **Things that flow (ObjectNodes)**
  - H2O
  - steam
  - hotDirty
  - pure
  - recovered
  - external
  - hiPress

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

States:
- Gas
- Liquid
- Solid

Transitions:
- Add Latent Heat of Vaporization
- Remove Latent Heat of Vaporization
- Add Latent Heat of Liquification
- Remove Latent Heat of Liquification
Distiller Example – Activity Diagram:
I/O Driven: Continuous Parallel Behavior

**Act [activity] DistillWater [Parallel Continuous Activities]**

- **coldDirty:** H$_2$O [liquid]
- **recovered:** Heat
- **hotDirty:** H$_2$O [liquid]
- **external:** Heat
- **steam:** H$_2$O [gas]
- **hiPress:** Residue
- **loPress:** Residue
- **a1:** HeatWater
- **a2:** BoilWater
- **a3:** CondenseSteam
- **a4:** DrainResidue
- **pure:** H$_2$O [liquid]

**Continuous Distiller**
Distiller Example – Activity Diagram: No Control Flow – Simultaneous Behavior
Distiller Example – Activity Diagram (with Swimlanes): DistillWater

Parts

- allocate
  - h1: HeatExchanger
  - b1: Boiler
  - drain Valve

- allocate
  - coldDirtyH2O [liquid]

- allocate
  - steamH2O [gas]

- allocate
  - highPress:Residue

- allocate
  - a1:HeatWater

- allocate
  - a3:CondenseSteam

- allocate
  - a2:BoilWater

- allocate
  - a4:DrainResidue

- allocate
  - lowPress Residue

- allocate
  - continuous: external Heat

- allocate
  - continuous: hotDirtyH2O [liquid]

- allocate
  - continuous: pureH2O [liquid]

- streaming
  - a1 HeatWater
  - a3 CondenseSteam

- streaming
  - a2 BoilWater

- continuous
  - recovered Heat

- continuous
  - lowPress Residue

- continuous
  - highPress Residue

- continuous
  - coldDirtyH2O [liquid]

- continuous
  - external Heat

- shutdown
Distiller Example – Block Definition Diagram: DistillerStructure.
Distiller Example – Block Definition
Diagram: Heat Exchanger Flow Ports

Constraints (on Ports)

Flow Ports (typed by things that flow)

Generic Things That Flow (Blocks)

Generic Subsystems (Blocks)
Distiller Example – Internal Block Diagram: Distiller Initial Design

Parts
(Block used in context)

Flow Ports

Connectors

Things That Flow In Context (ItemFlows)
Distiller Example – Internal Block Diagram: Distiller with Allocation
Distiller Example – Parametric Diagram:
Heat Balance Equations

par [block] Distiller [Simplified Isobaric Heat Balance Analysis]

Parts or Item Flows

Value Properties

Value Bindings

Constraints

Constraint callouts

Note: Underline = these are invariant properties of all uses of H2O
Distiller Example – Heat Balance Results

<table>
<thead>
<tr>
<th></th>
<th>water_in</th>
<th>hx_water_out</th>
<th>bx_water_in</th>
<th>bx_steam_out</th>
<th>water_out</th>
</tr>
</thead>
<tbody>
<tr>
<td>specific heat cal/gm-°C</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>latent heat cal/cm</td>
<td>540</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mass flow rate gm/sec</td>
<td>6.75</td>
<td>6.75</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>temp °C</td>
<td>20</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>dQ/dt cooling water cal/sec</td>
<td>540</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dQ/dt steam-condensate cal/sec</td>
<td>540</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>condenser efficiency</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>heat deficit</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dQ/dt condensate-steam cal/sec</td>
<td>540</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>boiler efficiency</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dQ/dt in boiler cal/sec</td>
<td>540</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: Cooling water needs to have 6x flow of steam! Need bypass between hx_water_out and bx_water_in!
Distiller Example – Activity Diagram: Updated DistillWater
Distiller Example – Use Case and Sequence Diagrams

seq OperateDistiller [Operational Sequence]

«actor» :User

«block» :Distiller

TurnOn

PowerLampOn

OperatingLampOn

loop

alt

LevelHighLampOn

DrainingLampOn

LevelLowLampOn

TurnOff

PowerLampOff

uc DistillerUseCases [Operate Distiller]

Operate Distiller

User

Distiller
Distiller Example – State Machine Diagram: Distiller Controller
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.
Systems Modeling Activities - OOSEM

Major SE Development Activities

- Analyze Needs
  - 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

Common Subactivities
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

Intruder Detection

ESS Logical Requirements

ESS Allocated Requirements

Entry/Exit Subsystem

Entry/Exit Detection Test
Operational View Depiction
ESS Operational Enterprise To-Be Model
System Use Cases - Operate

- Acticate/Deactivate
- Monitor Site
- Operate
- Respond
  - Respond to Break-In
  - Respond to Fire
  - Respond to Medical

uc [package] System Use Cases
ESS Logical Design – Example Subsystem

```
[ibd [subsystem]Entry/Exit Subsystem]

: Door Input
  : Door Input
  : Window Input
  : Window Input

  «logical» : Entry Sensor
  : SensedEntry

  «logical» : Exit Sensor
  : SensedExit

  «logical» : Entry/Exit Monitor

  «logical» : Event Monitor
  «store» : Event Log

  : Entry/Exit Alert Status

  : Alert Status
```

ESS Logical Design (Partial)
ESS Allocation Table (partial)

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

<table>
<thead>
<tr>
<th>Physical Components</th>
<th>Logical Components</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>software</strong></td>
<td>Entry Sensor</td>
</tr>
<tr>
<td>Device Mgr</td>
<td></td>
</tr>
<tr>
<td>SF Comm I/F</td>
<td></td>
</tr>
<tr>
<td>User I/F</td>
<td></td>
</tr>
<tr>
<td>Event Mgr</td>
<td>X</td>
</tr>
<tr>
<td>Site Status Mgr</td>
<td></td>
</tr>
<tr>
<td>Site RDBMS</td>
<td></td>
</tr>
<tr>
<td>CMS RDBMS</td>
<td></td>
</tr>
<tr>
<td><strong>data</strong></td>
<td>Video File</td>
</tr>
<tr>
<td>CMS Database</td>
<td></td>
</tr>
<tr>
<td>Site Database</td>
<td></td>
</tr>
<tr>
<td><strong>hardware</strong></td>
<td>Optical Sensor</td>
</tr>
<tr>
<td>DSL Modem</td>
<td></td>
</tr>
<tr>
<td>User Console</td>
<td></td>
</tr>
<tr>
<td>Video Camera</td>
<td></td>
</tr>
<tr>
<td>Alarm</td>
<td></td>
</tr>
</tbody>
</table>

ESS Deployment View
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))\}

MRT
OA
OC
CE

«moe» CostEffectiveness
Entry/Exit Test Case

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 Entry: Alert Status
Enter
:SensedEntry
Exit
:SensedExit
Intruder Exit: Alert Status

«testComponent»:IntruderEmulator
«sut» «hardware»:Door[1] /:Optical Sensor
«sut» «hardware»:Site Processor
«sut» «hardware»:DSL Modem
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
  - UPDM
  - HLA
  - MathML

- **Interchange & Metamodelling Standards**:
  - MOF
  - XMI
  - STEP/AP233

**Implemented By Tools**

**Data Repository**
Standards-based Tool Integration with SysML
Participating SysML Tool Vendors

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

|-------|-------------------------|-------------------------|----------------------------------|----------------------------------------|----------------|-------|-------------------|-------|-------------------|-----------------|--------------------------|-------------------------------|

11 July 2006 Copyright © 2006 by Object Management Group.
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
• 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