OMG SysML™ Specification

- **Specification status**
  - Adopted by OMG in May ’06
  - Available Specification v1.0 in Sept ’07
  - Available Specification v1.1 in Nov ‘08
  - Revision task force for v1.2 in process

- Multiple vendor implementations available

- 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/](http://www.omgsysml.org/)

- Refer to “A Practical Guide to SysML” by Friedenthal, Moore, and Steiner for language details and reference
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

• Class Exercise
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
- Engine
- Transmission
- Transaxle

Structural/Component Model
- Mass Properties
- Structural
- Safety
- Cost Model

Other Engineering Analysis Models

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
Modeling at Multiple Levels of the System
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)
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
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

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

- Behavior Diagram
- Requirement Diagram
- Structure Diagram

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

Modified from UML 2

Same as UML 2

New diagram type
4 Pillars of SysML – ABS Example

1. Structure

- **bdd** [Package] Structure [ABS Structure Hierarchy]
  - **<<block>>** Library: Anti-Lock Control
  - **<<block>>** ABS_Init
  - **<<block>>** ABS_ActivationSequence [Sequence Diagram]
    - d1: Traction Detector
    - m1: Brake Modulator
    - detTrkLos()
    - modBrkFrc()
    - sendSignal()
    - modBrkFrc(traction_signal:boolean)
    - sendAck()

2. Behavior

- Interaction state machine activity/function

3. Requirements

- **req** [Package] Vehicle Specifications [Braking Requirements]
  - **<<requirement>>** Vehicle System Specification
    - Id = "10.2"
    - Text = "The vehicle shall stop from 60 miles per hour within 150 ft on a clean dry surface."
  - **<<requirement>>** Braking Subsystem Specification
    - Id = "33.7"
    - Text = "The braking system shall prevent wheel lockup under all braking conditions."

4. Parametrics

- **par** [Block] Straight Line Vehicle Dynamics [Parameters]
  - e1: Braking Force Equation
    - \( f = (f^2 - f^4)(1 - f) \)
  - e2: Acceleration Equation
    - \( a = m/s^2 \)
  - e3: Velocity Equation
    - \( v = m/sec \)
  - e4: Distance Equation
    - \( x = m \)
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 Description
Version:
Description:
Completion status:
Reference:
(User-defined fields)
```

```
«diagram usage»
diagramKind [modelElementType] modelElementName [diagramName]
```

Header

Contents
Structural 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
- 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 kine (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

By Diagram Type
pkg SampleModel [by diagram type]
- Use Cases
- Requirements
- Behavior
- Structure
- EngrAnalysis

By Hierarchy
pkg SampleModel [by level]
- Enterprise
- System
- Logical Design
- Physical Design
- Verification

By IPT
pkg SampleModel [by IPT]
- Architecture Team
- Requirements Team
- IPT A
- IPT B
- IPT C
• 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

- 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

<table>
<thead>
<tr>
<th>Compartment Label</th>
<th>«block»</th>
<th>BrakeModulator</th>
</tr>
</thead>
<tbody>
<tr>
<td>allocatedFrom</td>
<td>«activity» Modulate BrakingForce</td>
<td></td>
</tr>
<tr>
<td>values</td>
<td>DutyCycle: Percentage</td>
<td></td>
</tr>
</tbody>
</table>
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

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

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

Enclosing Block
Connector
Item Flow

ibd [Block] Anti-Lock Controller [ Item Flow ]

c2:

d1 : Traction Detector
activate : 5vdc

m1 : Brake Modulator

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

**Standard Port and Flow Port**
**Support Different Interface Concepts**
Port Notation

**Standard Port**
- **Provided Interface**: (provides the operations)
- **Required Interface**: (calls the operations)

**Flow Port**
- **Item Flow**
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 × 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 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

Example

act Example

Activity

Input

Output

[x > 0]

[x <= 0]
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**
  - (Event Data Pin often elided)

- **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 $\Delta T$ime between tokens approaches zero

Actions are enabled by default when activity is enabled

Accept Event Action  
Will Terminate Execution

Interruptible Region

Continuous Flow
Example – Operate Car

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)

Activity Diagram (with Swimlanes)
Activity Decomposition

**Definition**

- Prevent Lockup
- Detect Loss of Traction
- Modulate Braking Force

**Use**

- a1 : Detect Loss of Traction
- p1 : TractLoss
- a2 : Modulate Braking Force
- p2 : TractLoss
SysML EFFBD Profile

EFFBD - Enhanced Functional Flow Block Diagram

Aligning SysML with Classical Systems Engineering Techniques

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)

```
black

drive: Driver

ref

StartVehicleBlackBox

par

alt controlSpeed

[ state = (idle) ]

ref

Idle

[ state = (accelerating/cruising) ]

ref

Accelerate/Cruise

[ state = (braking) ]

ref

Brake

ref

Steer

ref

Park/ShutdownVehicle
```

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

References Lifeline Decomposition For White Box Interaction

Simple Black 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
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

Drive_The_Vehicle

Accelerate

Steer

Park

Brake

Flat_Tire

Driver

«include»

«extend»

«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”
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
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**

- **HSUVRequirements**
  - **Eco-Friendliness**
  - **Performance**
  - **Braking**
  - **FuelEconomy**
  - **Acceleration**

- **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

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

The master cylinder in previous version leaked.

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

```
ModelingDomain [Establishing HSUV Model]

profile SysML

modelLibrary SI Definitions

apply {strict}

apply {strict}

import

HSUVModel
```
Cross Connecting Model Elements

1. Structure

- ibd [block] Anti-LockController
  - [Internal Block Diagram]
  - satisfies
    - requirements
    - Anti-Lock Performance

- d1: TractionDetector
- m1: BrakeModulator

allocatedFrom
- actions: a1: DetectLoss of Traction
- actions: a2: Modulate Braking Force

values: DutyCycle: Percentage

allocate

satisfy

2. Behavior

- act [Flow] TractionController
- objFlow1:
  - TractLoss
  - TractLoss

- a2: Modulate BrakingForce

- v.b.abs.m1.duty cycle : %
  - v.mass : Kg
  - v.c.fraction : N
  - v.b.f.braking force : N

- e1 : Braking Force Equation
  - \( f = (t^3 + tf) / (1-tf) \)

- e2 : Acceleration Equation
  - \( a = m/sec^2 \) \( f = m/a \)

- e3 : Velocity Equation
  - \( v = dx/dt \)

- e4 : Distance Equation
  - \( x = m \)

- v.position : m

- time : sec

3. Requirements

- Vehicle System Specification
- Braking Subsystem Specification

- requirement: StoppingDistance
  - id: 102
  - text: The vehicle shall stop from 50 mph within 150 ft on a clean dry surface.

- requirement: Anti-LockPerformance
  - id: 33.7
  - text: Braking subsystem shall prevent wheel lockup under all braking conditions.

Verify

4. Parametrics

OMG SYSTEML Modeling Language
SysML Modeling as Part of the SE Process
Distiller Sample Problem

Refer to Chapter 15
“A Practical Guide to SysML”
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 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

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
Distiller Example Requirements Diagram

Distiller Requirements (Top Level Requirements)

Source_Requirements
(C27:Distiller Requirements)

<<requirement>>
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>>
Purity 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:
- 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>>
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 [Decomposition of Original Statement]**

<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></td>
<td></td>
<td>S5.1 WaterInitialTemp      water has an initial temp 20 deg C</td>
</tr>
</tbody>
</table>

**Table: 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>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

Continuous Distiller Here

Batch Distiller
Distiller Example – Block Definition
Diagram: DistillerBehavior

Activities (Functions)

- Heat Water
- Boil Water
- Condense Steam
- Drain Residue

Control (not shown on BDD)

Need to consider phases of H₂O

Things that flow (ObjectNodes)

Distilled Water

Heat

Residue

H₂O

Variables:
- `sludge temp`: °C (unit = degrees celcius, dimension = temperature)
- `sludge press`: N/m² (unit = newtons per square meter, dimension = pressure)
- `water temp`: °C
- `specific heat`: cal/gm
- `latent heat`: cal/(g°C)
- `mass flow rate`: g/min
Distiller Example – State Machine Diagram: States of H2O

- **States**:
  - Gas
  - Liquid
  - Solid

- **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

Batch Distiller Here

Continuous Distiller

Distiller Example – Activity Diagram:
No Control Flow, ActionPin Notation,
Simultaneous Behavior

Distiller Example – Activity Diagram
(with Swimlanes): DistillWater
Distiller Example – Block Definition

Diagram: DistillerStructure

- Condenser
  - Heat Exchanger
    - Satisfies = Heat Exchanger

- Evaporator
  - Boiler
    - Satisfies = Boiler

- Drain
  - Valve
    - Satisfies = Drain
Distiller Example – Block Definition
Diagram: Heat Exchanger Flow Ports

```
bdd Initial Distiller Structure [distiller breakdown (ports)]

<<block>>
Distiller

condenser
Heat Exchanger
constraints
{h out.temp<=120,
c in.temp<=60,
h in.temp<=120,
c out.temp<=90}

h in : Fluid
h out : Fluid

in : Fluid
evaporator
middle : Fluid

out : Fluid

in : Fluid
Valve

out : Fluid

in : Fluid
Boiler

top : Fluid

bottom : Fluid

top : Heat

bottom : Heat

Constraints (on Ports)

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

ibd Distiller [1:distiller block diagram (initial)]

- **condenser**: Heat Exchanger
  - **h in**: Fluid
  - **h out**: Fluid
  - **main3**: H2O

- **evaporator**: Boiler
  - **top**: Fluid
  - **bottom**: Heat
  - **in**: Fluid
  - **out**: Fluid
  - **main4**: H2O

- **drain**: Valve
  - **purified**: H2O

- **sludge1**: Residue
  - **sludge2**: Residue

- **main1**: H2O
  - **dirty water**: H2O
  - **c in**: Fluid
  - **c out**: Fluid
Distiller Example – Table: Functional Allocation

<table>
<thead>
<tr>
<th>Initial Distiller Structure</th>
<th>Distiller</th>
<th>Condenser</th>
<th>Drain</th>
<th>Evaporator</th>
<th>Main1</th>
<th>Main2</th>
<th>Main3</th>
<th>Main4</th>
<th>Q1</th>
<th>Sludge1</th>
<th>Sludge2</th>
</tr>
</thead>
<tbody>
<tr>
<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>

Swimlane Diagram

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

```
par [Constraint Elcct] Distiller Isochoric Heat Balance [ composition of equations ]

<<block>>
: Distiller

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

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

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

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

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

<<constraint>>
s1 : Single Phase Heat Xfer Equation
(q rate = (h-tc) * m rate * s heat)

th : °C
q rate : cal/sec
s heat : cal(gm°°C)

<<block>>
: H2O
<<ValueType>>
specific heat : cal(gm°°C)

<<ValueType>>
latent heat : calgm

condensing : Phase Change Heat Xfer Equation
(q rate = m rate * I heat)

m rate : gm/sec

heating : Phase Change Heat Xfer Equation
(q rate = m rate * I heat)

m rate : gm/sec

q rate : cal/sec
1 heat : calgm
```

### Distiller Example – Heat Balance Results

**table** IsobaricHeatBalance [Results of Isobaric Heat Balance]

<table>
<thead>
<tr>
<th></th>
<th>main1 : H2O</th>
<th>main2 : H2O from condenser</th>
<th>main2 : H2O into evap</th>
<th>main3 : H2O</th>
<th>main4 : H2O</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>specific heat cal/gm⁻°C</strong></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>latent heat cal/cm</strong></td>
<td>540</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Satisfies «requirement» WaterSpecificHeat
Satisfies «requirement» WaterHeatOfVaporization
Satisfies «requirement» WaterInitialTemp

<table>
<thead>
<tr>
<th></th>
<th>6.8</th>
<th>6.8</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>mass flow rate gm/sec</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>temp °C</strong></td>
<td>20</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>540</th>
<th>540</th>
<th>540</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>dQ/dt cooling water cal/sec</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>dQ/dt steam-condensate cal/sec</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>condenser efficiency</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>heat deficit</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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!

1. Set these (steady state)
2. Solve for these
Distiller Example – Activity Diagram: Updated DistillWater
Distiller Example – Internal Block Diagram: Updated Distiller
Distiller Example – Use Case and Sequence Diagrams

**Operational Sequence (simple sequence)**

- **Operator**
  - 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

- **Distiller**
  - **while state=Operating**
  - **state=draining residue**
  - **level=high**
  - **level=low**

**Use Case (use case example)**

- **Distiller**
  - **Operator**
  - **Operate Distiller**
Distiller Example – State Machine
Diagram: Distiller Controller

stm Controller State Machine [ simple diagram ]

Filling
- do open feed : Valve
- [NOT bx1 level low]

Warming Up
- do bx1 heater on
- [bx1 temp = 100]

Operating
- do bx1 heater on

Level Low
- do open feed : Valve
- [bx1 level low]
  - [NOT bx1 level low]

Level OK
- do shut all Valves
- [bx1 level high]
  - [NOT bx1 level high]

Level High
- do open drain : Valve
  - [bx1 level low]
  - [NOT bx1 level high]

Building Up Residue
- do close drain : Valve
- [residue timer]

Purging Residue
- do open drain : Valve
- [drain timer]

Draining
- do open drain : Valve
- [bx1 temp = 30]

Cooling Off
- entry / bx1 heater OFF
- do open feed : Valve, open drain : Valve
- [shutdown command]

Off
- do Power Light Off
- [power = on]
- [bx level low]
OOSEM – ESS Example

Refer to Chapter 16
“A Practical Guide to SysML”
Integrated Product Development (IPD) is essential to improve communications

A Recursive V process that can be applied to multiple levels of the system hierarchy
System Modeling Activities – OOSEM
Integrating MBSE into the SE Process

Major SE Development Activities

- Analyze Needs
  - Causal analysis
  - Mission use cases/scenarios
  - Enterprise model

- Define System Requirements
  - System use cases/scenarios
  - Elaborated context

- Define Logical Architecture
  - Logical decomposition
  - Logical scenarios
  - Logical subsystems

- Optimize & Evaluate Alternatives
  - Parametric Diag
  - Trade study

- Manage Requirements
  - Reqt’s Diagram & tables

- Validate & Verify System
  - Test system
  - Test cases

- Synthesize Allocated Architecture
  - Node diagram
  - HW, SW, Data arch
  - System deployment

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 (early version) for the SysML artifacts
System shall detect intruder entry and exit ...

IntruderDetection

id# = SS102
bt = System shall detect intruder entry and exit ...

deriveReq

satisfiedBy
Entry/Exit Subsystem

verifiedBy
Entry/Exit Detection Test

ESS Logical Requirements

id# = LR1

R111

id# = SS111

satisfy

ESS System Models

ESS Logical Design Models

ESS Allocated Design Models

Market Needs

ESS System Models

ESS Logical Requirements

ESS Allocated Requirements

«document»

«requirement»

«deriveReqt»

«refine»

«satisfy»

«trace»

«refine»

«satisfy»

«trace»

req [package] ESS Requirements Flowdown
Operational View Depiction

[Image of a diagram showing a central monitoring station, communication network, residence, dispatcher, police car, and intruder.]
ESS Enterprise As-Is Model
ESS Operational Enterprise To-Be Model

Diagram showing relationships between various entities such as Intruder, Customer, Protected Site, Emergency Services, ESS, Comm Network, Physical Environment, Property, Single-family Residence, Multi-family Residence, Business, Dispatcher, Responder, Police, Fire, and Paramedic.
System Use Cases - Operate

- Activate/Deactivate
- Monitor Site
- Operate
- Respond
  - Respond to Break-In
  - Respond to Fire
  - Respond to Medical
System Scenario: Activity Diagram
Monitor Site (Break-In)

**System Scenario:**

- **Monitor Site (Break-In)**

**Act:** Monitor Site (break in)

**System:**
- Intruder
  - Enter Property
  - Conduct Theft
  - Exit Property
- ESS
  - Status Update
  - DetectEntry
  - ValidateEntry
  - GenerateAlarm
  - ReportEntry
- InternalMonitor
  - DetectExit
  - ReportExit

**External:**
- Emergency Services
  - Assess Report
  - Report Update
  - Dispatch Police

**Events:**
- System On
- Validated Entry
- [Alert]

**Status:**
- System On
- System Off

Detect Entry Subsystem Scenario

act detectEntry

«subsystem» entry/exit subsystem

«logical» Entry Sensor

Door Input

«continuous»

Window Input

Sense State Change

wi : Window Input

ee : SensedEntry

di : Door Input

Detect Event

sensor : SensorOutput

[State=BreakInResponse]

[Else]

Record Event

log : Event

status

estatus

Event Log

Alert Status

«store»

«logical»

Entry/Exit Monitor

Event Monitor

«subsystem»

«logical»

«continuous»

Copyright © 2006-2009 by Object Management Group.

4/01/2009
Elaborating Logical Component

- Added operations from Detect Entry / Detect Exit logical scenario
- These operations support entry/exit subsystem
ESS Logical Design – Example Subsystem
ESS Logical Design (Partial)
### Logical Components

- 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>Device Mgr</strong></td>
<td>Entry Sensor</td>
</tr>
<tr>
<td><strong>SF Comm I/F</strong></td>
<td>Exit Sensor</td>
</tr>
<tr>
<td><strong>User I/F</strong></td>
<td>Perimeter Sensor</td>
</tr>
<tr>
<td><strong>Event Mgr</strong></td>
<td>Entry/Exit Monitor</td>
</tr>
<tr>
<td><strong>Site Status Mgr</strong></td>
<td>Event Monitor</td>
</tr>
<tr>
<td><strong>Site RDBMS</strong></td>
<td>Site Comms I/F</td>
</tr>
<tr>
<td><strong>CMS RDBMS</strong></td>
<td>Event Log</td>
</tr>
<tr>
<td><strong>Video File</strong></td>
<td>Customer I/F</td>
</tr>
<tr>
<td><strong>CMS Database</strong></td>
<td>System Status</td>
</tr>
<tr>
<td><strong>Site Database</strong></td>
<td>Customer Output Mgr</td>
</tr>
<tr>
<td><strong>Optical Sensor</strong></td>
<td>Fault Mgr</td>
</tr>
<tr>
<td><strong>DSL Modem</strong></td>
<td>Alarm Generator</td>
</tr>
<tr>
<td><strong>User Console</strong></td>
<td>Alarm I/F</td>
</tr>
<tr>
<td><strong>Video Camera</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Alarm</strong></td>
<td></td>
</tr>
</tbody>
</table>
ESS Parametric Diagram
To Support Trade-off Analysis

\[ CE = \text{Sum}(w_1 \times \text{OA} + w_2 \times \text{MRT} + w_3 \times \text{OC}) \]
**Entry/Exit Test Case**

**sd Entry/Exit Detection Test**

**Description**

- **testComponent** : IntruderEmulator

- **sut** : hardware
  - Door[1] : Optical Sensor
  - Site Processor : hardware
  - DSL Modem : hardware

**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

**seq**

- IntruderEntry : Alert Status
- SensedEntry

- Exit

- Intruder Exit : Alert Status
- SensedExit
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

- **Interchange & Metamodelling Standards**
  - MOF
  - XMI
  - STEP/AP233
## ISO/IEC 15288

### System Life Cycle Processes

#### Enterprise Processes
- **5.3.2** Enterprise Environment Management Process
- **5.3.3** Investment Management Process
- **5.3.4** System Life Cycle Processes Management
- **5.3.5** Quality Management Process
- **5.3.6** Resource Management Process

#### Project Processes
- **5.4.2** Project Planning Process
- **5.4.3** Project Assessment Process
- **5.4.4** Project Control Process
- **5.4.5** Decision-Making Process
- **5.4.6** Risk Management Process
- **5.4.7** Configuration Management Process
- **5.4.8** Information Management Process

#### Technical Processes
- **5.5.2** Stakeholder Reqs Definition Process
- **5.5.3** Reqs Analysis Process
- **5.5.4** Architectural Design Process
- **5.5.5** Implementation Process
- **5.5.6** Integration Process
- **5.5.7** Verification Process
- **5.5.8** Transition Process
- **5.5.9** Validation Process
- **5.5.10** Operation Process
- **5.5.11** Maintenance Process
- **5.5.12** Disposal Process

#### Agreement Processes
- **5.2.2** Acquisition Process
- **5.2.3** Supply Process
Standards-based Tool Integration with SysML

Systems Modeling Tool

Model/ Data Interchange

Other Engineering Tools

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 (Tau and Rhapsody)
- TopCased
- Visio SysML template
Transitioning to SysML
Using Process Improvement
To Transition to SysML

Plan Improvement

Assess & Measure Improvement

Continuous Improvement Cycle

Define 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

<table>
<thead>
<tr>
<th>CM/DM</th>
<th>Requirements Management</th>
<th>Verification &amp; Validation</th>
<th>Project Management</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product Data Management</td>
<td>SoS/ DoDAF / Business Process Modeling</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SoS/ DoDAF / Business Process Modeling</td>
<td>System Modeling</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UML 2.0</td>
<td>Hardware Modeling</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VHDL, CAD, ..</td>
<td>Engineering Analysis</td>
<td></td>
<td></td>
</tr>
<tr>
<td>System Modeling</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SysML</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software Modeling</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hardware Modeling</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VHDL, CAD, ..</td>
<td>Simulation &amp; Visualization</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Engineering Analysis</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Copyright © 2006-2009 by Object Management Group.
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
- **A Practical Guide to SysML** (Morgan Kaufmann) by Friedenthal, Moore, Steiner
- **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
- **OMG SysML Information Days Presentations (Dec 8-11, 2008)**

- **Artisan website**
  - Presenter details: Matthew.Hause@ArtisanSoftwareTools.com
References

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)
Class Exercise
Dishwasher Example
Sample Artifacts

Primary
• Requirement diagram
• Block definition diagram – top level
• Internal block diagram – dishwasher black box
• Use case diagram
• Activity diagram
• Block definition diagram – input/output definitions
• Block definition diagram – dishwasher hierarchy
• Internal block diagram – dishwasher white box

Optional
• Parametric diagram
• State machine diagram
• Sequence diagram