



# OMG Systems Modeling Language (OMG SysML™) Tutorial

25 June 2007

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

- Specification status
  - Adopted by OMG in May '06
  - Finalization Task Force Report in March '07
  - Available Specification v1.0 expected June '07
  - Revision task force chartered for SysML v1.1 in March '07
- This tutorial is based on the OMG SysML adopted specification (ad-06-03-01) and changes proposed by the Finalization Task Force (ptc/07-03-03)
- 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:**

- Benefits of model driven approaches for systems engineering
- 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

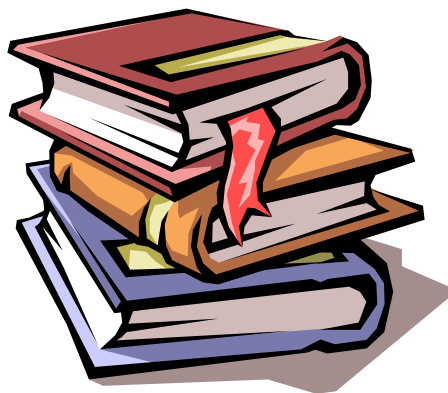
- Motivation & Background
- Diagram Overview and Language Concepts
- SysML Modeling as Part of SE Process
  - Structured Analysis – Distiller Example
  - OOSEM – Enhanced Security System Example
- SysML in a Standards Framework
- Transitioning to SysML
- Summary



## Motivation & Background

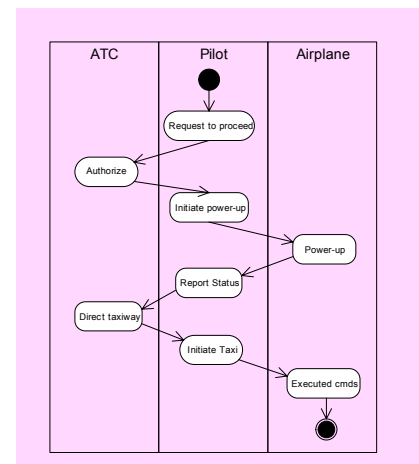
# SE Practices for Describing Systems

**Past**



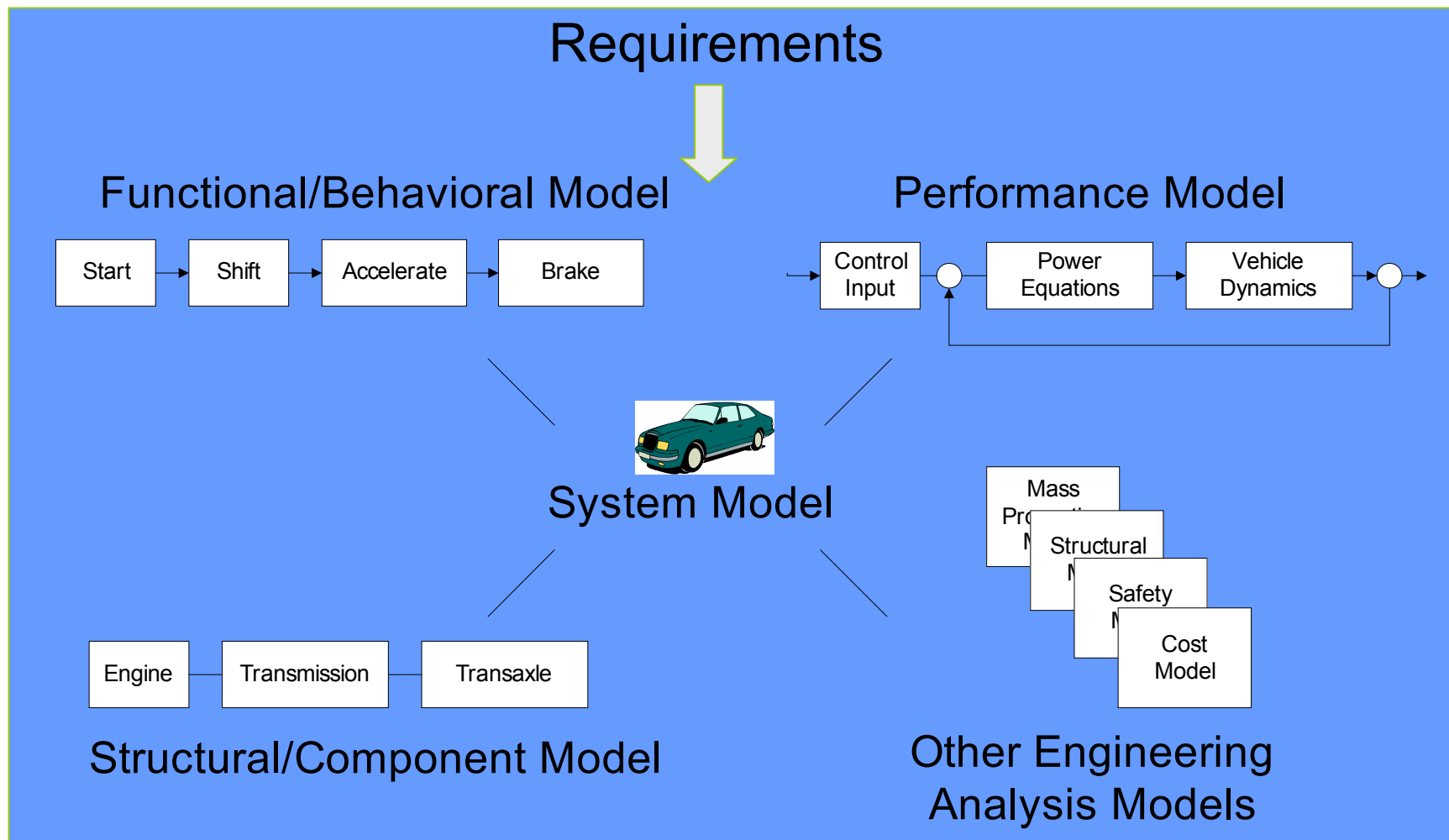
- Specifications
- Interface requirements
- System design
- Analysis & Trade-off
- Test plans

**Future**



**Moving from Document centric to Model centric**

# System Modeling



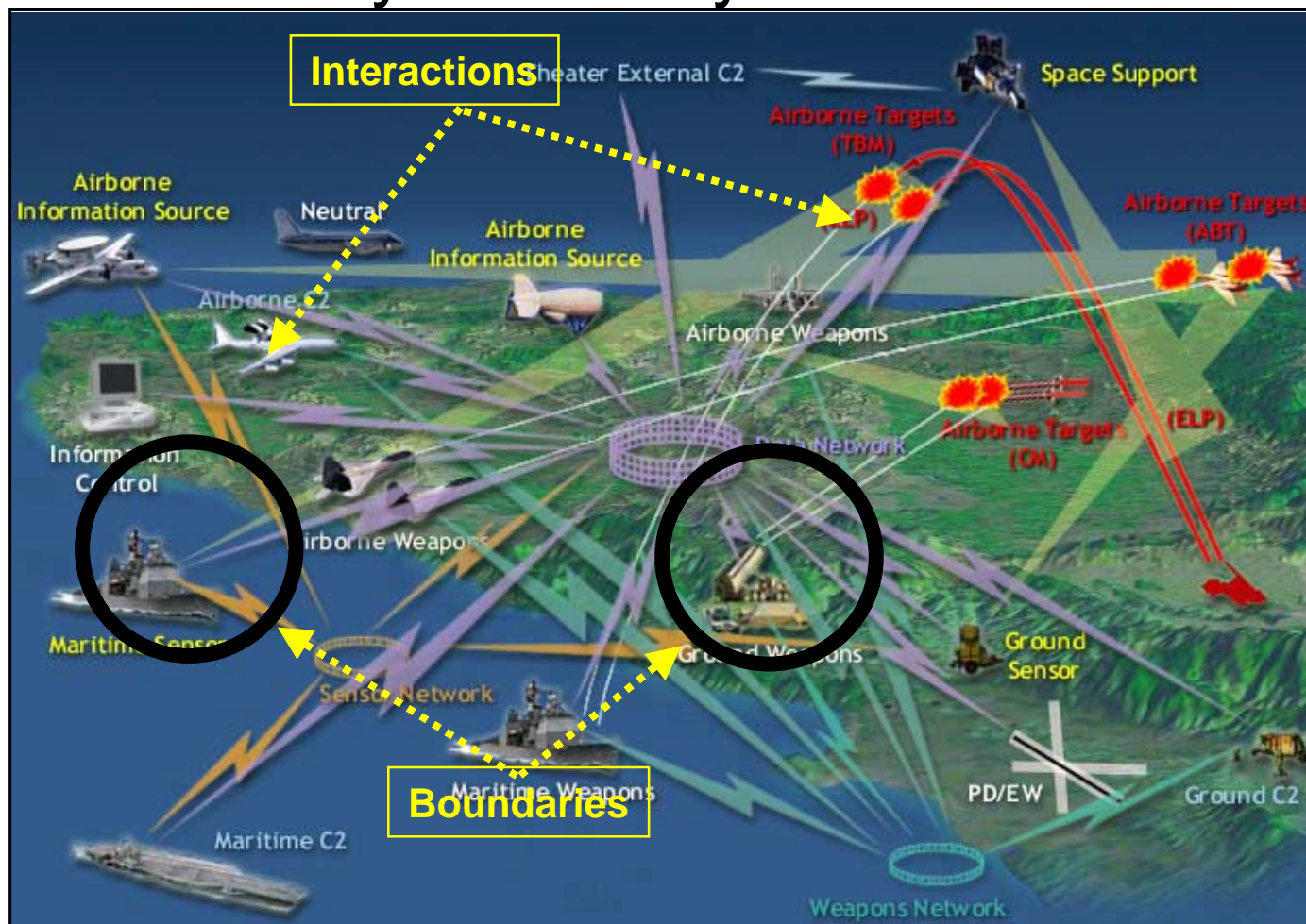
**Integrated System Model Must Address Multiple Aspects of a System**

# 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

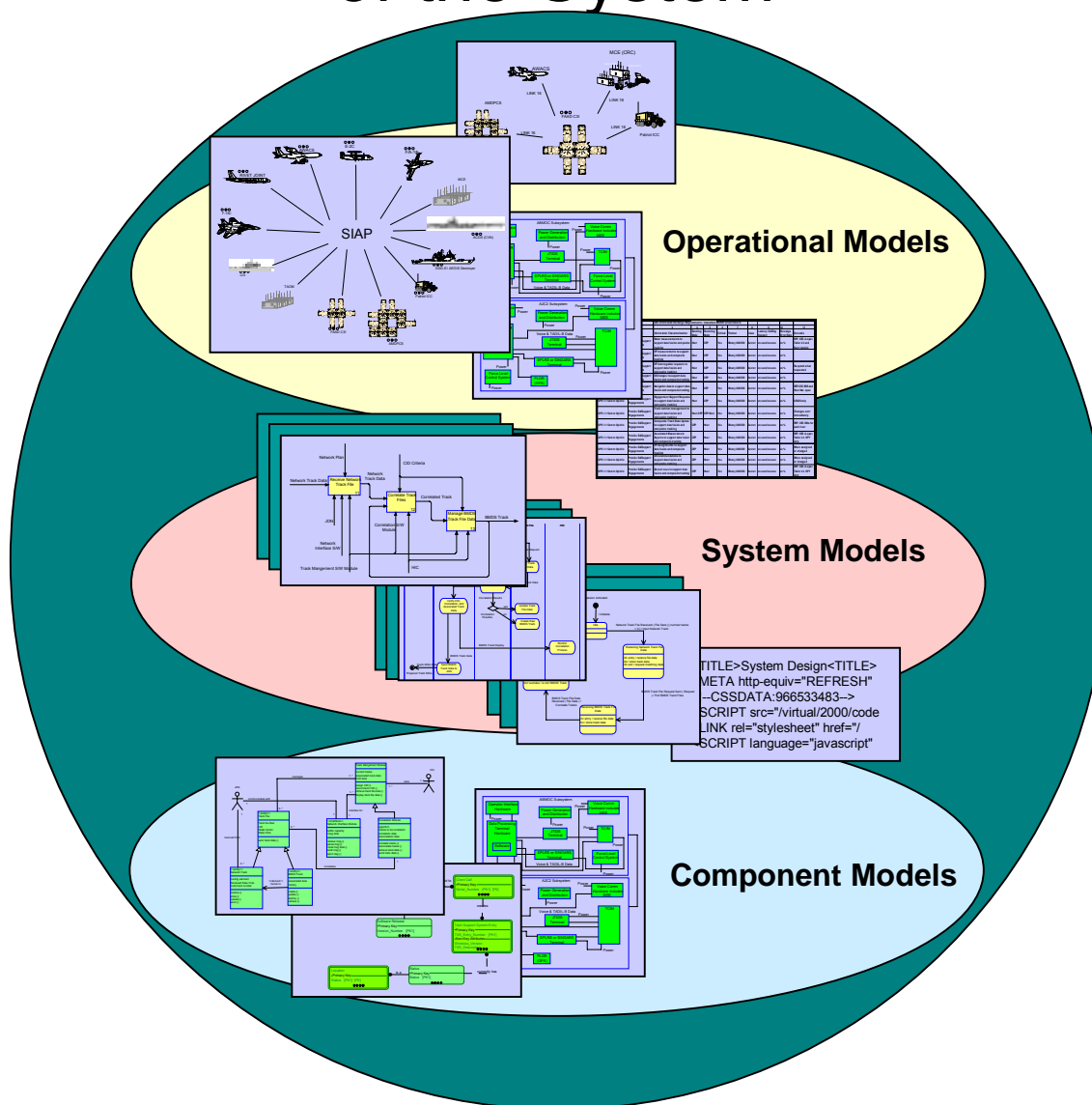


# System-of-Systems

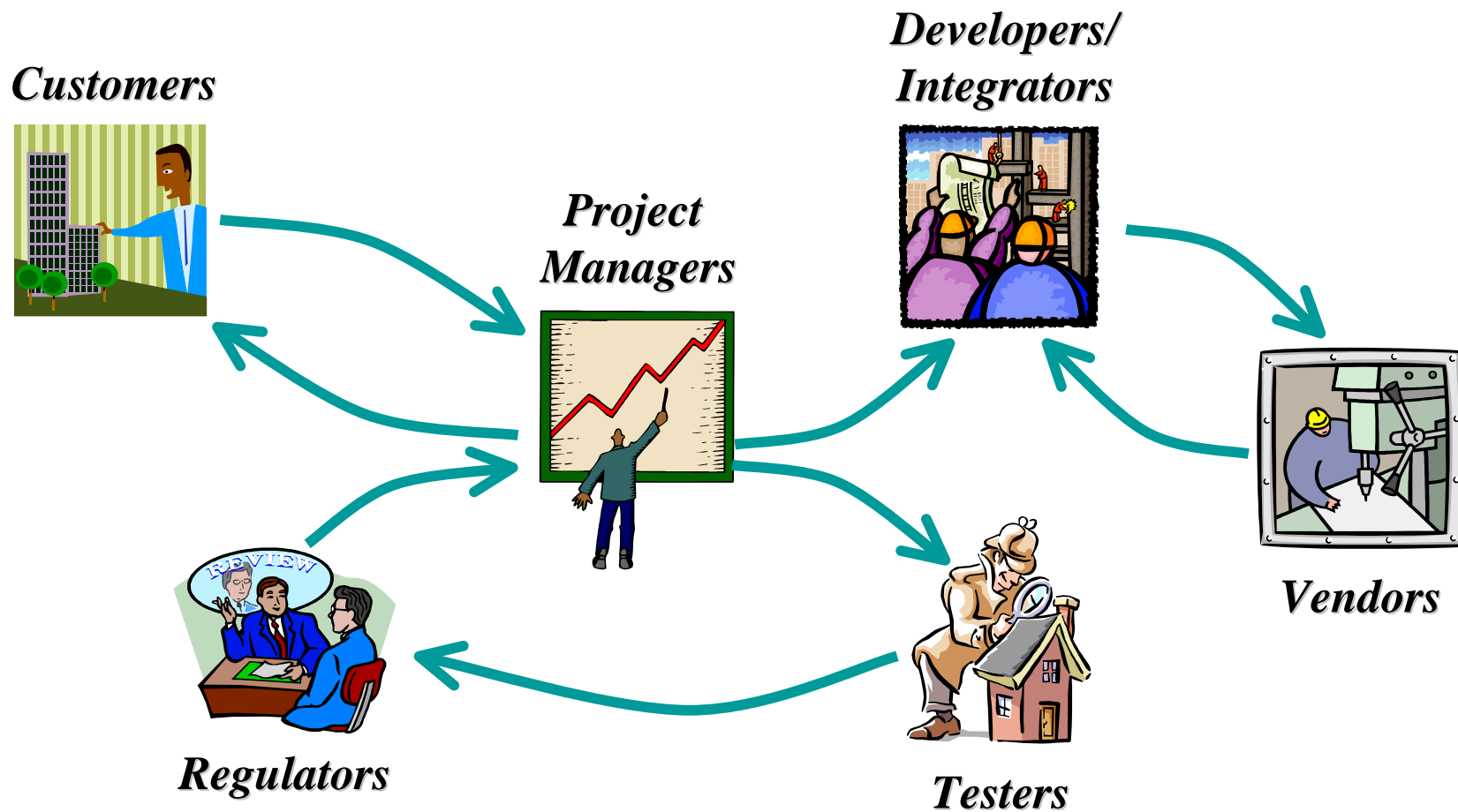


**Modeling Needed to Manage System Complexity**

# Modeling at Multiple Levels of the System



# Stakeholders Involved in System Acquisition



**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
  - Current version (formal/07-02-05)
- 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

# SysML Participants

- Industry & Government
  - American Systems, BAE SYSTEMS, Boeing, Deere & Company, EADS-Astrium, Eurostep, Lockheed Martin, Motorola, NIST, Northrop Grumman, oose.de, Raytheon, THALES
- Vendors
  - Artisan, EmbeddedPlus, Gentleware, IBM, I-Logix, Mentor Graphics, No Magic, PivotPoint Technology, Sparx Systems, Telelogic, Vitech Corp,
- Academia
  - Georgia Institute of Technology
- Liaison Organizations
  - INCOSE, ISO AP233 Working Group

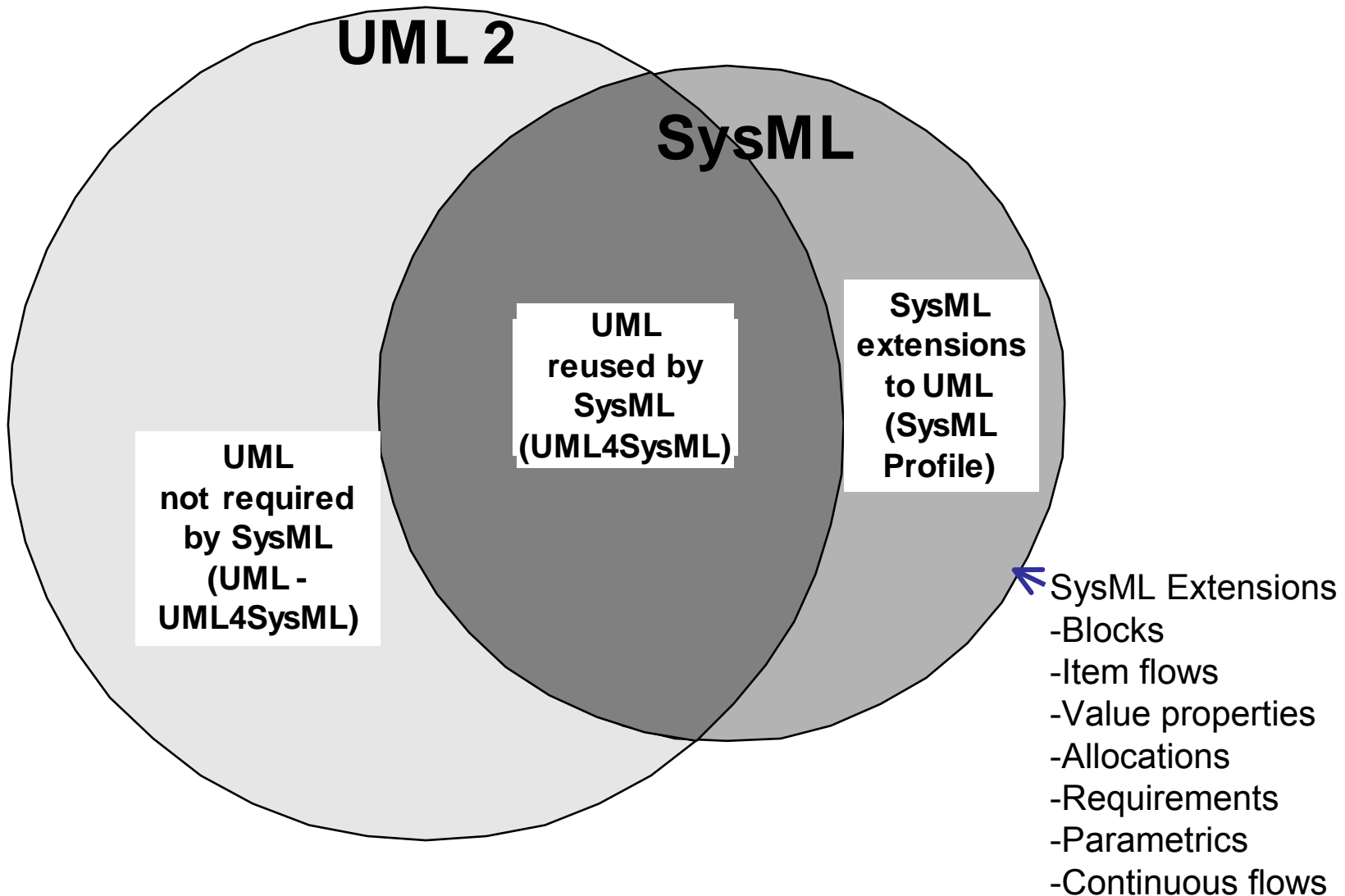




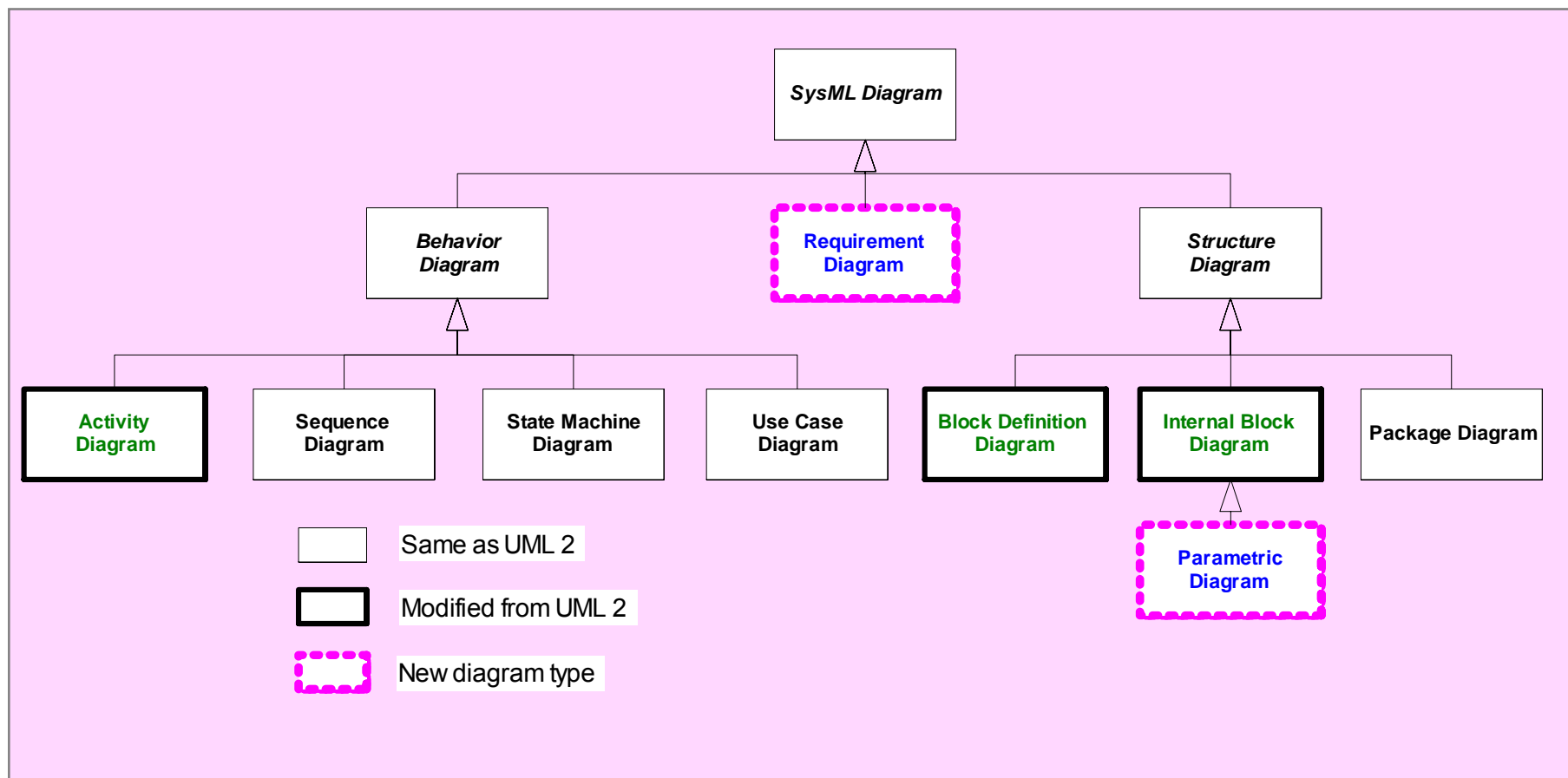
## Diagram Overview & Language Concepts



# Relationship Between SysML and UML

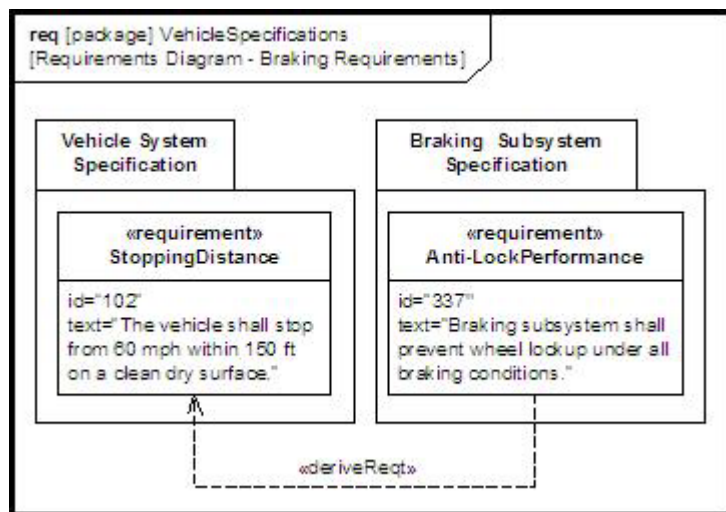
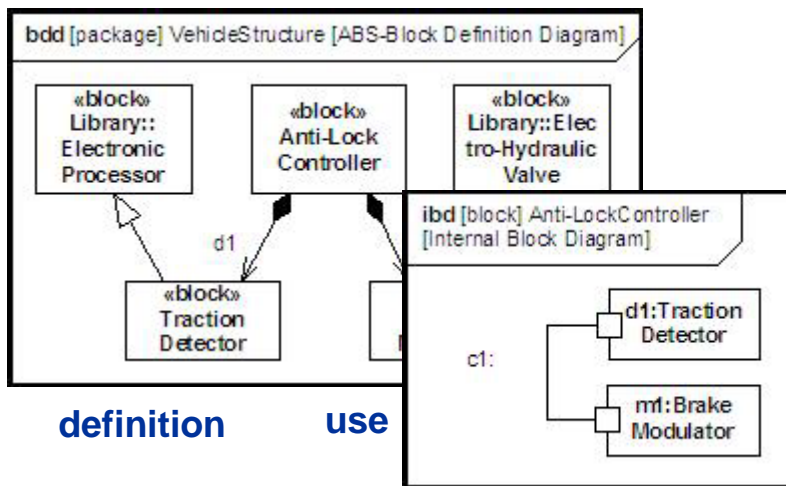


# SysML Diagram Taxonomy



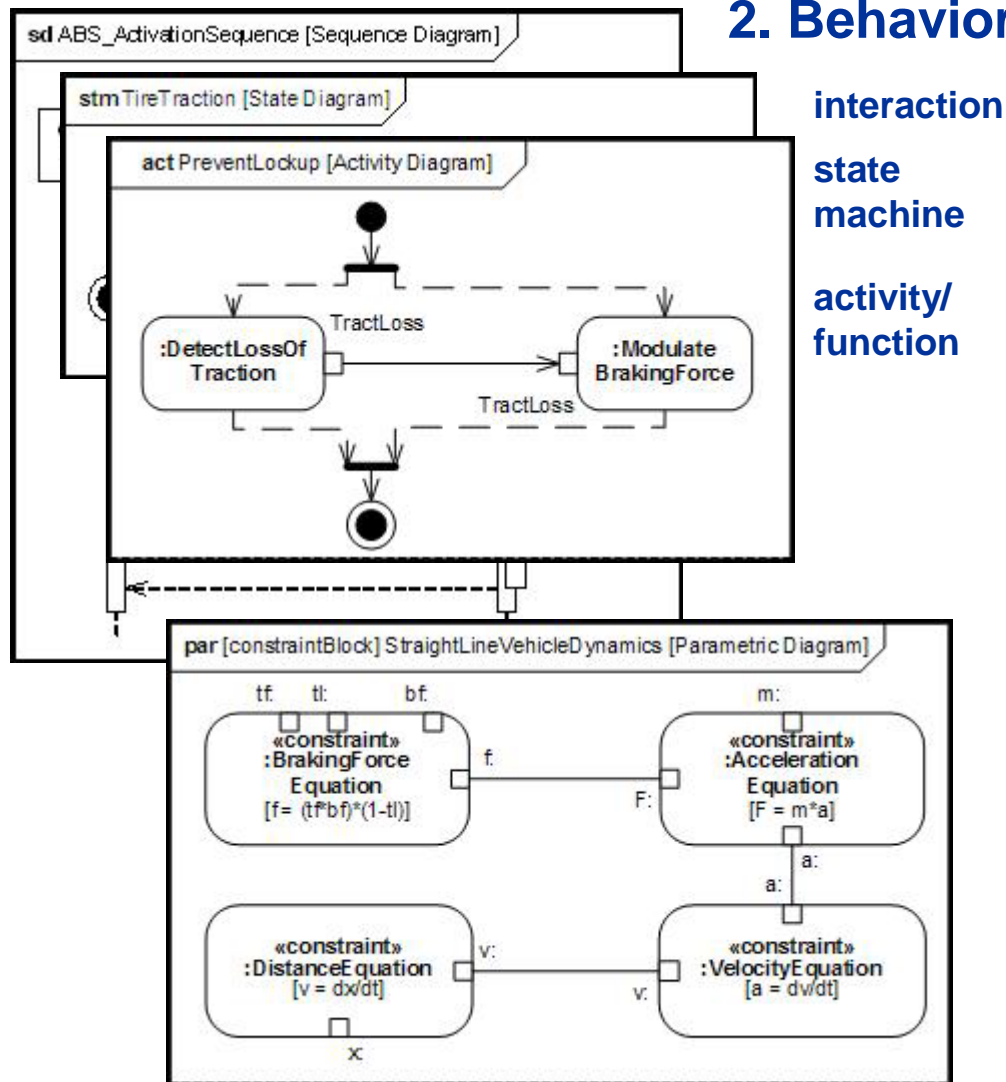
# 4 Pillars of SysML – ABS Example

## 1. Structure



## 3. Requirements

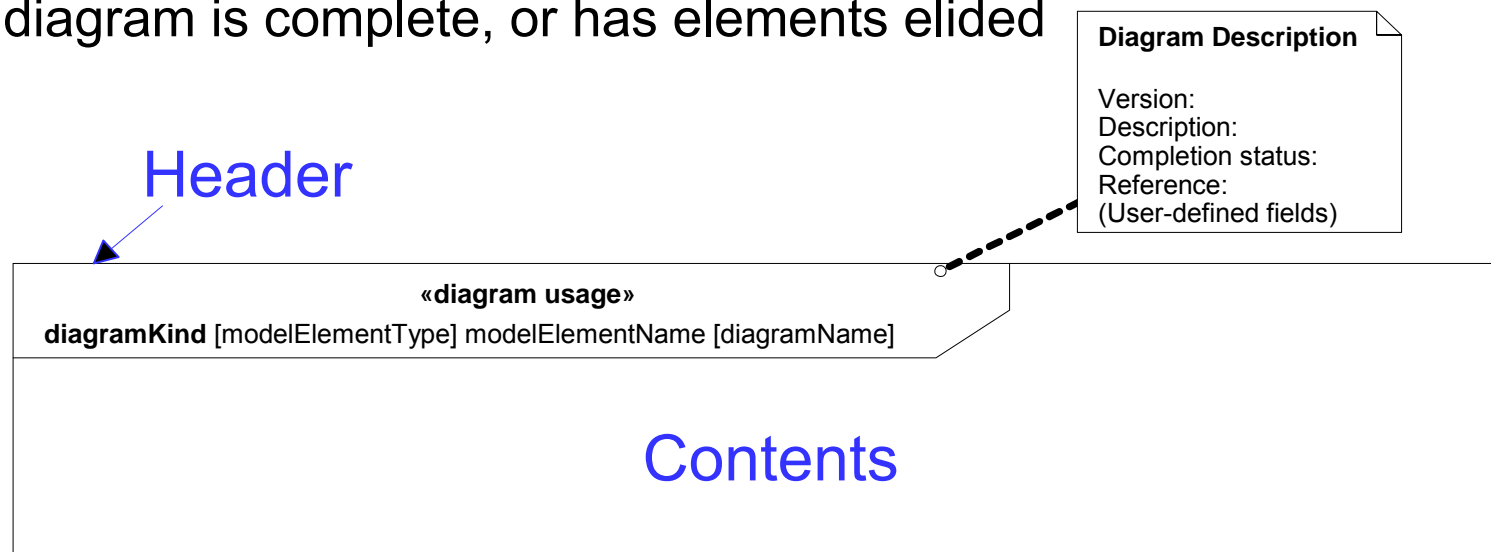
## 2. Behavior



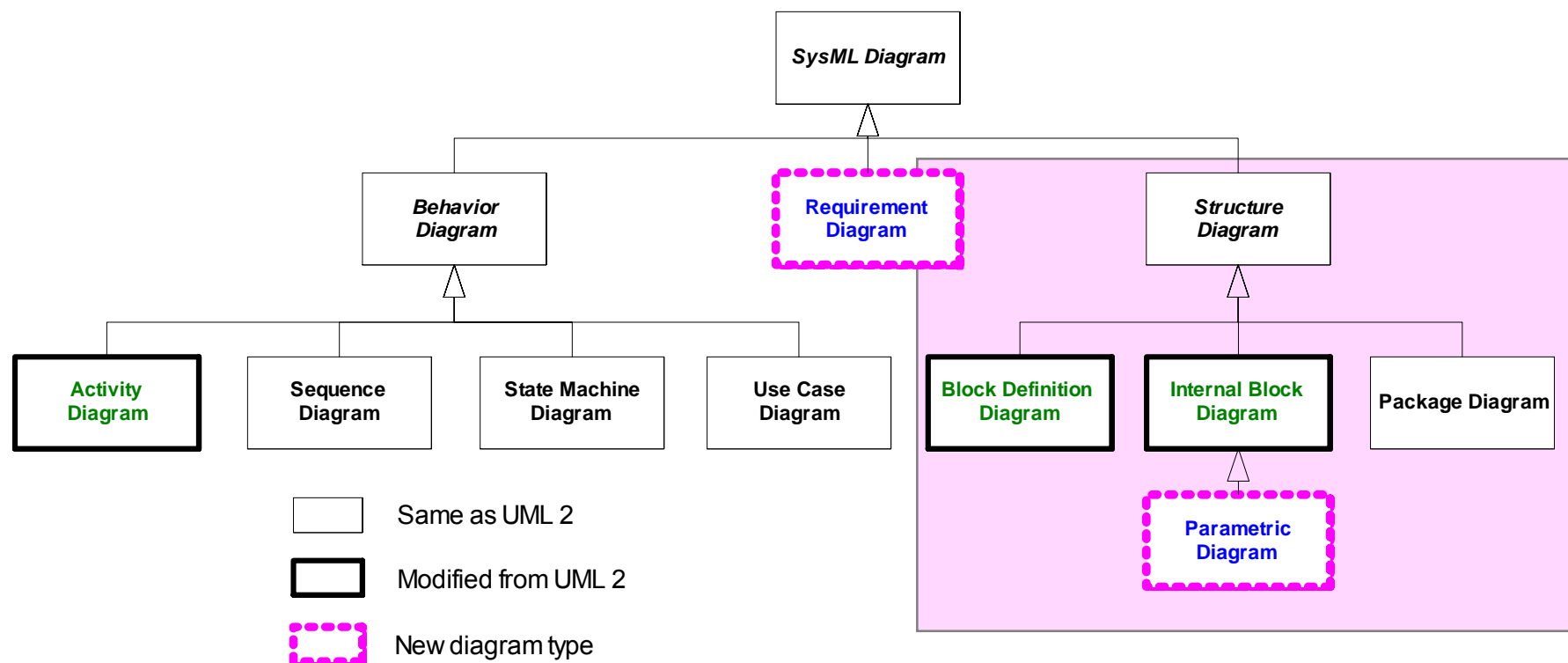
## 4. Parametrics

# 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 (activity, block, interaction, 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



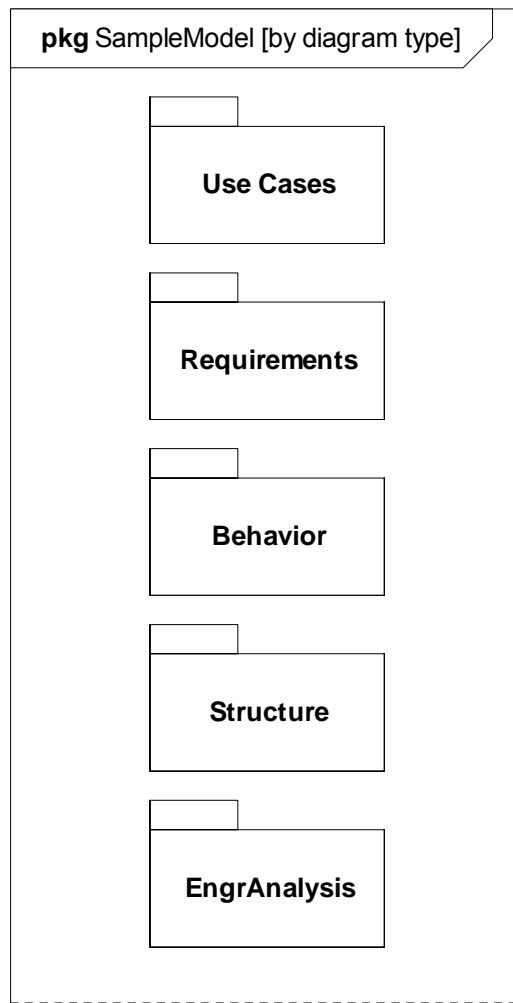
# Structural Diagrams



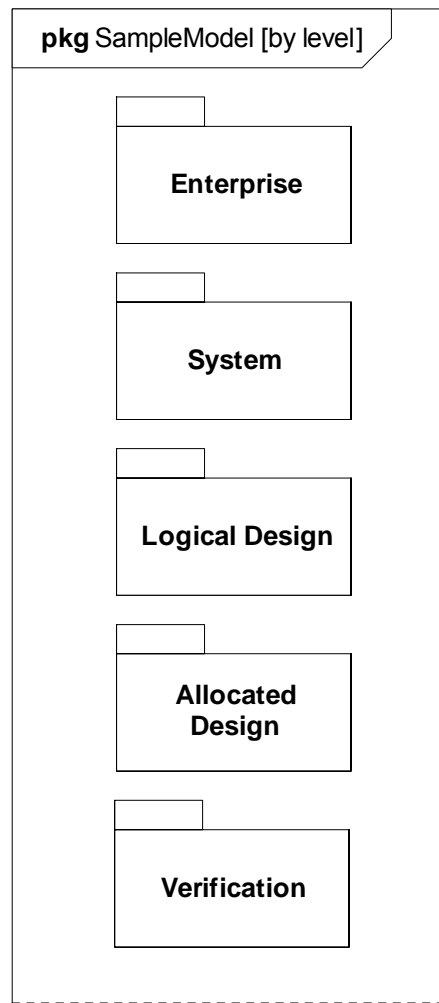
# Package Diagram

- 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 domain (e.g., requirements, use cases, behavior)
  - Use viewpoints to augment model organization
- Import relationship reduces need for fully qualified name (package1::class1)

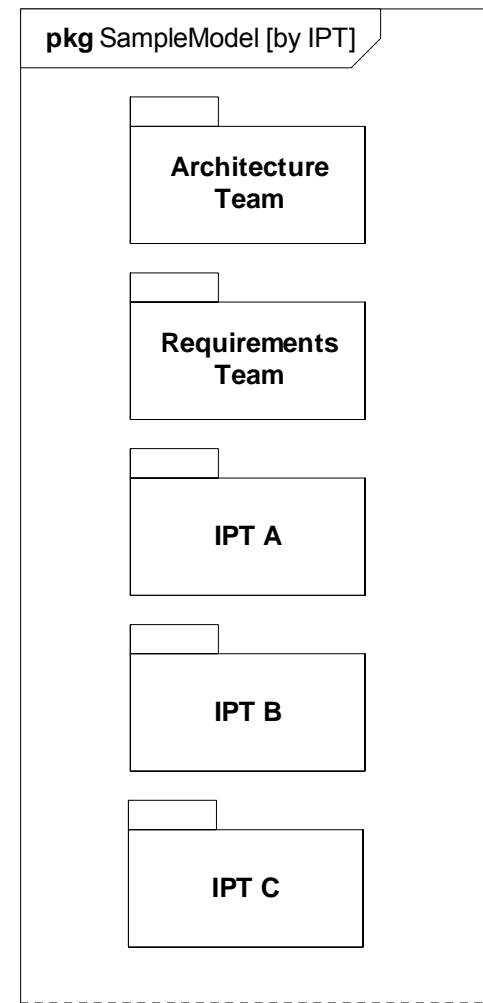
# Package Diagram Organizing the Model



By Diagram Type

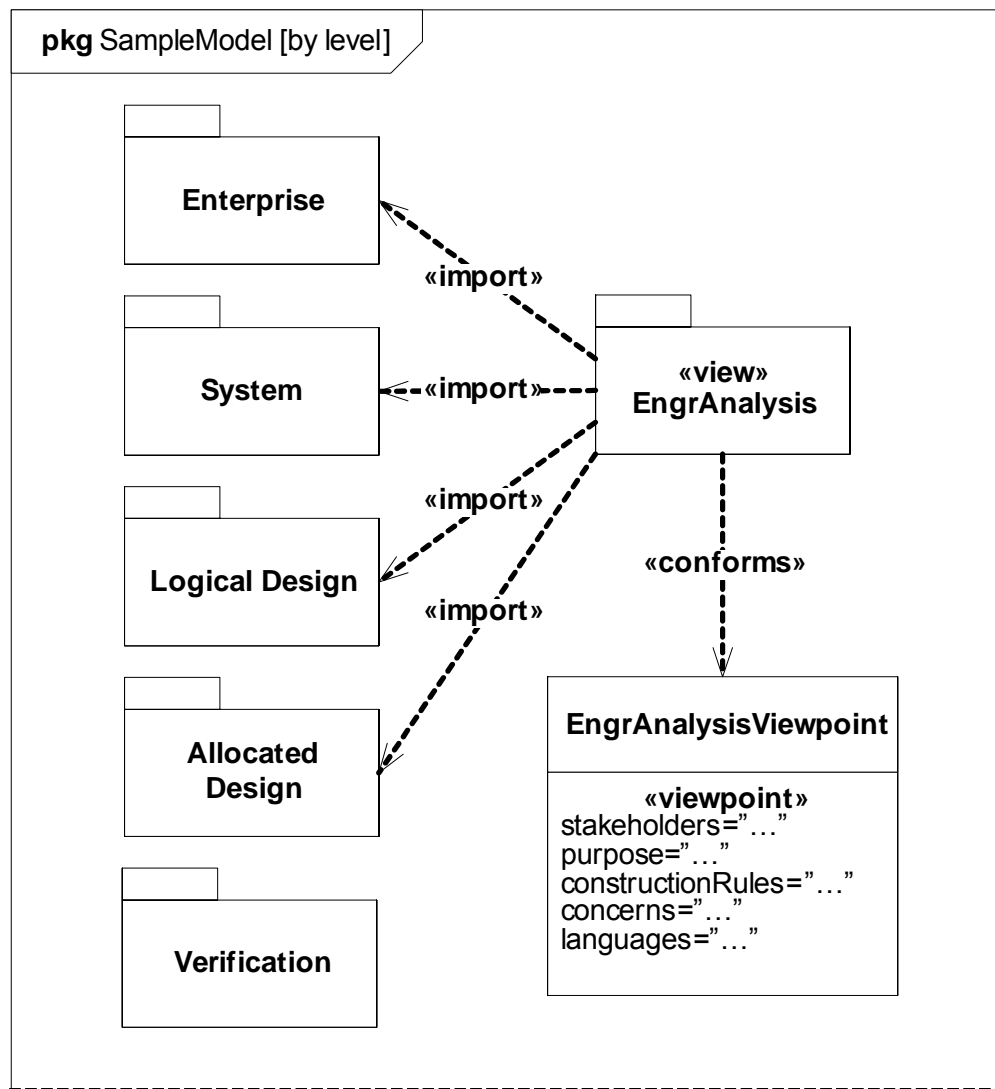


By Hierarchy



By IPT

# Package Diagram - Views



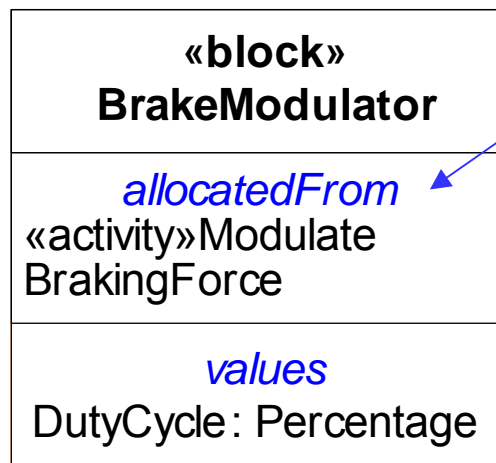
- Viewpoint represents the stakeholder perspective
- View conforms to a particular viewpoint
  - Imports model elements from multiple packages
  - Can represent a model query based on query criteria
- View and Viewpoint consistent with IEEE 1471 definitions



# 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



Compartment  
Label

- 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 - 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

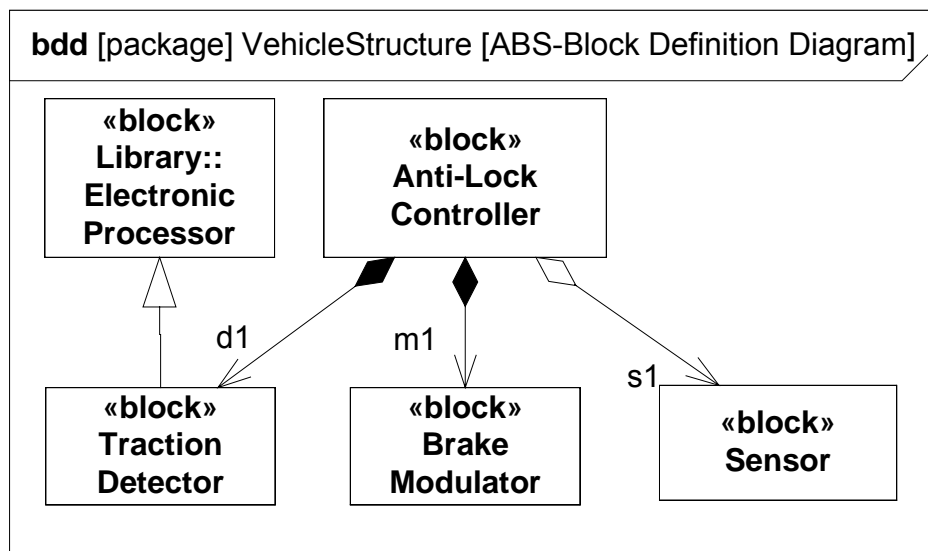
# Using Blocks

- Based on UML Class from UML Composite Structure
  - Supports unique features (e.g., flow ports, value properties)
- 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

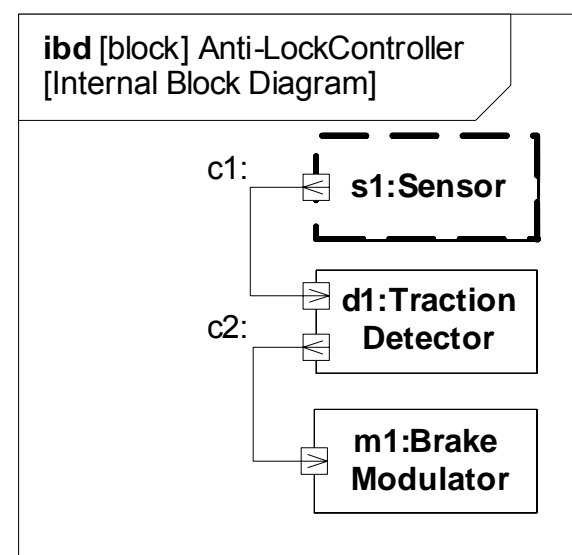
## Block Definition Diagram



## Definition

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

## Internal Block Diagram

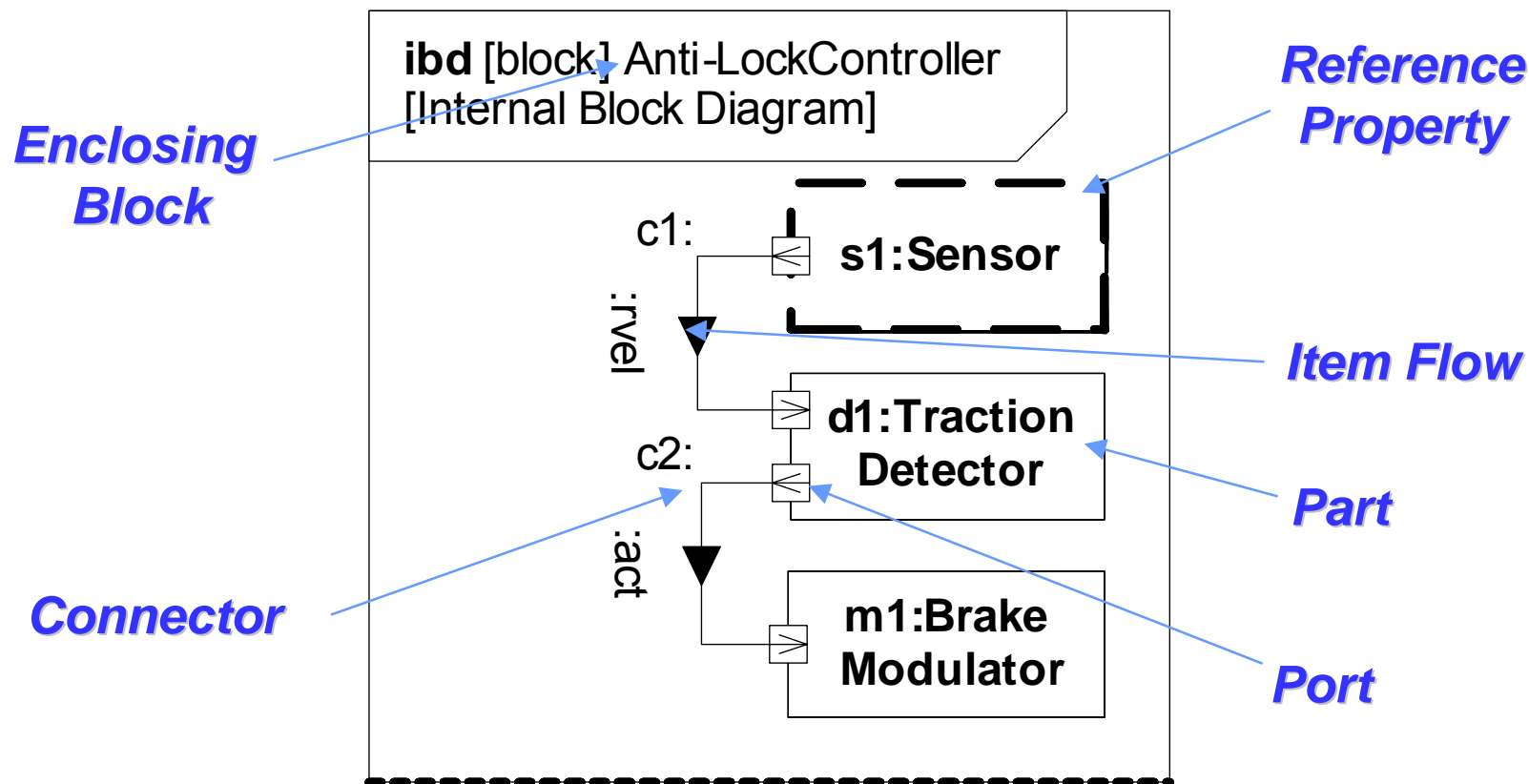


## Usage

- Part is the usage in a particular context
- Typed by a block
- Also known as a role

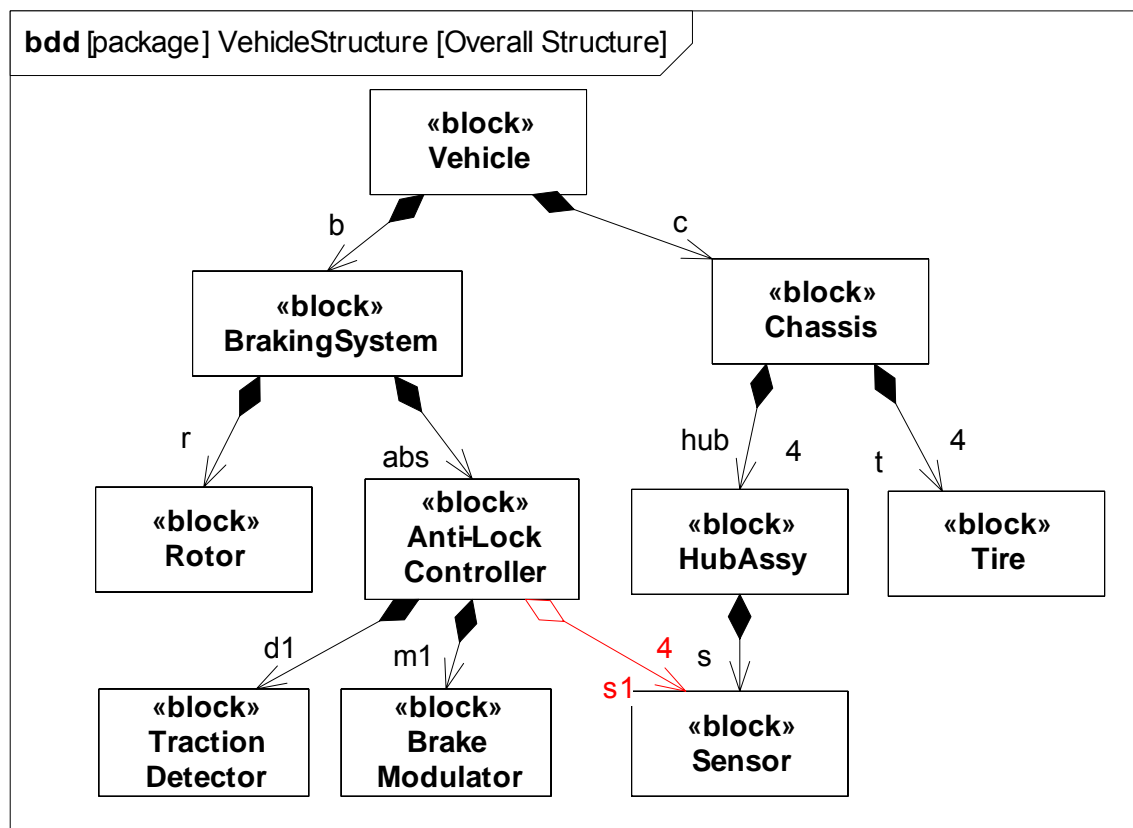
# Internal Block Diagram (ibd)

## Blocks, Parts, Ports, Connectors & Flows

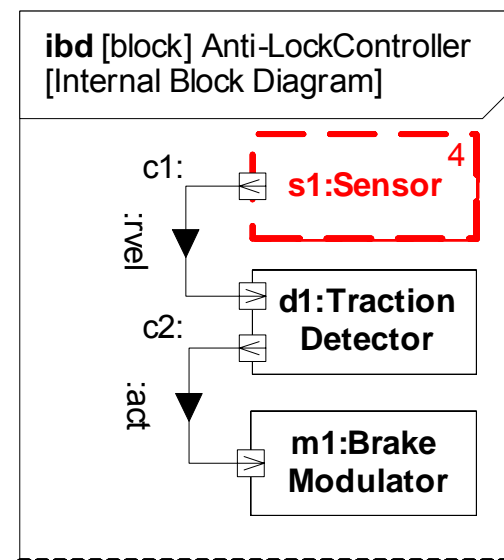


**Internal Block Diagram Specifies Interconnection of Parts**

# Reference Property Explained



**S1 is a reference part\* in ibd shown in dashed outline box**



\*Actual name is reference property

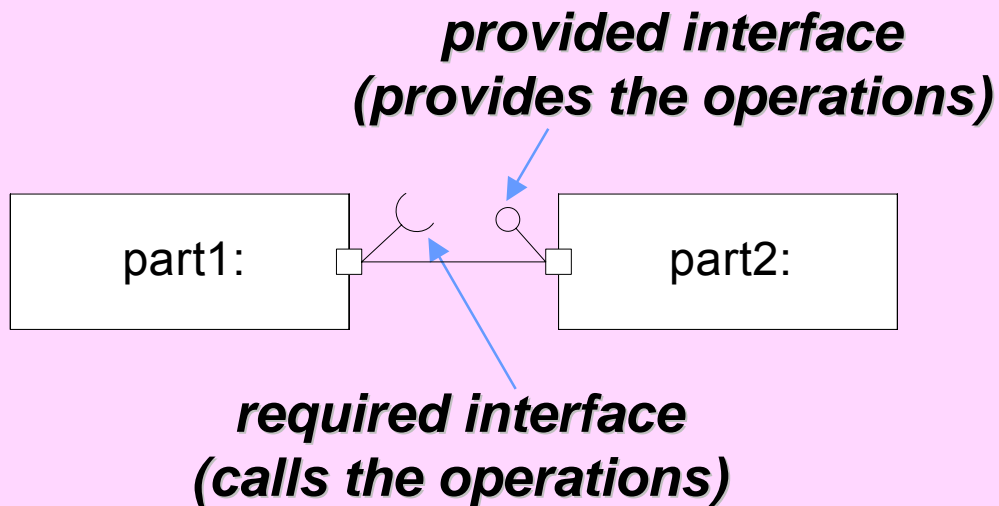
# SysML Ports

- 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 flow specification
    - Atomic, non-atomic, and conjugate variations

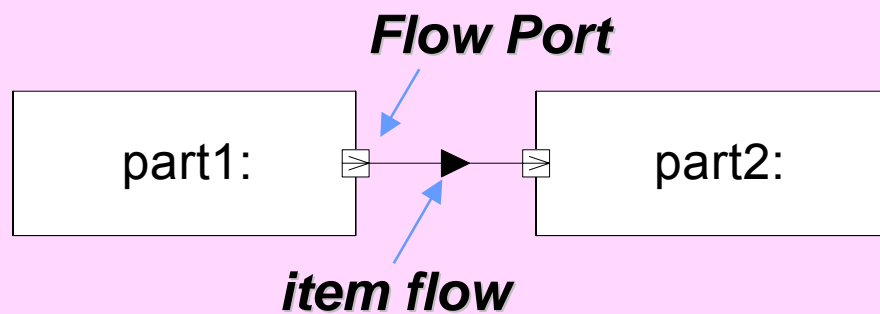
**Standard Port and Flow Port  
Support Different Interface Concepts**

# Port Notation

## **Standard Port**



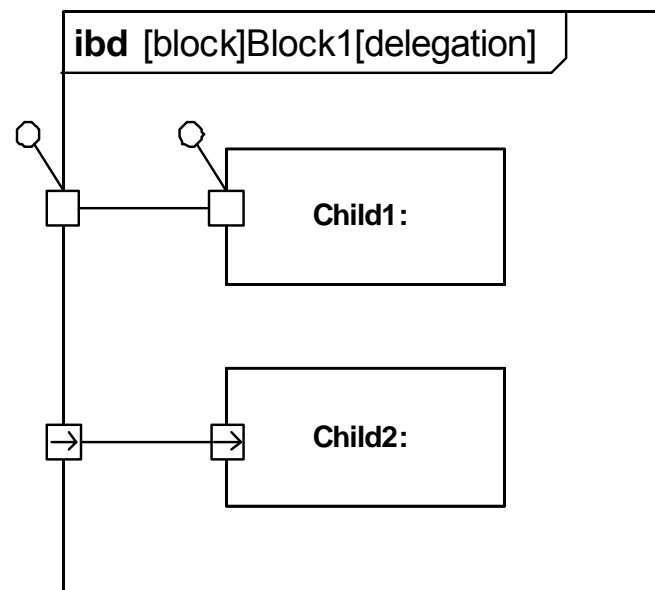
## **Flow Port**





# 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 hierarchy (e.g. tire to road)

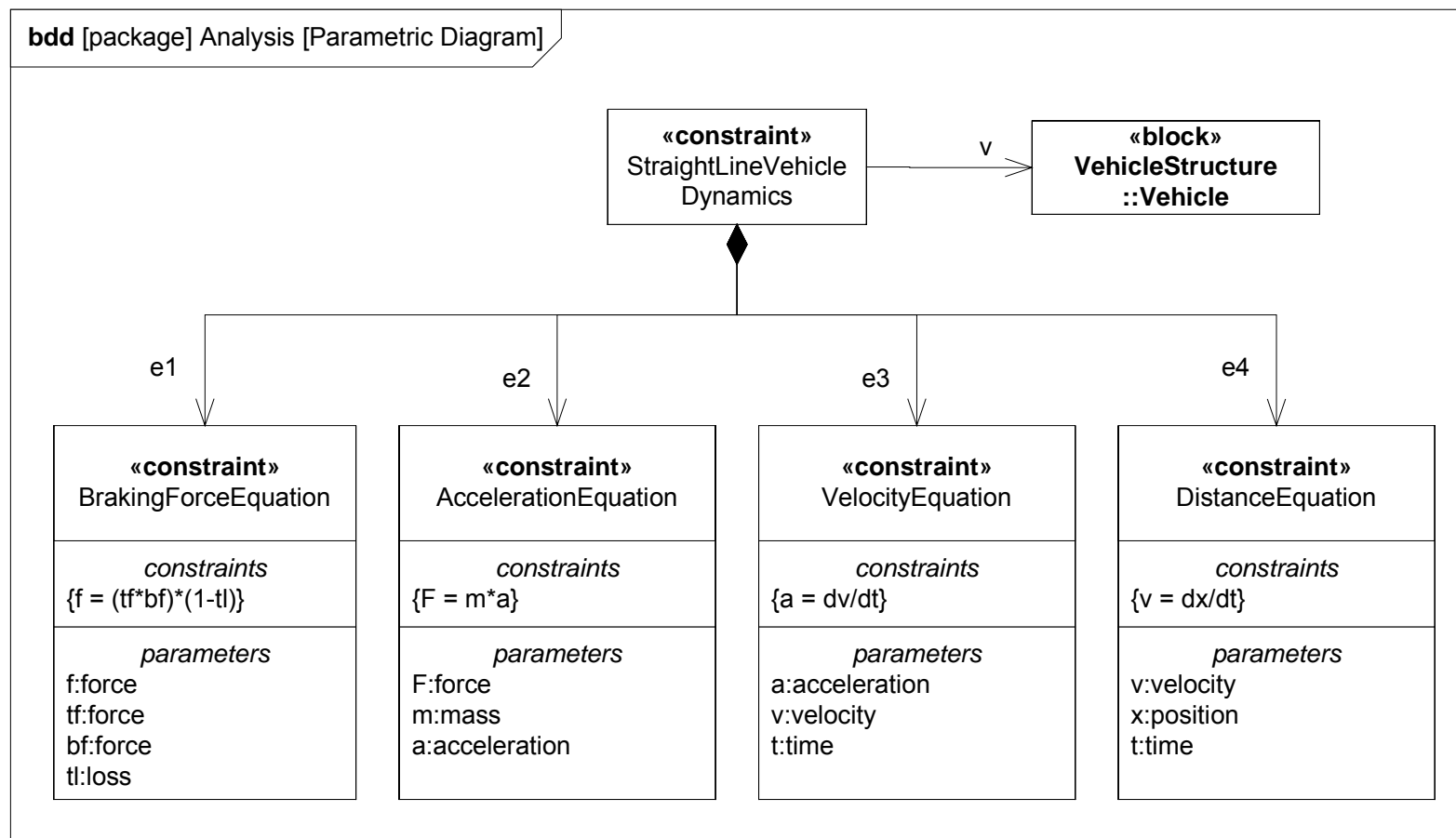


# Parametrics

- 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 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  $F = m \times a$ )

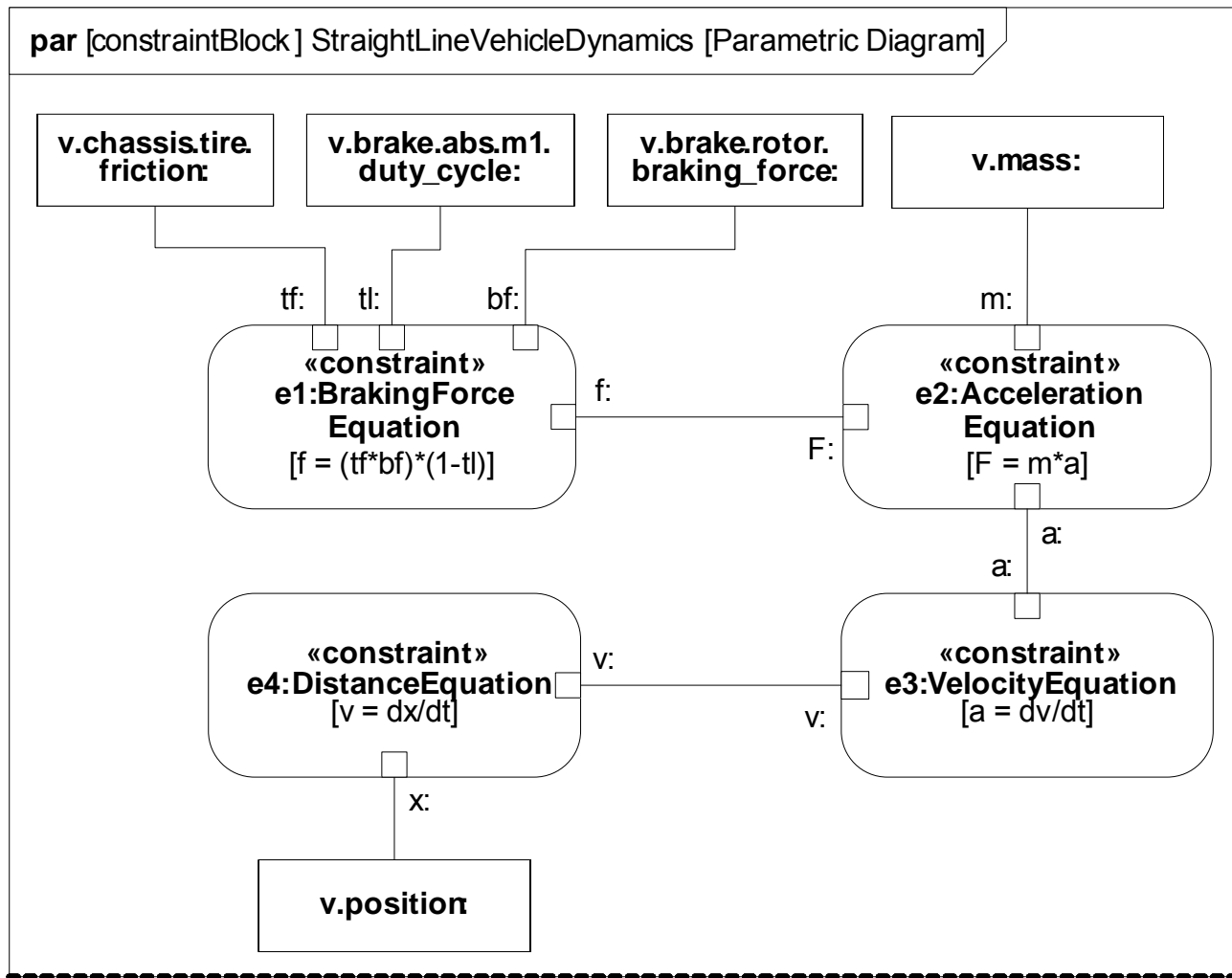
**Parametrics Enable Integration of Engineering Analysis with Design Models**

# Defining Vehicle Dynamics



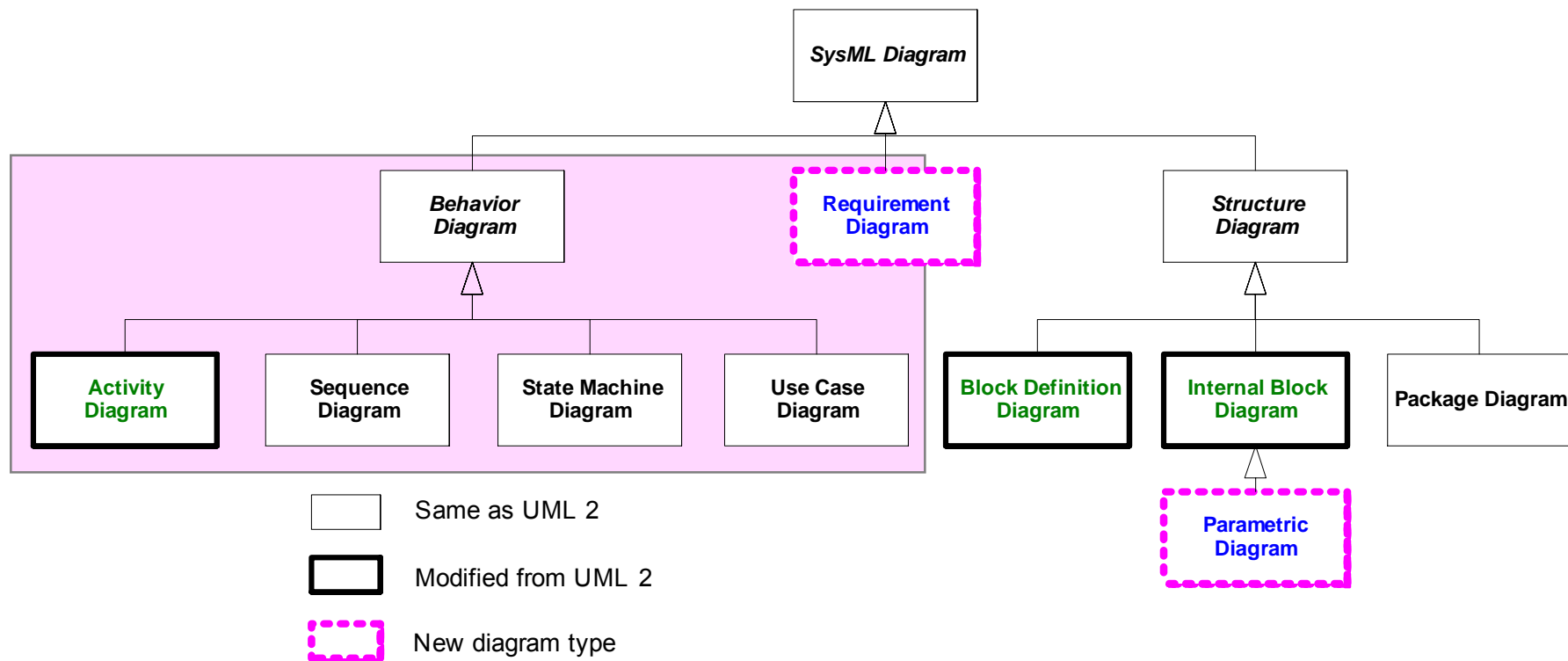
## Defining Reusable Equations for Parametrics

# Vehicle Dynamics Analysis



Using the Equations in a Parametric Diagram to Constrain Value Properties

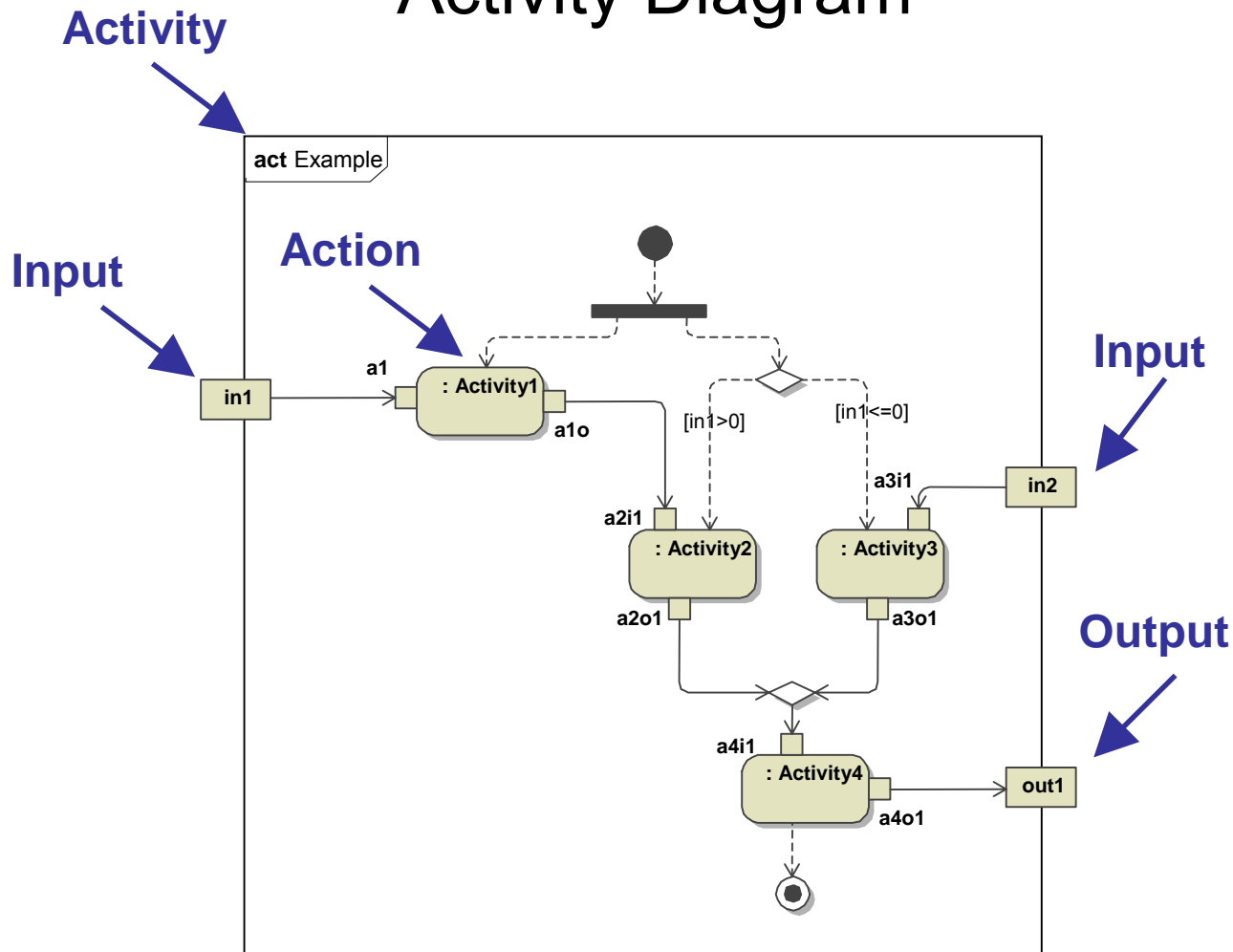
# Behavioral Diagrams



# Activities

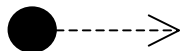
- Activity used to specify the flow of inputs/outputs and control, including sequence and conditions for coordinating activities
- Secondary constructs show responsibilities for the activities using swim lanes
- SysML extensions to Activities
  - Support for continuous flow modeling
  - Alignment of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

# Activity Diagram

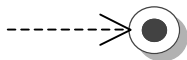


Activity Diagram Specifies Controlled Sequence of Actions

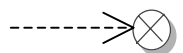
# Routing Flows



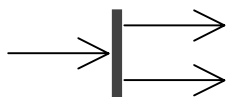
**Initial Node** – On execution of parent control token placed on outgoing control flows



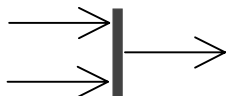
**Activity Final Node** – Receipt of a control token terminates parent



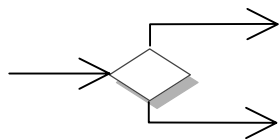
**Flow Final Node** – Sink for control tokens



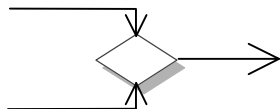
**Fork Node** – Duplicates input (control or object) tokens from its input flow onto all outgoing flows



**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



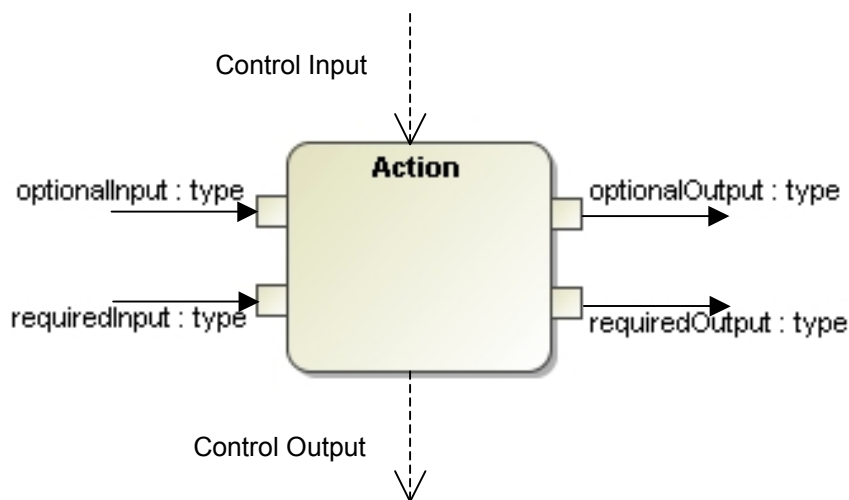
**Merge Node** – Waits for an input (control or object) token on any input flows and then places it on the outgoing flow

**Guard expressions can be applied on all flows**



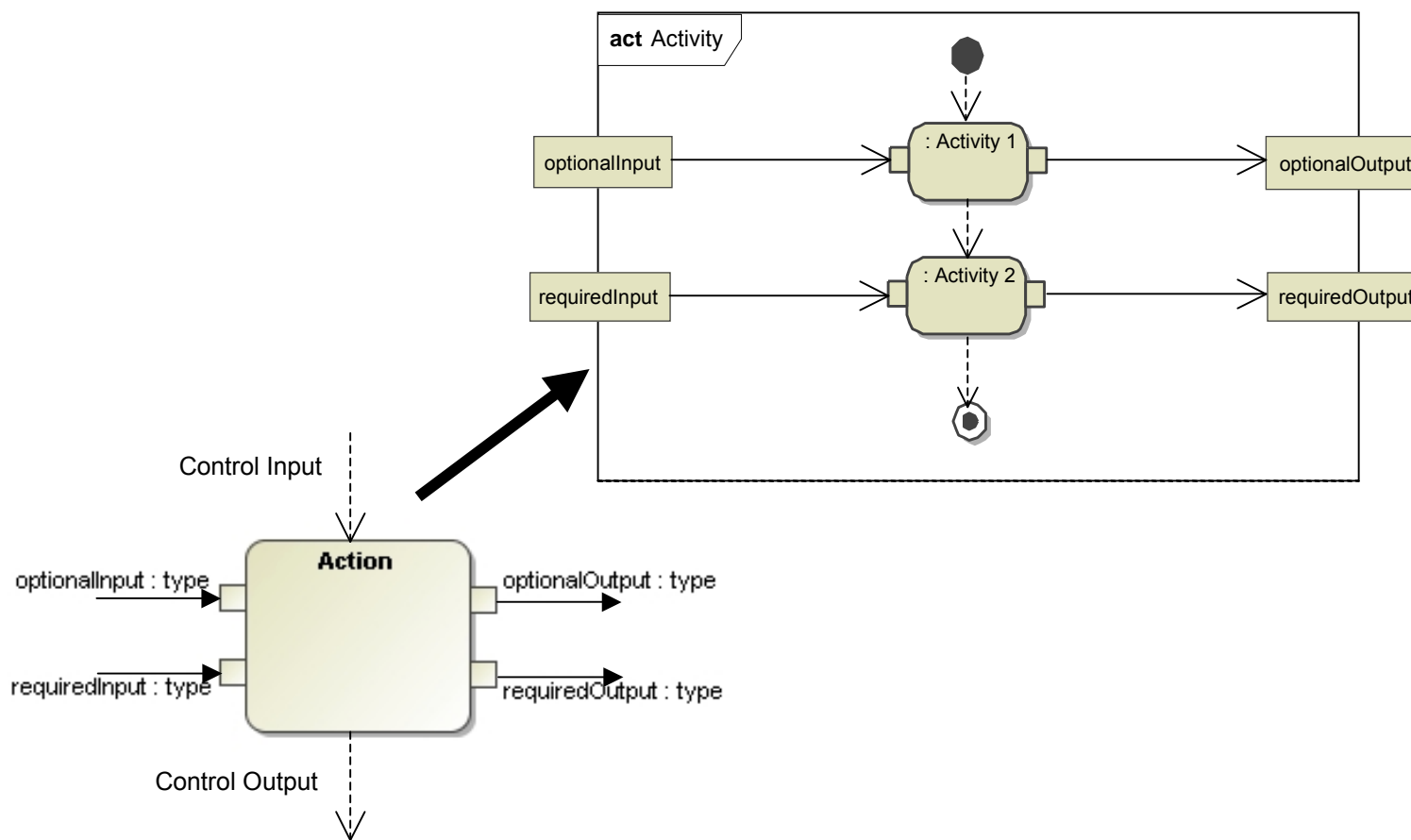
# Actions Process Flow of Control and Data

- **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

Note: The summary is an approximation of the semantics.

The detailed semantics are specified in the UML and SysML specification.

## 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

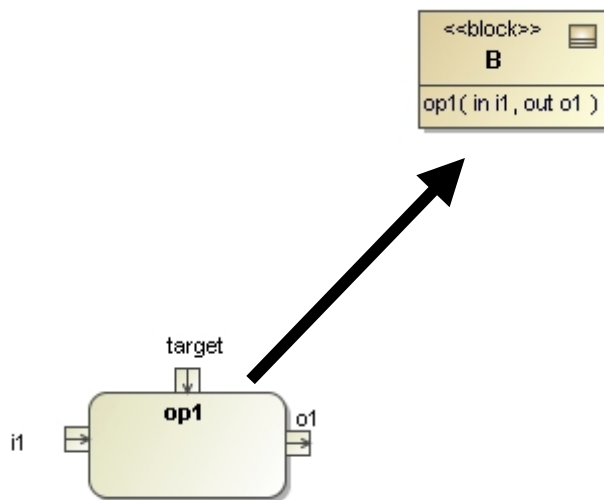
## Terminating an action:

- 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

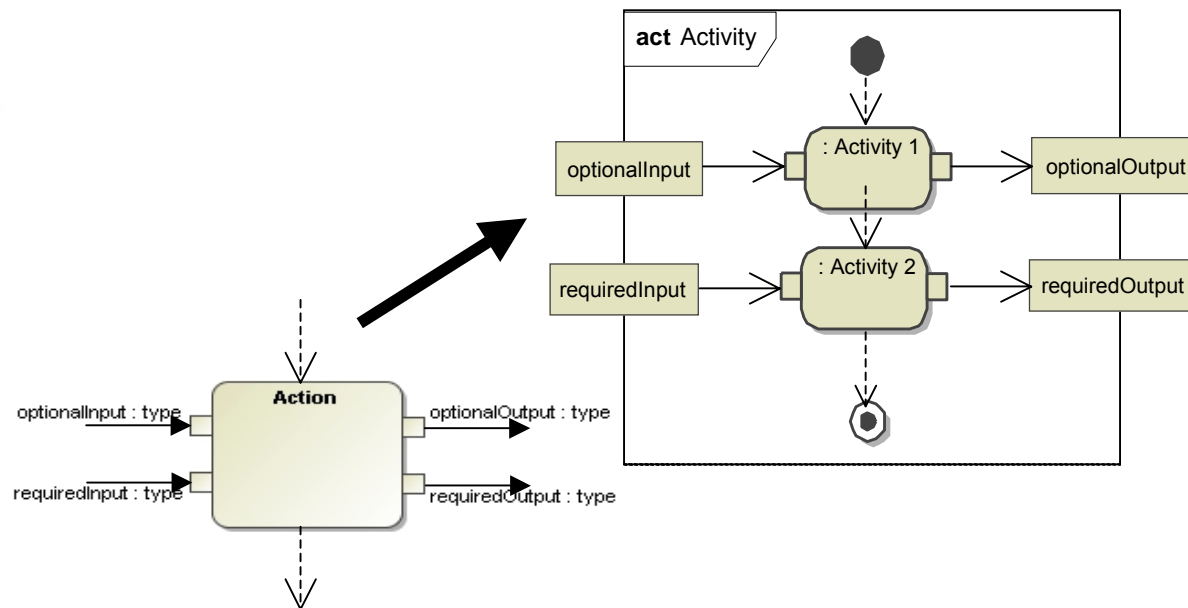
## Following action termination:

- 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

# Common Actions



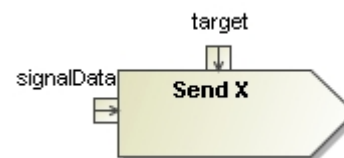
**Call Operation Action**  
(can call leaf level function)



**Call Behavior Action**

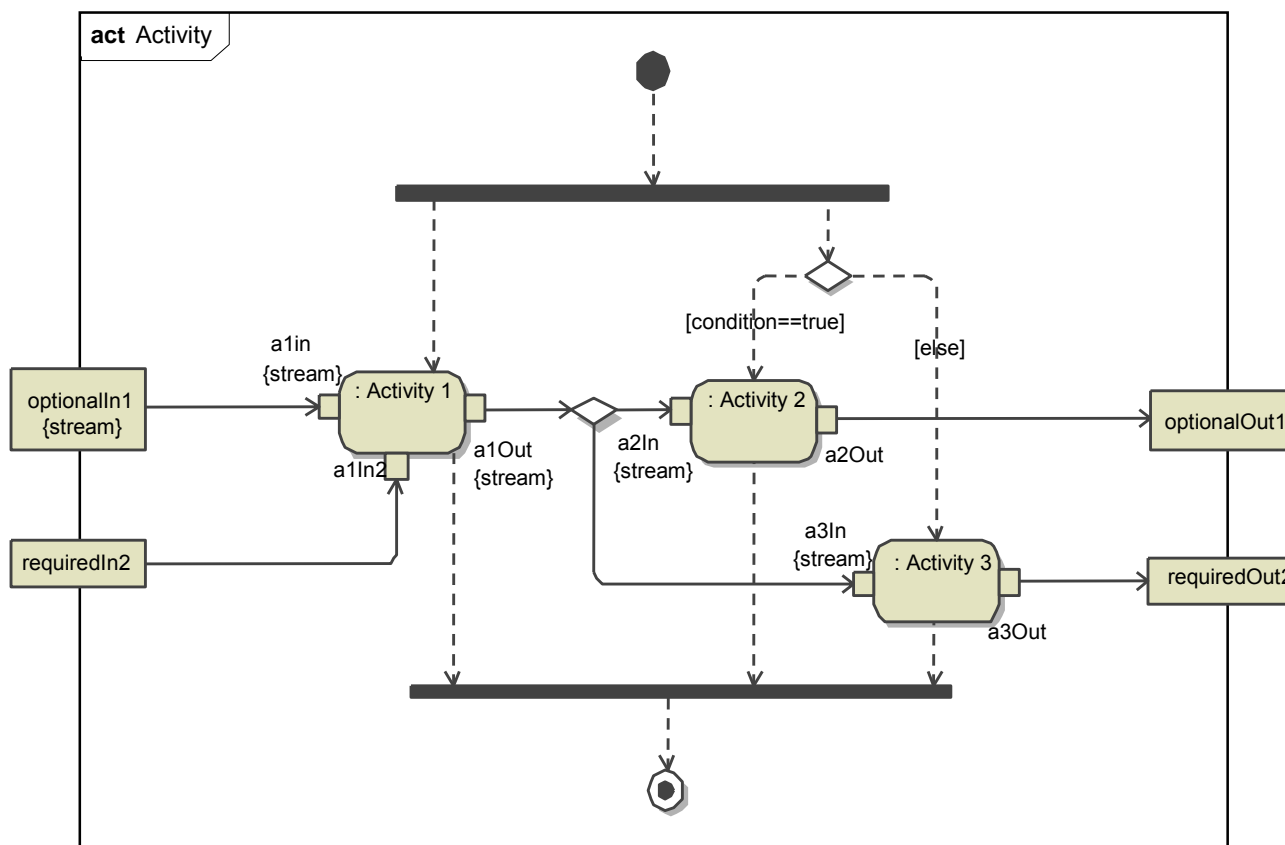


**Accept Event Action**  
(Event Data Pin often elided)

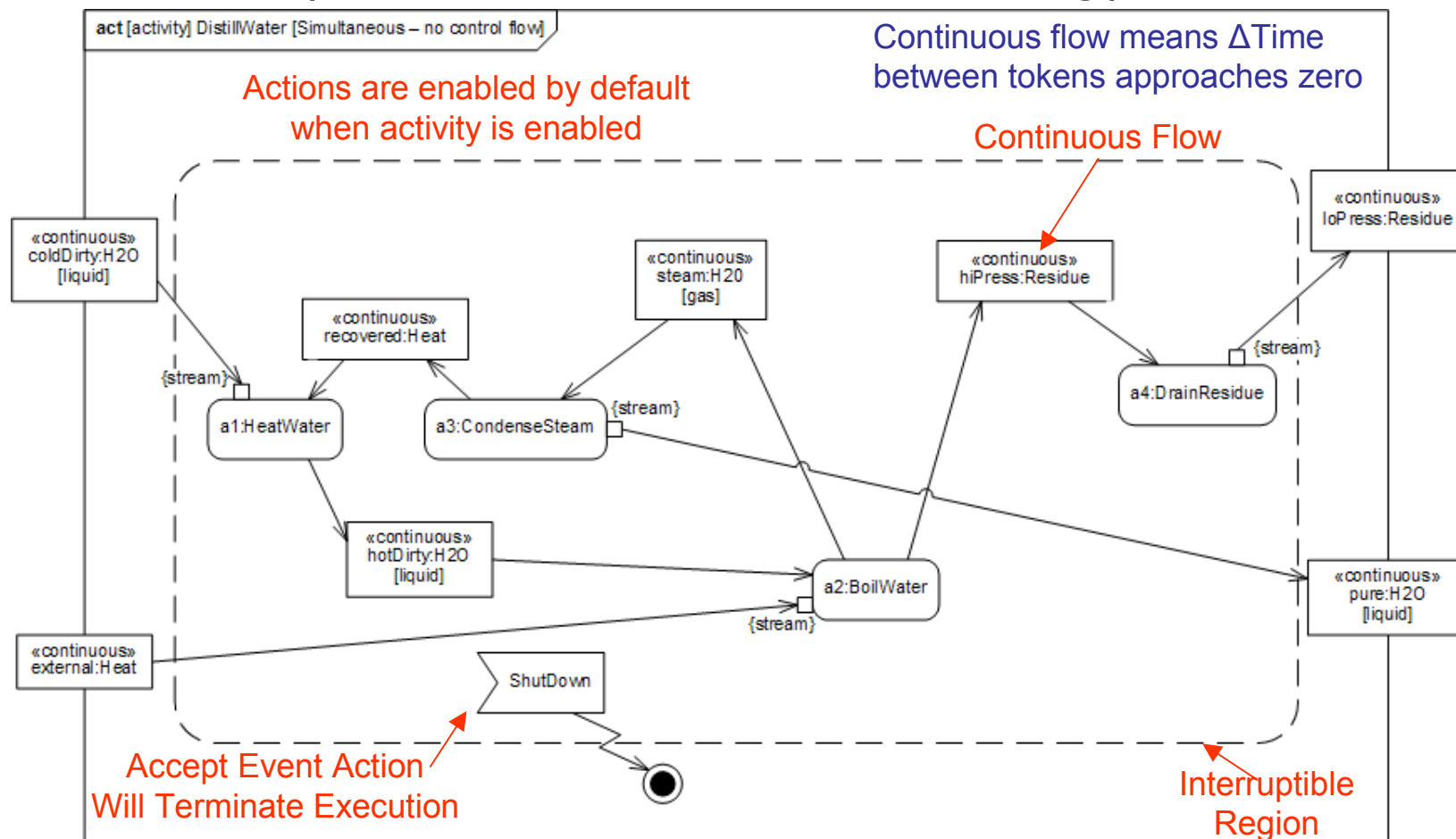


**Send Signal Action**  
(Pins often elided)

# Activity Diagram Example With Streaming Inputs and Outputs

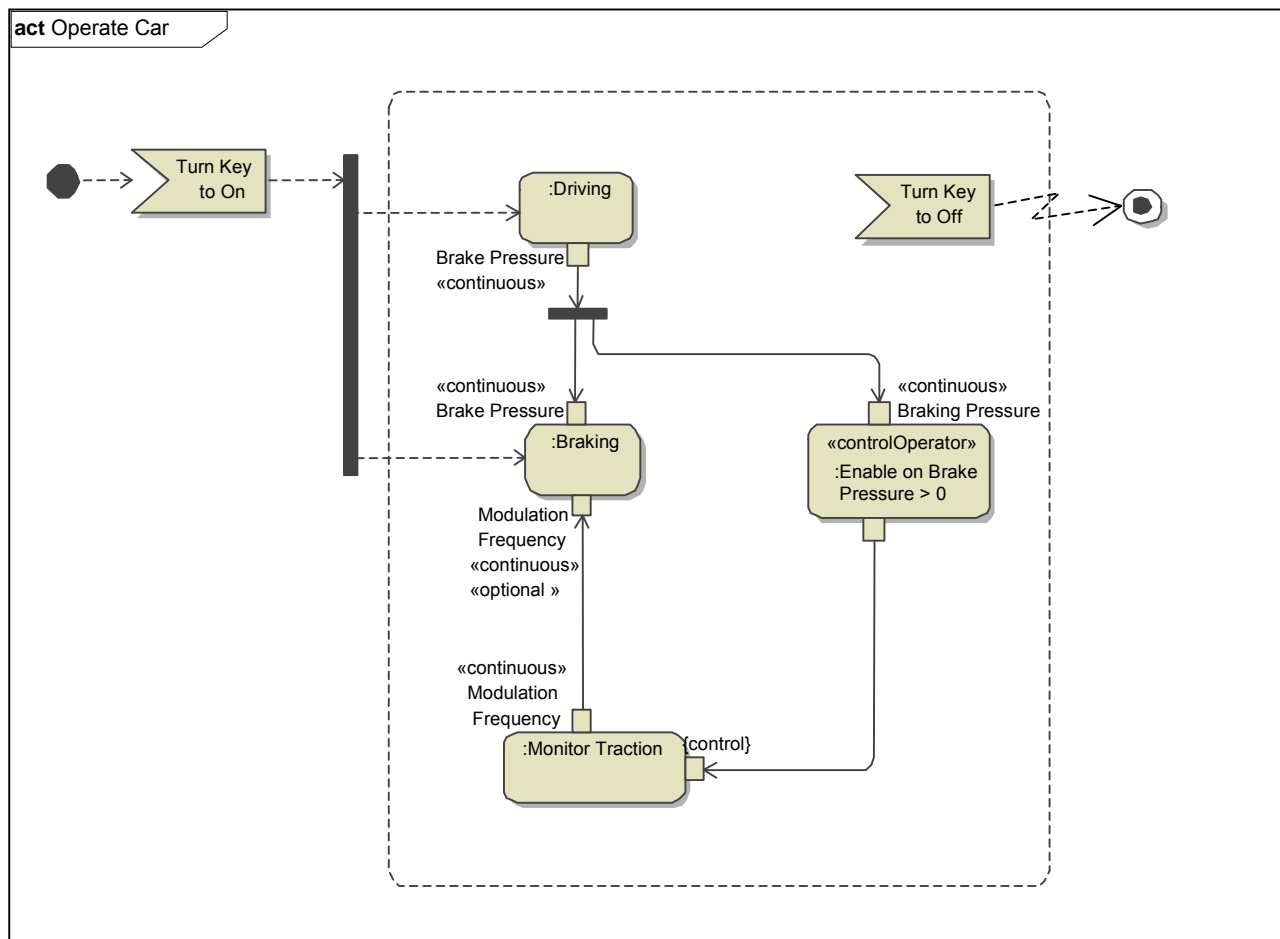


**Streaming Inputs and Outputs Continue to Be Consumed and Produced While the Action is Executing**



## Continuous Flow Is Representative of Many Physical Processes

# Example – Operate Car

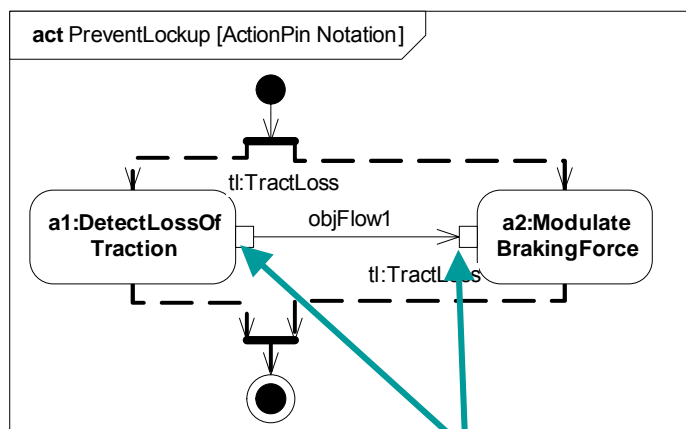


**Enabling and Disabling Actions  
With Control Operators**

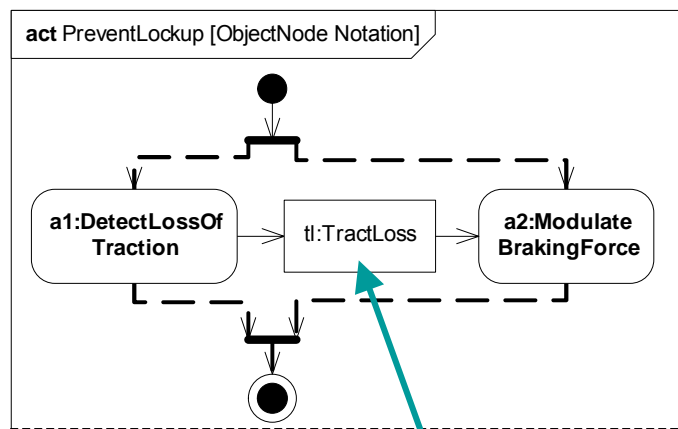
# Activity Diagrams

## 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



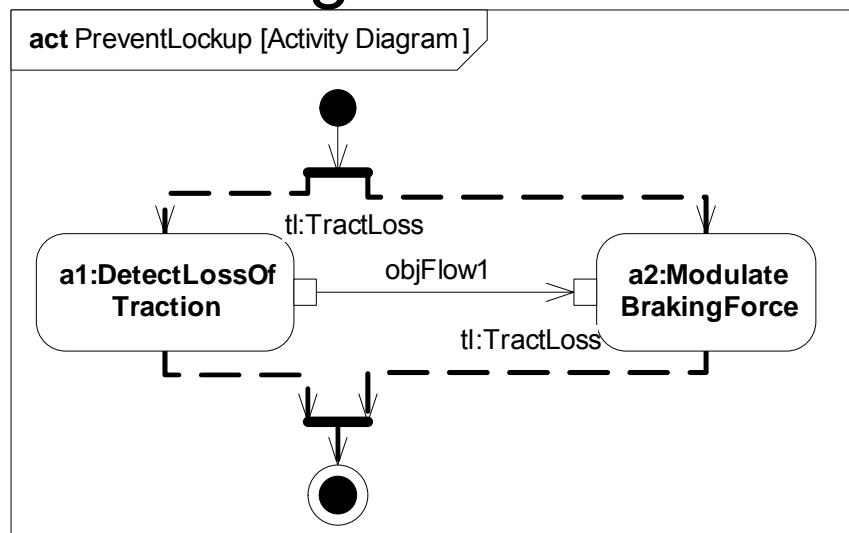
ObjectNode

Pins must have same characteristics (name, type etc.)

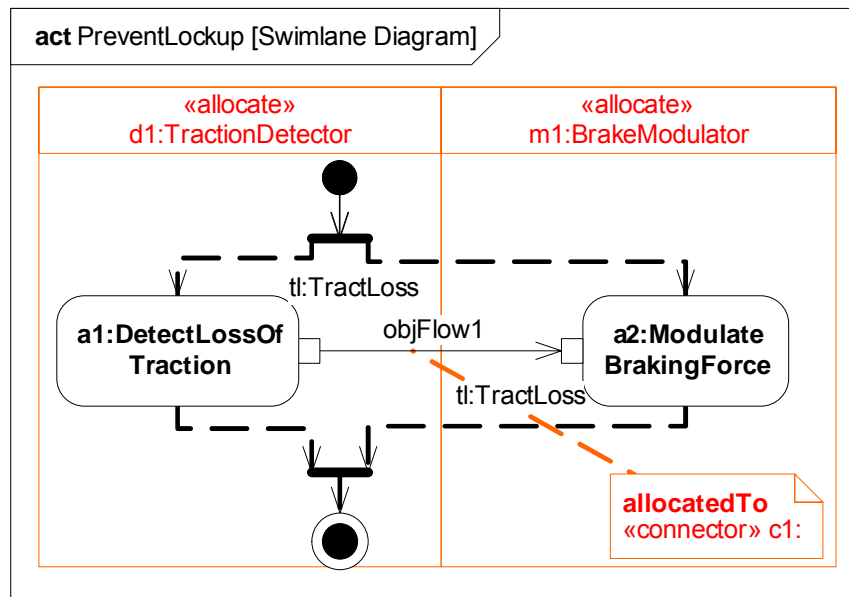


# Explicit Allocation of Behavior to Structure Using Swimlanes

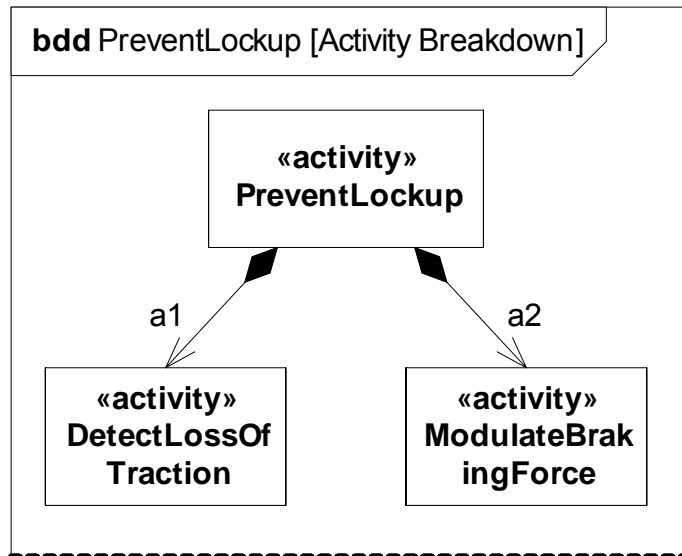
Activity Diagram  
(without Swimlanes)



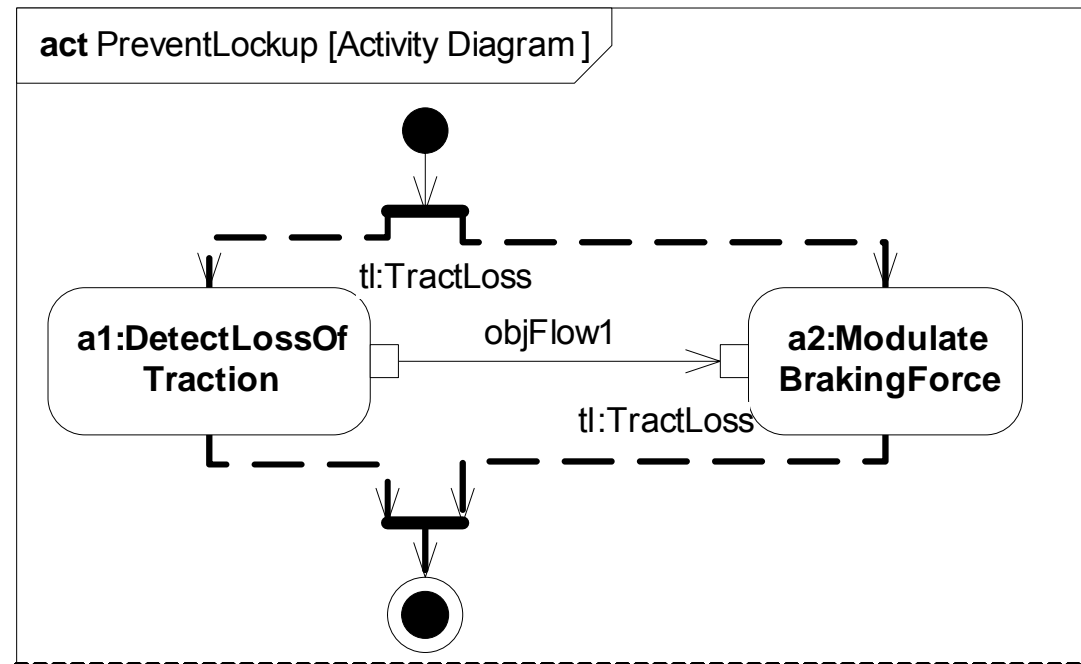
Activity Diagram  
(with Swimlanes)



# Activity Decomposition



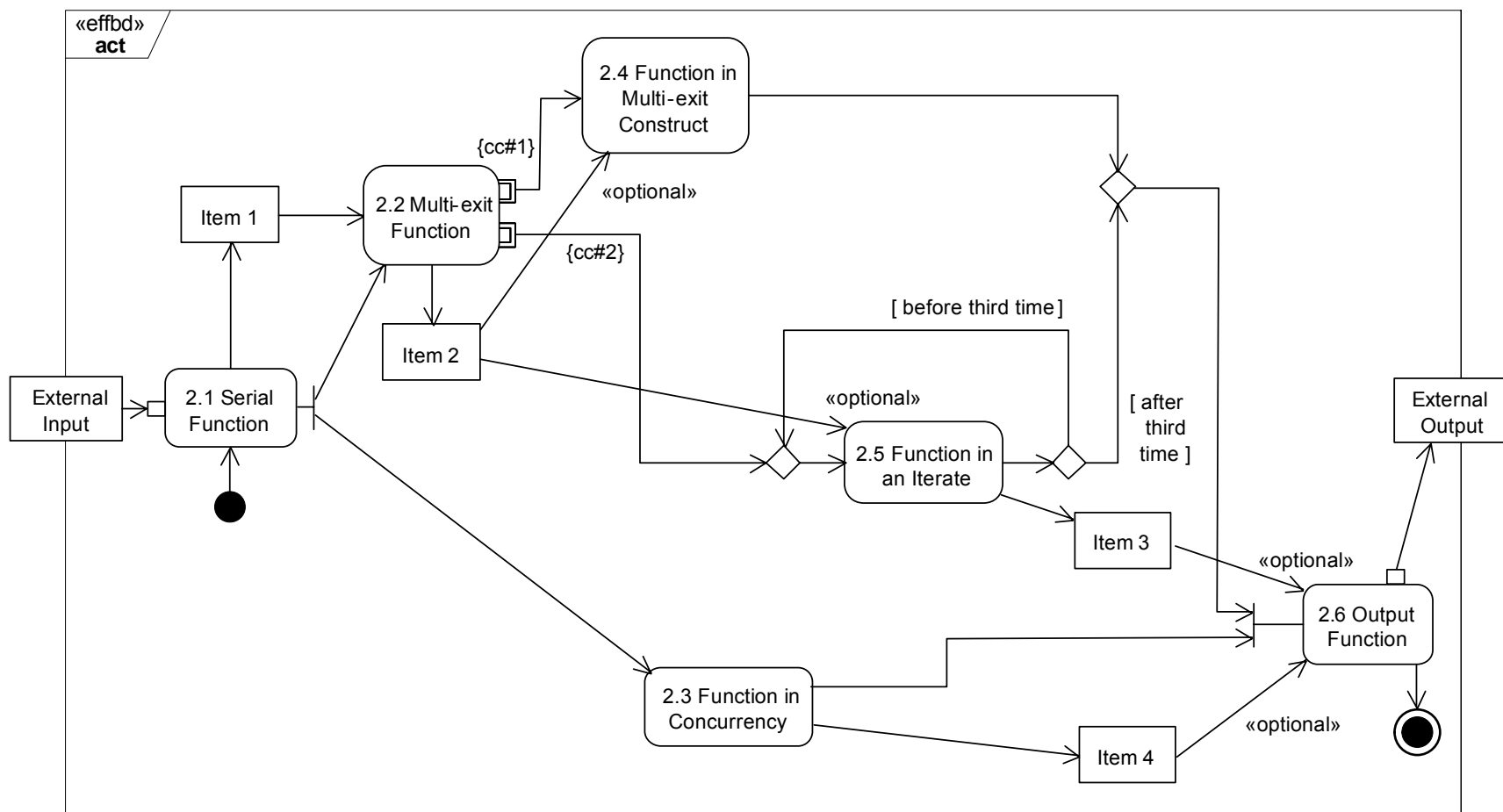
Definition



Use

# SysML EFFBD Profile

## EFFBD - Enhanced Functional Flow Block Diagram

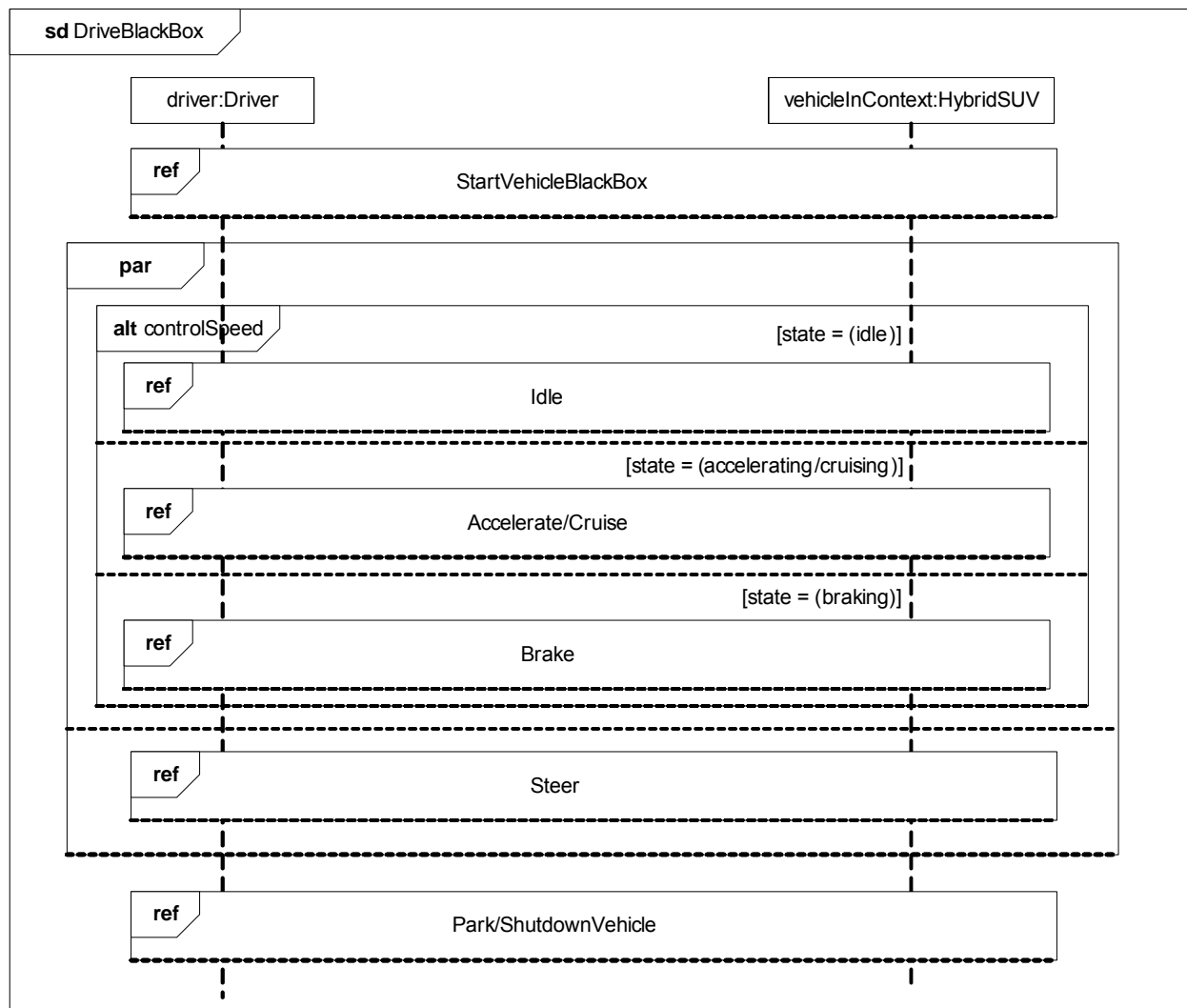


Aligning SysML with Classical Systems Engineering Techniques

# Interactions

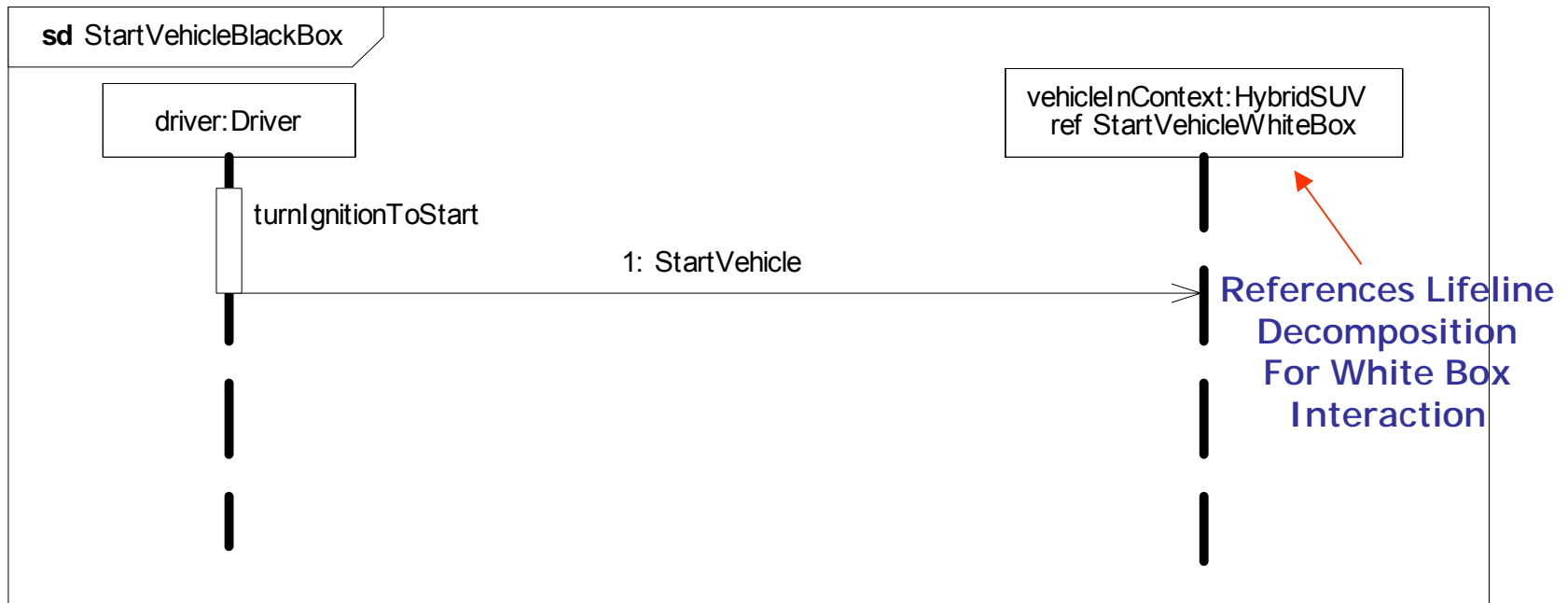
- Sequence diagrams provide representations of message based behavior
  - represent flow of control
  - describe interactions between parts
- Sequence diagrams provide mechanisms for representing complex scenarios
  - reference sequences
  - control logic
  - lifeline decomposition
- SysML does not include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

# Black Box Interaction (Drive)



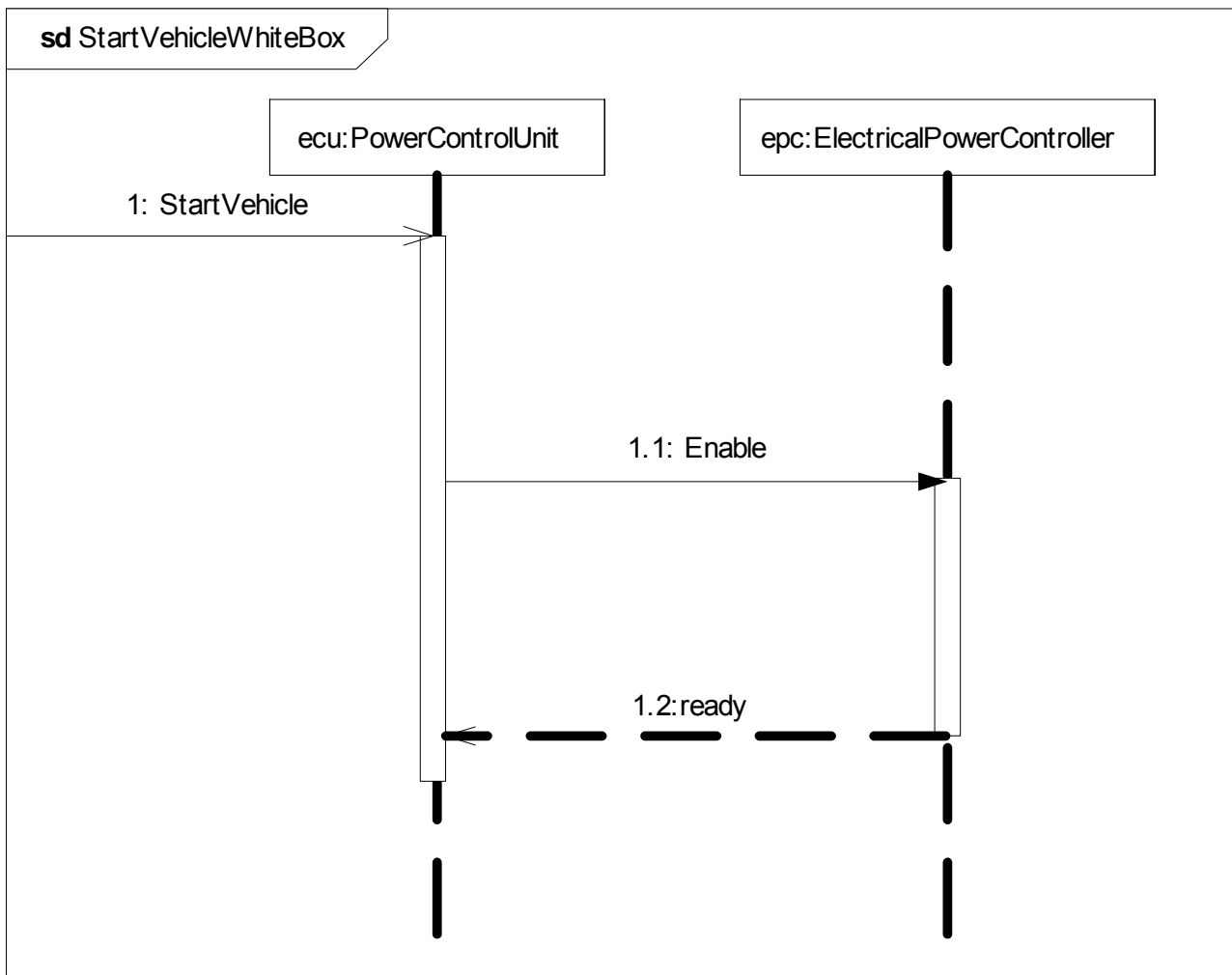
UML 2 Sequence Diagram Scales  
by Supporting Control Logic and Reference Sequences

# Black Box Sequence (StartVehicle)



**Simple Black Box Interaction**

# White Box Sequence (StartVehicle)



**Decomposition of Black Box Into White Box Interaction**

# 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

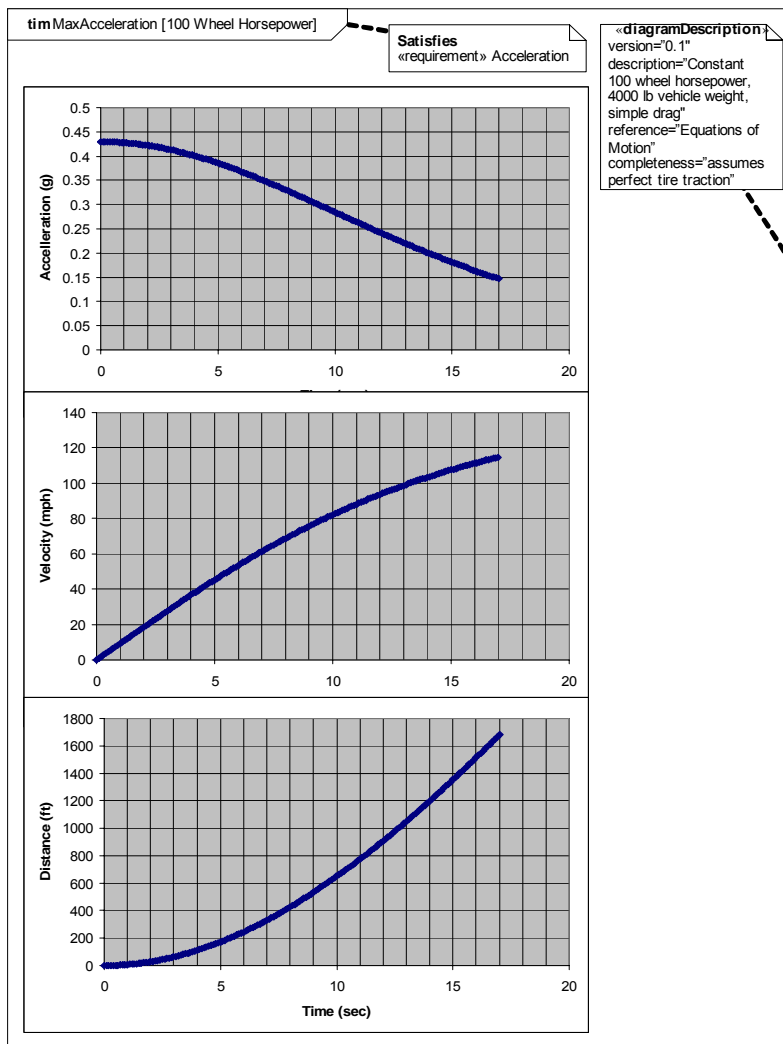


# Other Interaction Operators

- **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

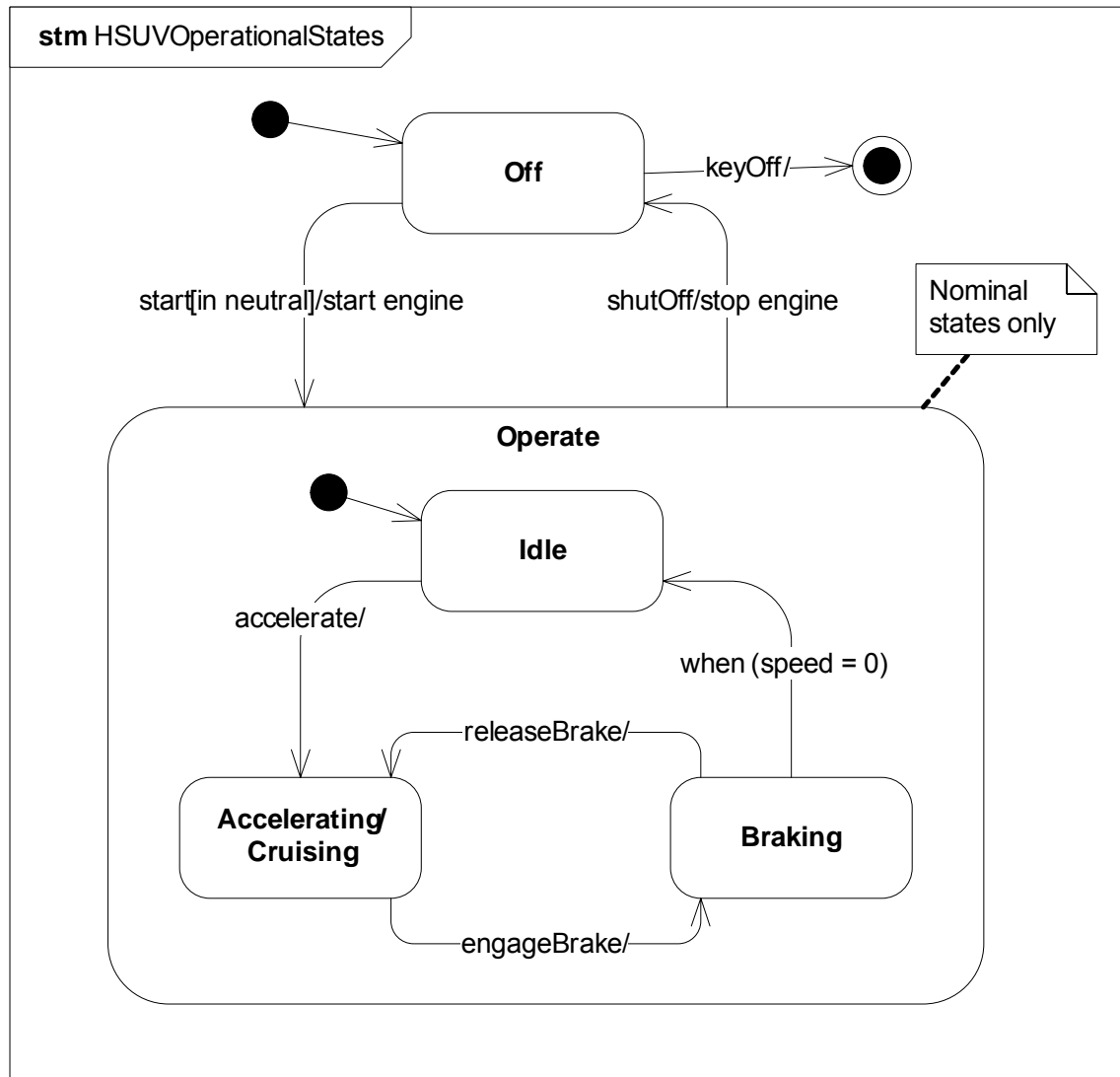
Timing Diagram Not  
Part of SysML

Typical Example of a Timing Diagram

# 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)

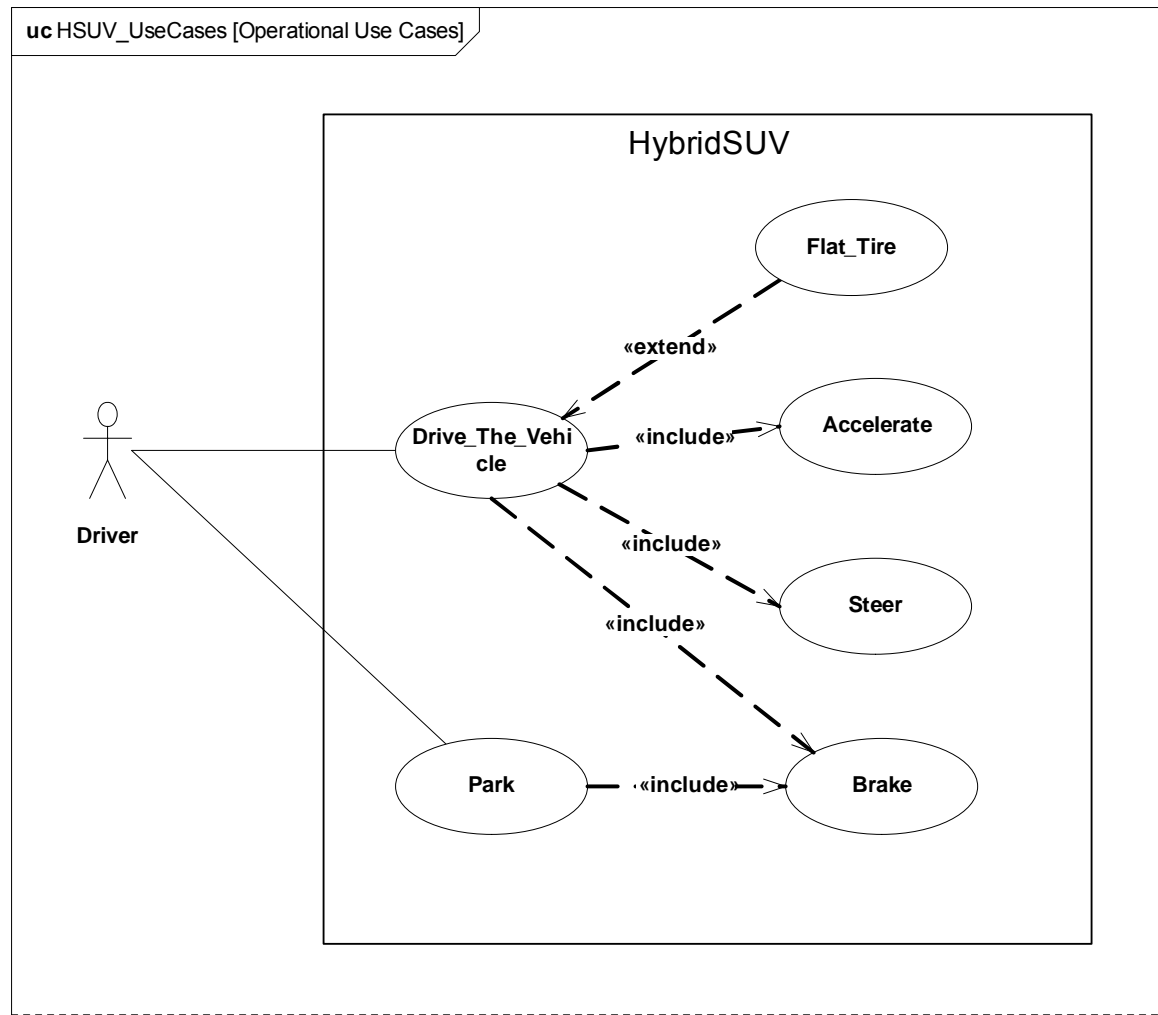


Transition notation:  
trigger[guard]/action

# Use Cases

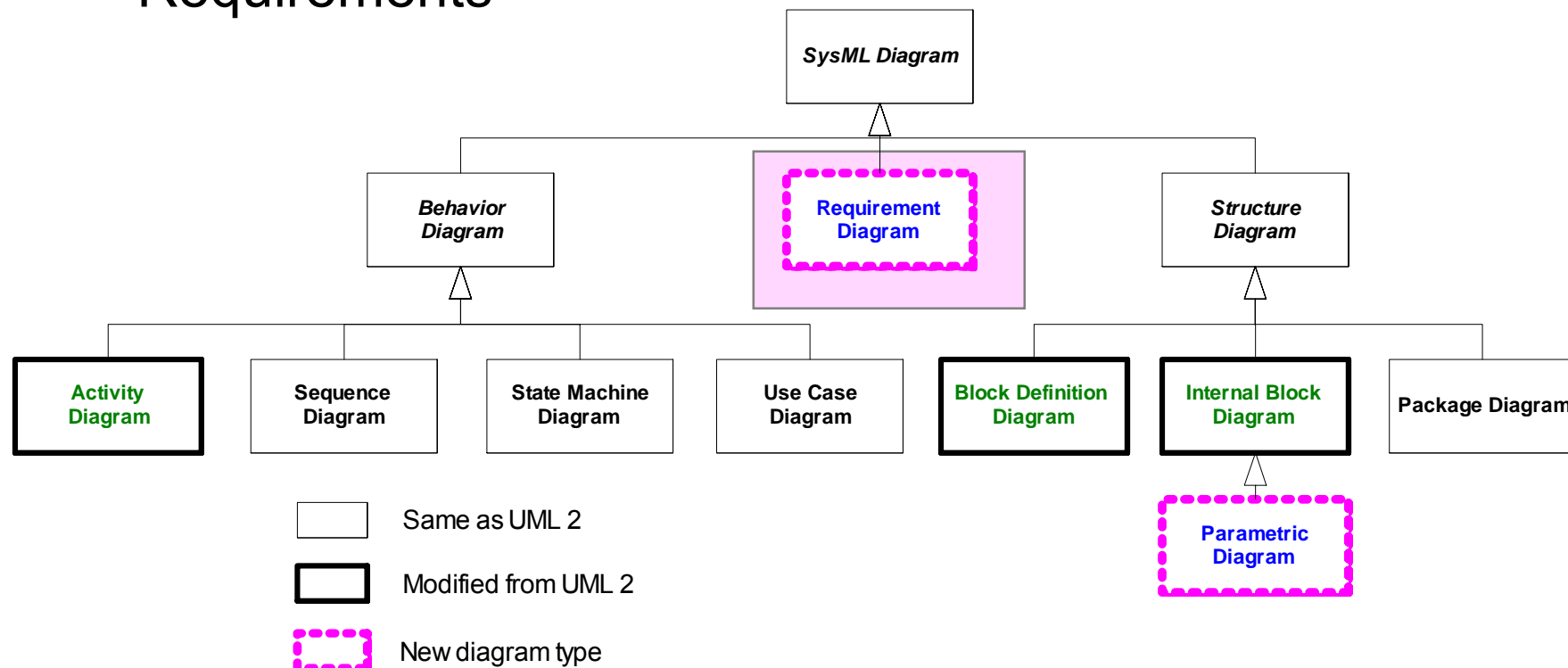
- Provide means for describing basic functionality in terms of usages/goals of the system by actors
- Common functionality can be factored out via «include» and «extend» relationships
- Elaborated via other behavioral representations to describe detailed scenarios
- No change to UML

# Operational Use Cases



# Cross-cutting Constructs

- Allocations
- Requirements

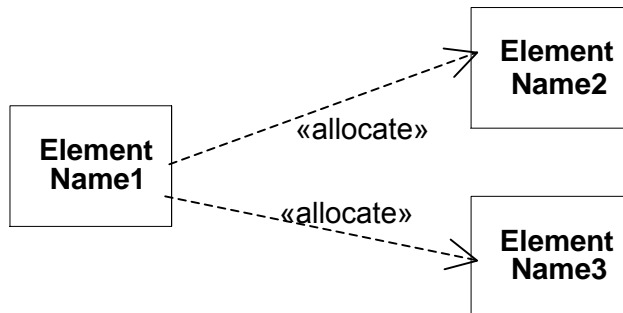


# Allocations

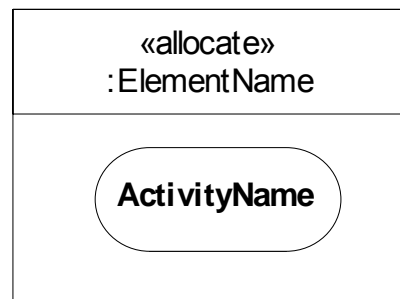
- Represent general relationships that map one model element to another
- Different types of allocation are:
  - Behavioral (i.e., function to component)
  - Structural (i.e., logical to physical)
  - Software to Hardware
  - ....
- Explicit allocation of activities to structure via swim lanes (i.e., activity partitions)
- Both graphical and tabular representations are specified



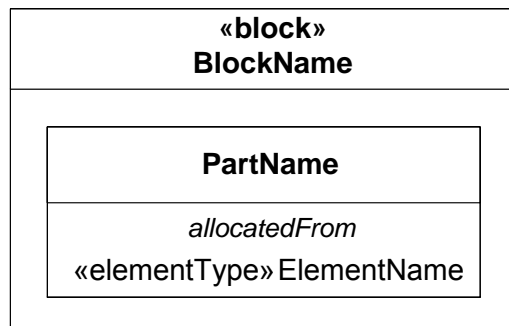
# Different Allocation Representations (Tabular Representation Not Shown)



Allocate Relationship

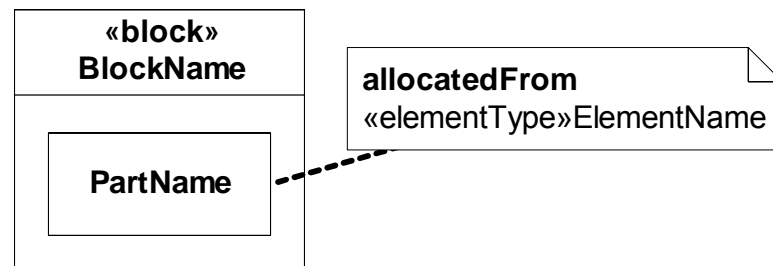


Explicit Allocation of  
Activity to Swim Lane



Compartment Notation

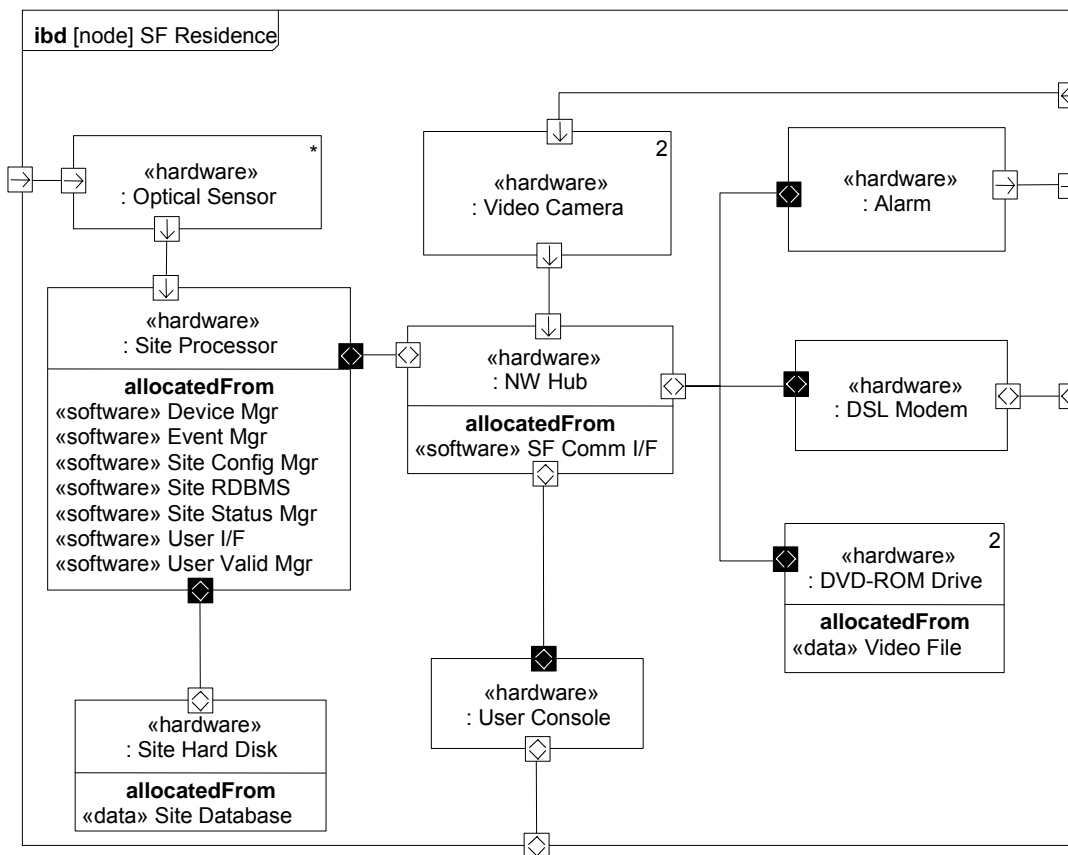
*Read as follows: "PartName has an<<element type>> allocatedFrom ElementName"*



Callout Notation

# SysML Allocation of SW to HW

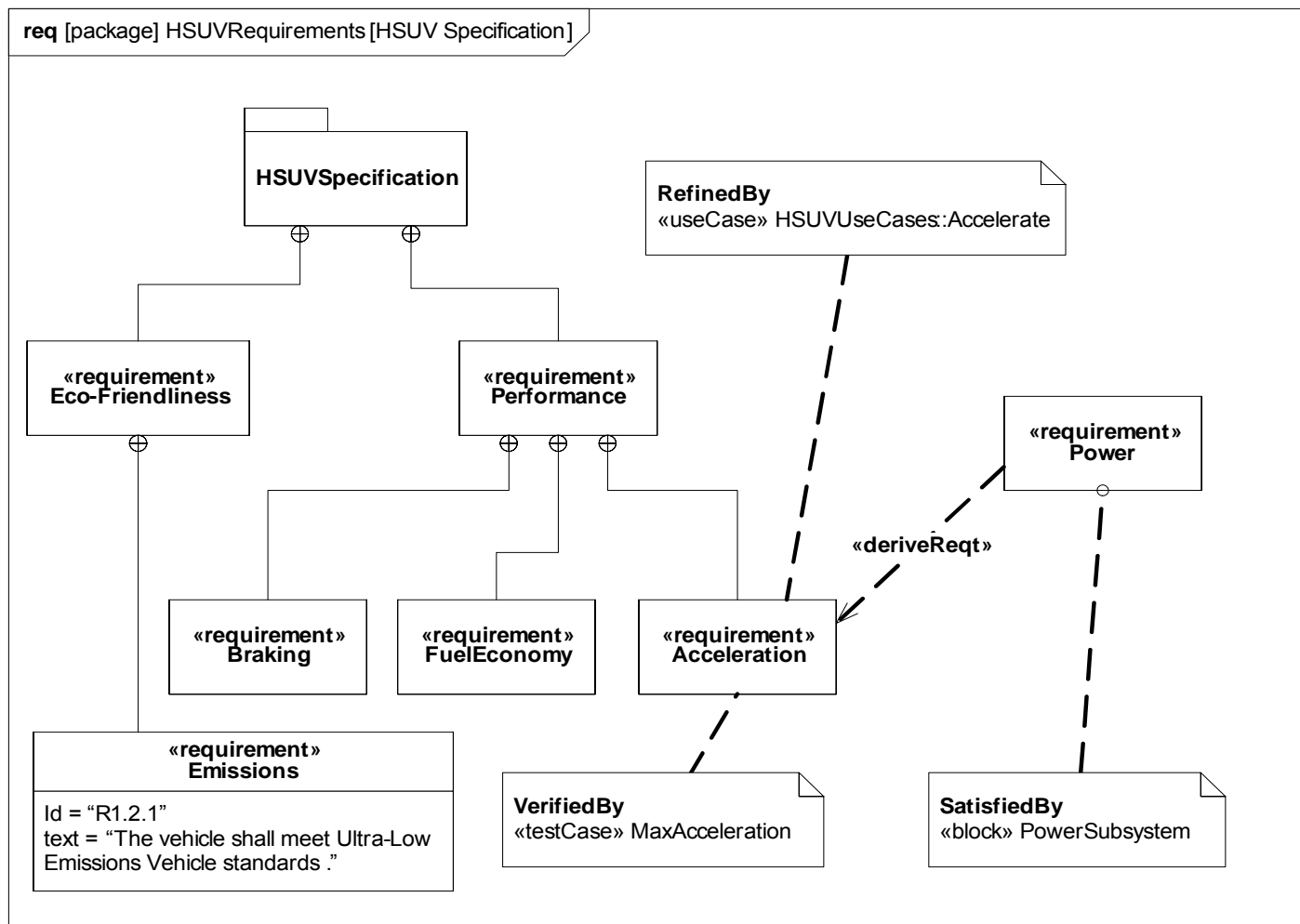
- In UML, the deployment diagram is used to deploy artifacts to nodes
- In SysML, «allocation» on an **ibd** and **bdd** is used to deploy software/data to hardware



# Requirements

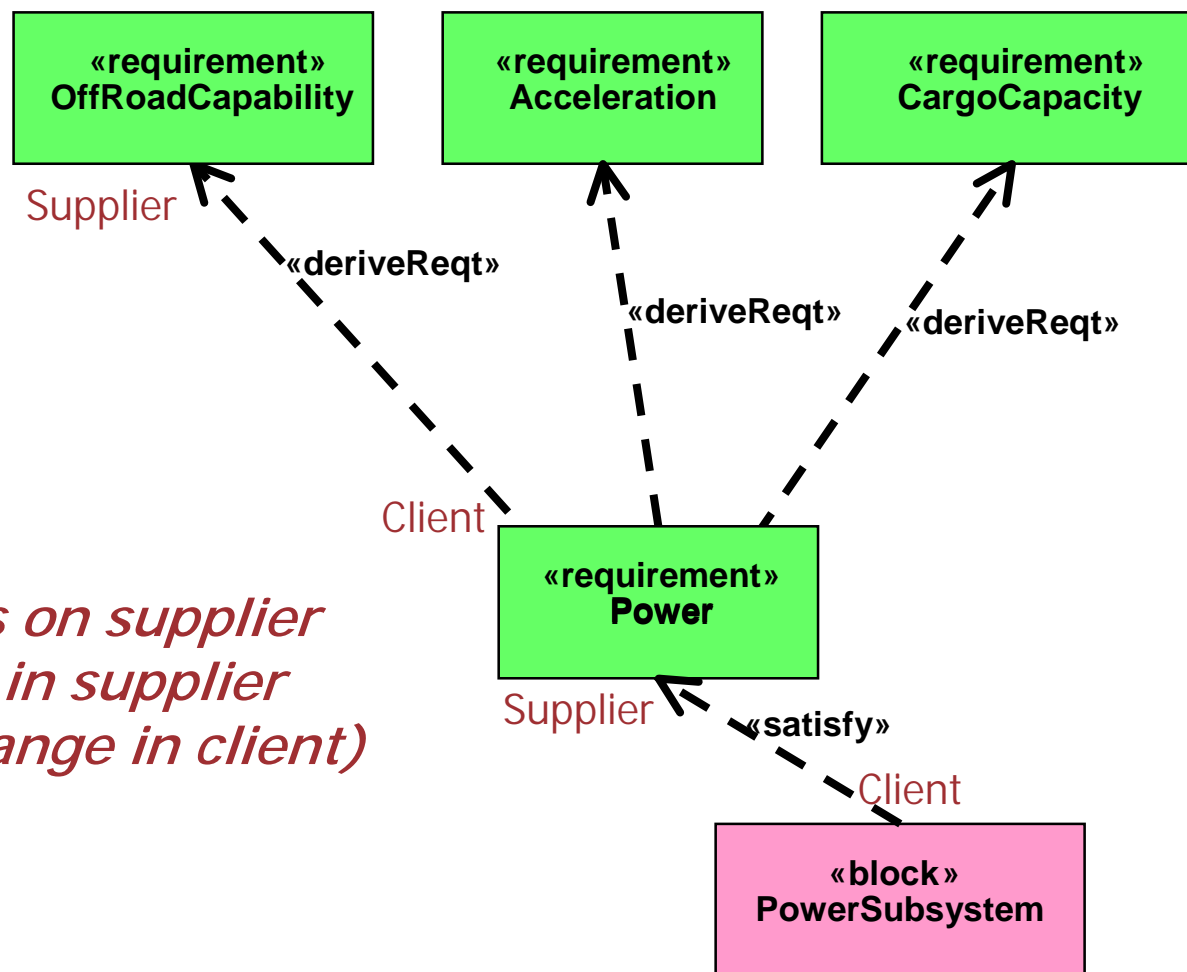
- The «requirement» stereotype represents a text based requirement
  - Includes id and text properties
  - Can add user defined properties such as verification method
  - Can add user defined requirements categories (e.g., functional, interface, performance)
- Requirements hierarchy describes requirements contained in a specification
- Requirements relationships include DeriveReq, Satisfy, Verify, Refine, Trace, Copy

# Requirements Breakdown



Requirement Relationships Model the Content of a Specification

# Example of Derive/Satisfy Requirement Dependencies

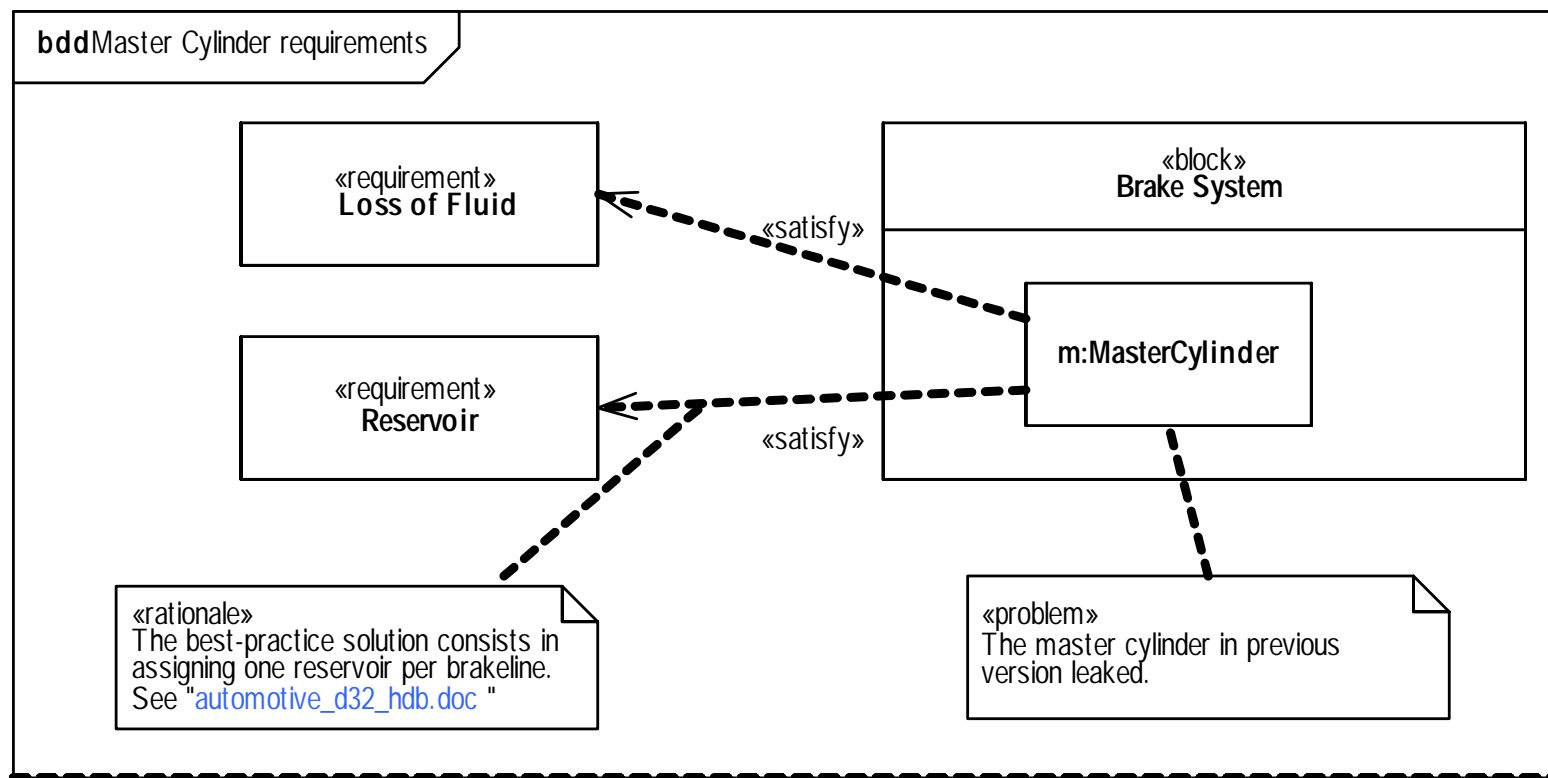


*Client depends on supplier  
(i.e., a change in supplier  
results in a change in client)*

from OMG

Arrow Direction Opposite Typical Requirements Flow-Down

# Problem and Rationale

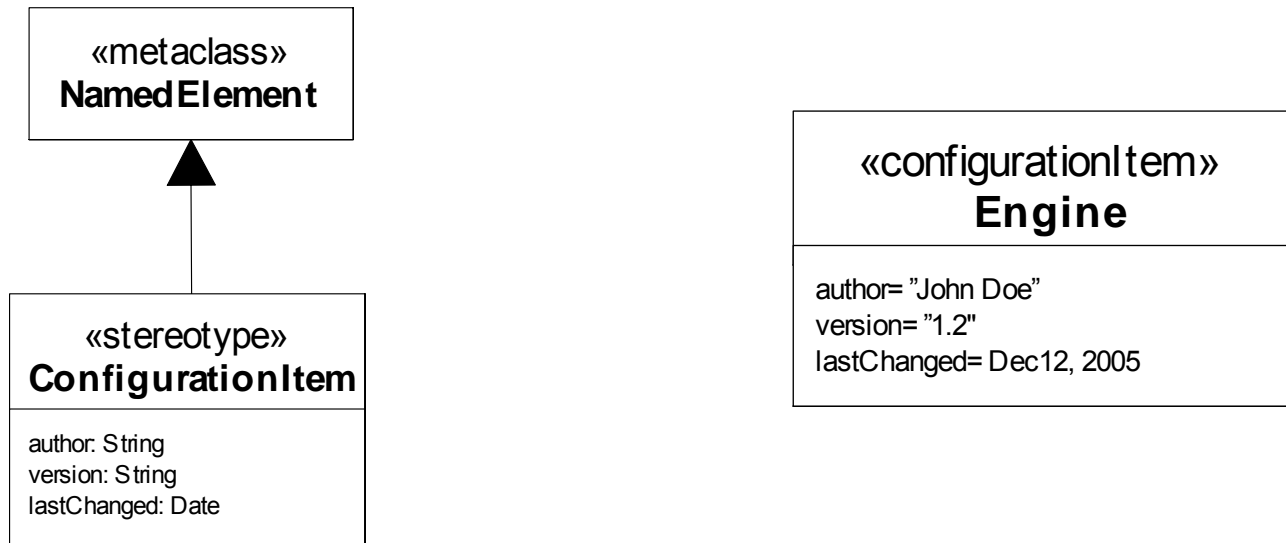


**Problem and Rationale can be attached to any Model Element to Capture Issues and Decisions**

# 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

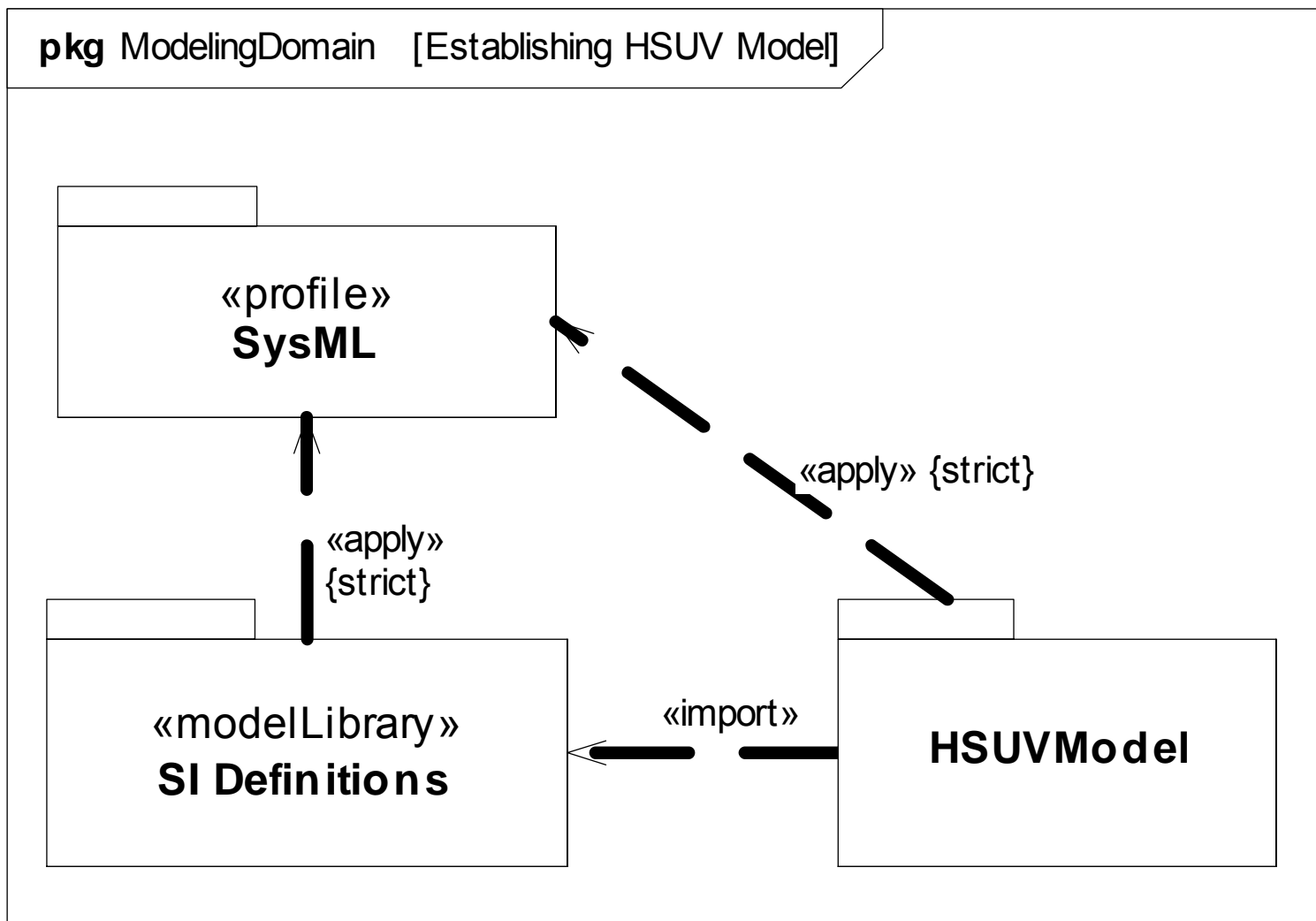


Defining the Stereotype

Applying the Stereotype

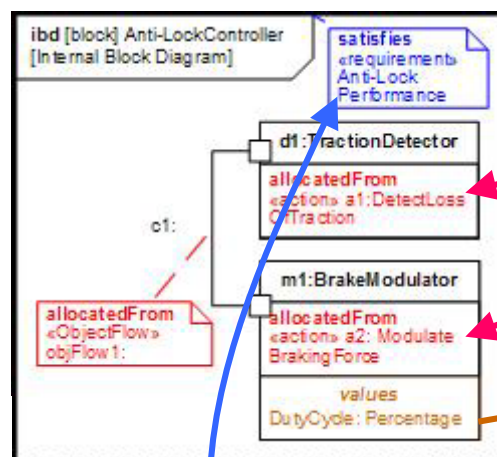


# Applying a Profile and Importing a Model Library

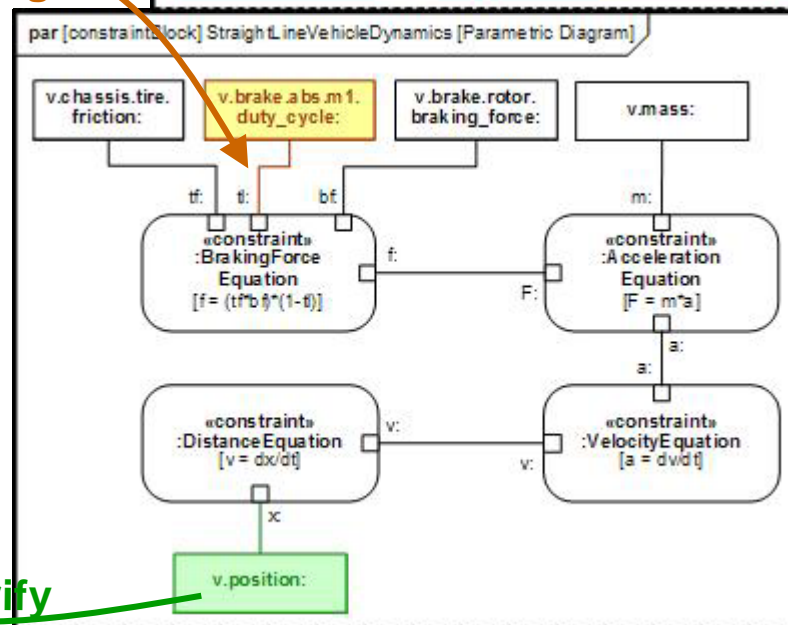
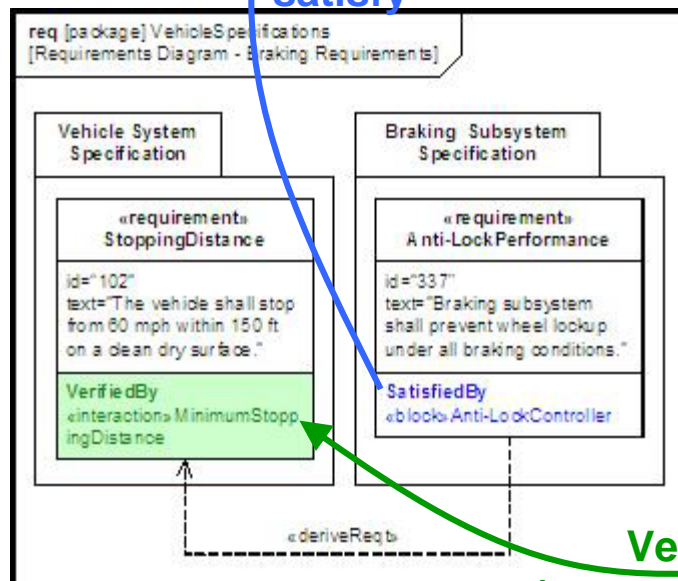
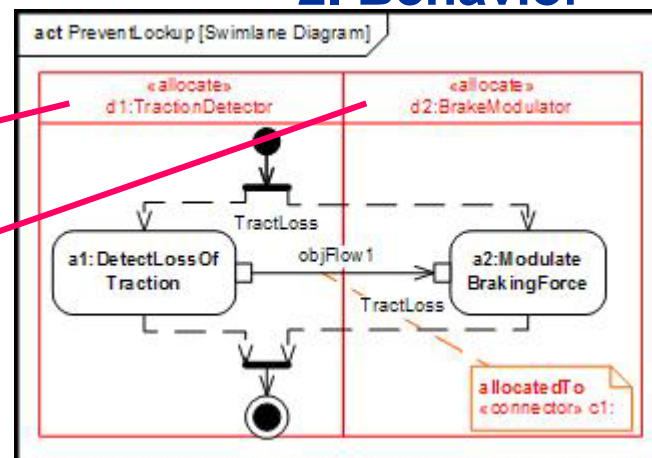


# Cross Connecting Model Elements

## 1. Structure



## 2. Behavior



## 3. Requirements (via interaction)

## 4. Parametrics



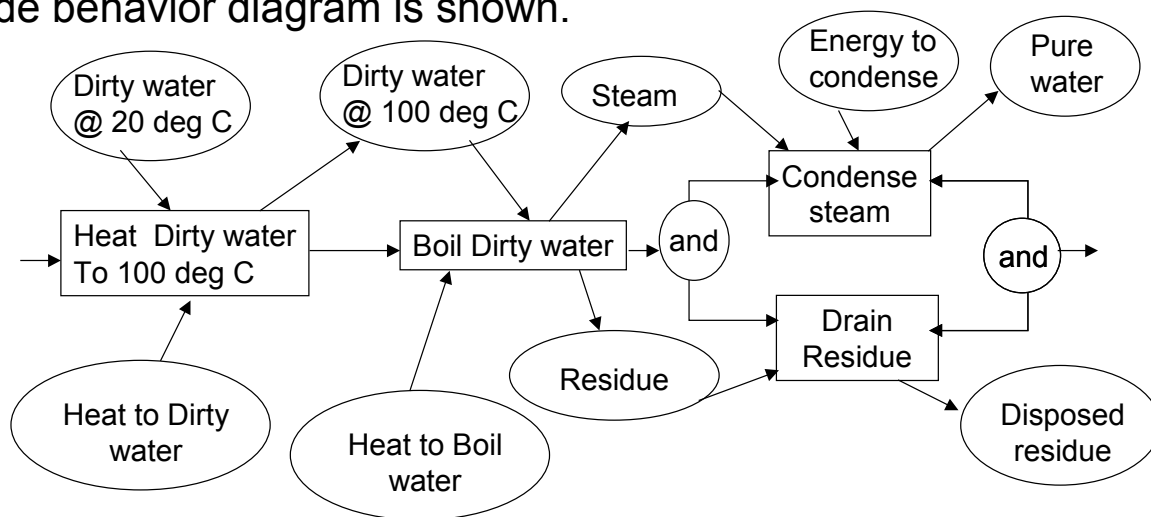
# SysML Modeling as Part of the SE Process



## Distiller Sample Problem

# Distiller Problem Statement

- The following problem was posed to the SysMLteam 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<sup>3</sup>, temp 20 deg C, specific heat 1cal/gm deg C, heat of vaporization 540 cal/gm.
- A crude behavior diagram is shown.



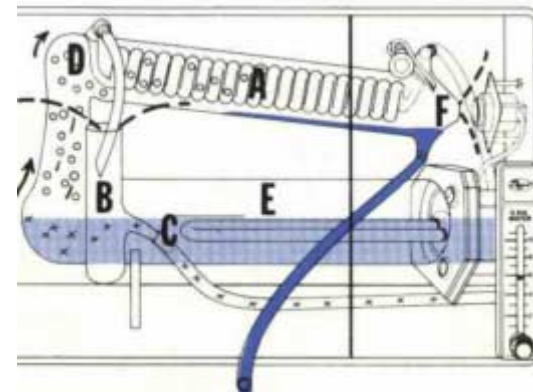
What are the real requirements?  
How do we design the system?

# Distiller Types

## Batch Distiller



## Continuous Distiller

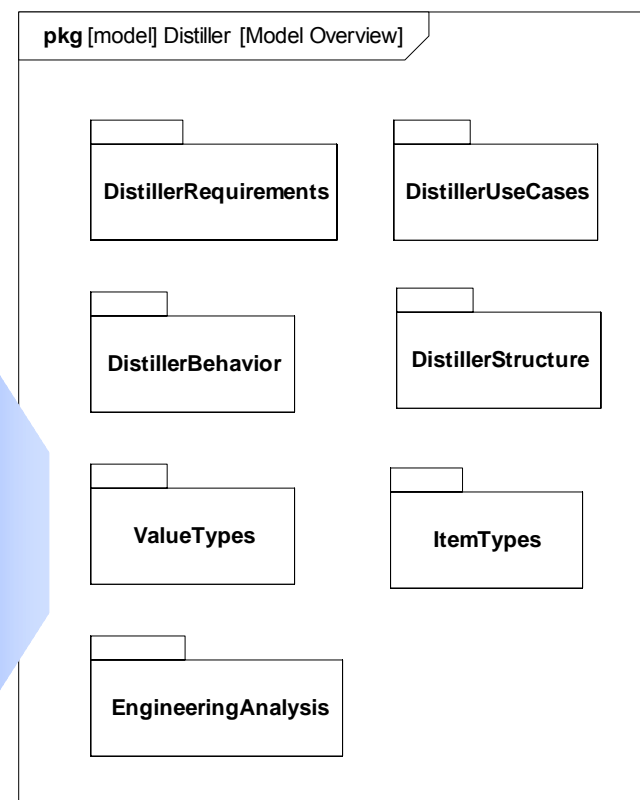
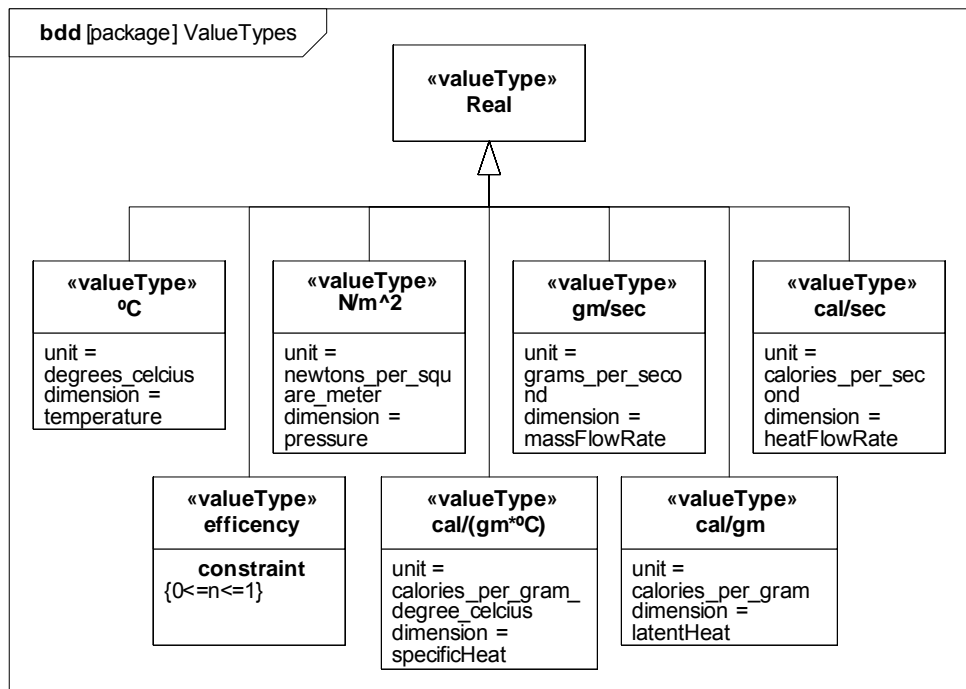


Note: Not all aspects of the distiller are modeled in the example

# Distiller Problem – Process Used

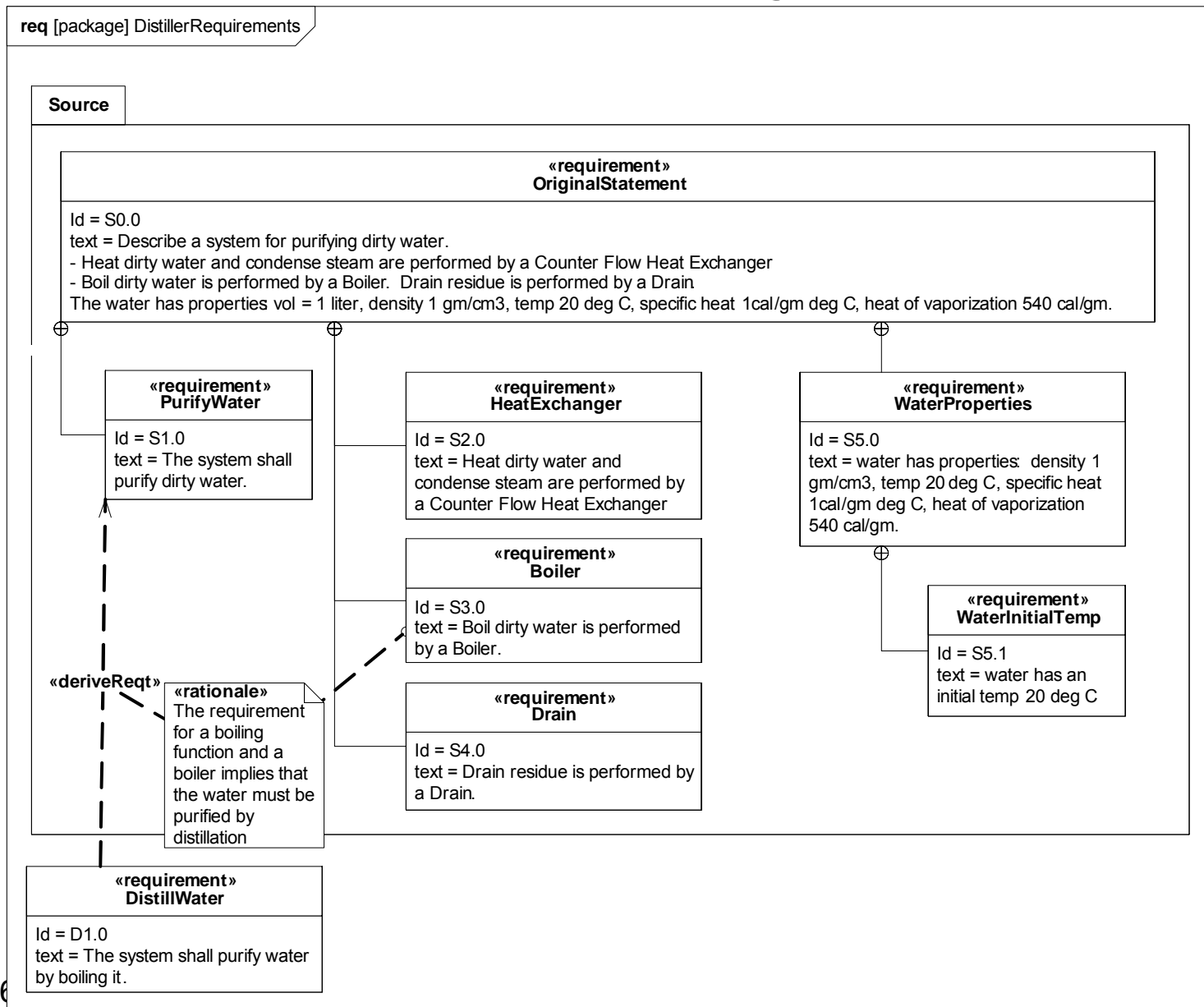
- Organize the model, identify libraries needed
- List requirements and assumptions
- Model behavior
  - In similar form to problem statement
  - Elaborate as necessary
- Model structure
  - Capture implied inputs and outputs
    - segregate I/O from behavioral flows
  - Allocate behavior onto structure, 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 Example: Requirements Tables

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

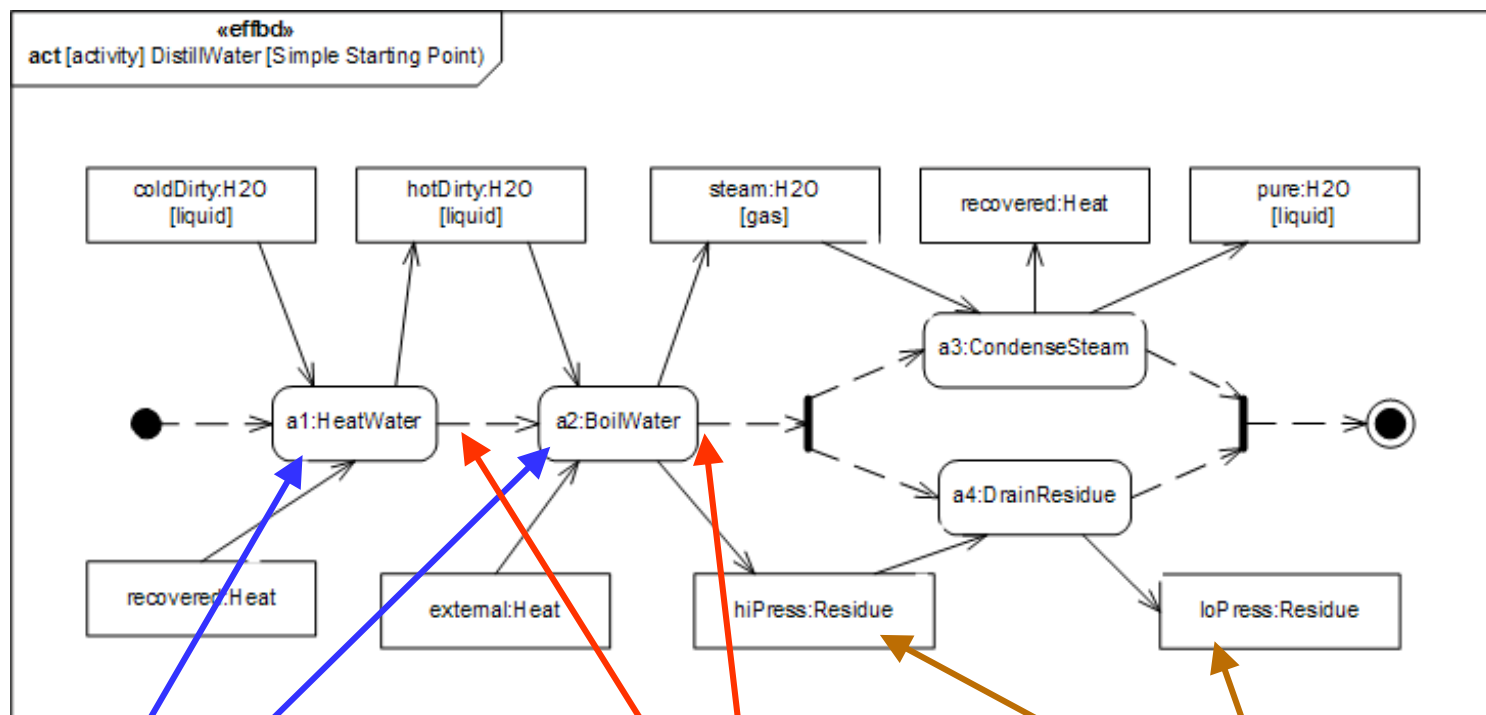
id	name	text
S0.0	OriginalStatement	Describe a system for purifying dirty water. ...
S1.0	PurifyWater	The system shall purify dirty water.
S2.0	HeatExchanger	Heat dirty water and condense steam are performed by a ...
S3.0	Boiler	Boil dirty water is performed by a Boiler.
S4.0	Drain	Drain residue is performed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

**table** [requirement] PurifyWater [Requirements Tree]

id	name	relation	id	name	Rationale
S1.0	PurifyWater	deriveReq	D1.0	DistillWater	The requirement for a boiling function and a boiler implies that the water must be purified by distillation

# Distiller Example – Activity Diagram: Initial Diagram for DistillWater

- This activity diagram applies the SysML EFFBD profile, and formalizes the diagram in the problem statement.

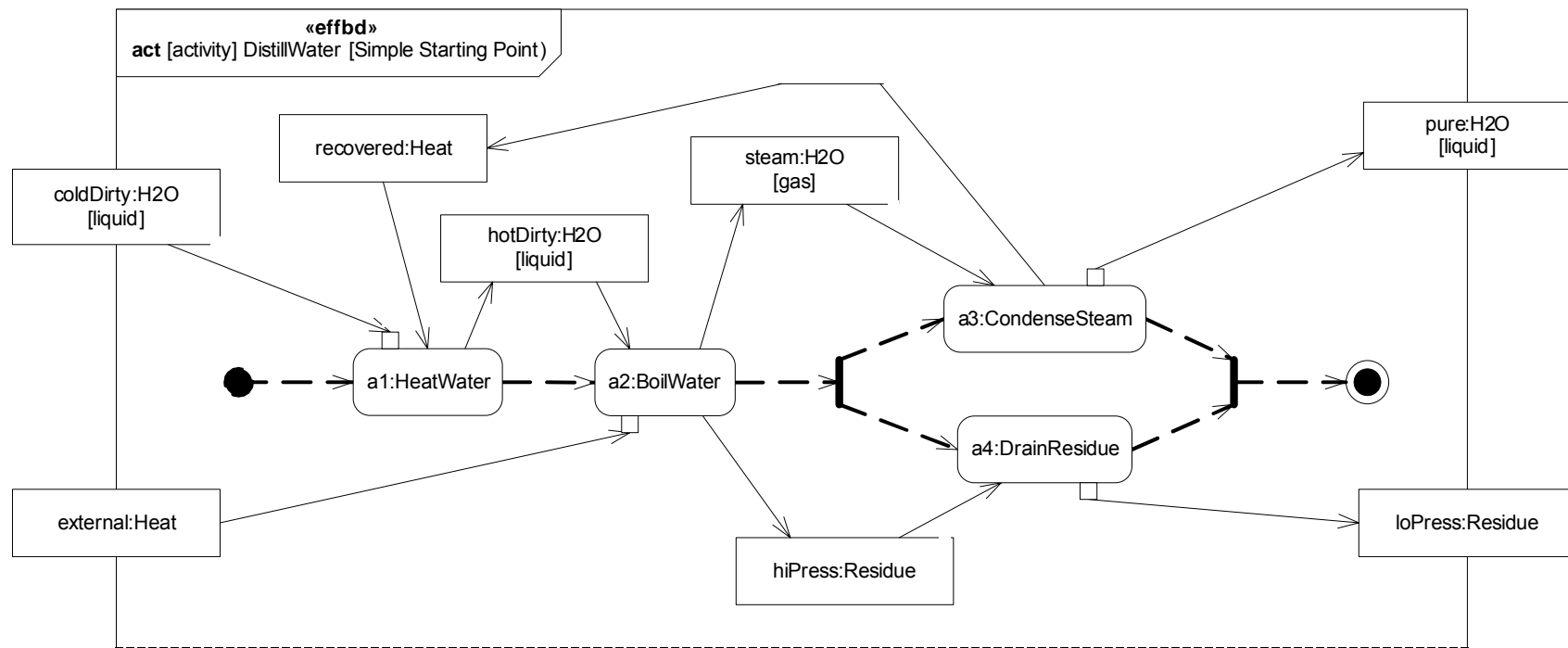


**Activities (Functions)**

**Control (Sequence) Things that flow (ObjectNodes)**



# Distiller Example – Activity Diagram: Control-Driven: Serial Behavior

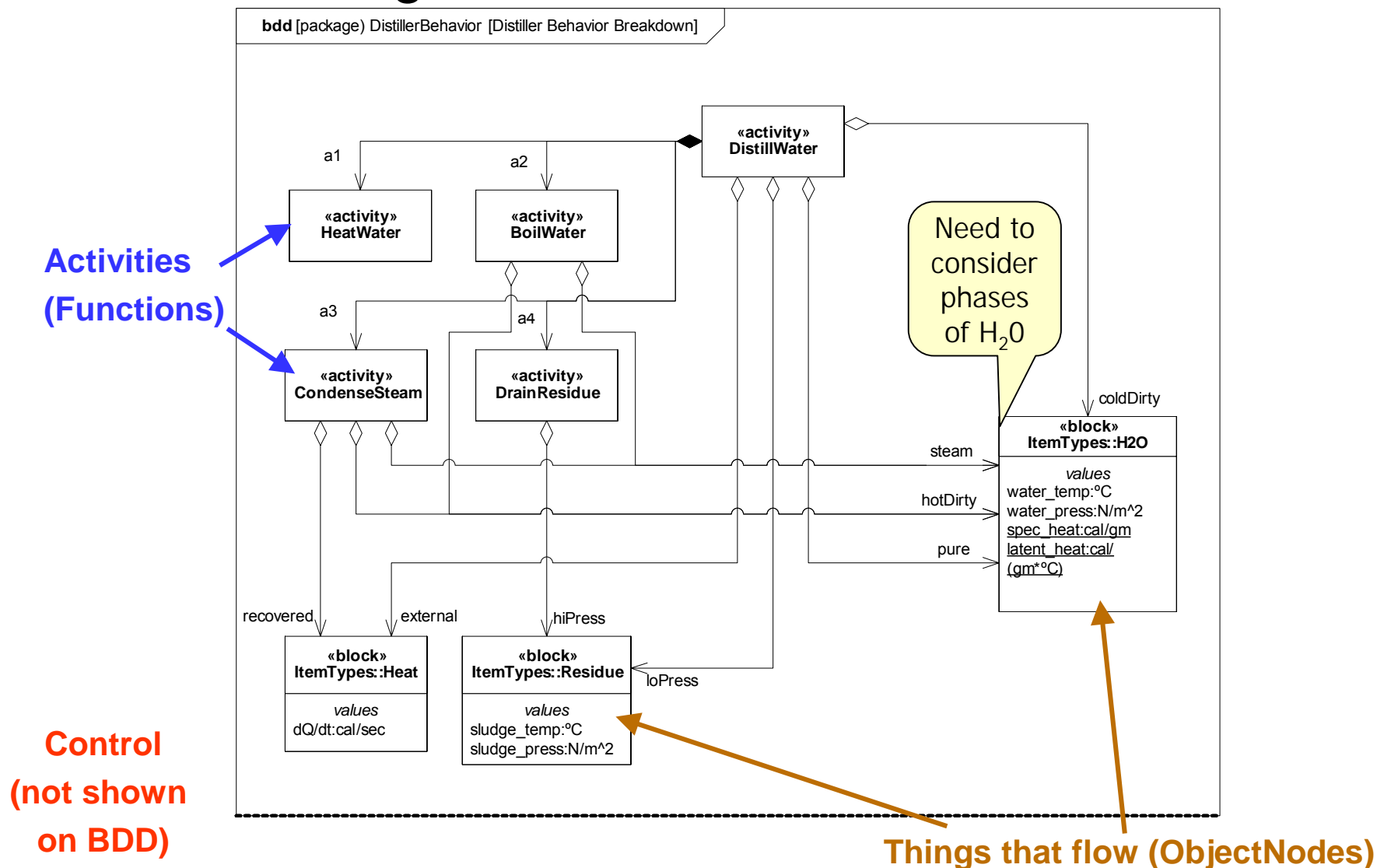


Continuous Distiller Here

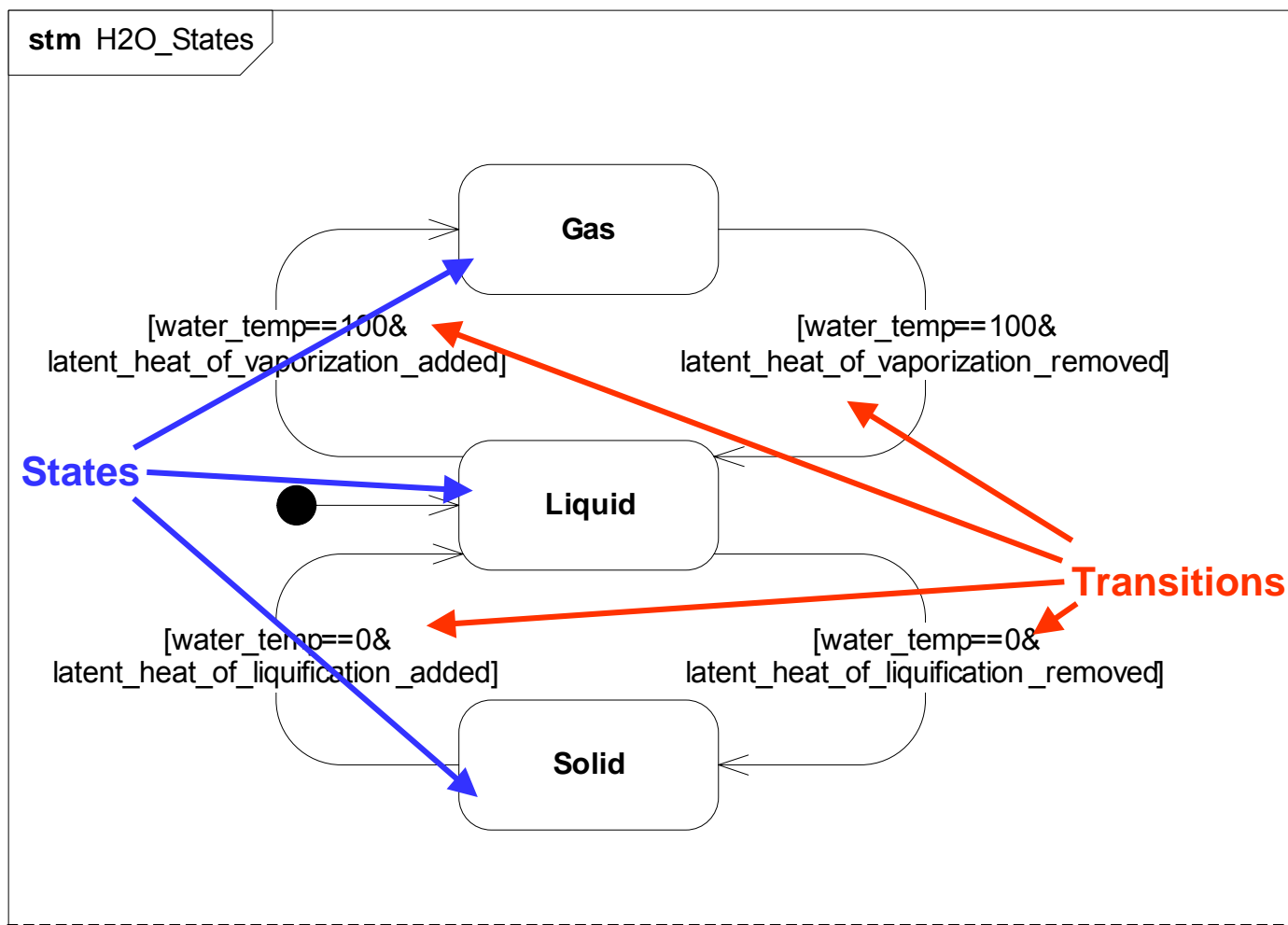
Batch  
Distiller



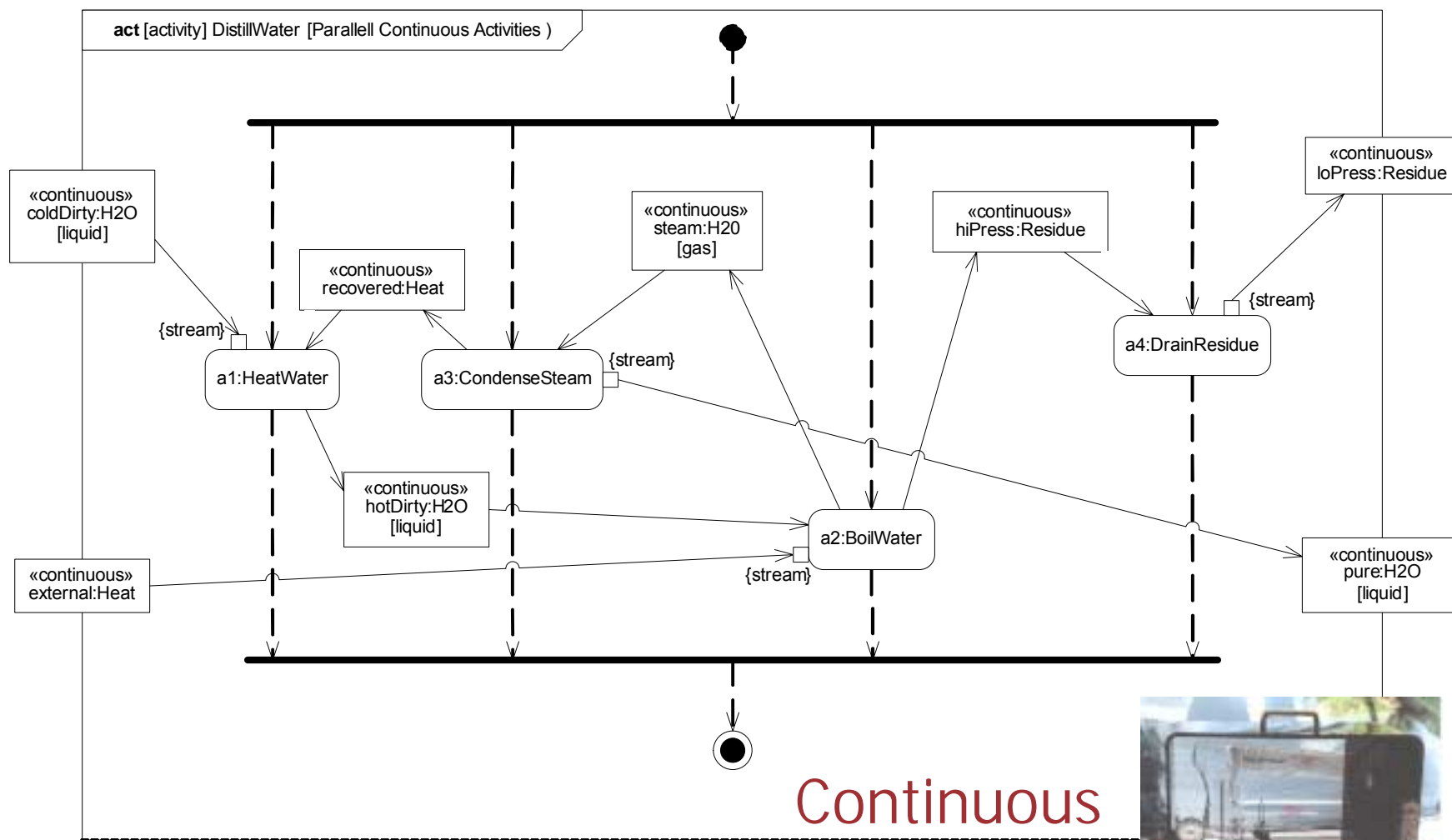
# Distiller Example – Block Definition Diagram: DistillerBehavior



# Distiller Example – State Machine Diagram: States of H2O



# Distiller Example – Activity Diagram: I/O Driven: Continuous Parallel Behavior

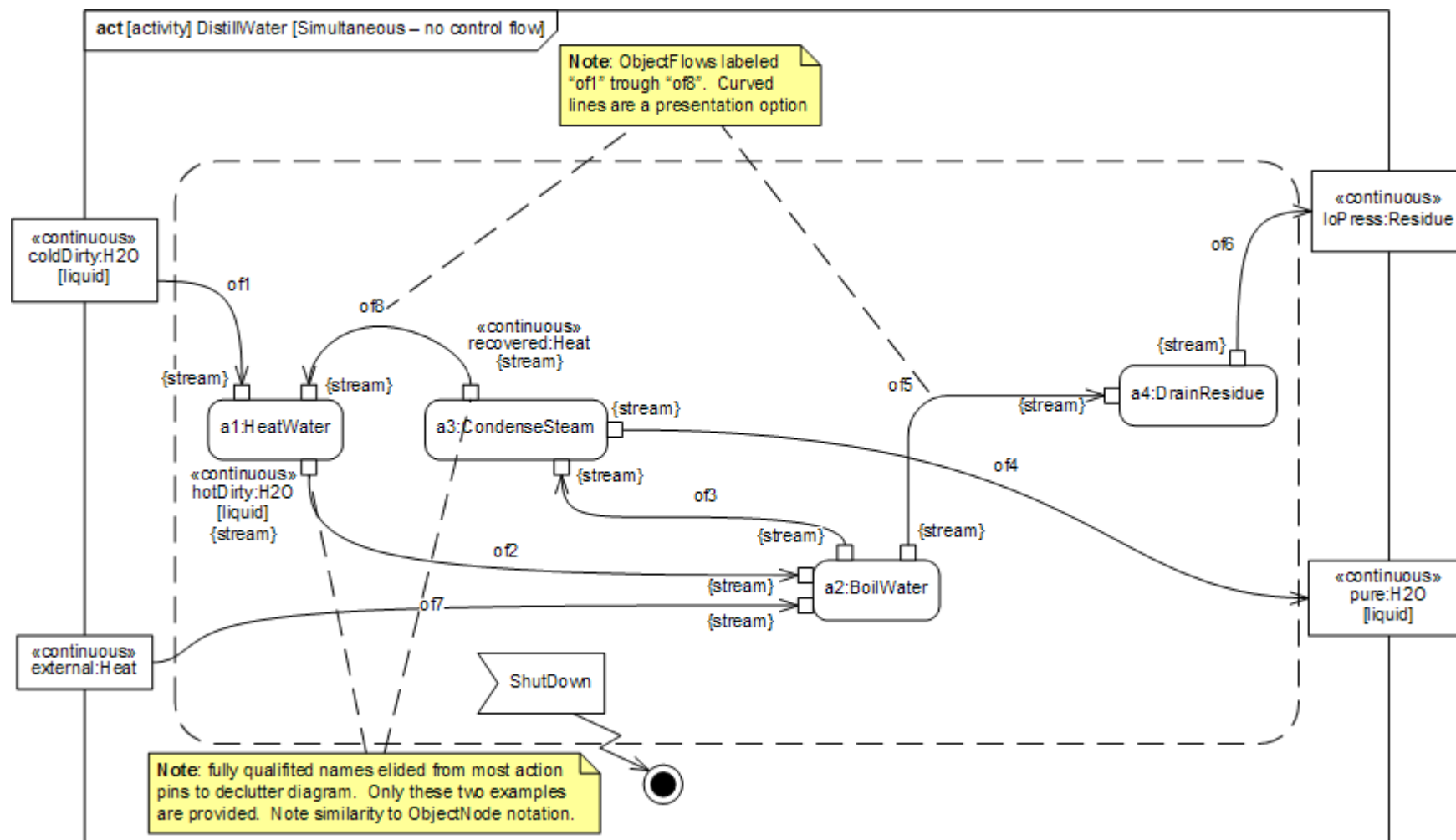


Batch Distiller Here

Continuous  
Distiller



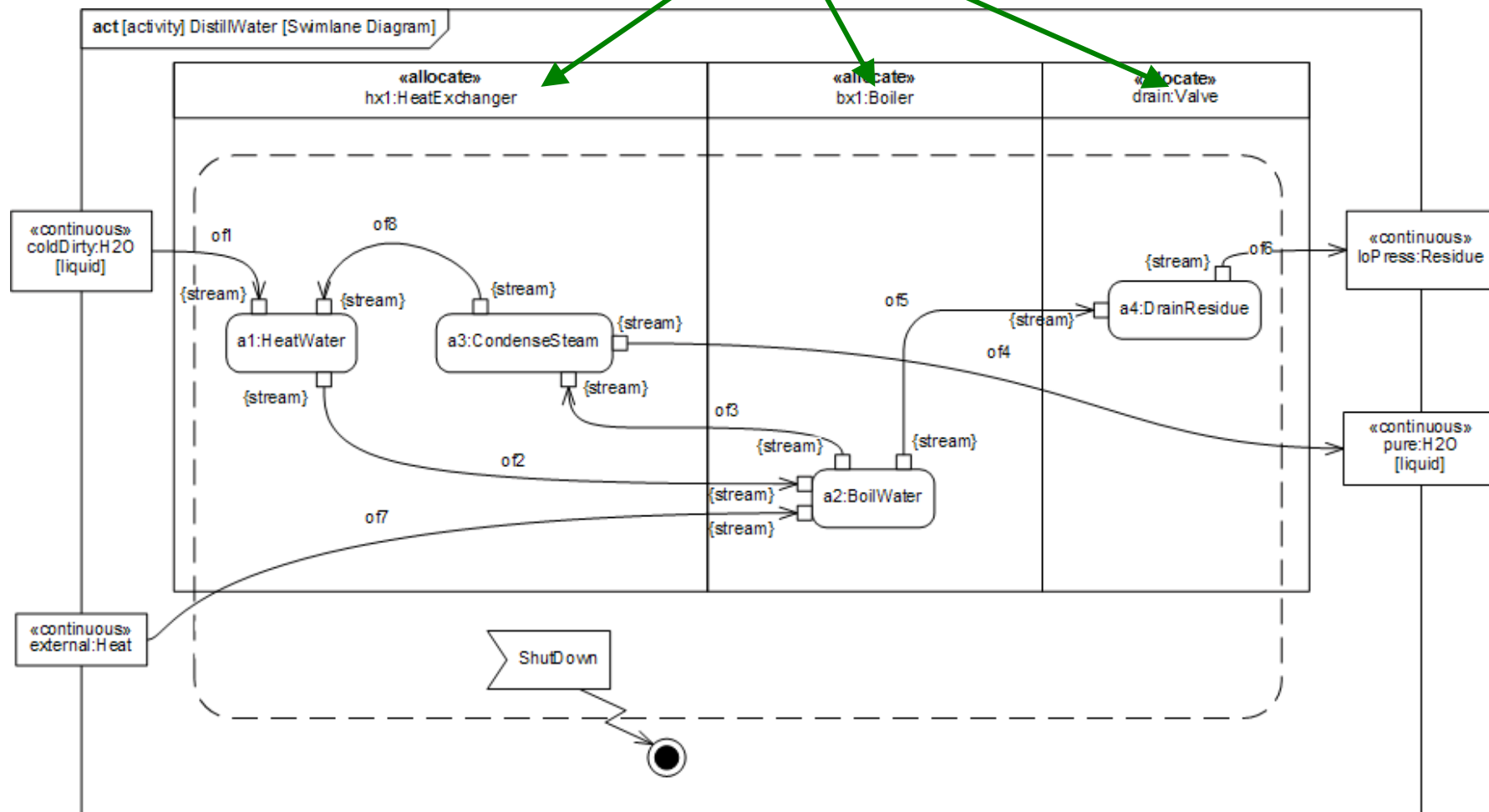
# Distiller Example – Activity Diagram: No Control Flow, ActionPin Notation, Simultaneous Behavior





# Distiller Example – Activity Diagram (with Swimlanes): DistillWater

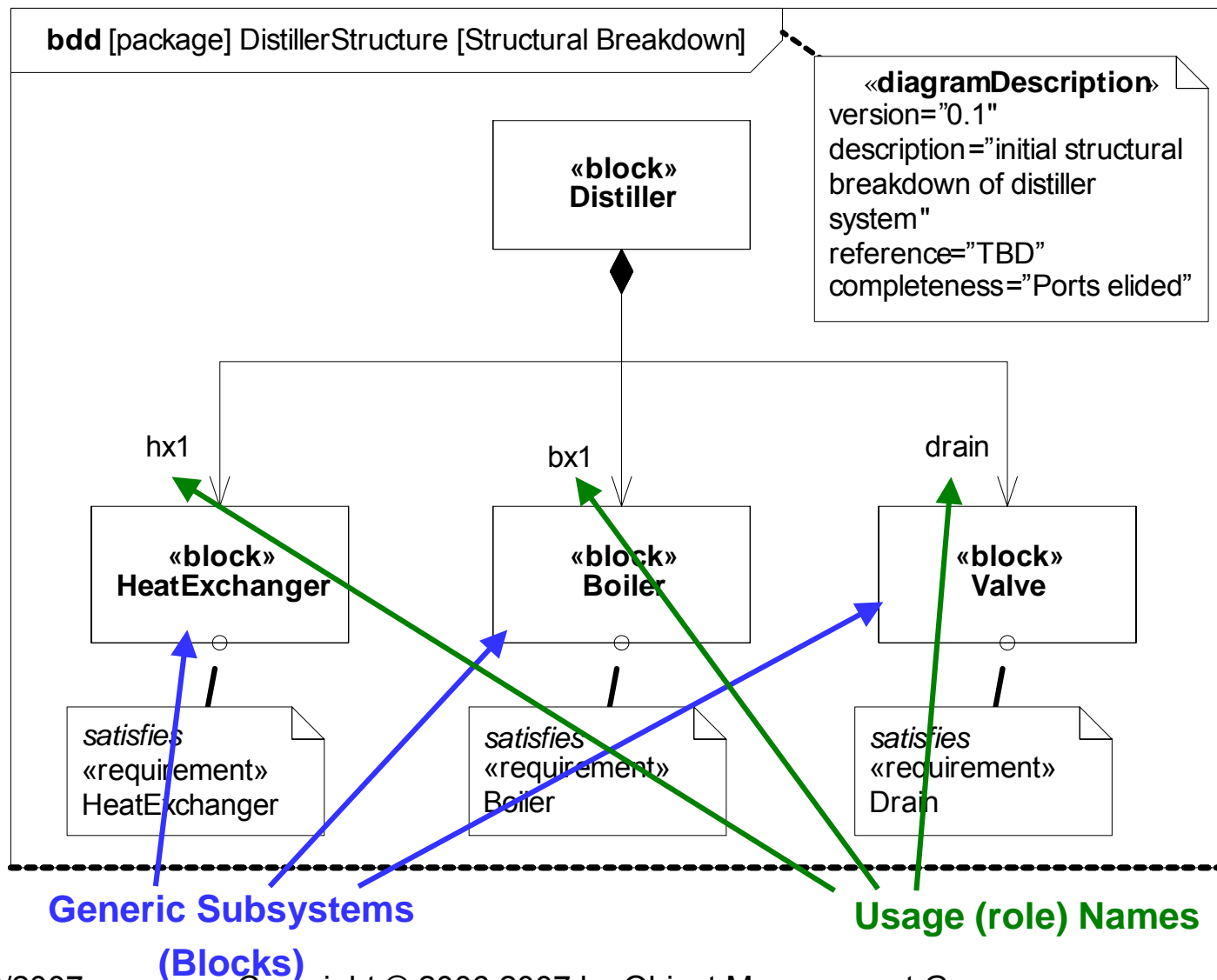
**Parts**



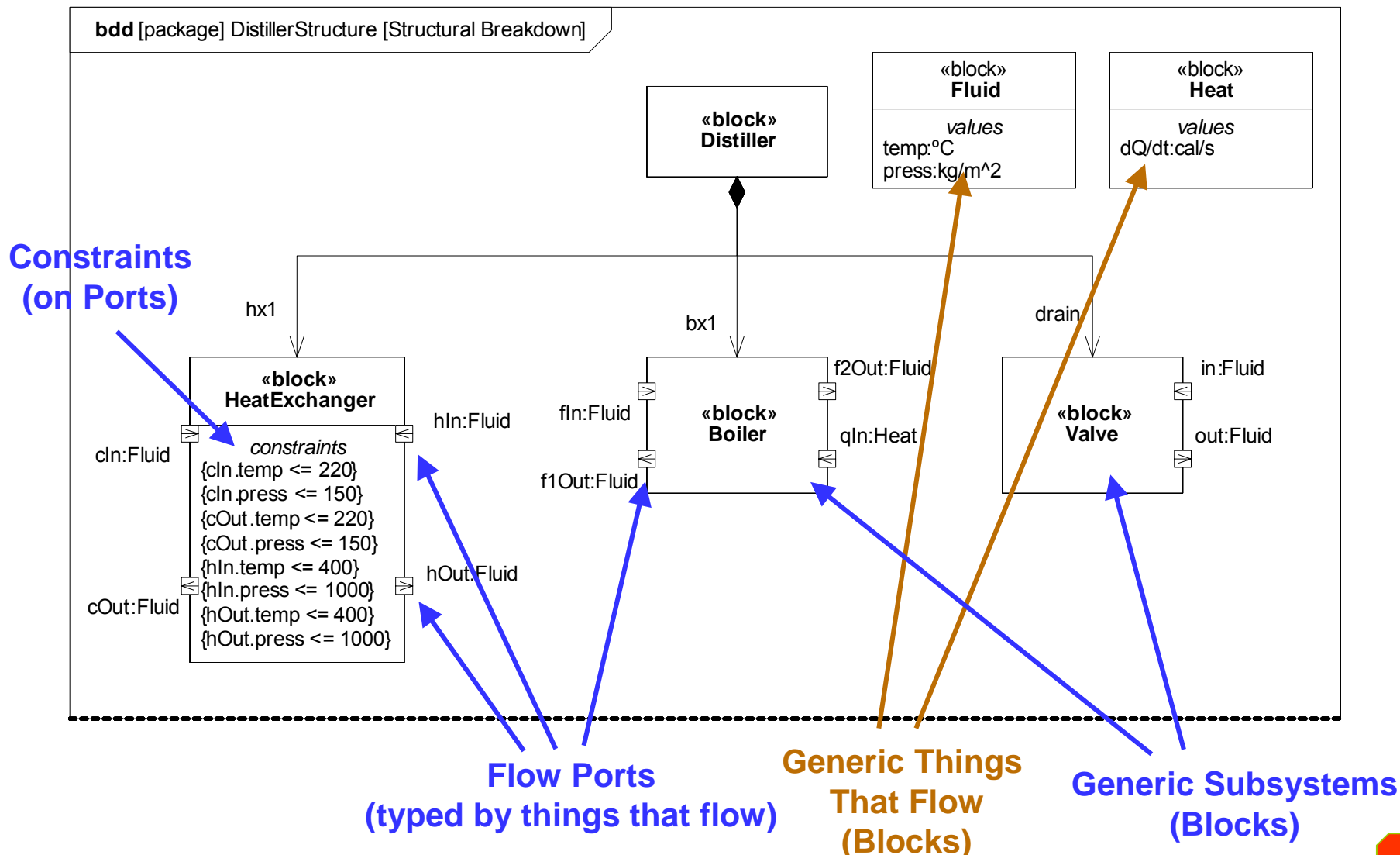
Allocated ibd



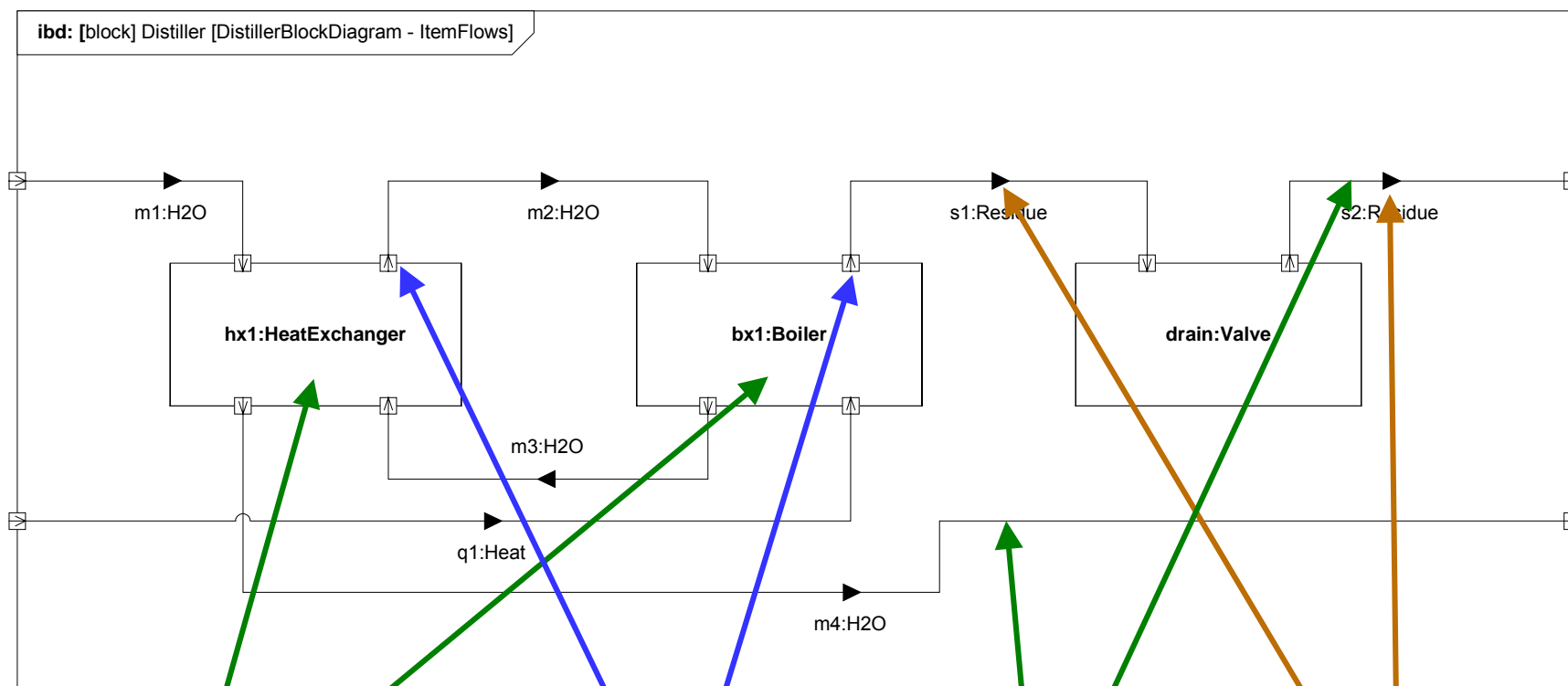
# Distiller Example – Block Definition Diagram: DistillerStructure



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



# Distiller Example – Internal Block Diagram: Distiller Initial Design



**Parts**  
(Blocks used  
in context)

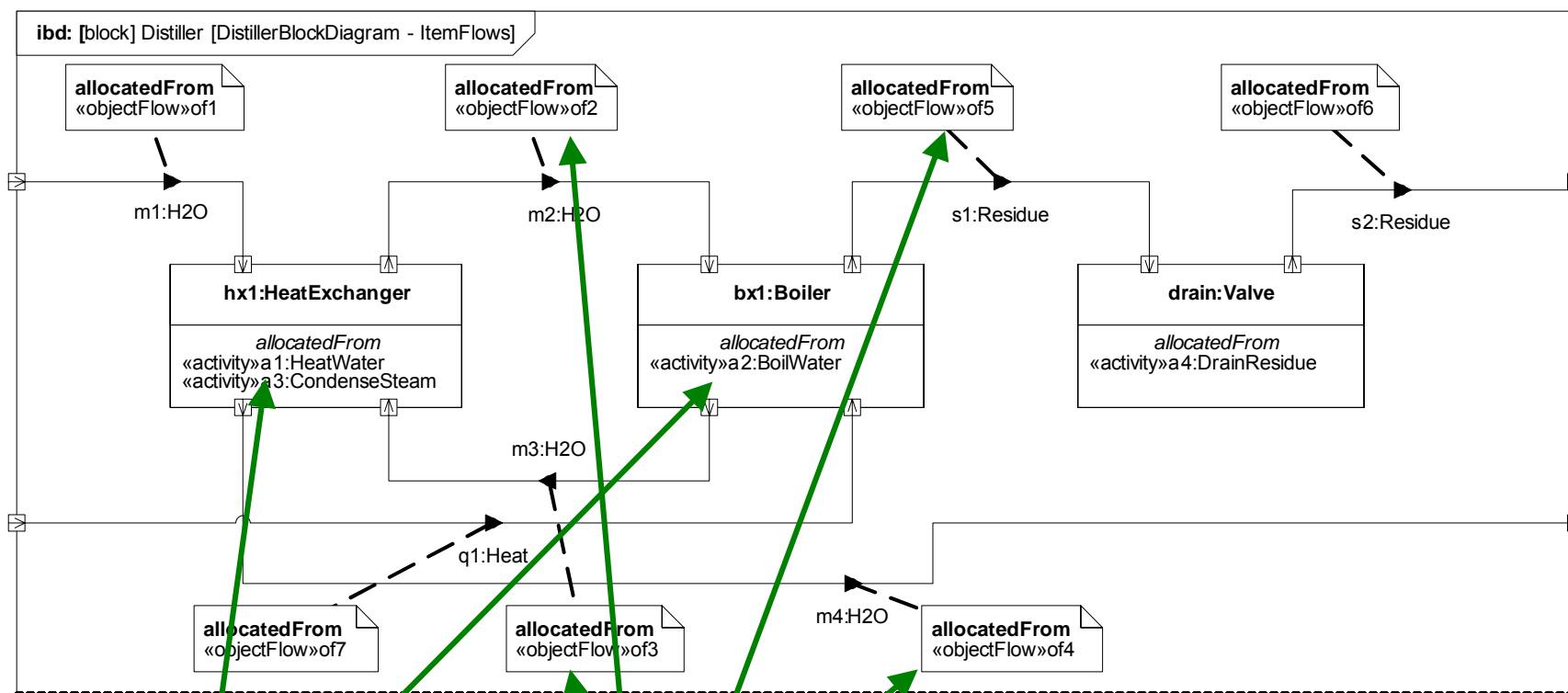
**Flow Ports**

**Connectors**

**Things That Flow  
In Context  
(ItemFlows)**



# Distiller Example –Internal Block Diagram: Distiller with Allocation



**Exercise for student:**  
 Is allocation complete?  
 Where is “«objectFlow»of8”?

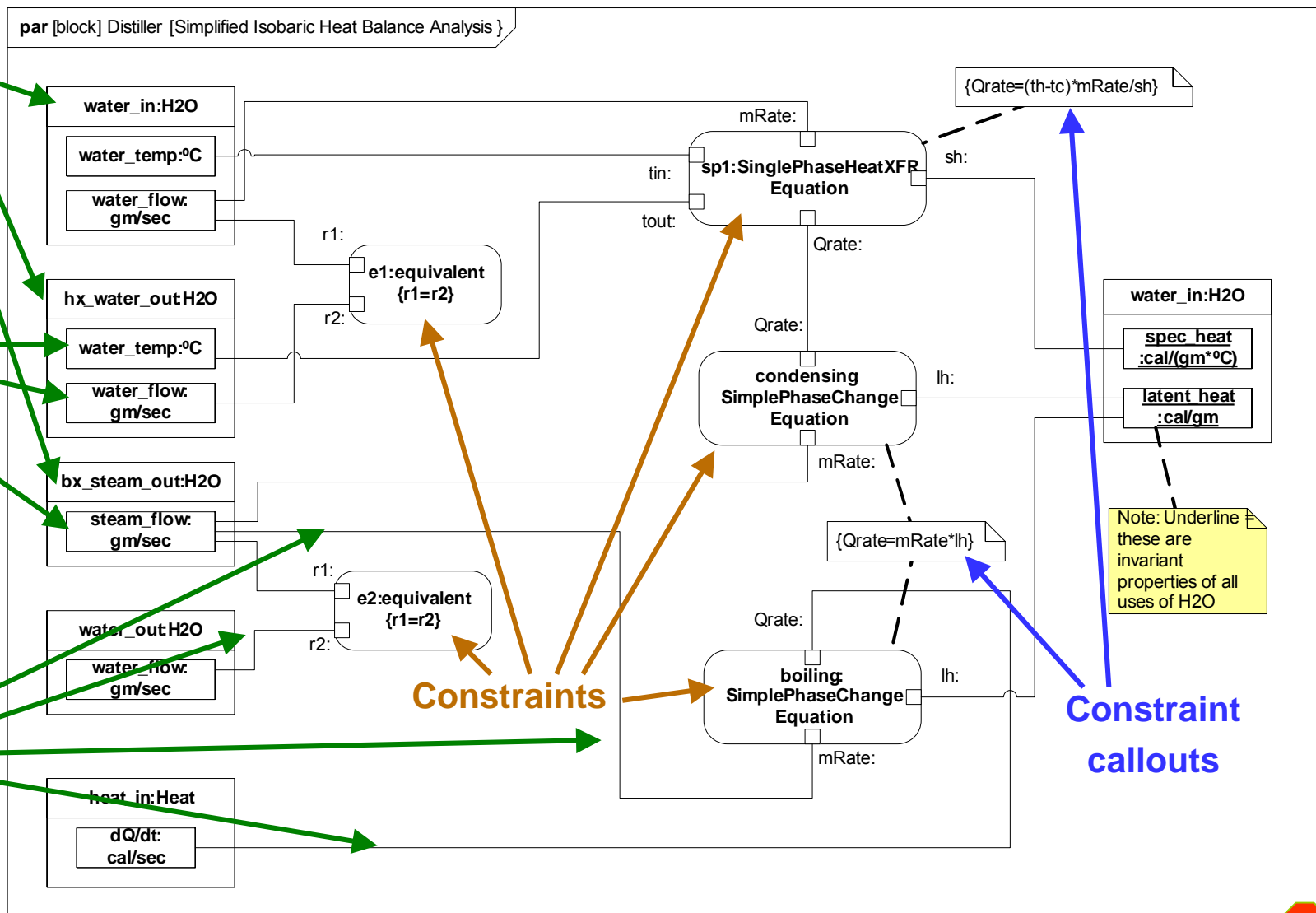


# Distiller Example – Parametric Diagram: Heat Balance Equations

Parts or  
ItemFlows

Value  
Properties

Value  
Bindings



# Distiller Example – Heat Balance Results

**table** IsobaricHeatBalance1 [Results of Isobaric Heat Balance]

specific heat cal/gm-°C	1
latent heat cal/cm	540

Satisfies «requirement»  
WaterSpecificHeat

Satisfies «requirement»  
WaterHeatOfVaporization

Satisfies «requirement»  
WaterInitialTemp

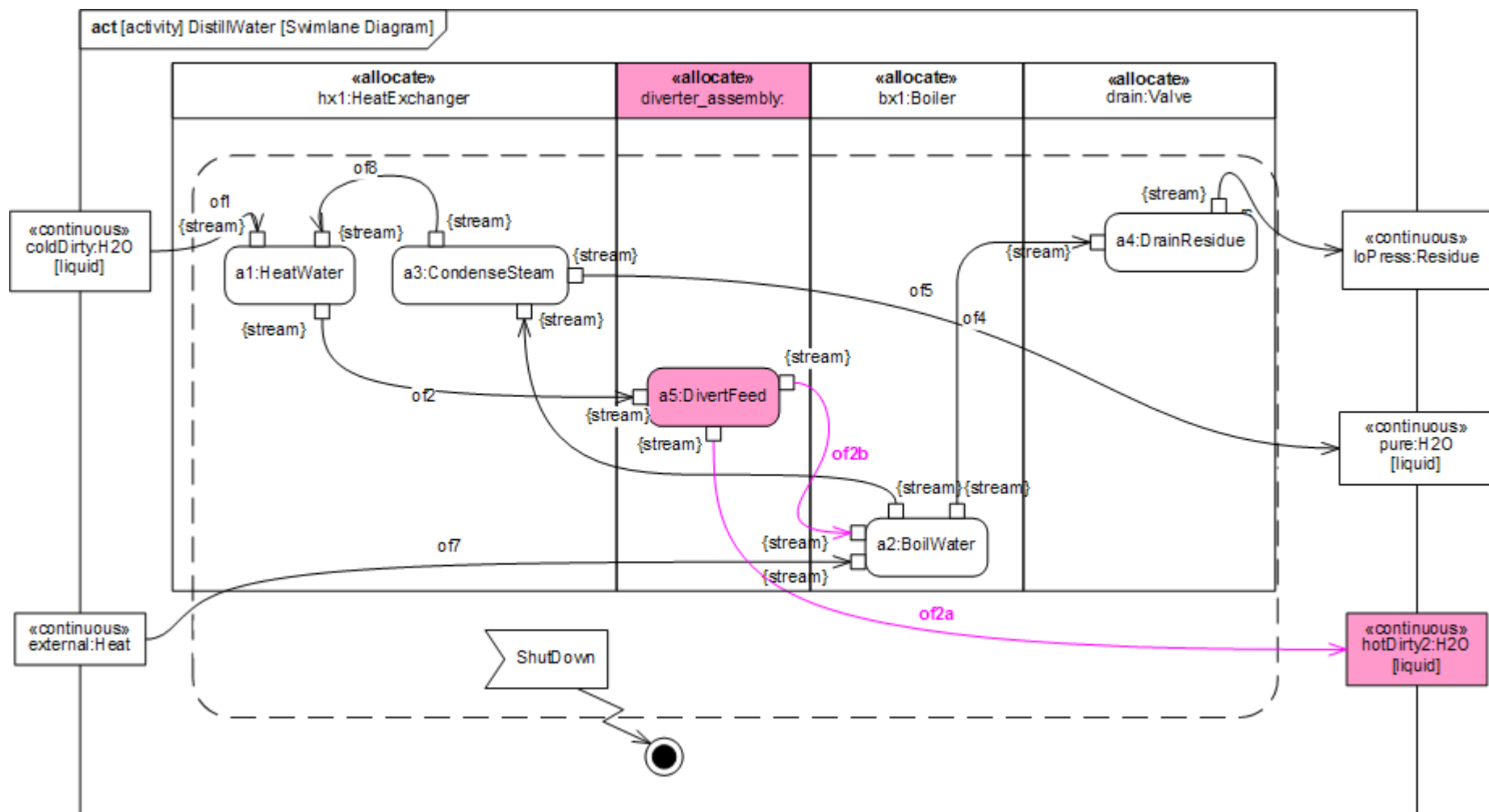
	water_in	hx_water_out	bx_water_in	bx_steam_out	water_out
mass flow rate gm/sec	6.75	6.75	1	1	1
temp °C	20	100	100	100	100

dQ/dt cooling water cal/sec	540
dQ/dt steam-condensate cal/sec	540
condenser efficiency	1
heat deficit	0

dQ/dt condensate-steam cal/sec	540
boiler efficiency	1
dQ/dt in boiler cal/sec	540

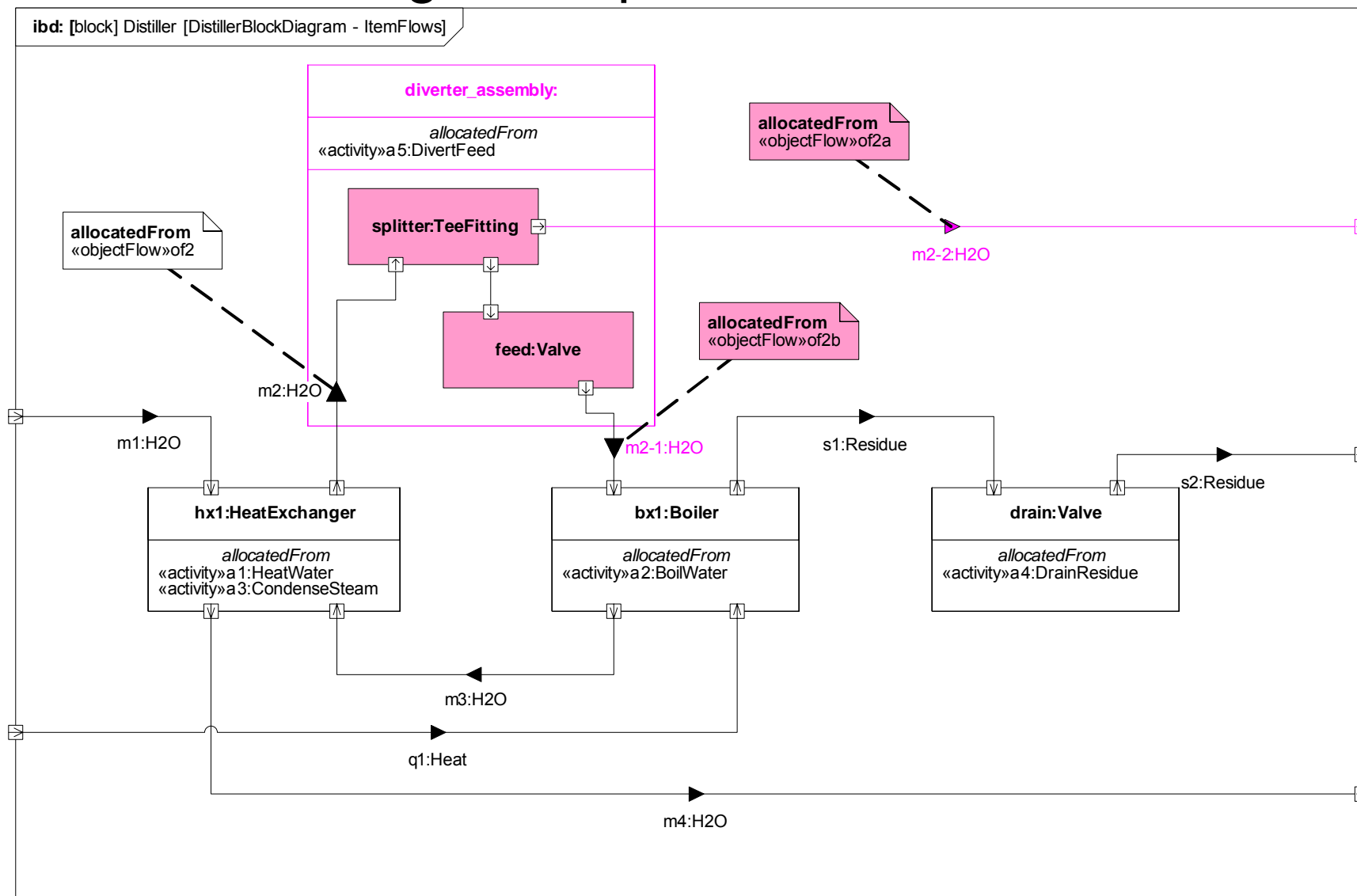
Note: Cooling water  
needs to have 6x flow  
of steam!  
Need bypass between  
hx\_water\_out and  
bx\_water\_in!

# Distiller Example – Activity Diagram: Updated DistillWater

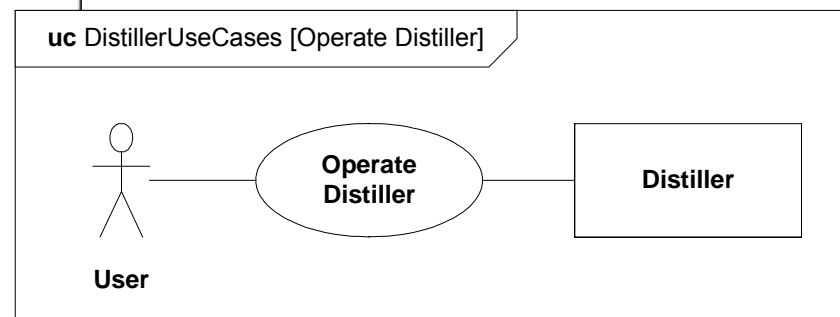
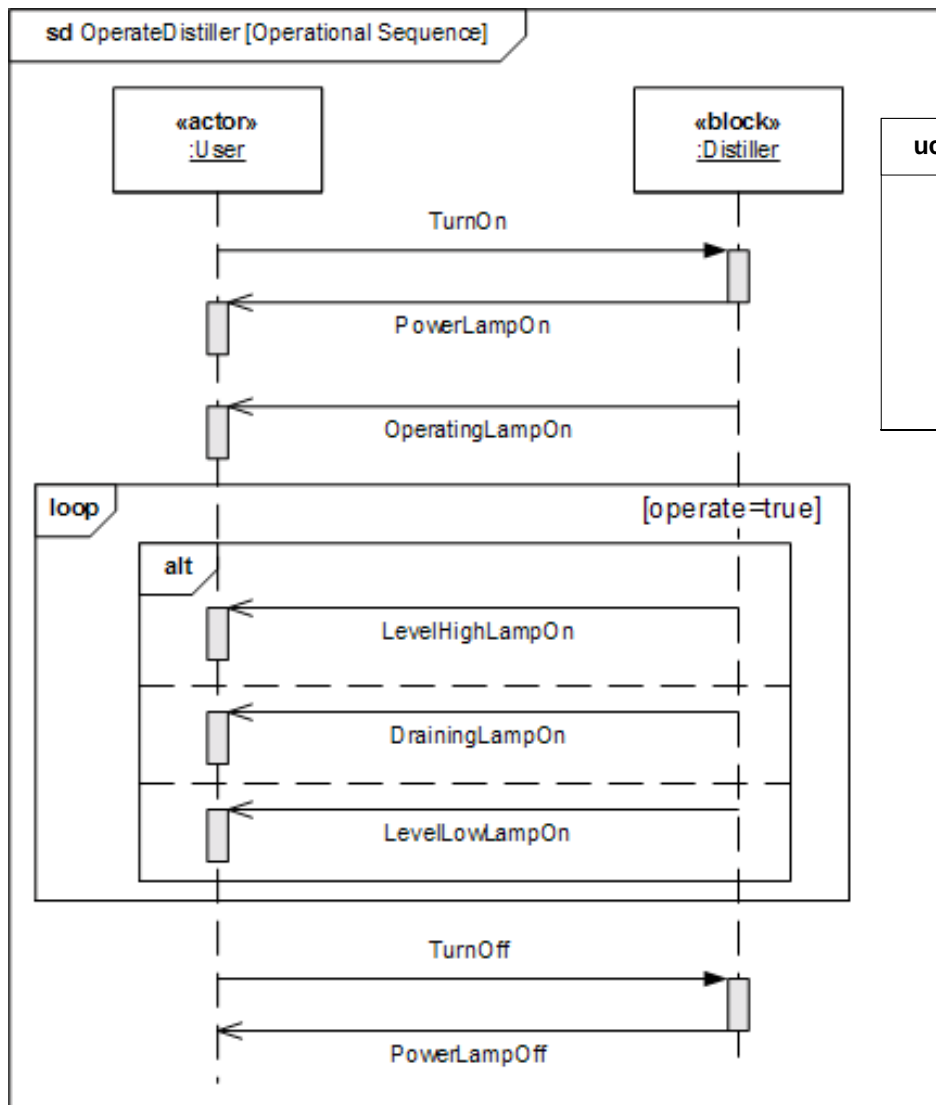




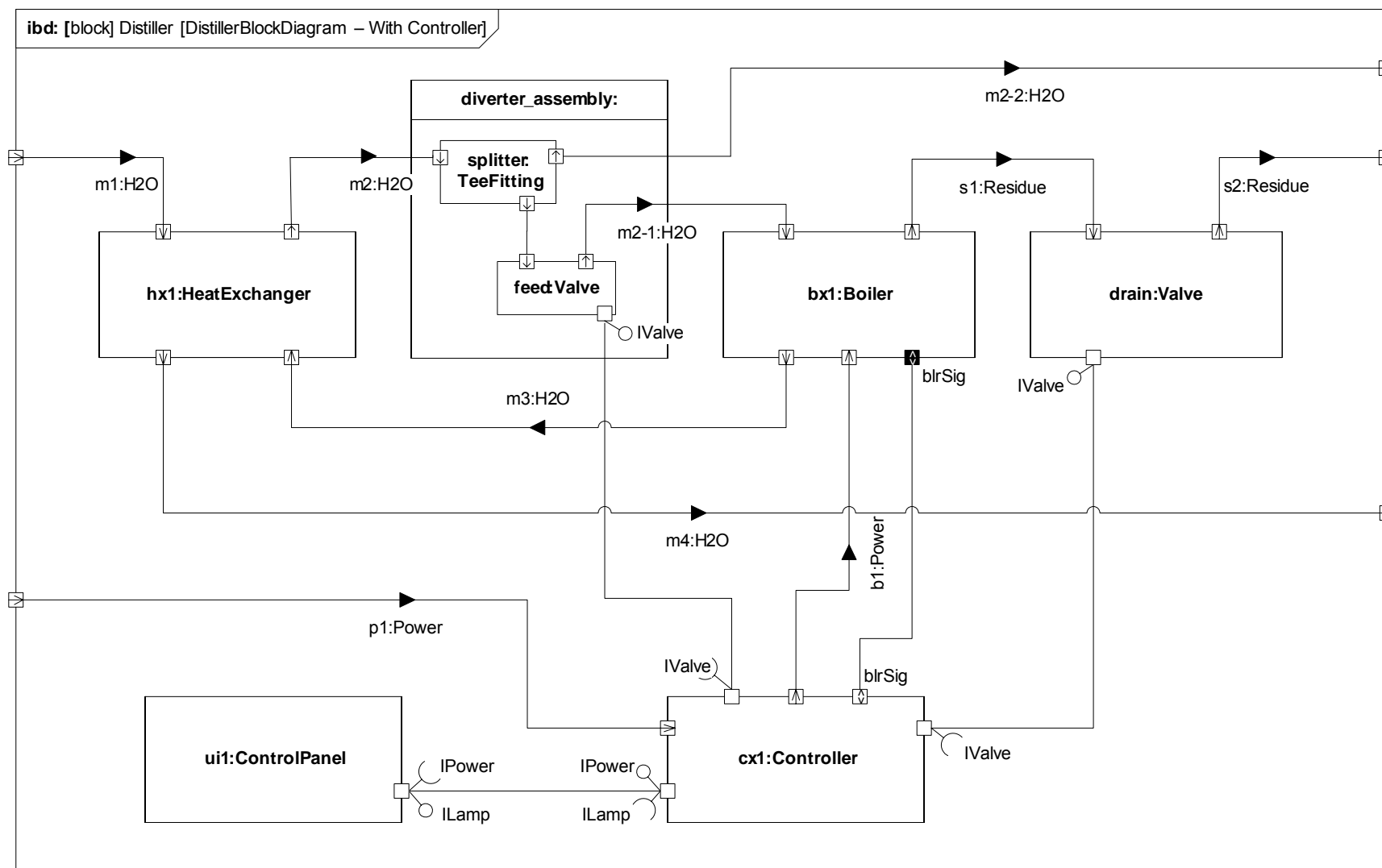
# Distiller Example – Internal Block Diagram: Updated Distiller



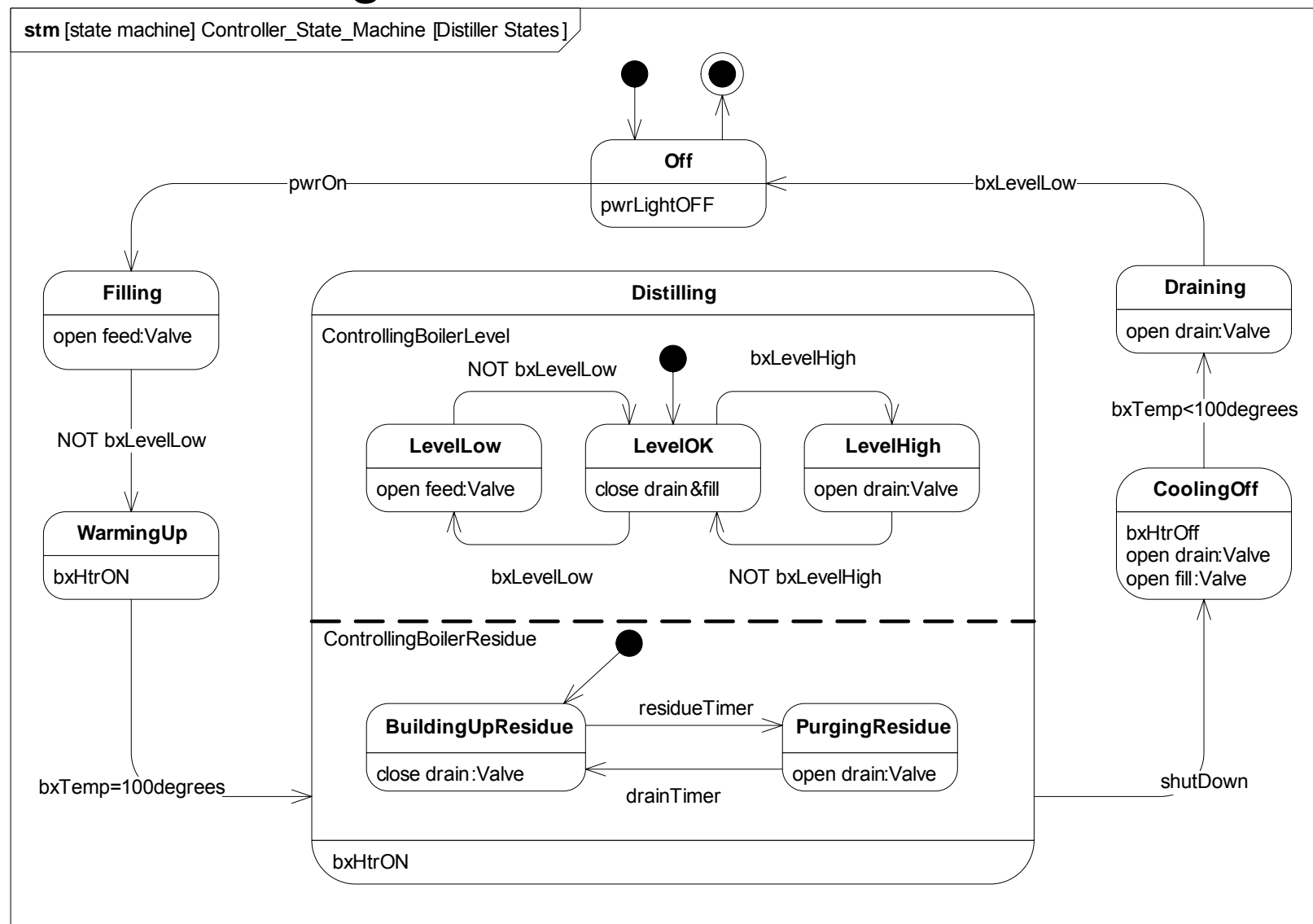
# Distiller Example – Use Case and Sequence Diagrams



# Distiller Example – Internal Block Diagram: Distiller Controller

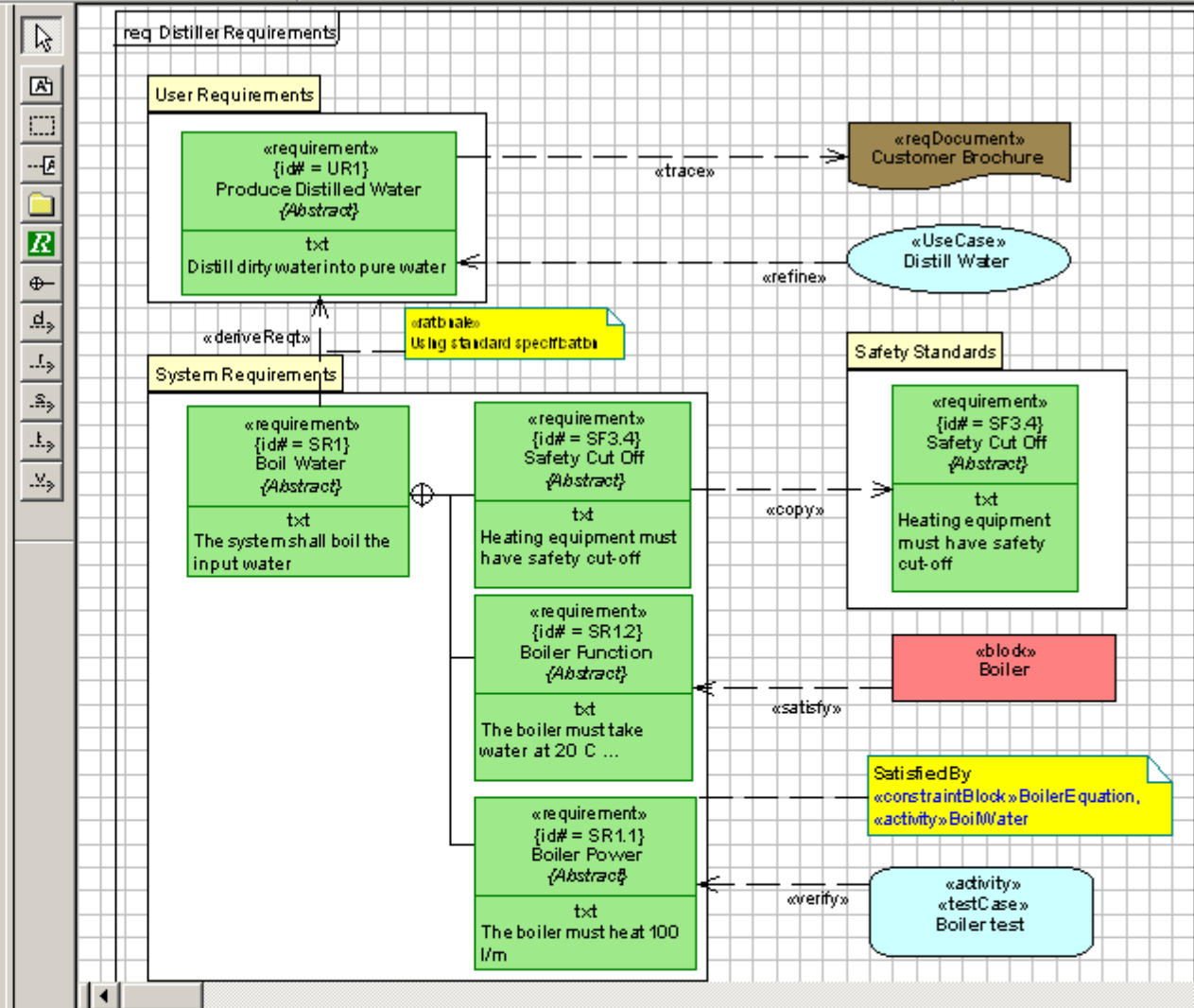


# Distiller Example – State Machine Diagram: Distiller Controller



requirement

- Acceleration (HSUWModel::HSUW Requirements::HSUW Requirements)
  - derivedRequirements
  - refinedBy
  - requirementTables
  - satisfactionMatrices
  - satisfiedBy
  - subRequirements
- Boil Water (INCOSE Challenge::Requirements::System Requirements)
  - derivedRequirements
  - refinedBy
  - requirementTables
  - satisfactionMatrices
  - satisfiedBy
  - subRequirements
  - tracedTo
  - verifiedBy
- Boiler Function (INCOSE Challenge::Requirements::System Requirements)
  - derivedRequirements
  - refinedBy
  - requirementTables
  - satisfactionMatrices
  - satisfiedBy
  - subRequirements
  - tracedTo
  - verifiedBy
- Boiler Power (INCOSE Challenge::Requirements::System Requirements)
  - derivedRequirements
  - refinedBy
  - requirementTables
  - satisfactionMatrices
  - satisfiedBy
  - subRequirements
  - tracedTo
  - verifiedBy



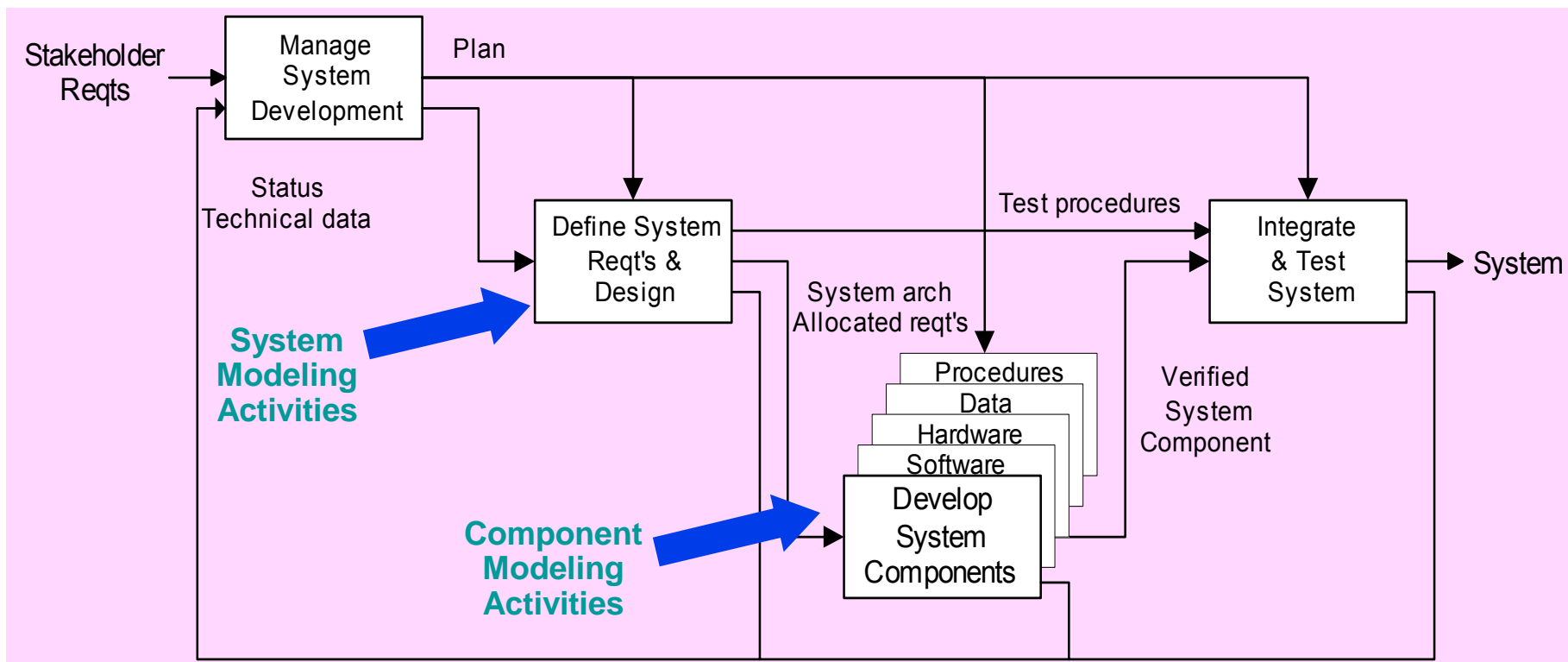
Tag Definition Name	Tag Value
id#	SR1.1
txt	The boiler must heat 100 l/m
satisfiedBy	BoilerEquation,BoilWater
refinedBy	
derivedRequirements	
verifiedBy	Boiler test
tracedTo	

Name	Type	Visibility	Changed By
Boiler Function	requirement	Public	ARTISAN_UK\AI...
Boiler Power	requirement	Public	ARTISAN_UK\AI...
Safety Cut Off	requirement	Public	ARTISAN_UK\AI...

**Sample - Artisan Tool**



## OOSEM – ESS Example



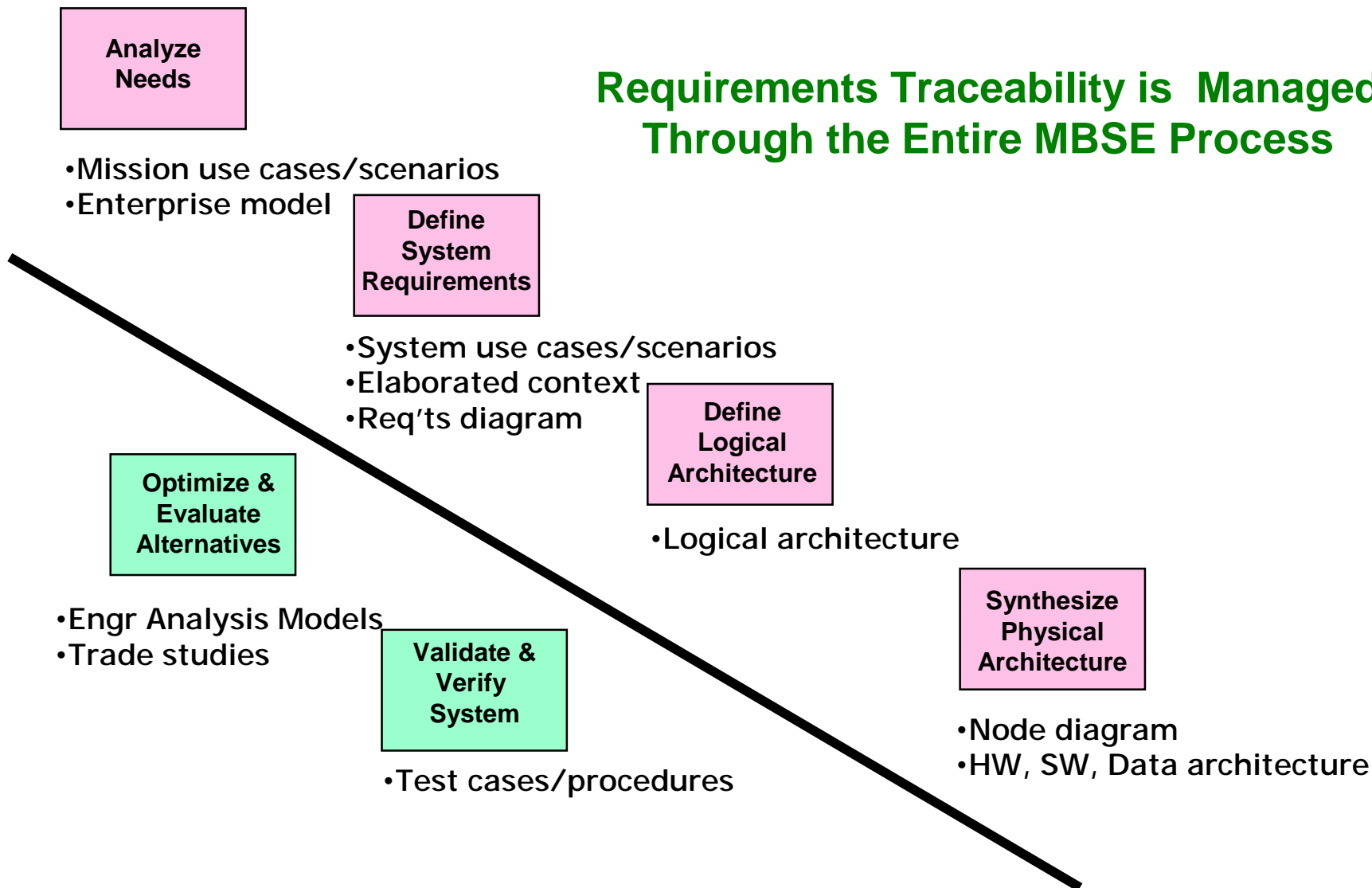
**Integrated Product Development (IPD) is essential to improve communications**

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

# System Modeling Activities – OOSEM

## Integrating MBSE into the SE Process

**Requirements Traceability is Managed Through the Entire MBSE Process**

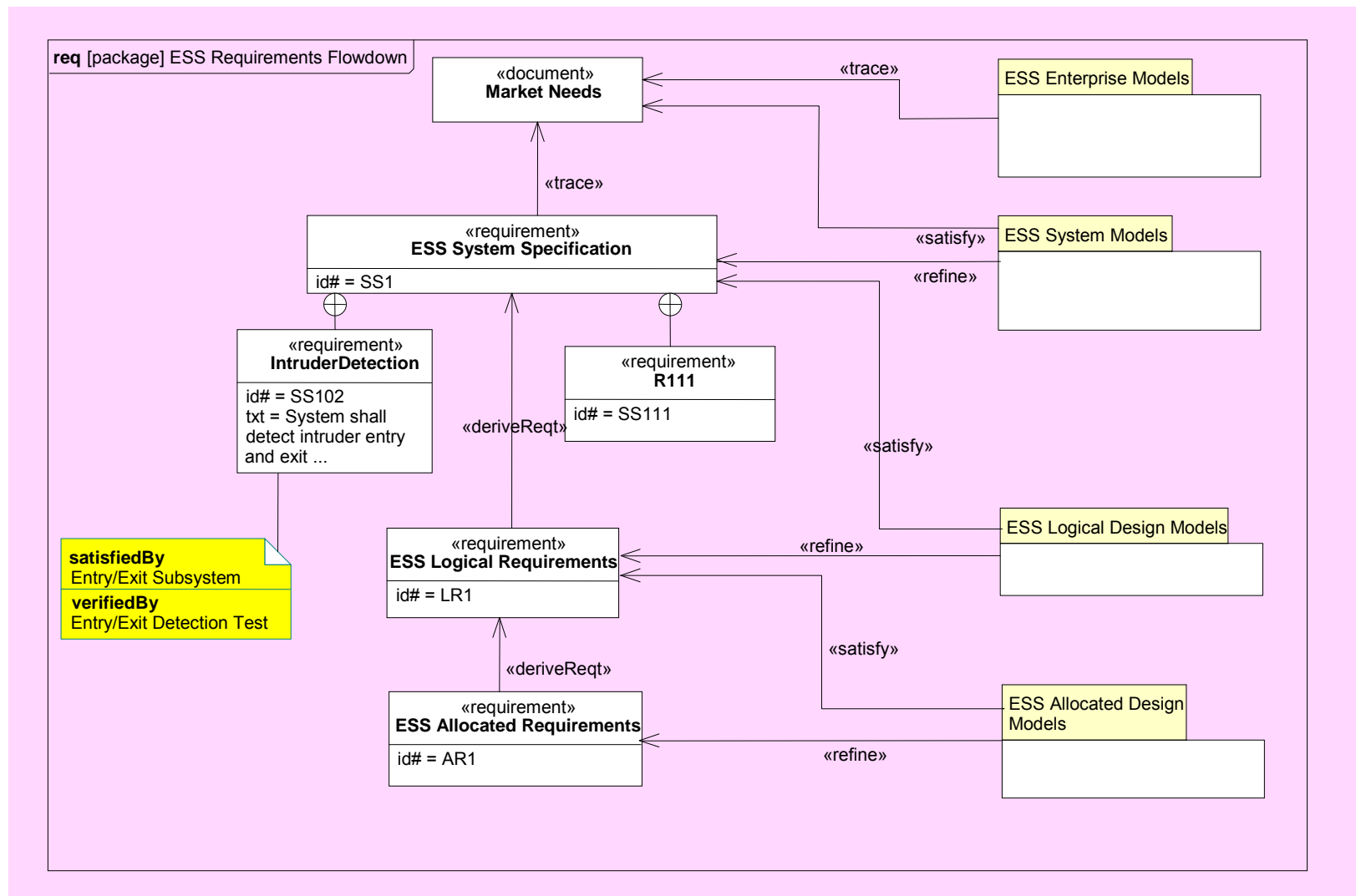




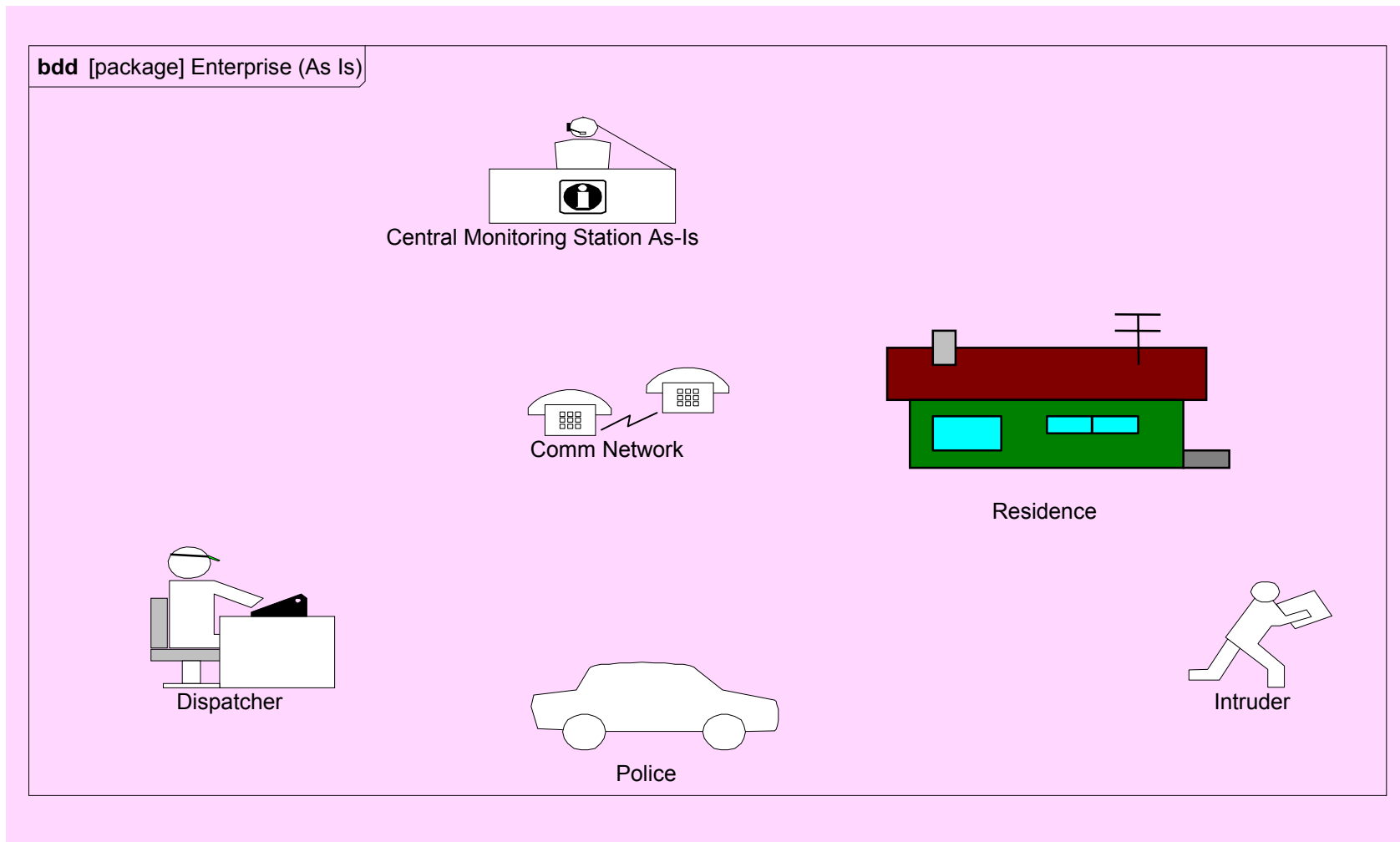
# Enhanced Security System Example

- 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

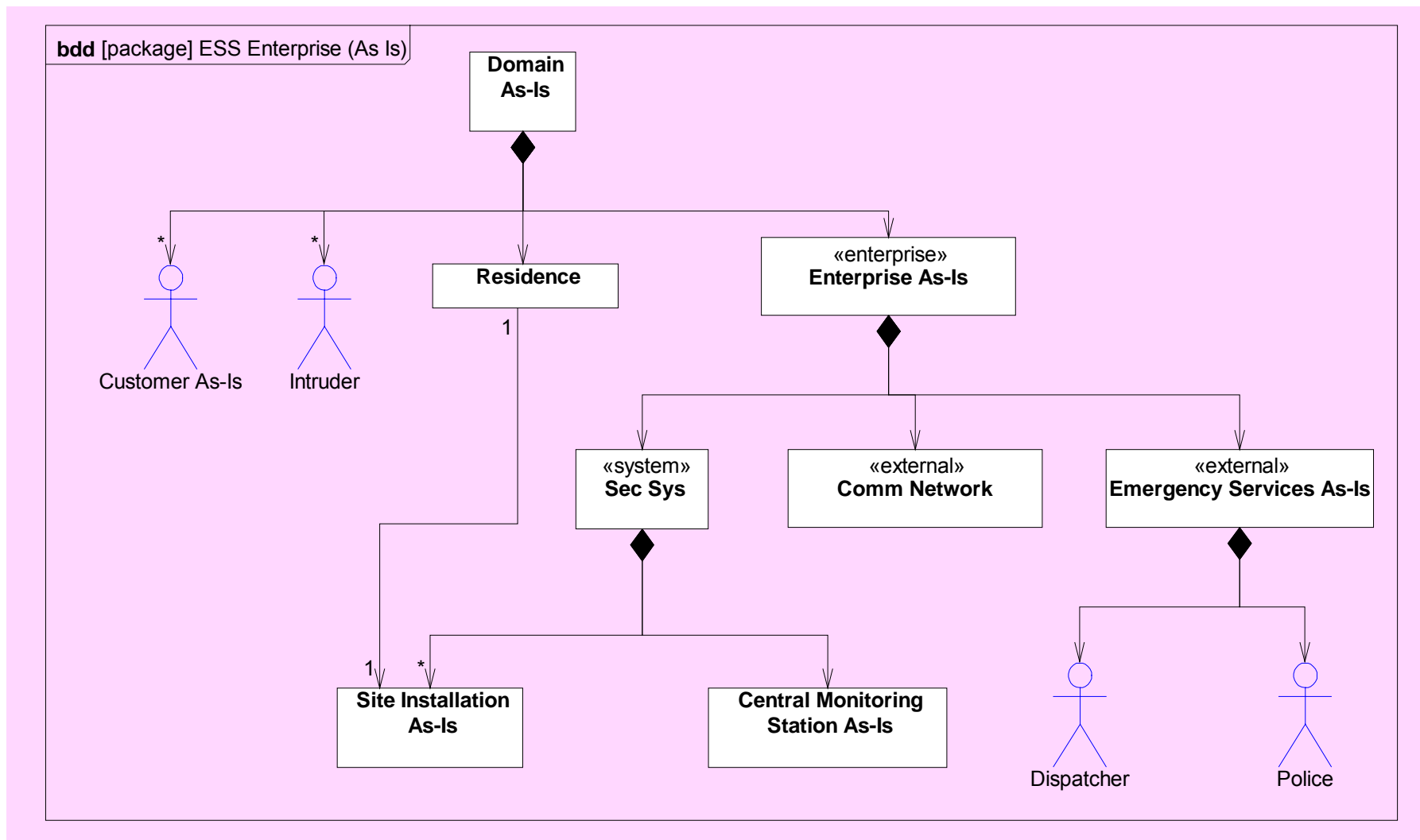
# ESS Requirements Flowdown



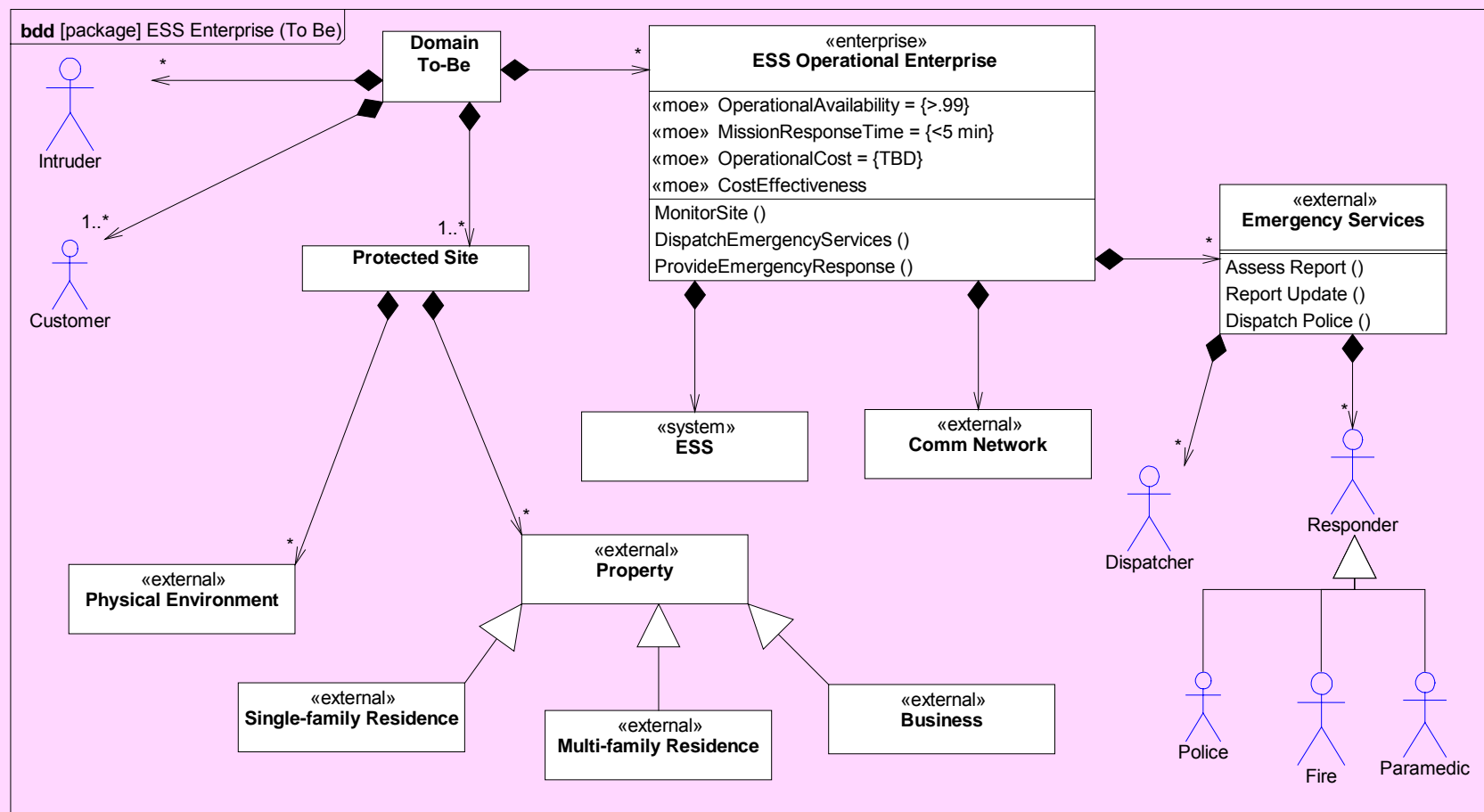
# Operational View Depiction



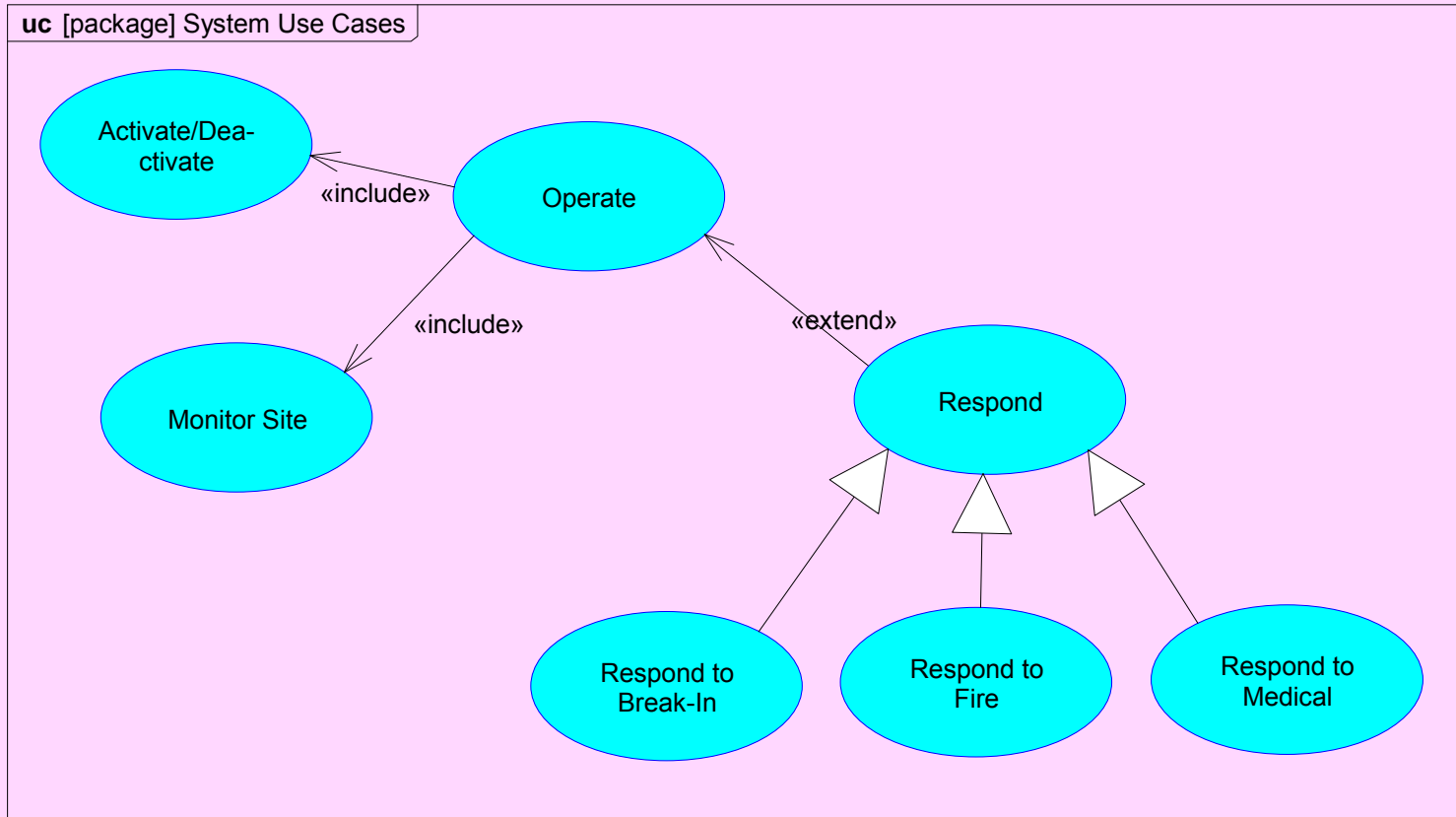
# ESS Enterprise As-Is Model



# ESS Operational Enterprise To-Be Model

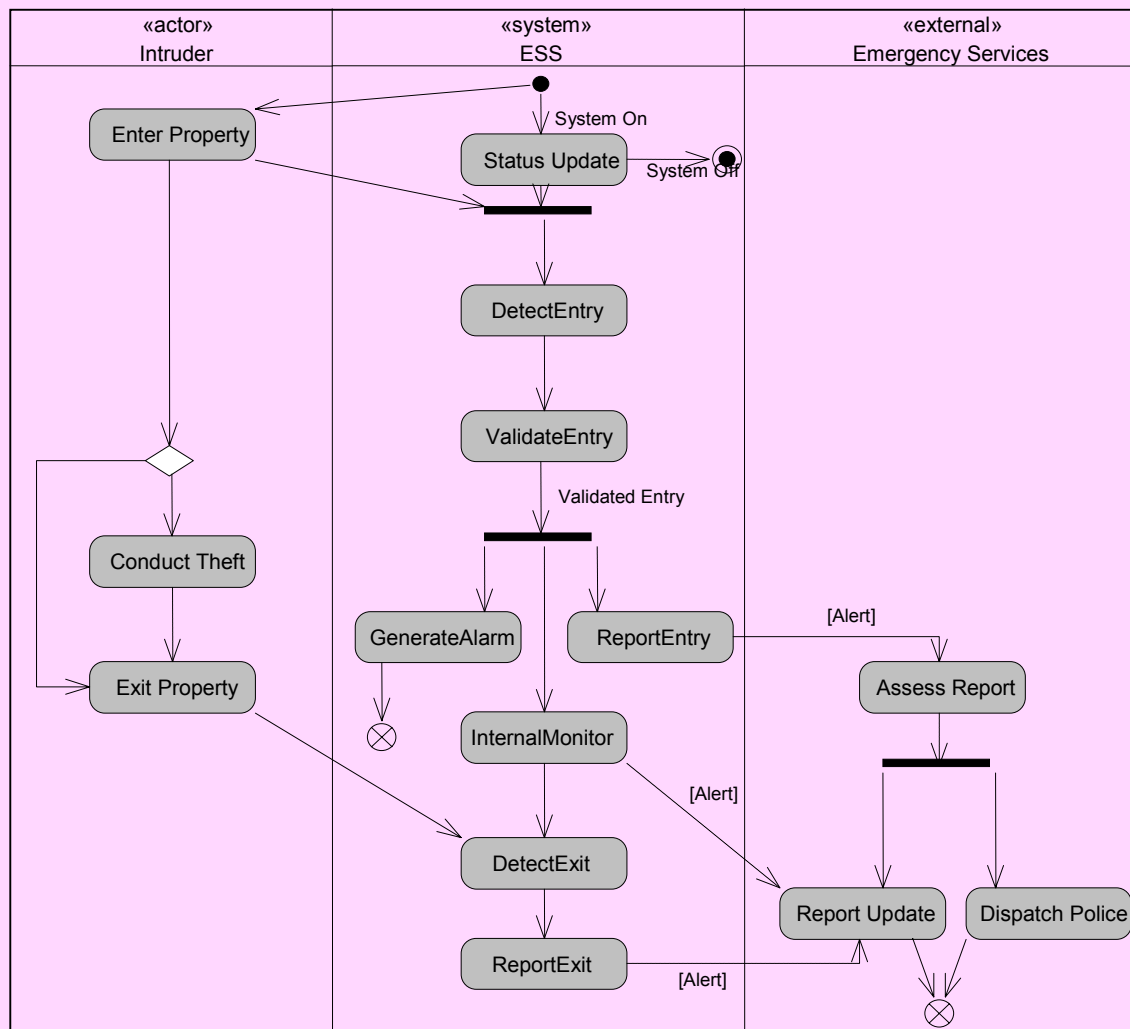


# System Use Cases - Operate

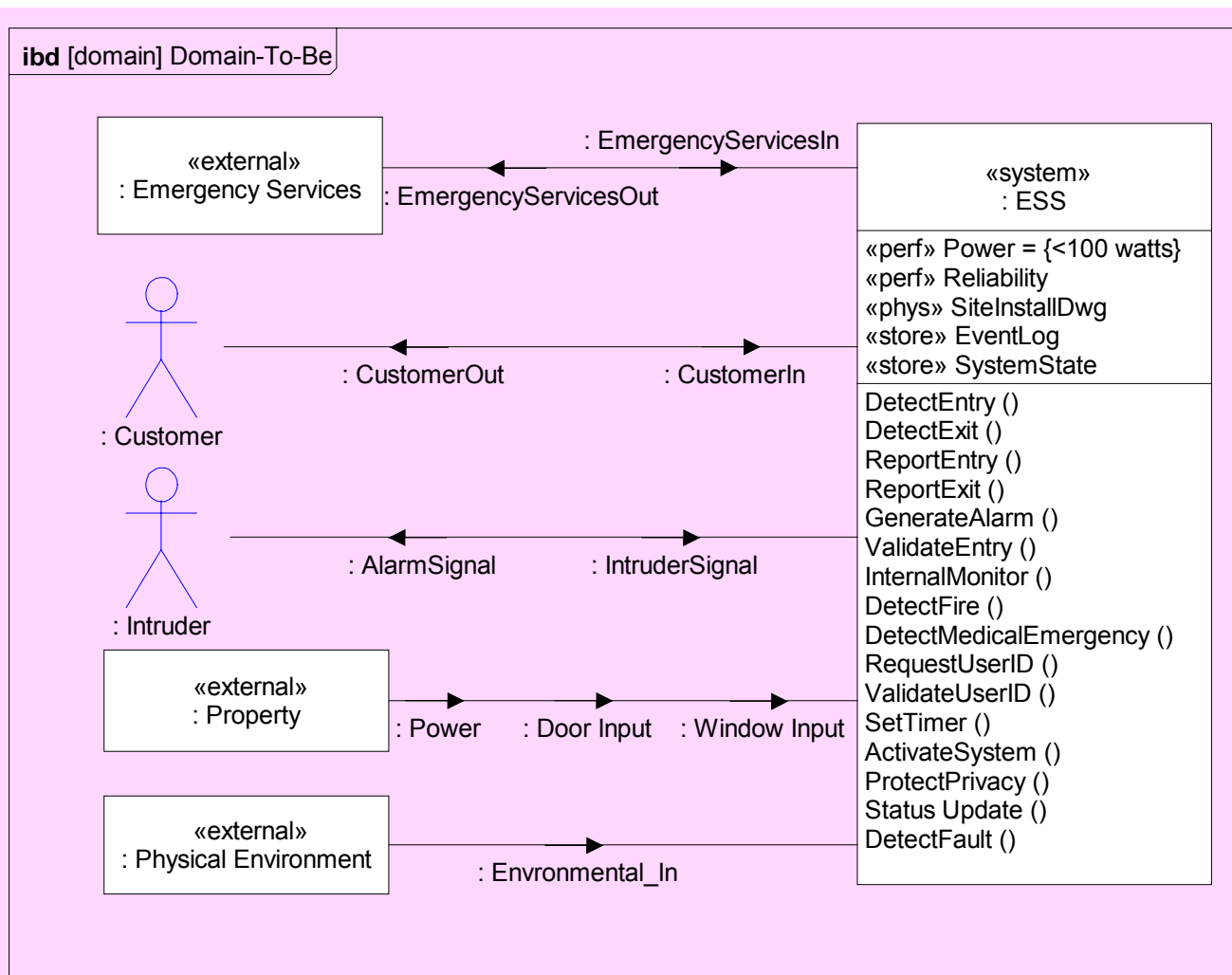


# System Scenario: Activity Diagram Monitor Site (Break-In)

act Monitor Site (break in)

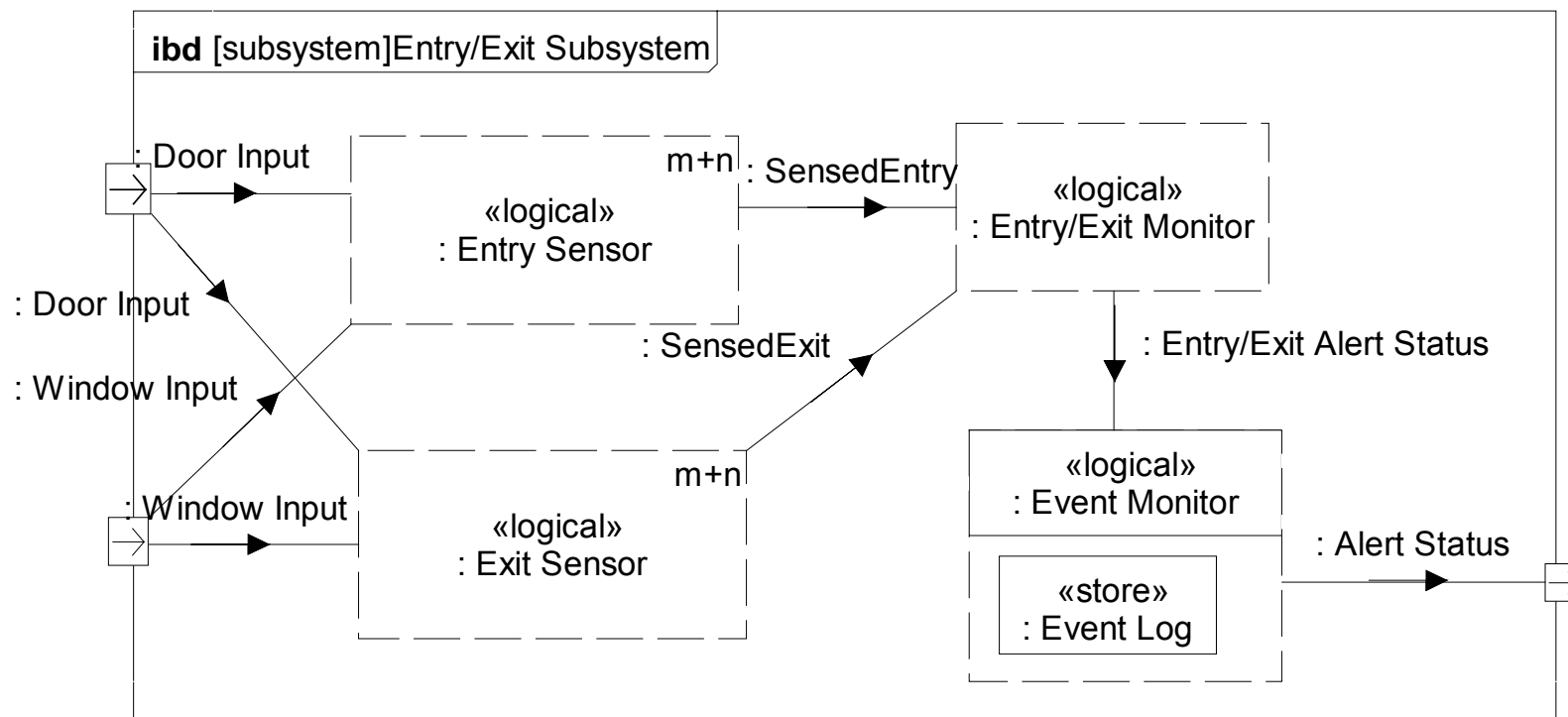


# ESS Elaborated Context Diagram

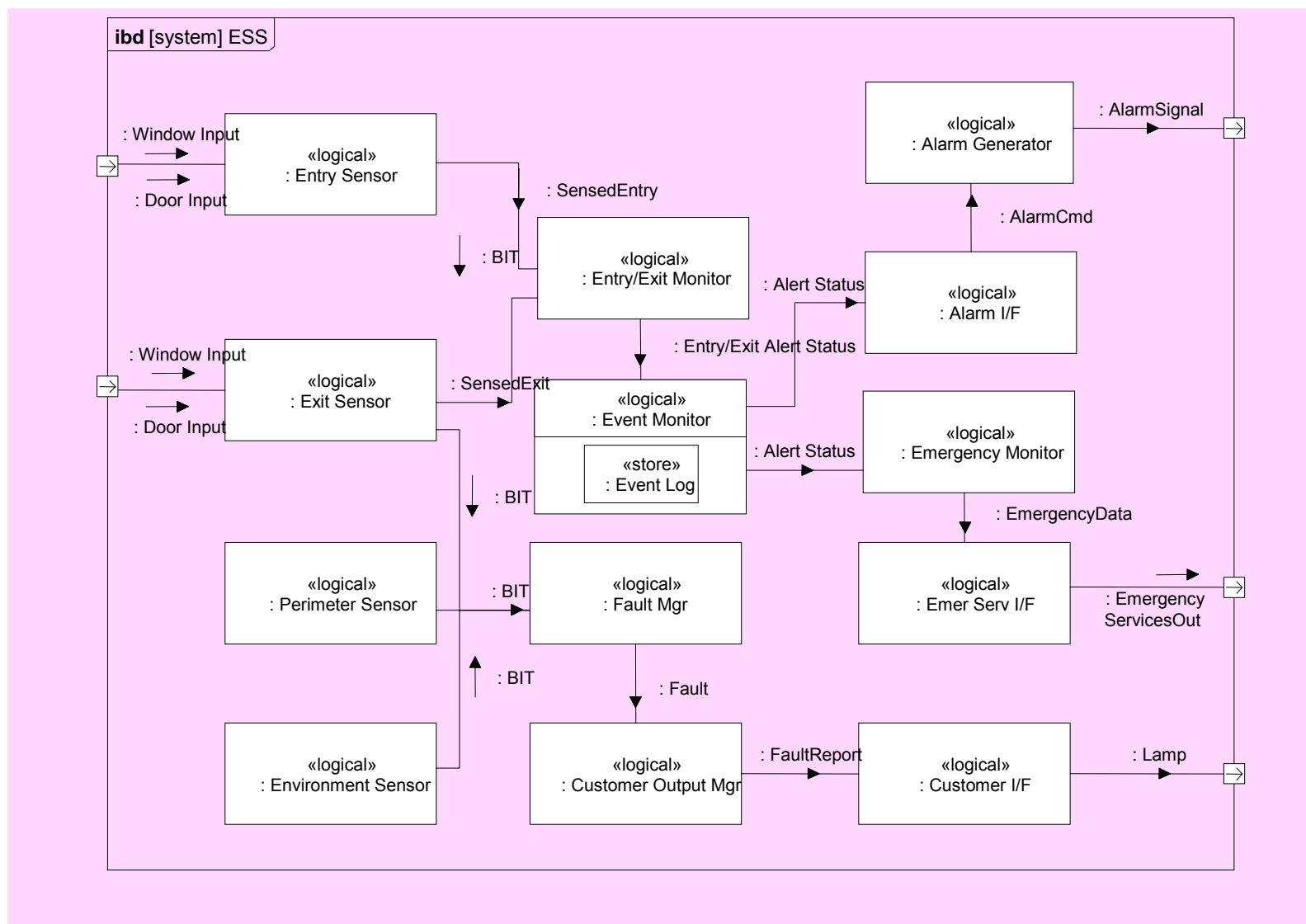




# ESS Logical Design – Example Subsystem



# ESS Logical Design (Partial)

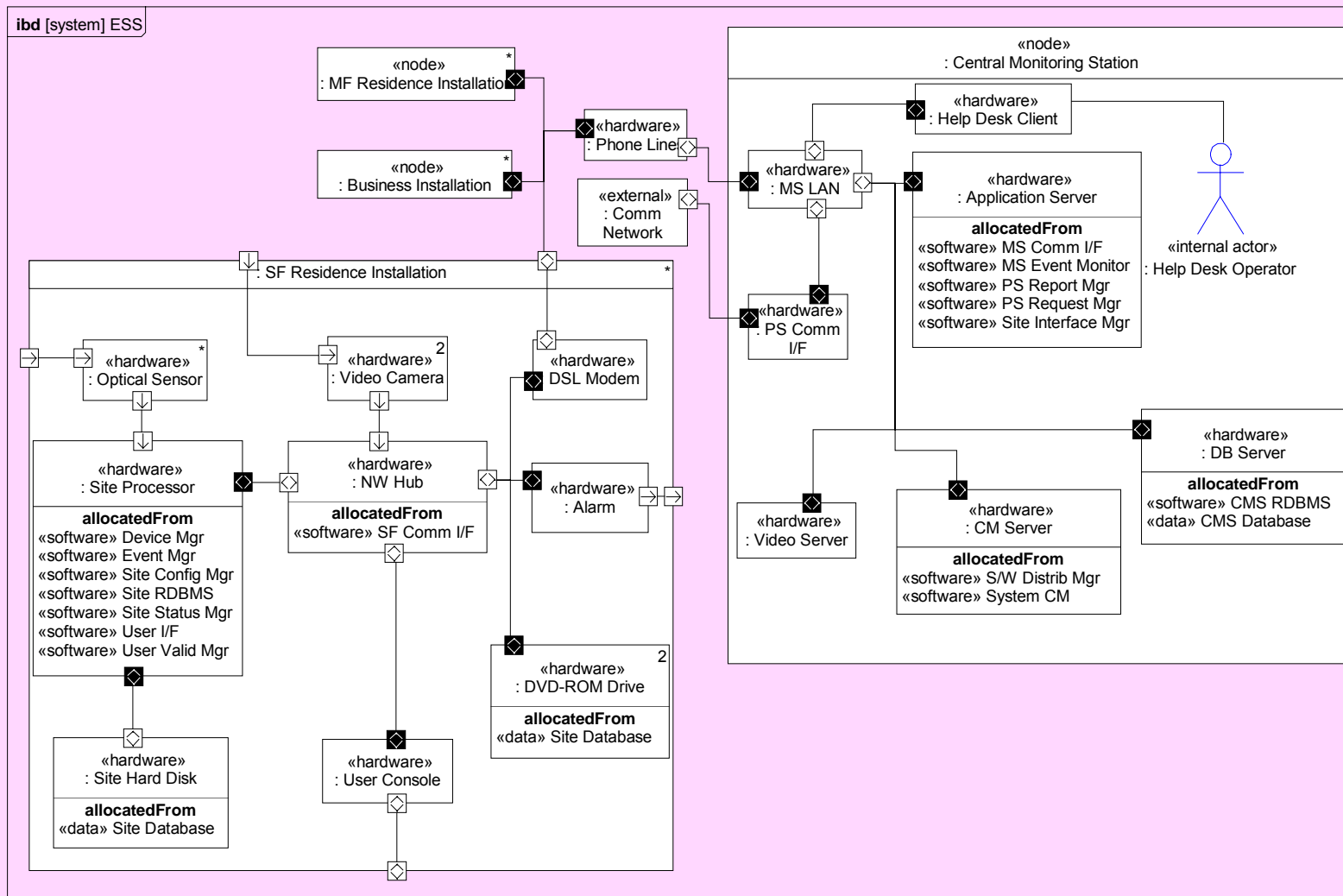


# ESS Allocation Table (partial)

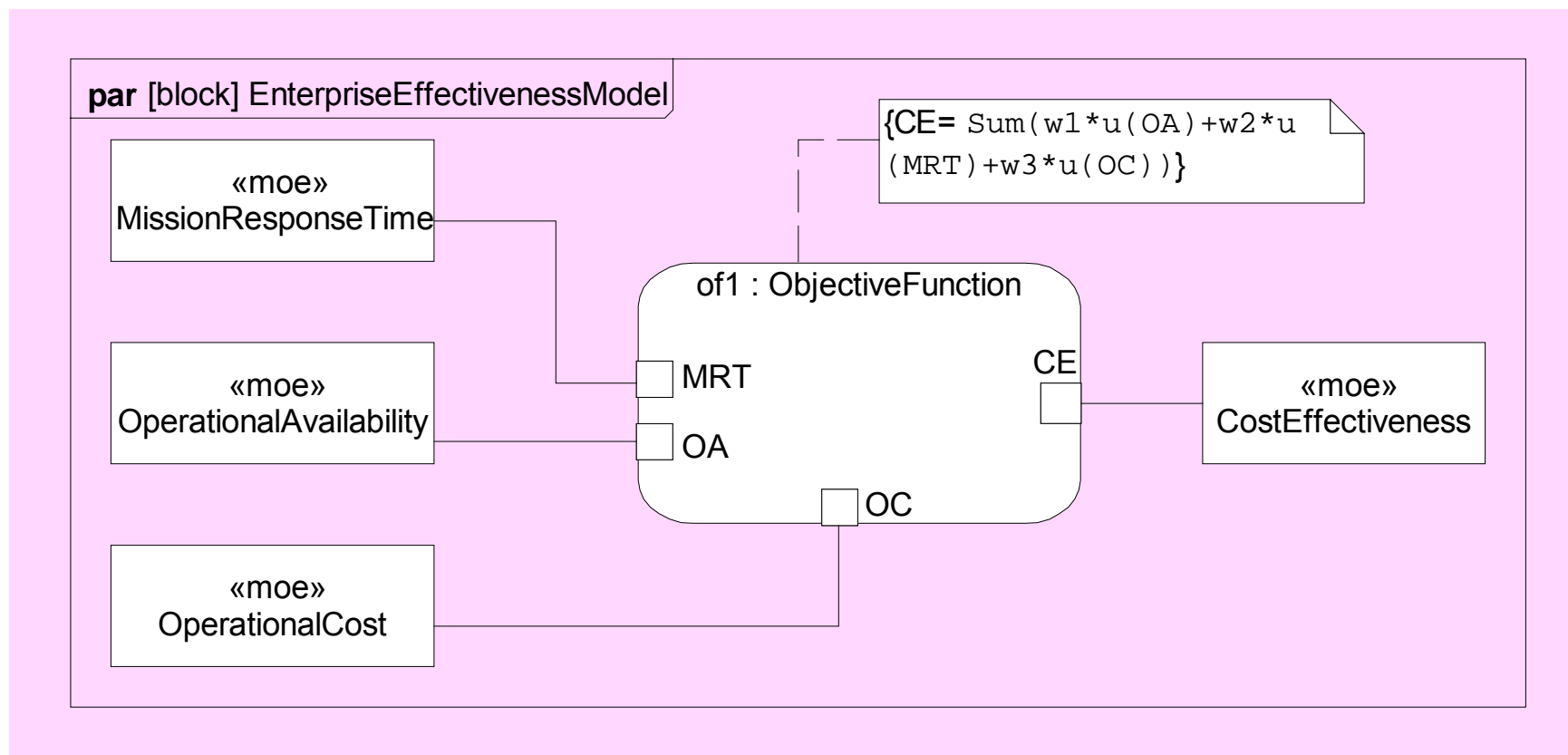
- Allocating Logical Components to HW, SW, Data, and Procedures components

			Logical Components													
Physical Components	Type		Entry Sensor	Exit Sensor	Perimeter Sensor	Entry/Exit Monitor	Event Monitor	Site Comms I/F	Event Log	Customer I/F	Customer Output Mgr	System Status	Fault Mgr	Alarm Generator	Alarm I/F	
	«software»	Device Mgr														X
		SF Comm I/F							X							
		User I/F										X				
		Event Mgr				X	X									
		Site Status Mgr												X		
		Site RDBMS								X			X			
		CMS RDBMS								X						
	«data»	Video File								X						
		CMS Database								X						
		Site Database								X			X			
«hardware»	Optical Sensor	X	X													
	DSL Modem							X								
	User Console									X						
	Video Camera			X												
	Alarm													X		

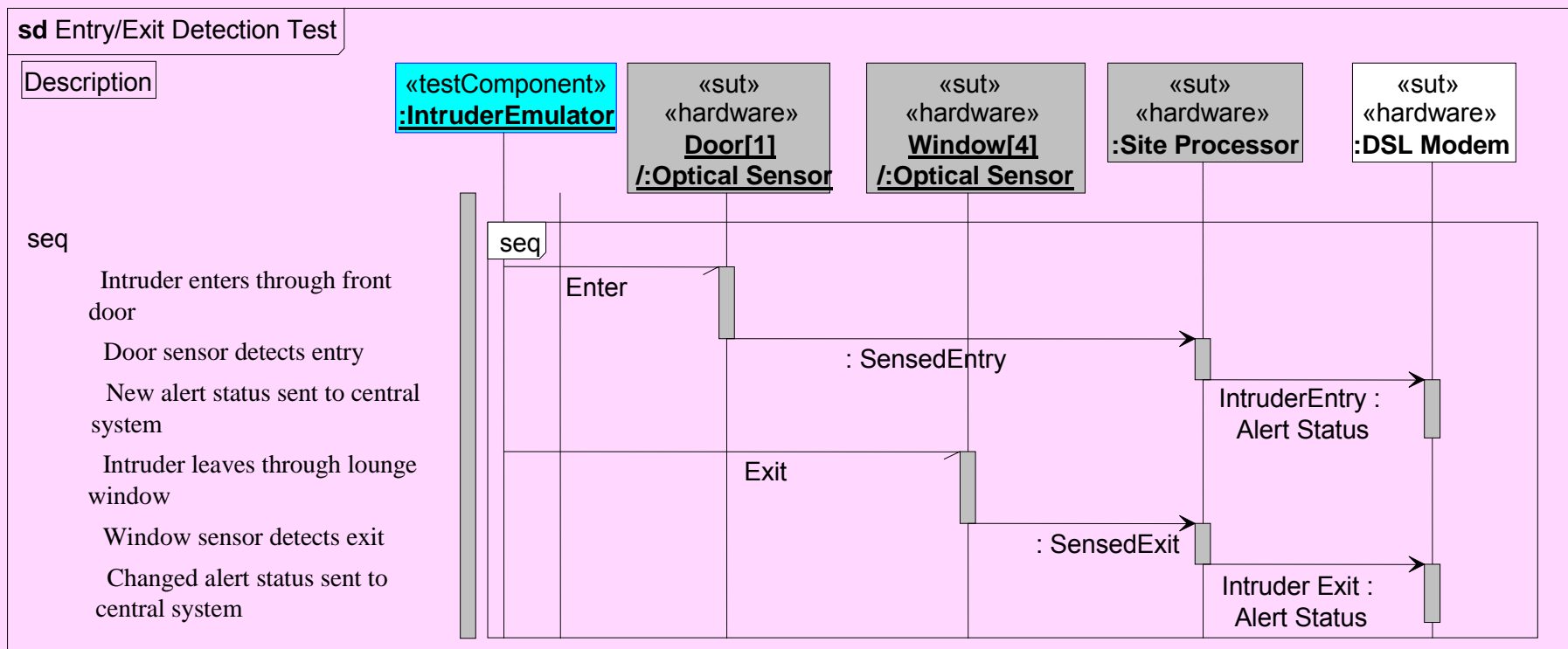
# ESS Deployment View

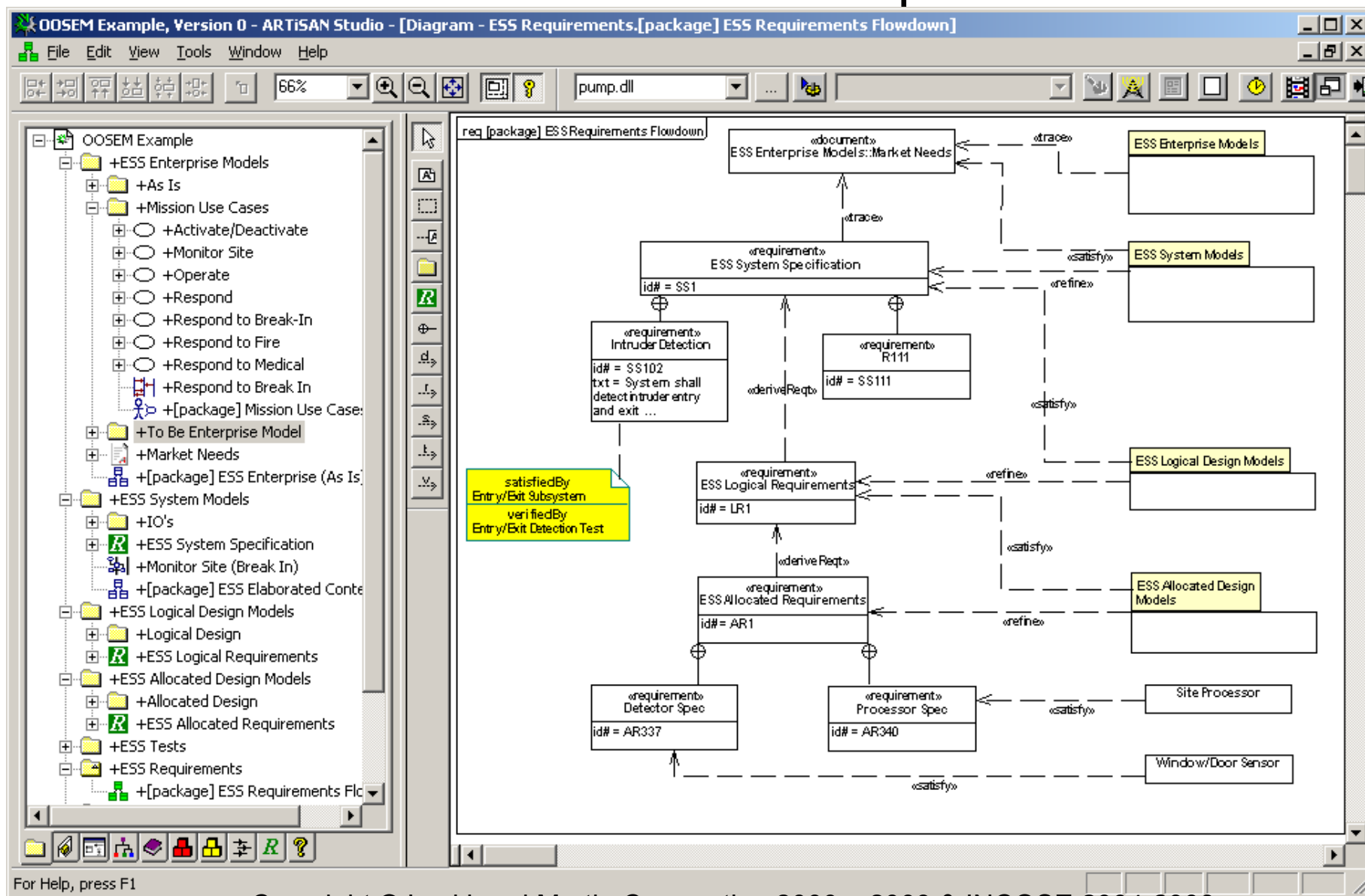


# ESS Parametric Diagram To Support Trade-off Analysis



# Entry/Exit Test Case



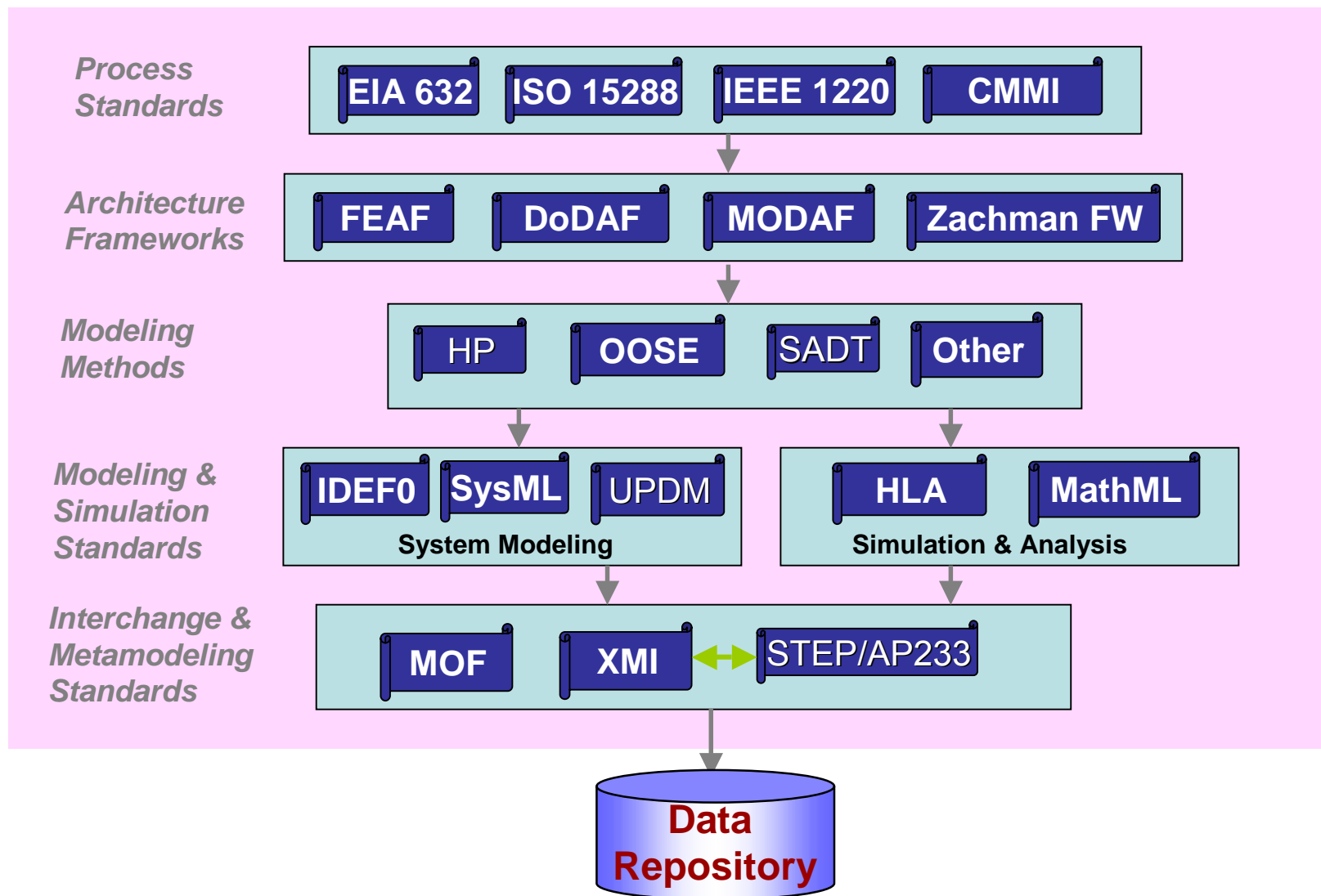




## SysML in a Standards Framework

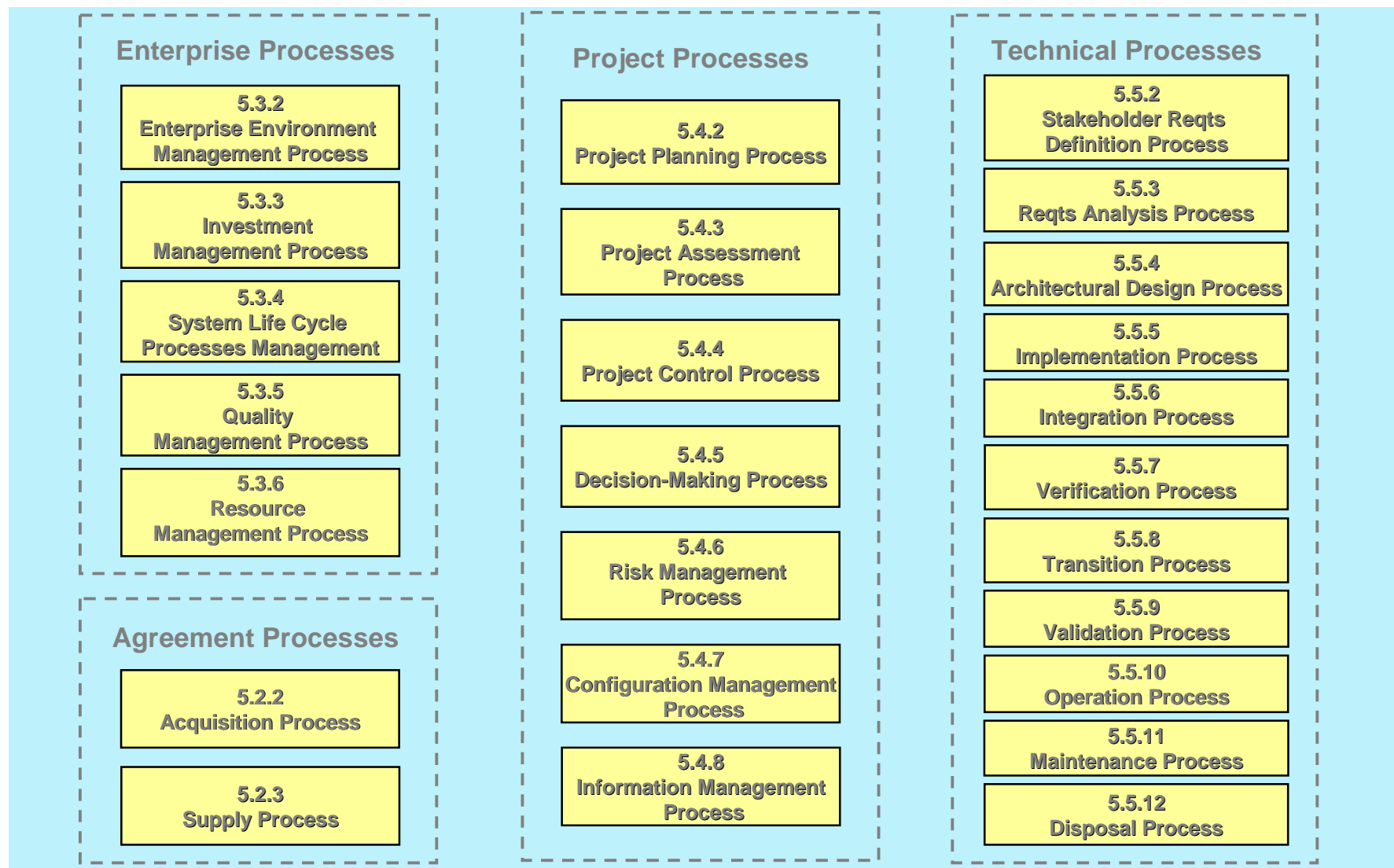


# Systems Engineering Standards Framework (Partial List)



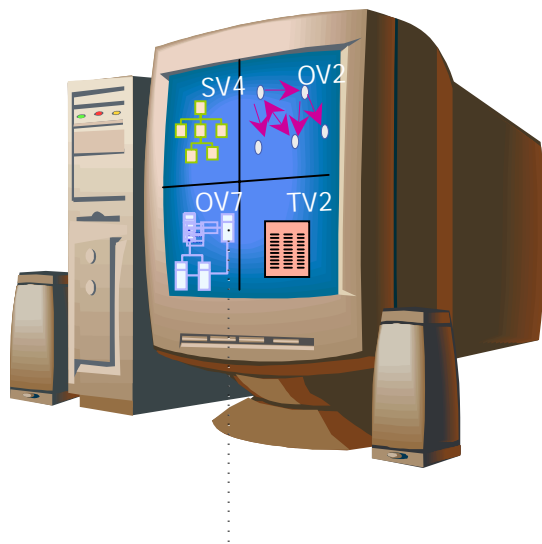
# ISO/IEC 15288

## System Life Cycle Processes

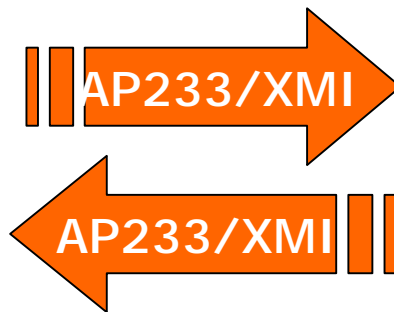


# Standards-based Tool Integration with SysML

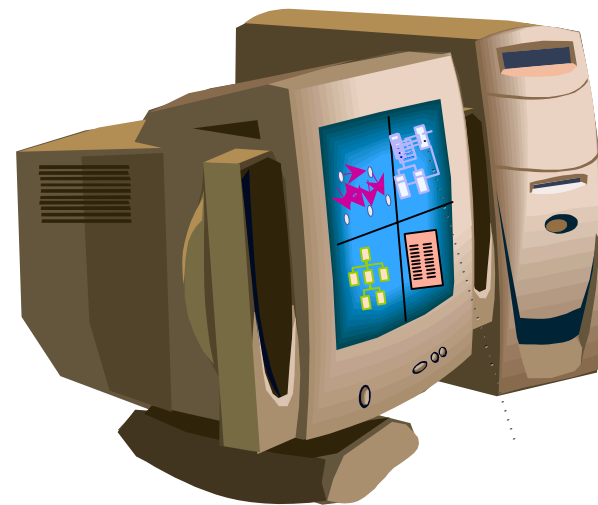
Systems Modeling  
Tool



Model/Data  
Interchange



Other Engineering  
Tools



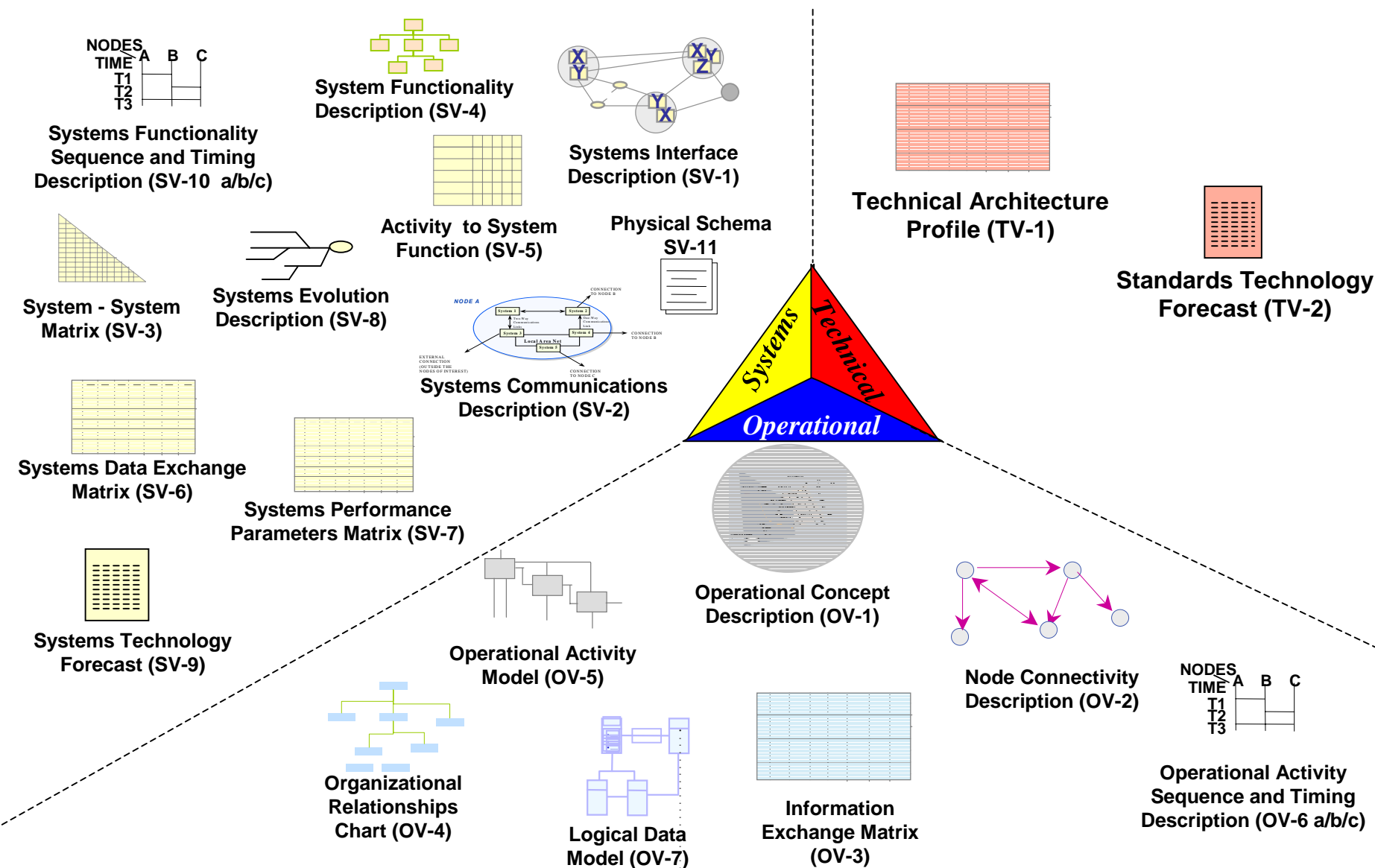
# Participating SysML Tool Vendors

- Artisan
- EmbeddedPlus
  - 3rd party IBM vendor
- No Magic
- Sparx Systems
- Telelogic (Tau and Rhapsody)
- Visio SysML template
- Vitech

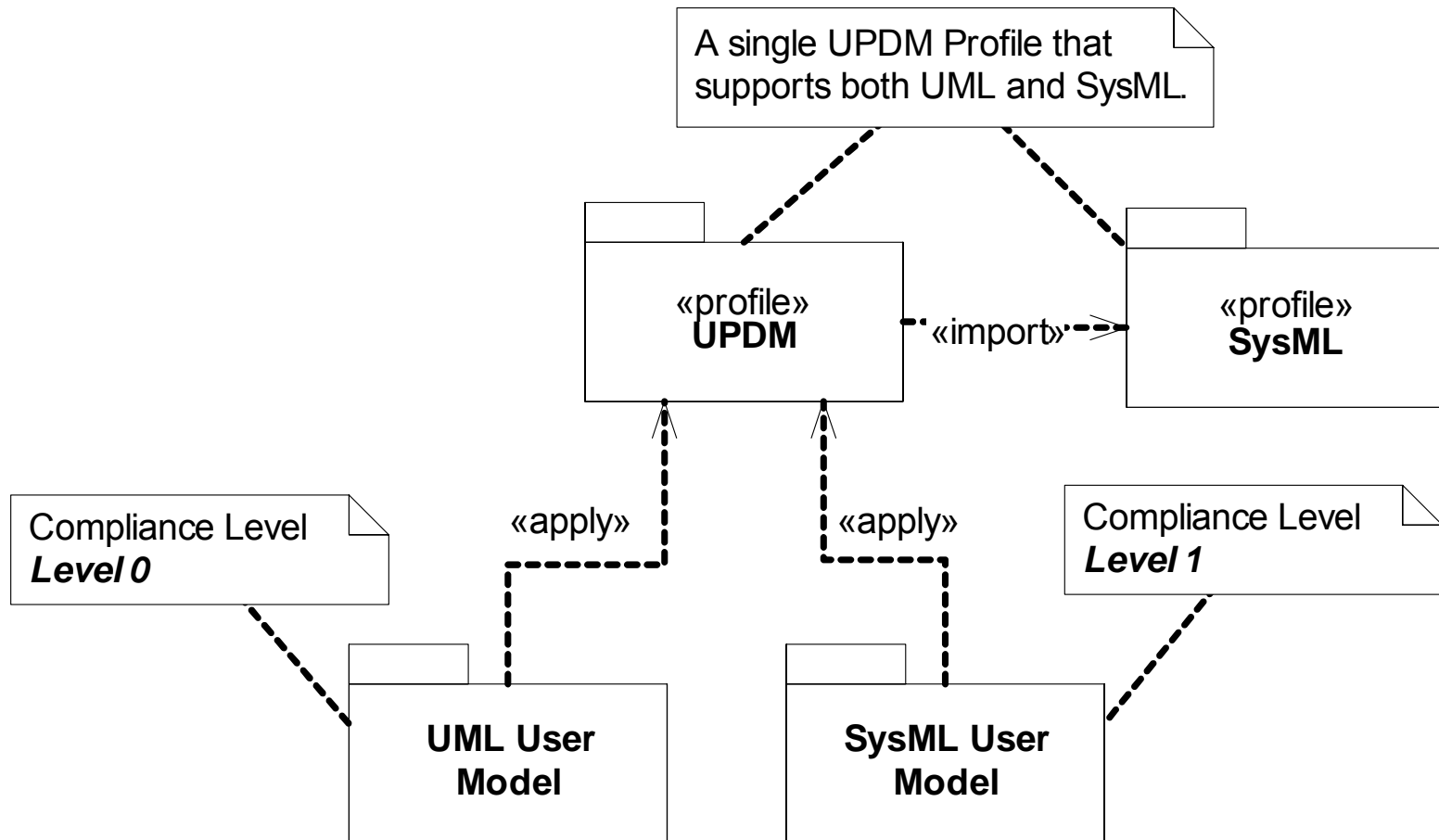
# UML Profile for DoDAF/MODAF (UPDM) Standardization

- Current initiative underway to develop standard profile for representing DODAF and MODAF products
  - Requirements for profile issued Sept 05
  - Final submissions presented in March '07
  - Begin vote for adoption in June '07
- Goal is to provide robust architecture modeling capability, improve communications and tool interoperability, and reduce re-training
- Multiple vendors and users participating
- Includes a UML and SysML compliance level

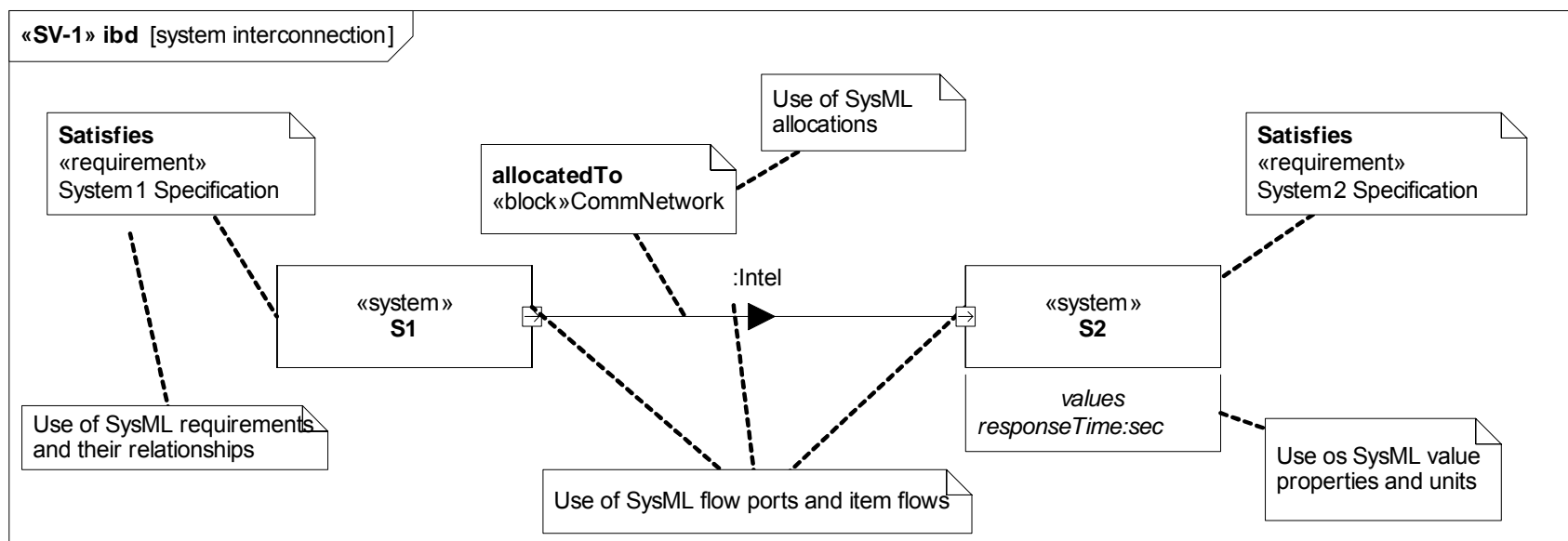
# Summary of DoDAF



# Using SysML with UPDM



# Leveraging SysML Features With Compliance Point Level 1

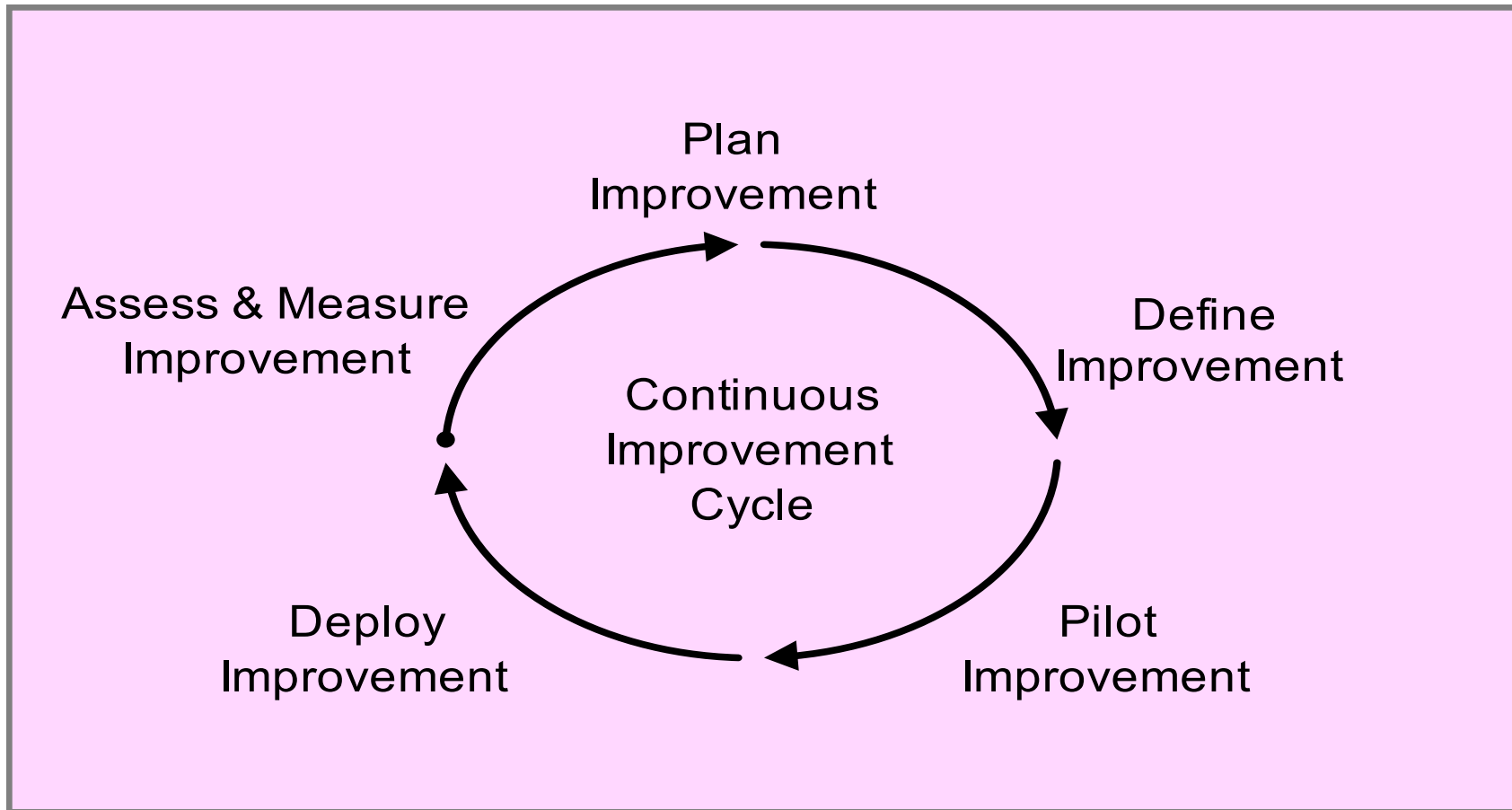






Transitioning to SysML

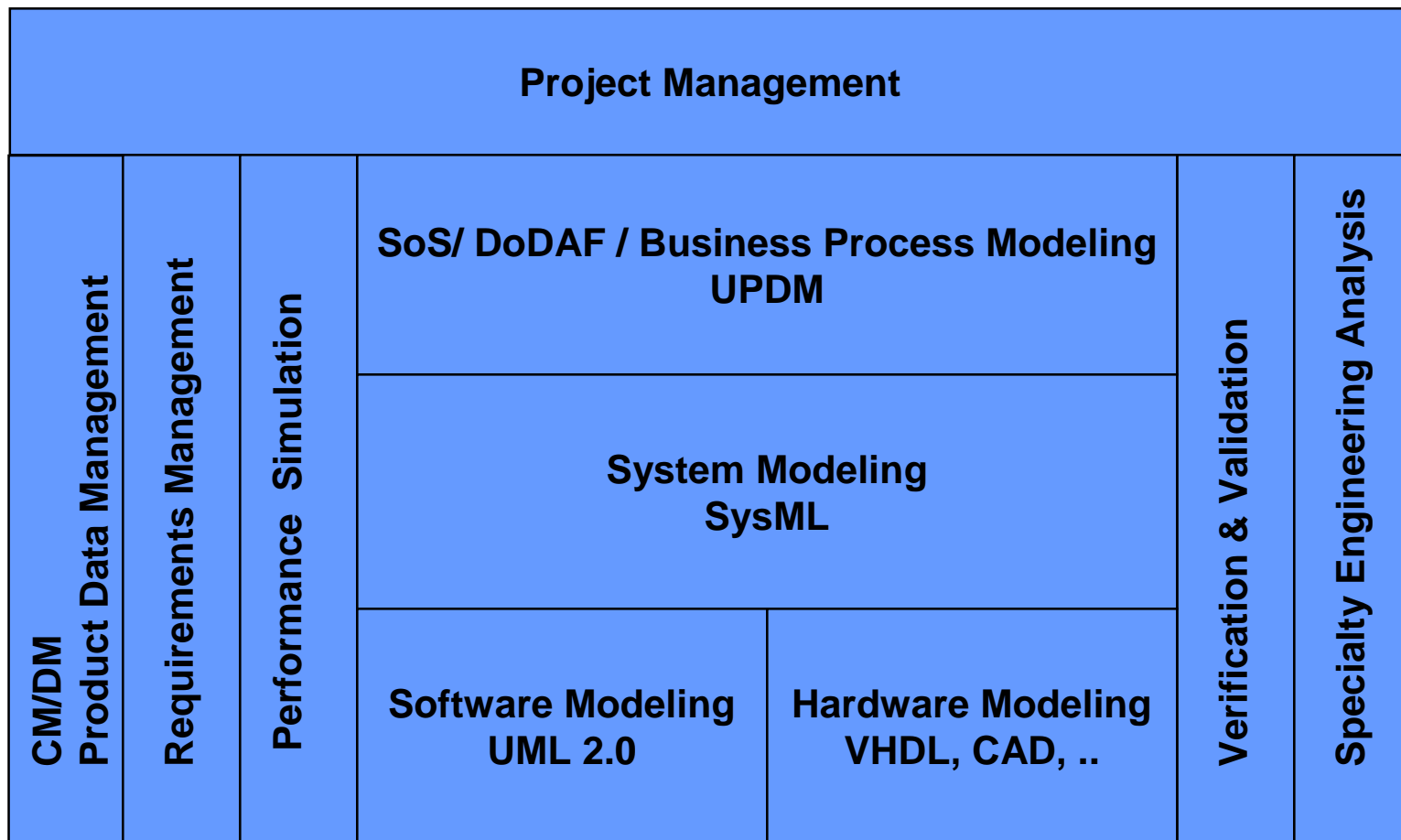
# Using Process Improvement To Transition to SysML



# 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





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 and Finalized in April 2007
- 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
- Evolve language based on user/vendor/researcher feedback and lessons learned

- **OMG SysML website**
  - <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 v2.1.1
  - OMG doc# formal/07-02-05
- UML 2 Infrastructure v2.1.1
  - OMG doc# formal/07-02-06

## PAPERS

- 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,
  - Bock. C., vol. 9 no.2, pp. 160-186, Journal of International Council of Systems Engineering, 2006.
- The Systems Modeling Language,
  - Matthew Hause, Alan Moore, June ' 2006.
- An Overview of the Systems Modelling Language for Products and Systems Development,
  - Laurent Balmelli, Oct ' 2006.
- Model-driven systems development,
  - L. Balmelli, D. Brown, M. Cantor, M. Mott, July ' 2006.

## TUTORIAL AUTHORS

- Sanford Friedenthal ([sanford.friedenthal@lmco.com](mailto:sanford.friedenthal@lmco.com))
- Alan Moore ([alan.moore@mathworks.co.uk](mailto:alan.moore@mathworks.co.uk))
- Rick Steiner ([fsteiner@raytheon.com](mailto:fsteiner@raytheon.com))