Introduction
OMG Systems Modeling Language (OMG SysML™) and OOSEM Tutorial

By
Abe Meilich, Ph.D.
abraham.w.meilich@lmco.com

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

Acknowledged Original Authors:
Sanford Friedenthal
Alan Moore
Rick Steiner

Caveat

- These materials have been modified slightly from the original Tutorial given at INCOSE 2006
  - Softcopy of Full Tutorial available at :

    http://www.omgsysml.org/SysML-Tutorial-Baseline-to-INCOSE-060524-low_res.pdf
- This material is based on version 1.0 of the SysML specification (ad-06-03-01)
  - Adopted by OMG in May ’06
  - *Going through finalization process*
- OMG SysML Website
  - http://www.omgsysml.org/

Objectives & Intended Audience

At the end of this tutorial, you should understand the:

- Benefits of model driven approaches to systems engineering
- Types of SysML diagrams and their basic constructs
- Cross-cutting principles for relating elements across diagrams
- Relationship between SysML and other Standards
- Introduction to principles of a OO System Engineering Method

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

Intended Audience:

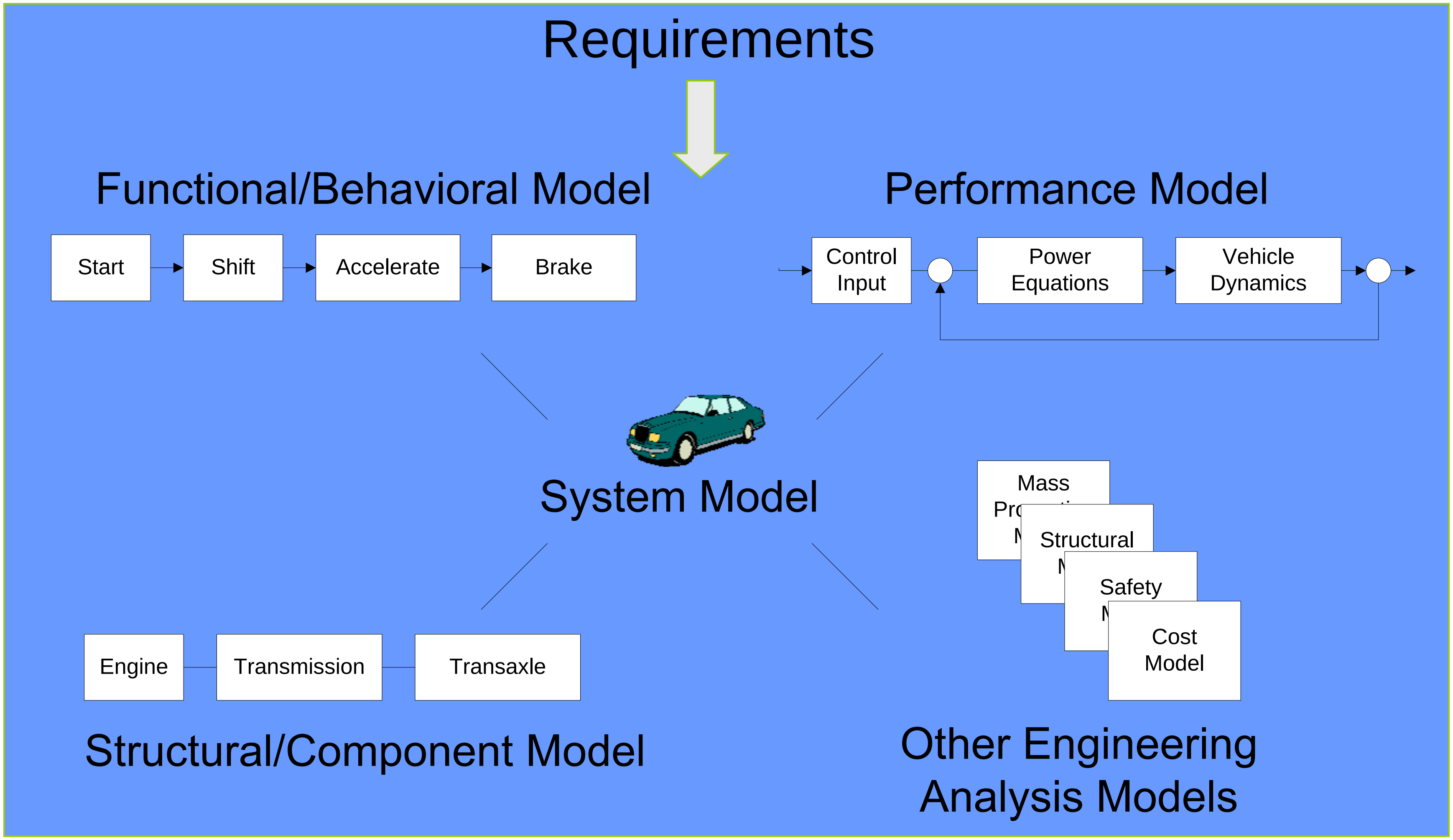
- Practicing Systems Engineers interested in system modeling
  - Already familiar with system modeling & tools, or
  - Want to learn about systems modeling
- Software Engineers who want to express systems concepts
- Familiarity with UML is not required, but it will help

Topics

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

Background

System Modeling

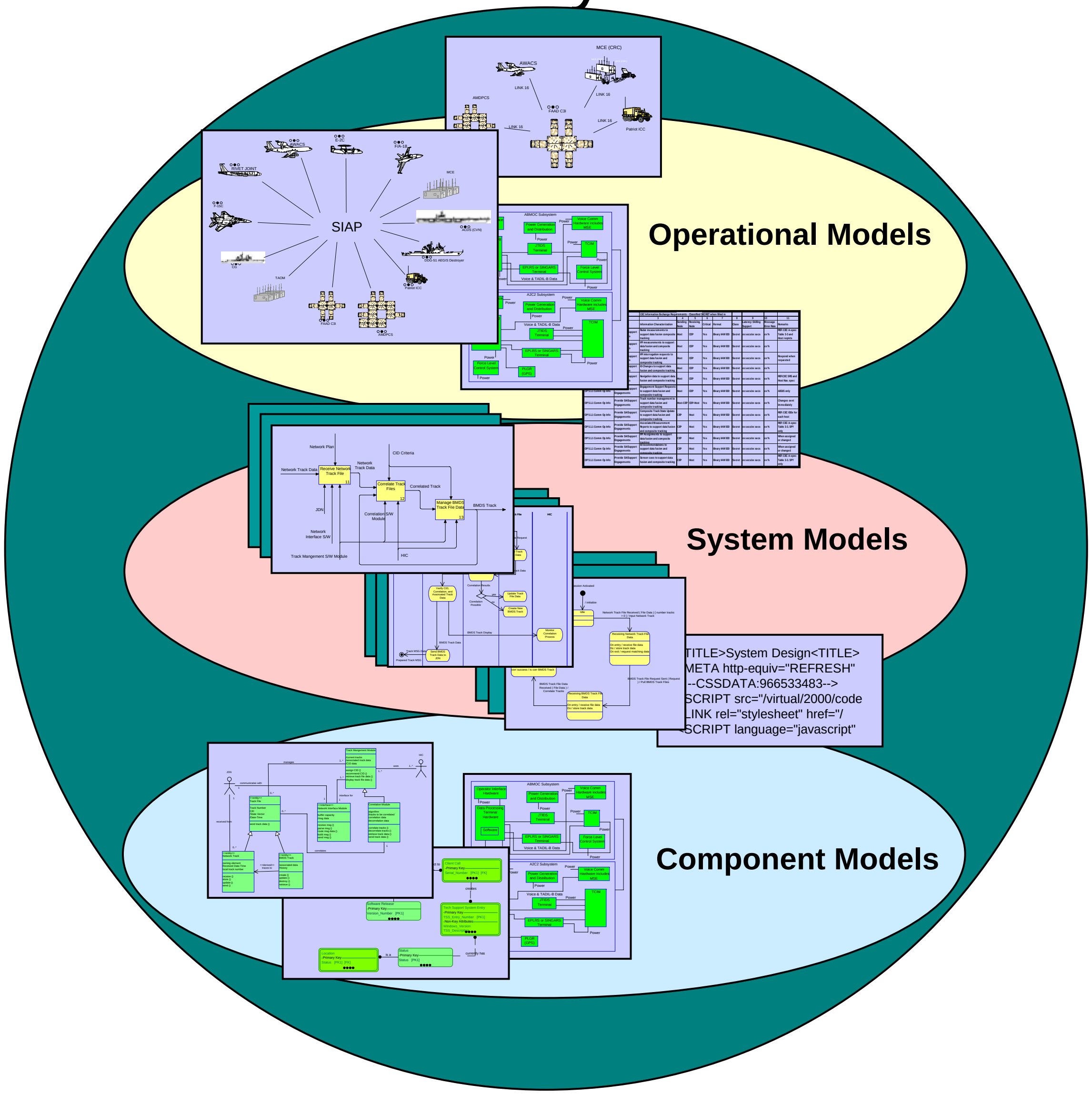


Integrated System Model Must Address Multiple Aspects of a System

Model Based Systems Engineering Benefits

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

Modeling at Multiple Levels of the System



What is SysML?

- A graphical modelling language in response to the UML for Systems Engineering RFP developed by the OMG, INCOSE, and AP233
  - a UML Profile that represents a subset of UML 2 with extensions
- Supports the specification, analysis, design, verification, and validation of systems that include hardware, software, data, personnel, procedures, and facilities
- Supports model and data interchange via XMI and the evolving AP233 standard (in-process)

SysML is Critical Enabler for Model Driven SE

What is SysML (cont.)

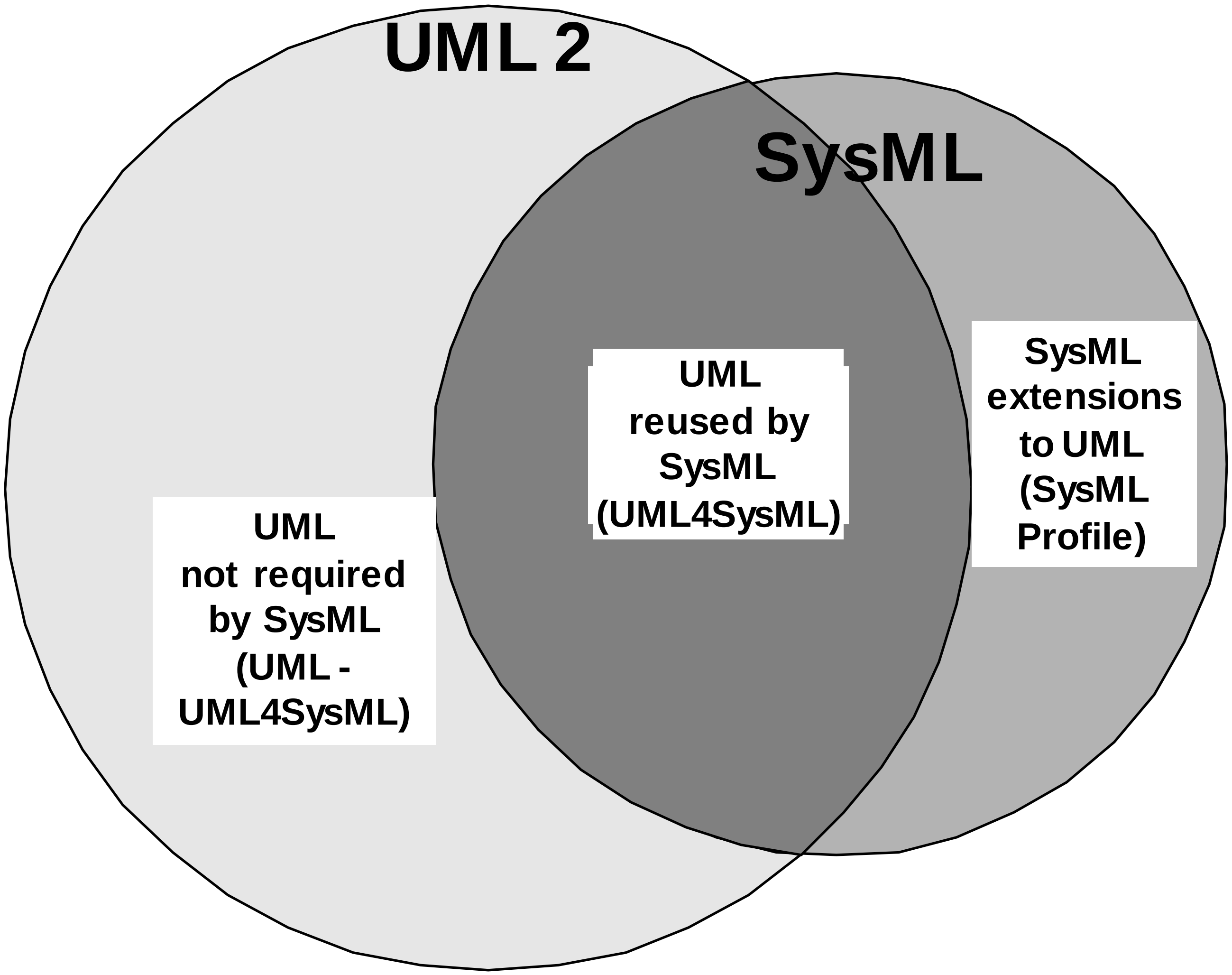
- ***Is*** a visual modeling language that provides
  - Semantics = meaning
  - Notation = representation of meaning
- ***Is not*** a methodology or a tool
  - SysML is methodology and tool independent

UML/SysML Status

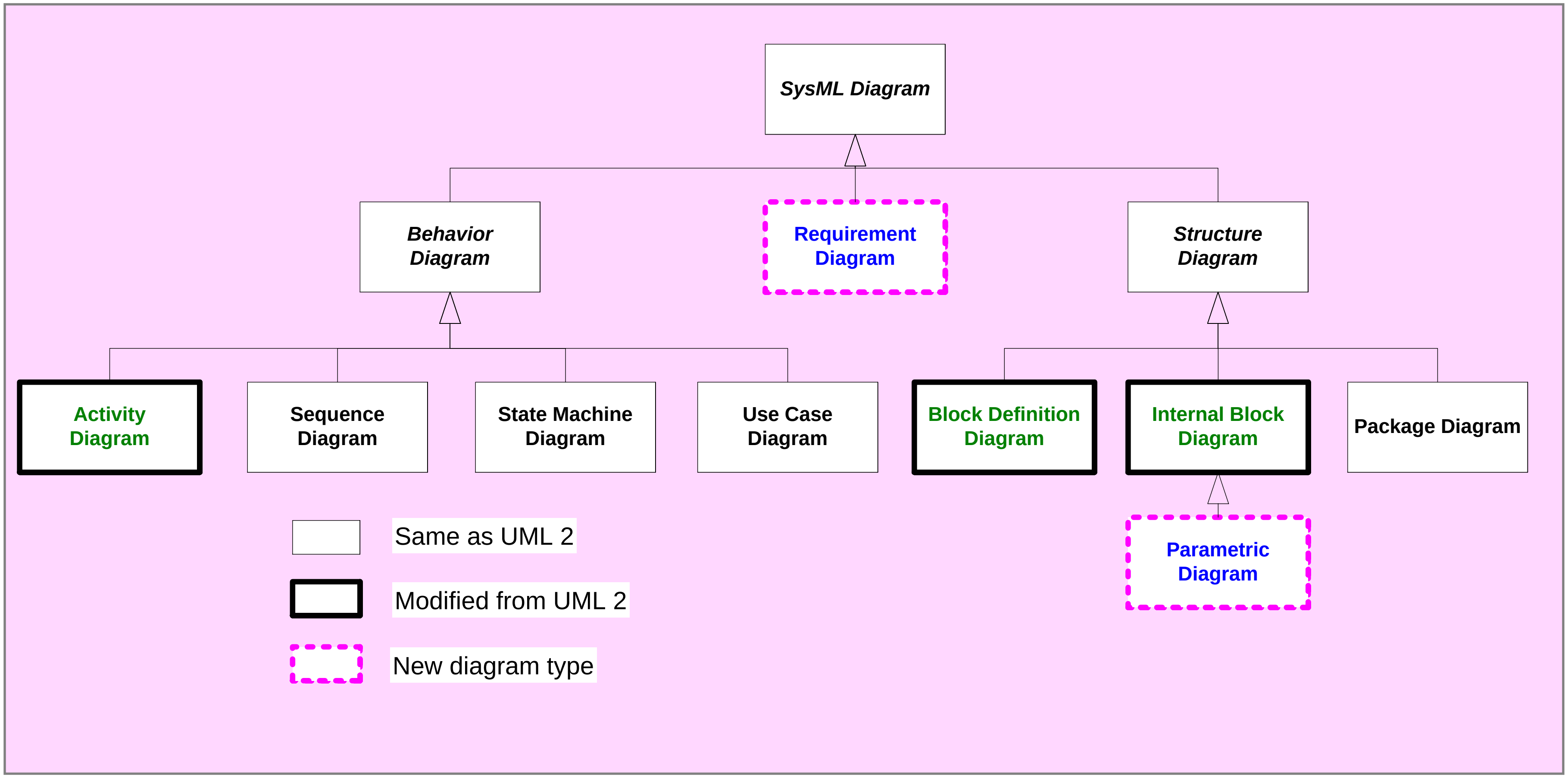
- UML V2.0
  - Updated version of UML that offers significant capability for systems engineering over previous versions
  - Finalized in 2005 (formal/05-07-04)
- UML for Systems Engineering (SE) RFP
  - Established the requirements for a system modeling language
  - Issued by the OMG in March 2003
- SysML
  - Industry Response to the UML for SE RFP
  - Addresses most of the requirements in the RFP
  - Version 1.0 adopted by OMG in May ’06 / In finalization
  - Being implemented by multiple tool vendors

Diagram Overview

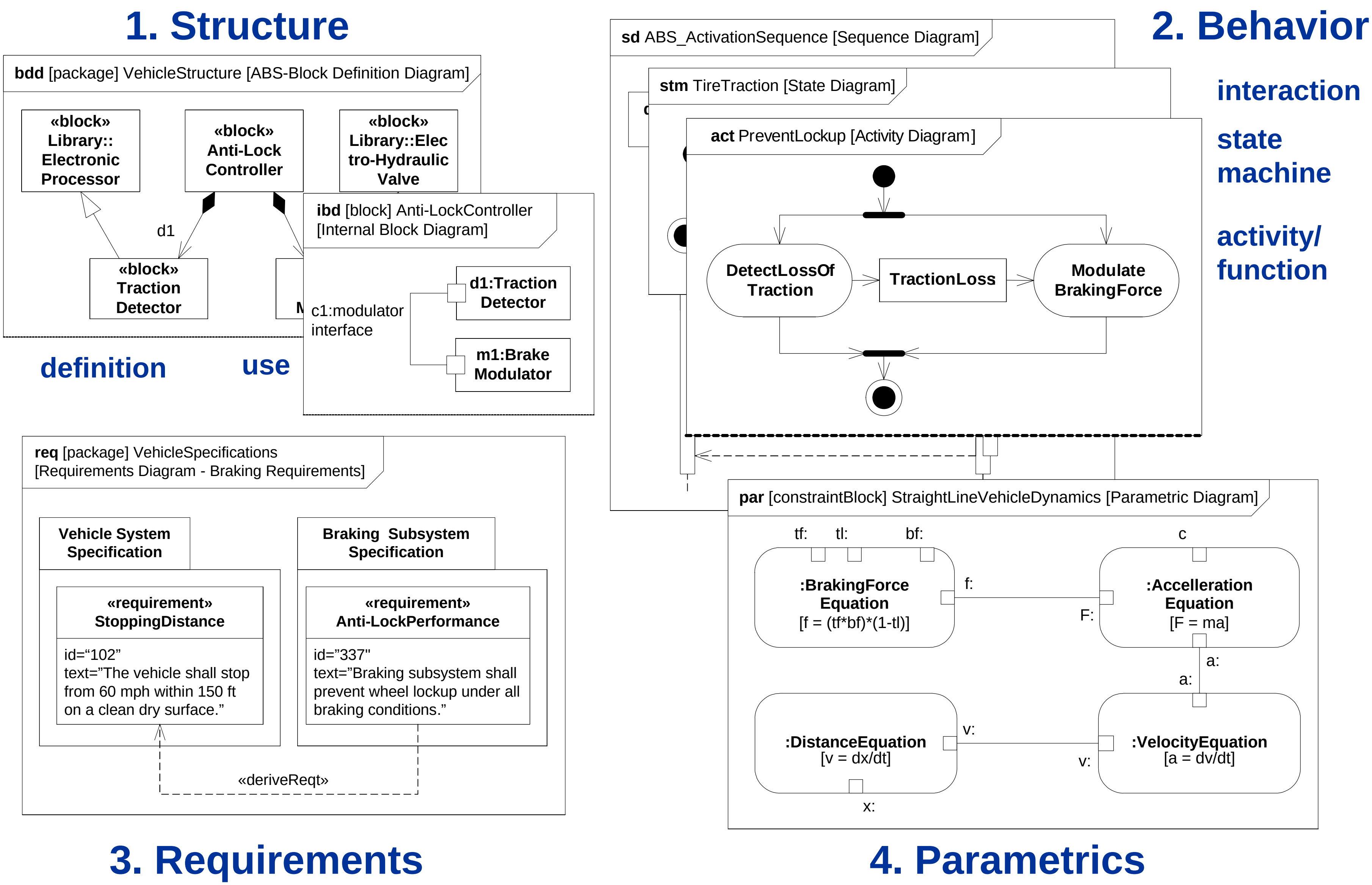
Relationship Between SysML and UML



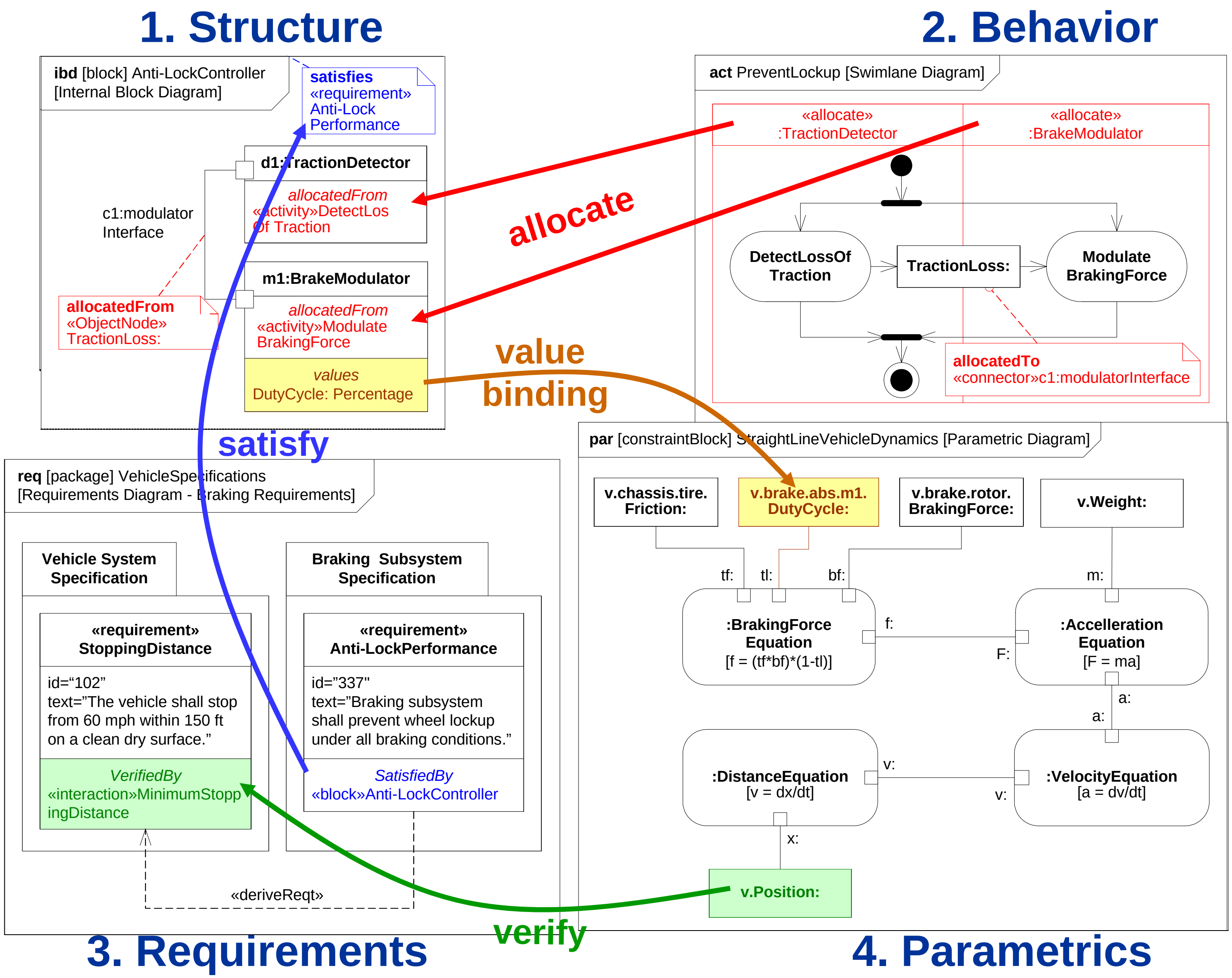
SysML Diagram Taxonomy



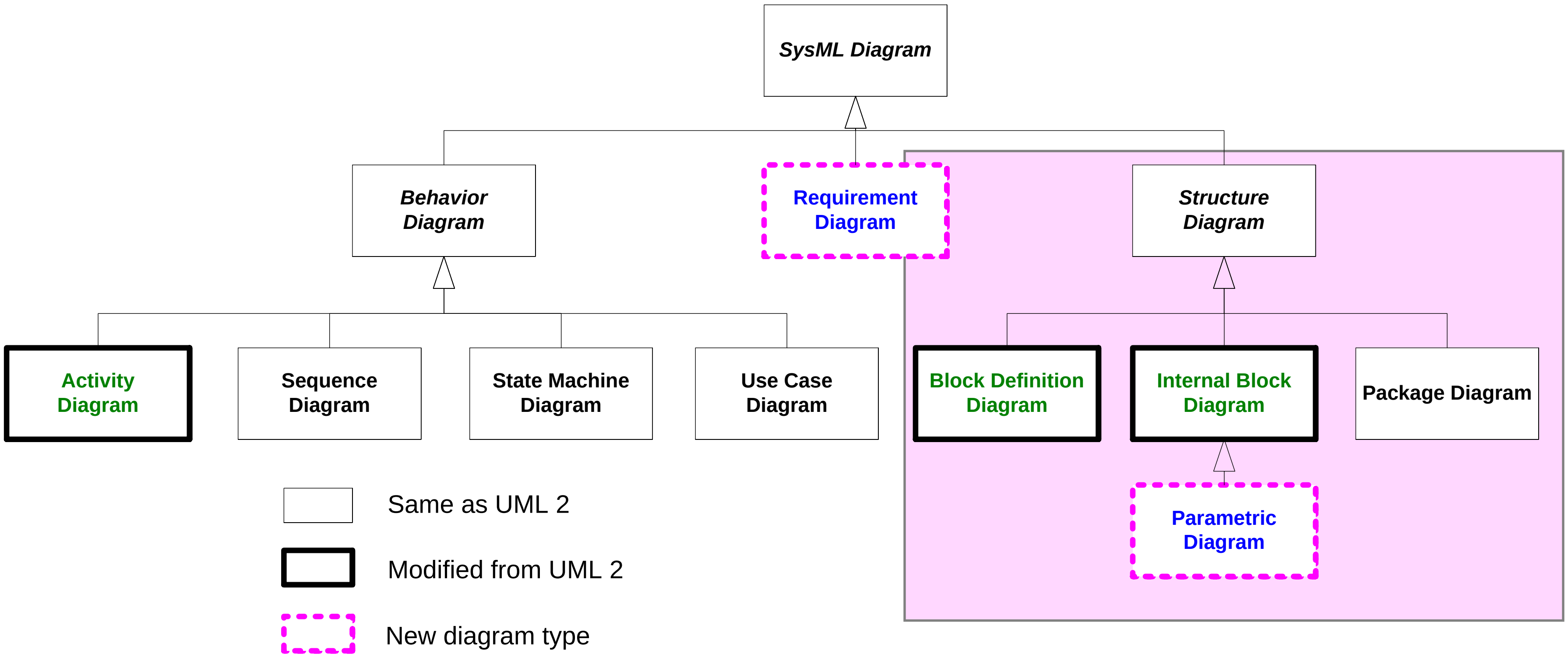
4 Pillars of SysML – ABS Example



Cross Connecting Model Elements



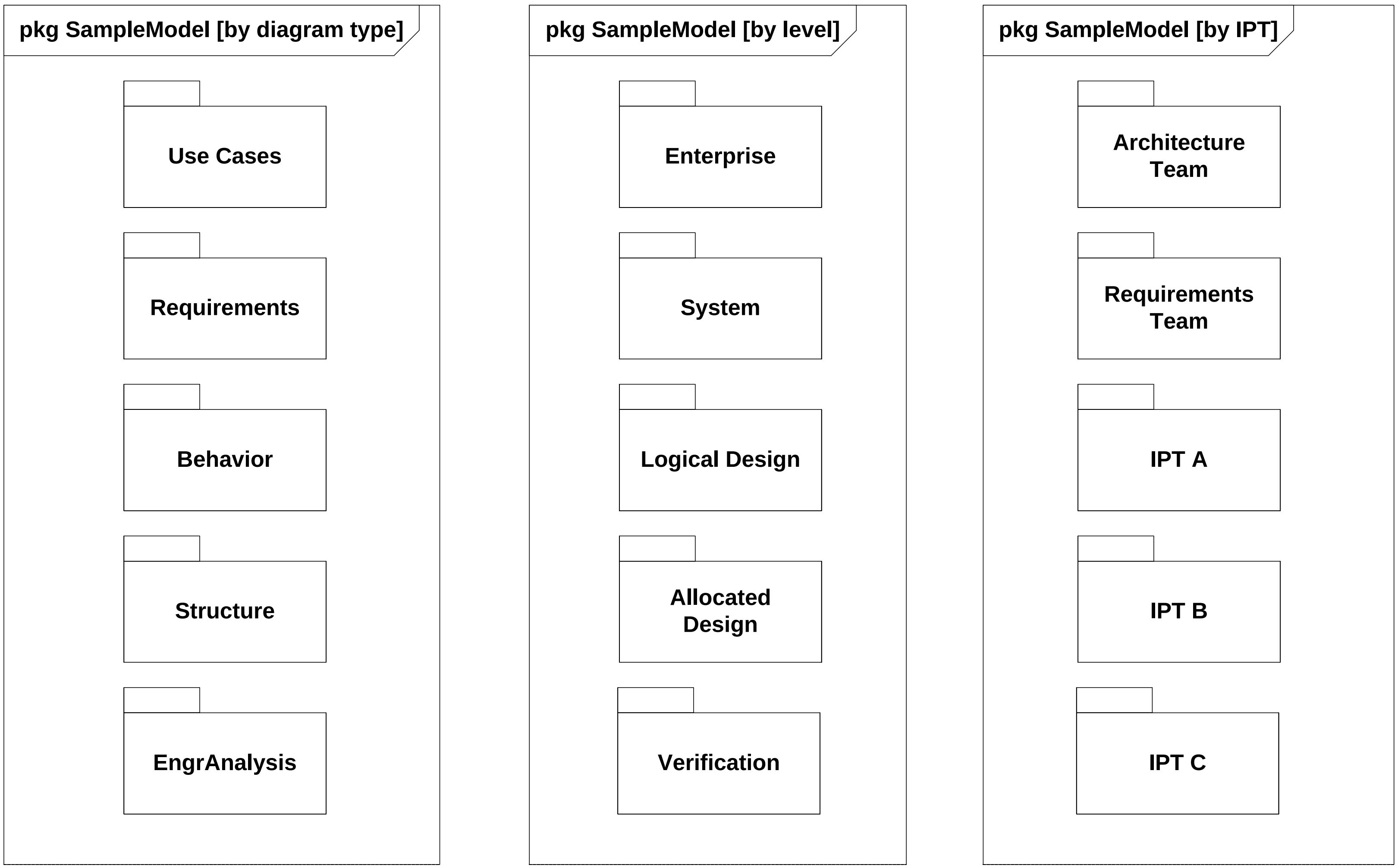
Structural Diagrams



Package Diagram

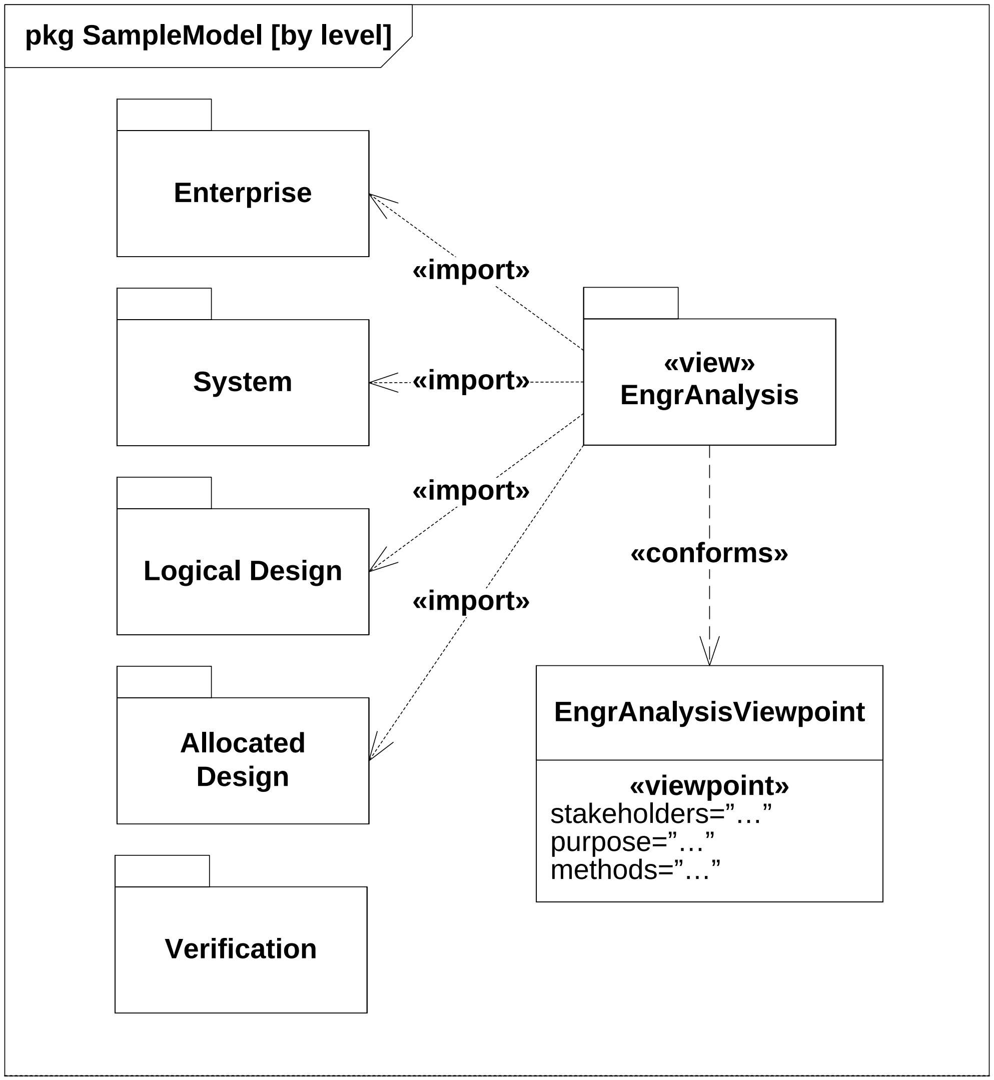
- Package diagram is used to organize the model
  - Groups model elements into a name space
  - Often represented in tool browser
- Model can be organized in multiple ways
  - By System hierarchy (e.g., enterprise, system, component)
  - By domain (e.g., requirements, use cases, behavior)
  - Use viewpoints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

Package Diagram
Organizing the Model



By Diagram Type | By Hierarchy | By IPT

Package Diagram - Views



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

Blocks are Basic Structural Elements

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

«block» BrakeModulator
allocatedFrom «activity»Modulate BrakingForce
values DutyCycle: Percentage

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

Block Property Types

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

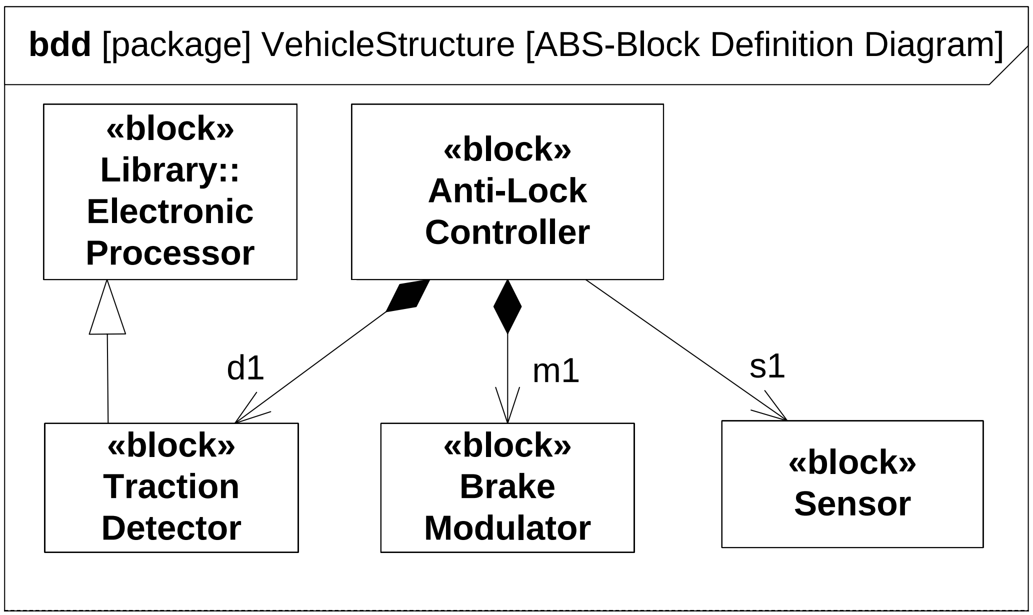
Using Blocks

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

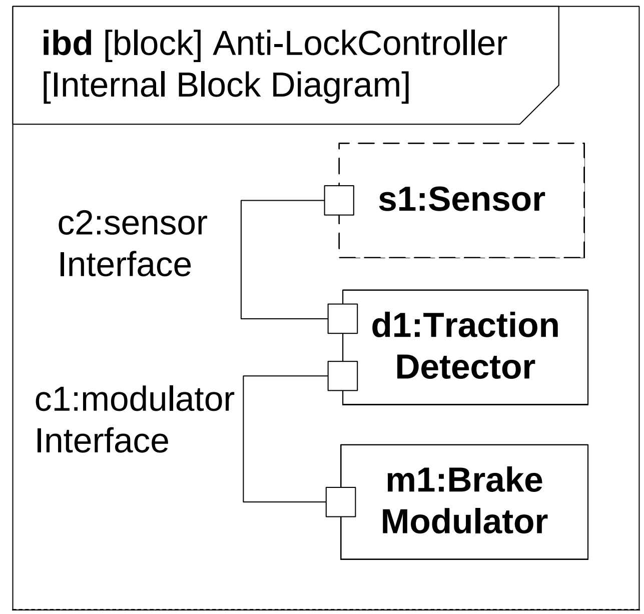
Blocks Used to Specify Hierarchies and Interconnection

Block Definition vs. Usage

Block Definition Diagram



Internal Block Diagram



Definition

– Block is a definition/type

– Captures properties, etc.

– Reused in multiple contexts

Usage

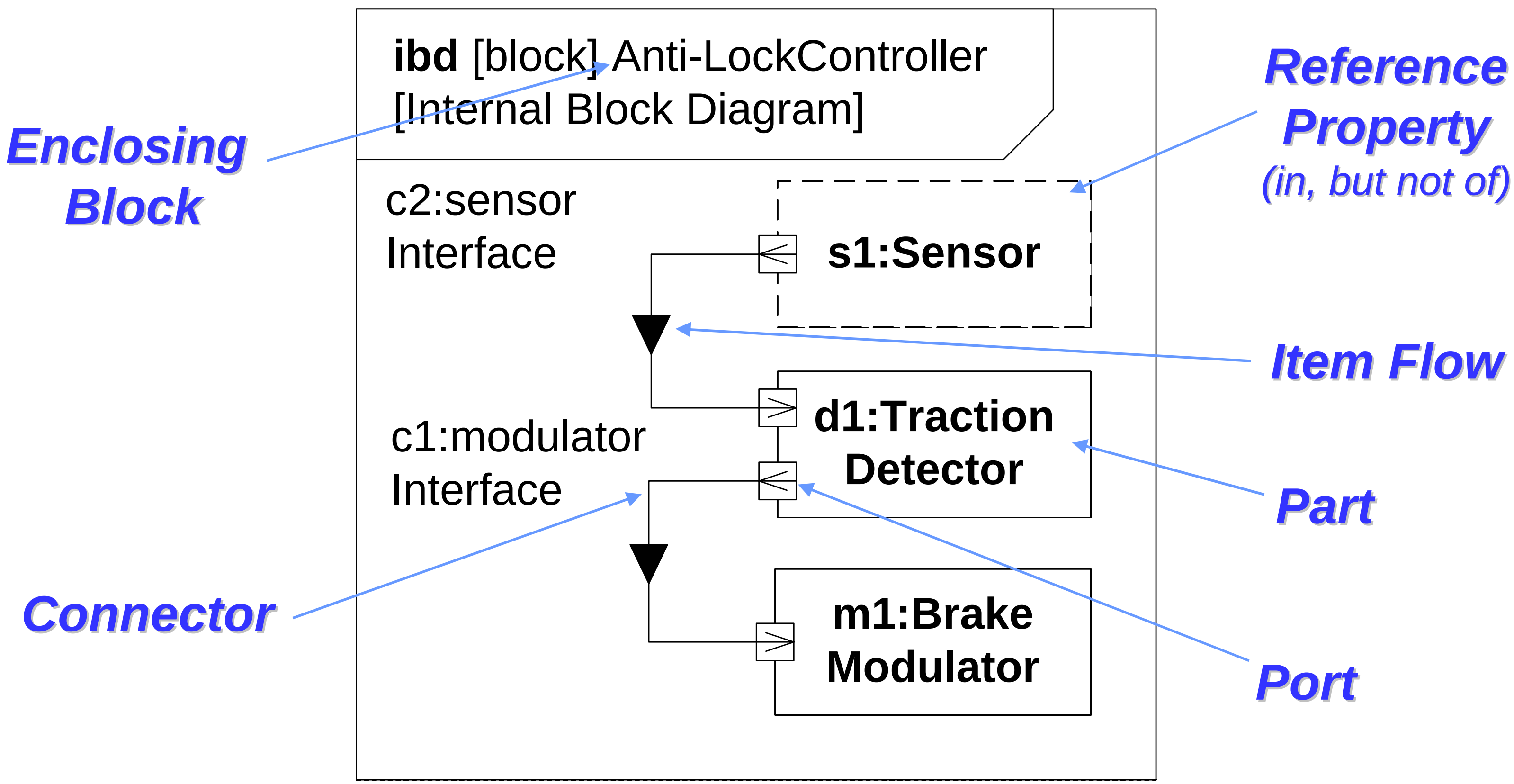
– Part is the usage in a particular context

– Typed by a block

– Also known as a role

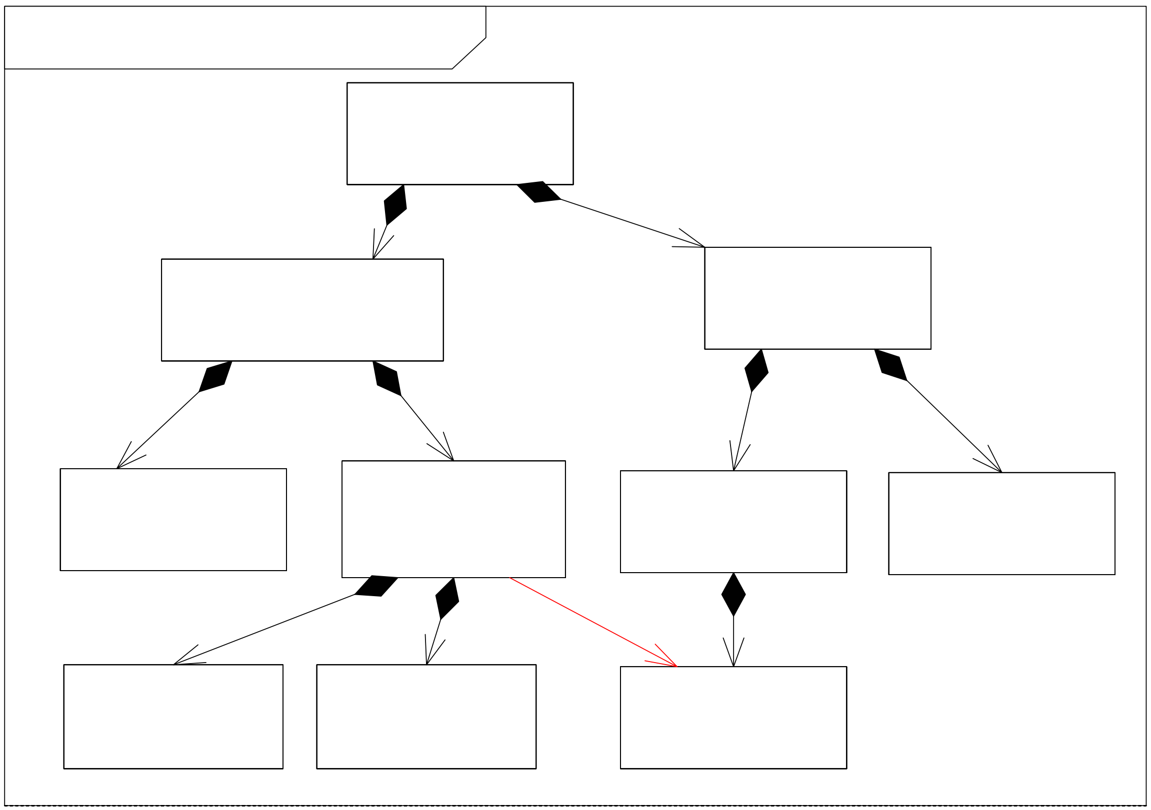
Internal Block Diagram (ibd)

Blocks, Parts, Ports, Connectors & Flows

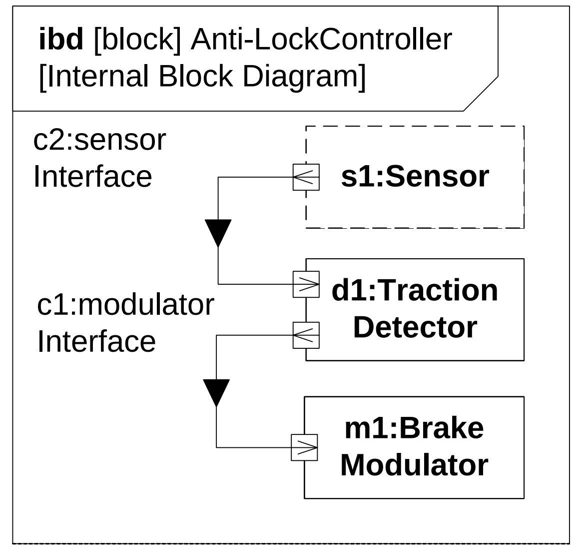


Internal Block Diagram Specifies Interconnection of Parts

Reference Property Explained



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



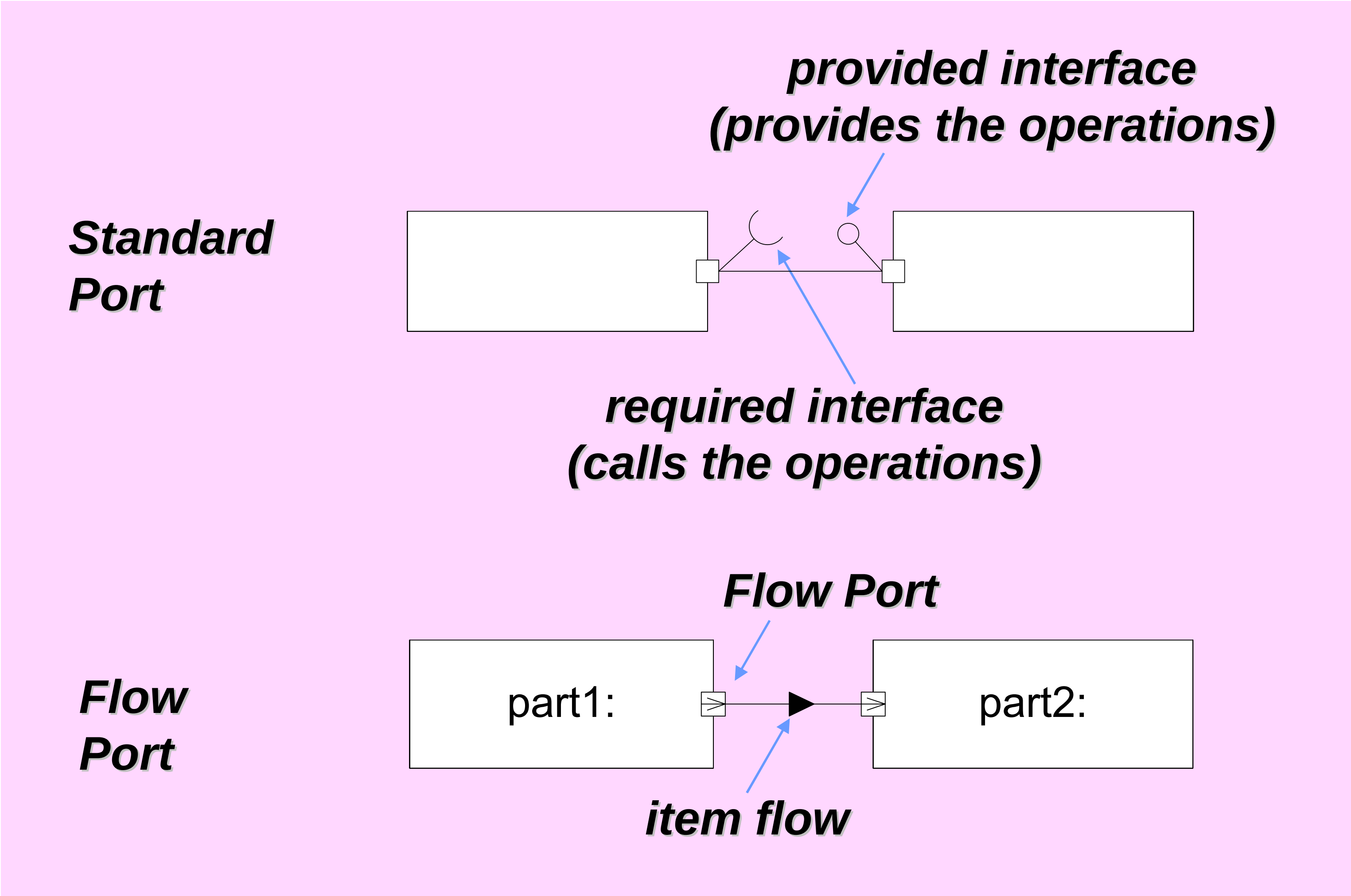
bdd [package] VehicleStructure

SysML Port

- Specifies interaction points on blocks and parts
  - Supports integration of behavior and structure
- Port types
  - Standard (UML) Port
    - Specifies a set of operations and/or signals
    - Typed by a UML interface
  - Flow Port
    - Specifies what can flow in or out of block/part
    - Typed by a flow specification

2 Port Types Support Different Interface Concepts

Port Notation

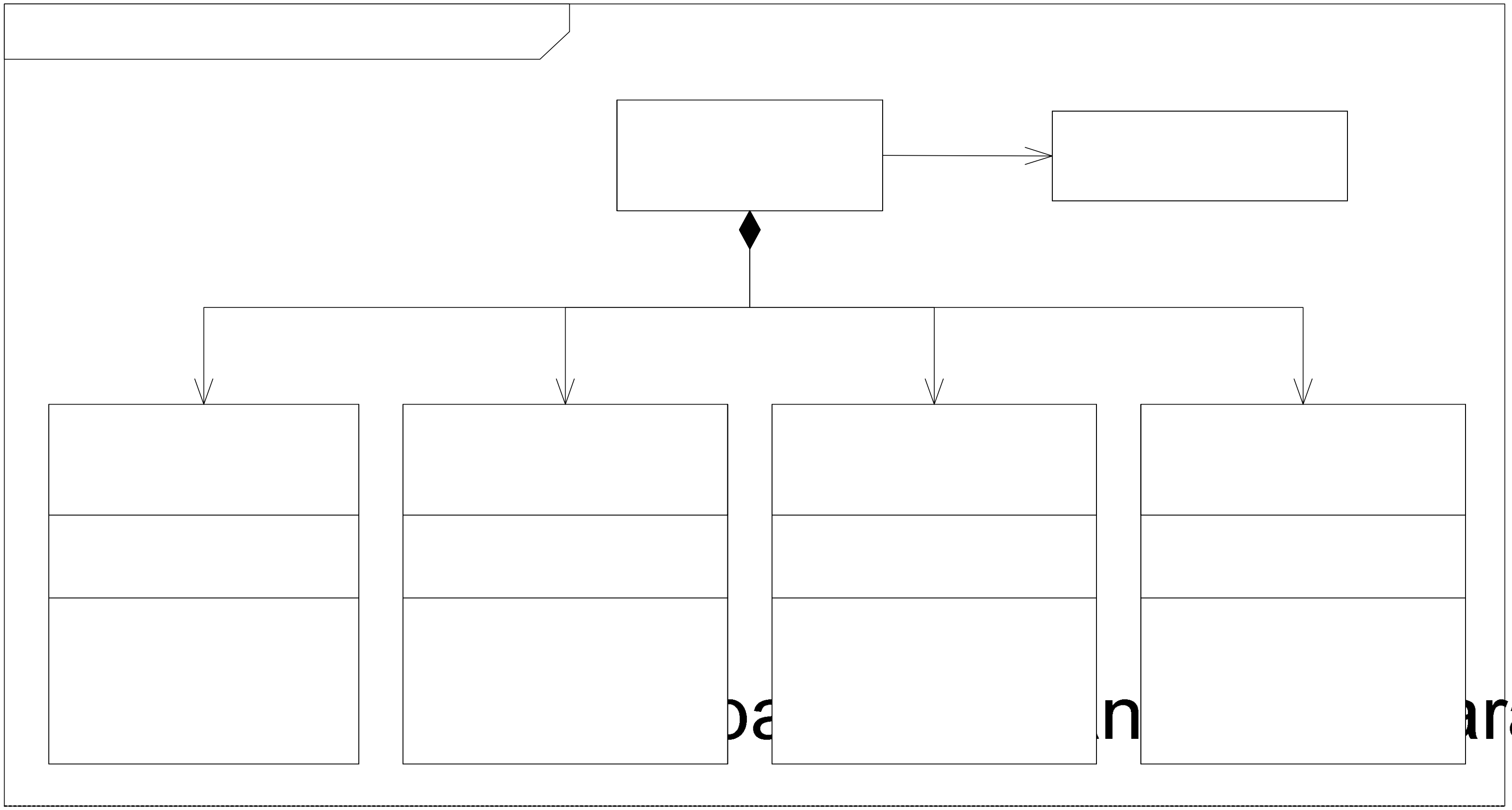


Parametrics

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

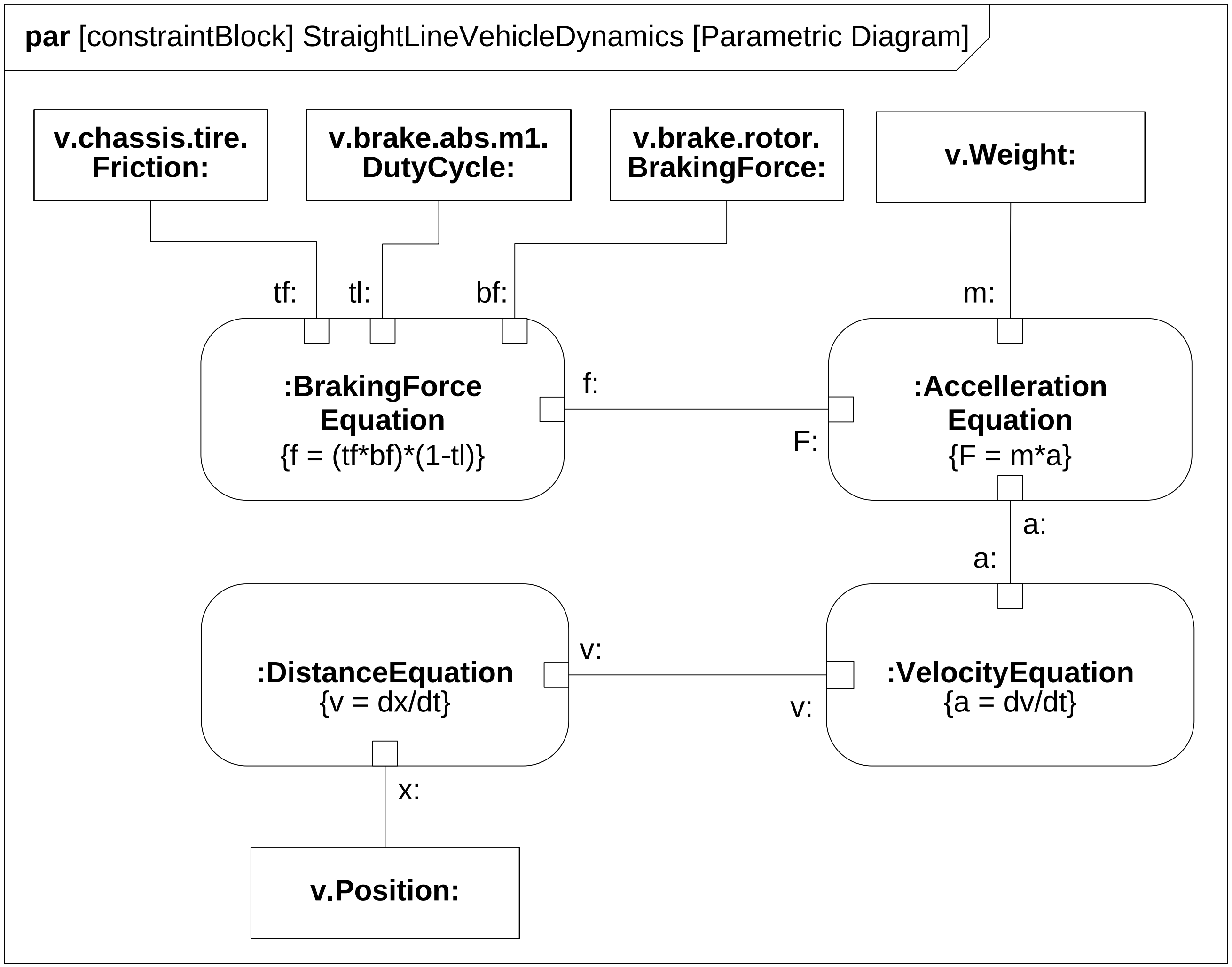
Parametrics Enable Integration of Engineering Analysis with Design Models

Defining Vehicle Dynamics



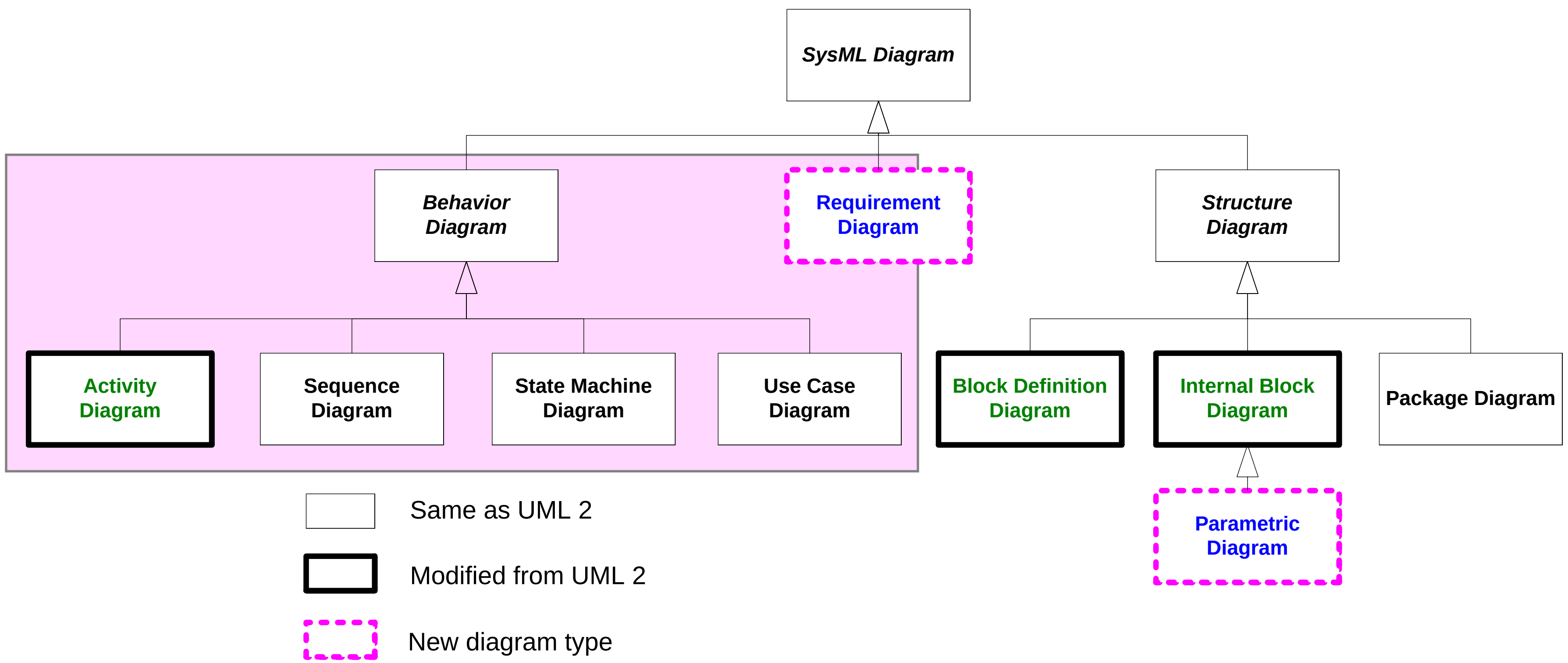
Defining Reusable Equations for Parametrics

Vehicle Dynamics Analysis



Using the Equations in a Parametric Diagram to Constrain Value Properties

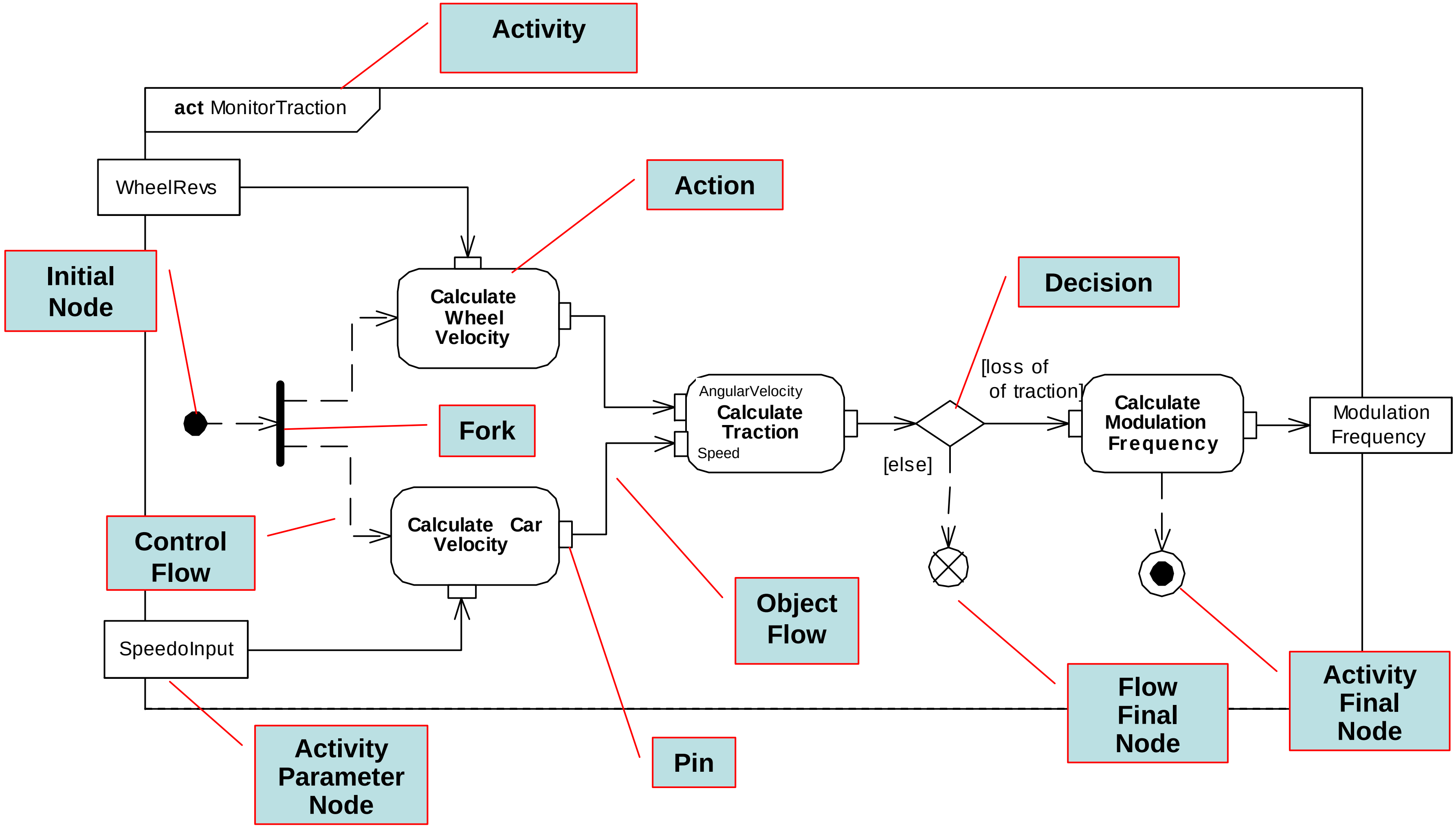
Behavioral Diagrams



Activities

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

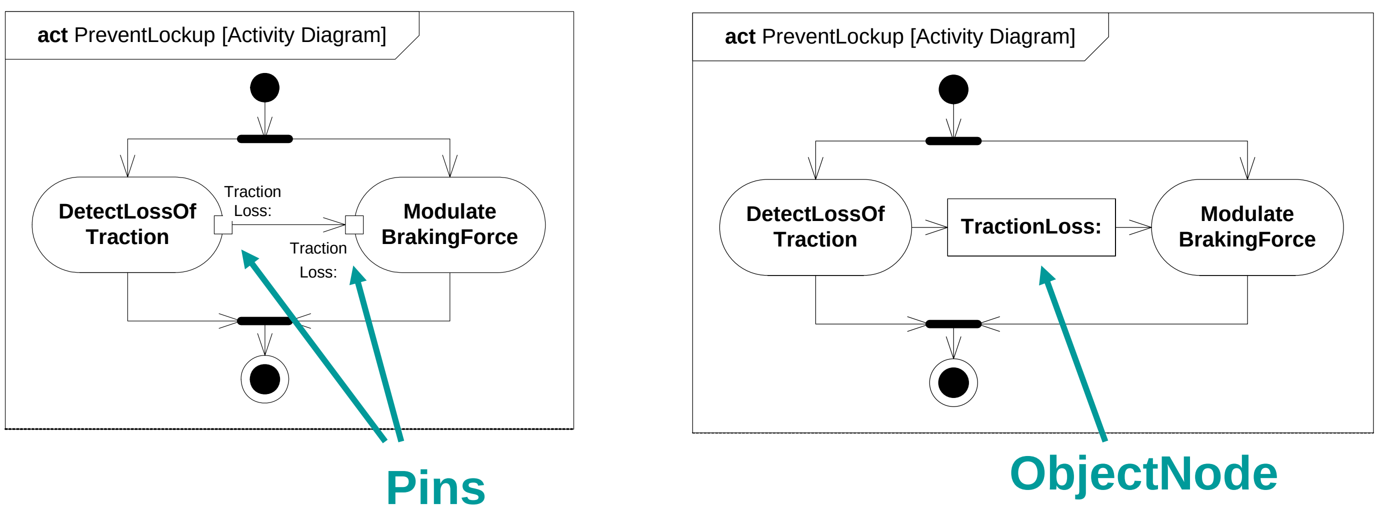
Activity Diagram Notation



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

Activity Diagrams
Pin vs. Object Node Notation

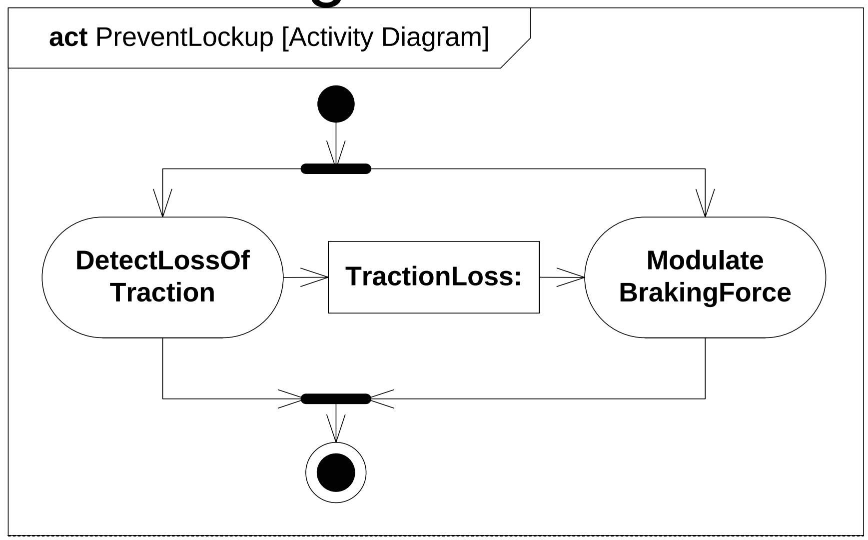
- Pins are kinds of Object Nodes
  - Used to specify inputs and outputs of actions
  - Typed by a block or value type
  - Object flows connect object nodes
- Object flows between pins have two diagrammatic forms
  - Pins shown with object flow between them
  - Pins elided and object node shown with flow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



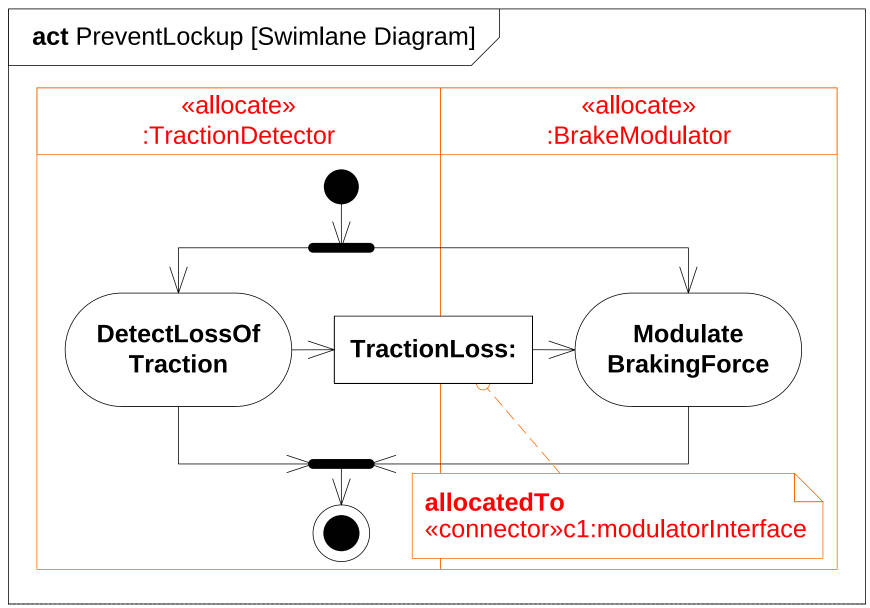
Pins must have same characteristics (name, type etc.)

Explicit Allocation of Behavior to Structure Using Swimlanes

Activity Diagram
(without Swimlanes)

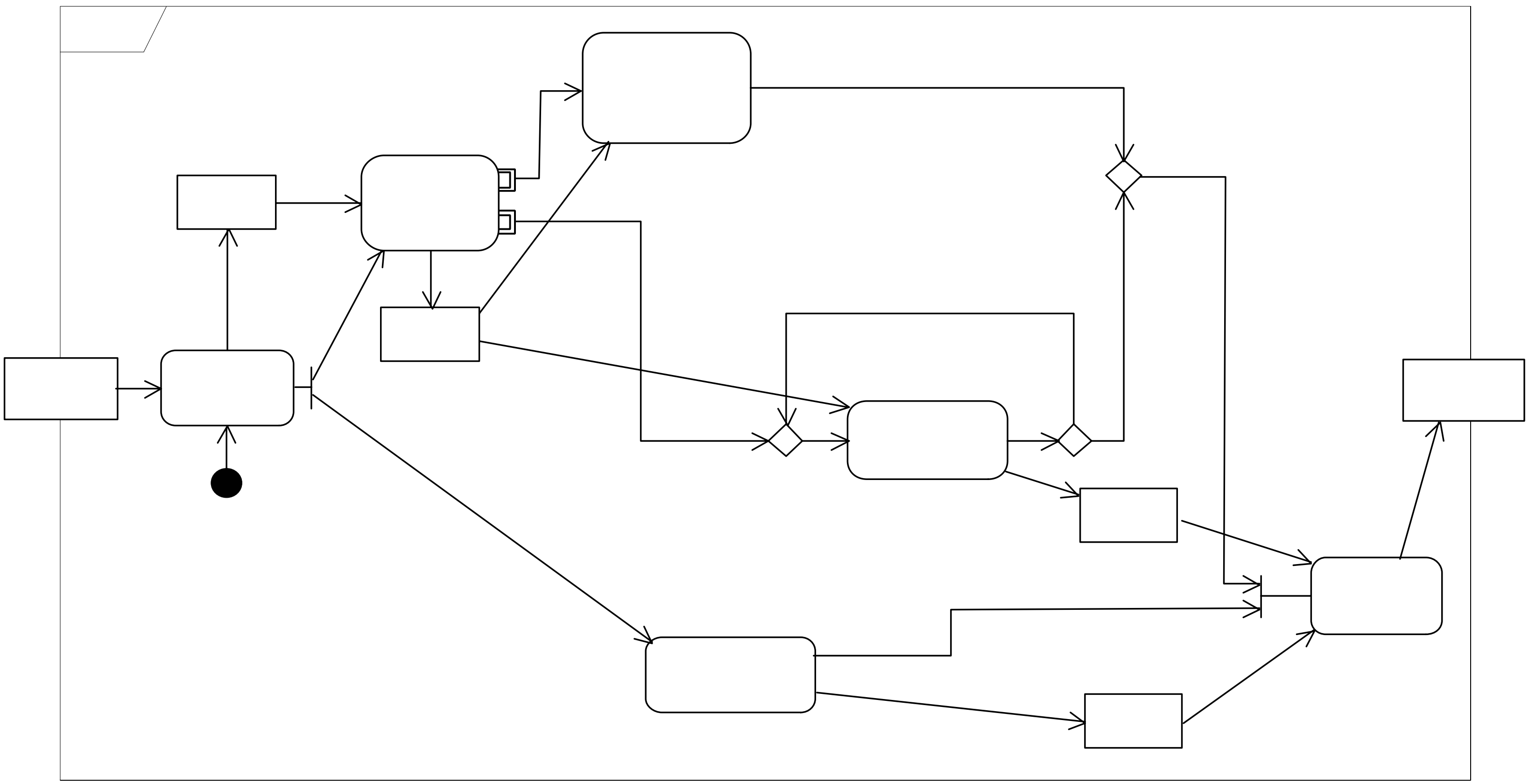


Activity Diagram
(with Swimlanes)



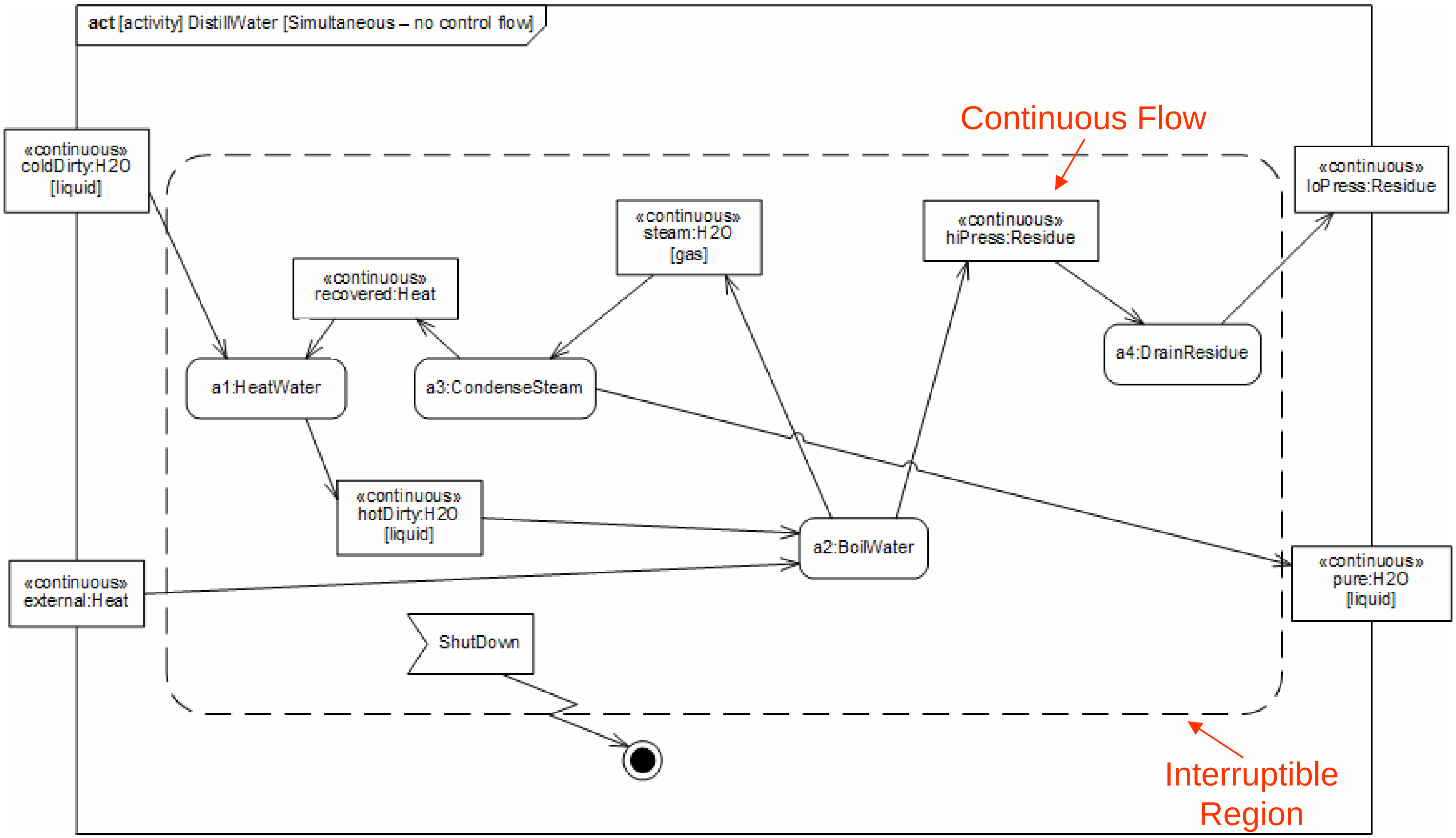
SysML EFFBD Profile

EFFBD - Enhanced Functional Flow Block Diagram



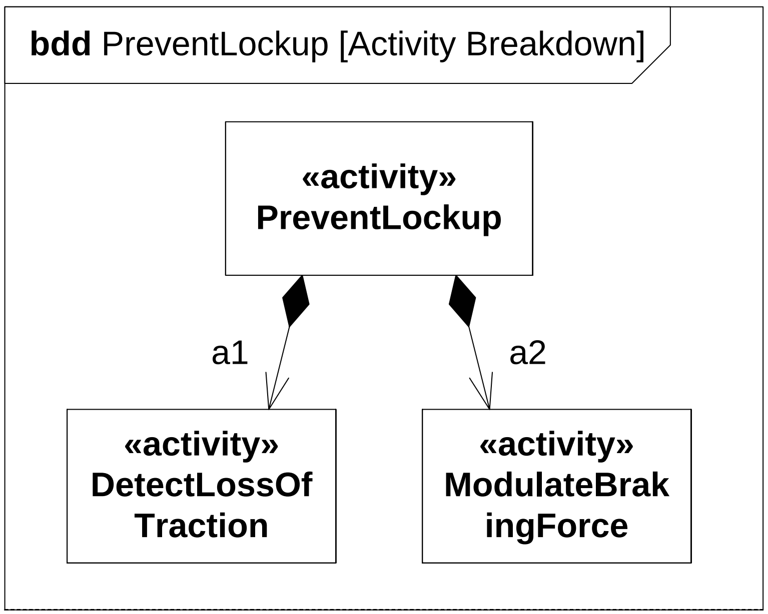
Aligning SysML with Classical Systems Engineering Techniques

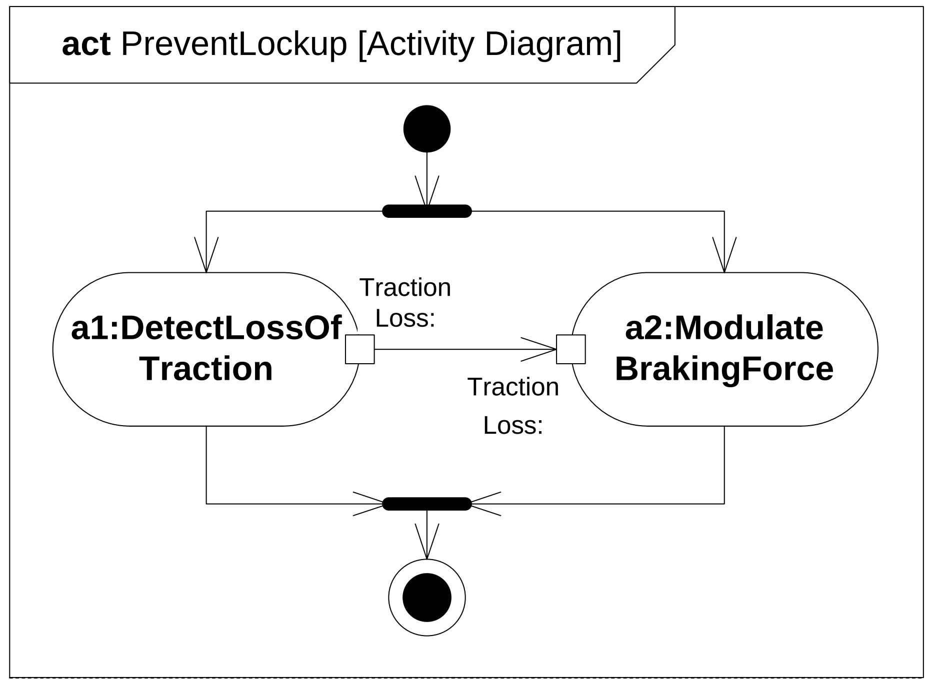
Distill Water Activity Diagram (Continuous Flow Modeling)



Representing Distiller Example in SysML Using Continuous Flow Modeling

Activity Decomposition





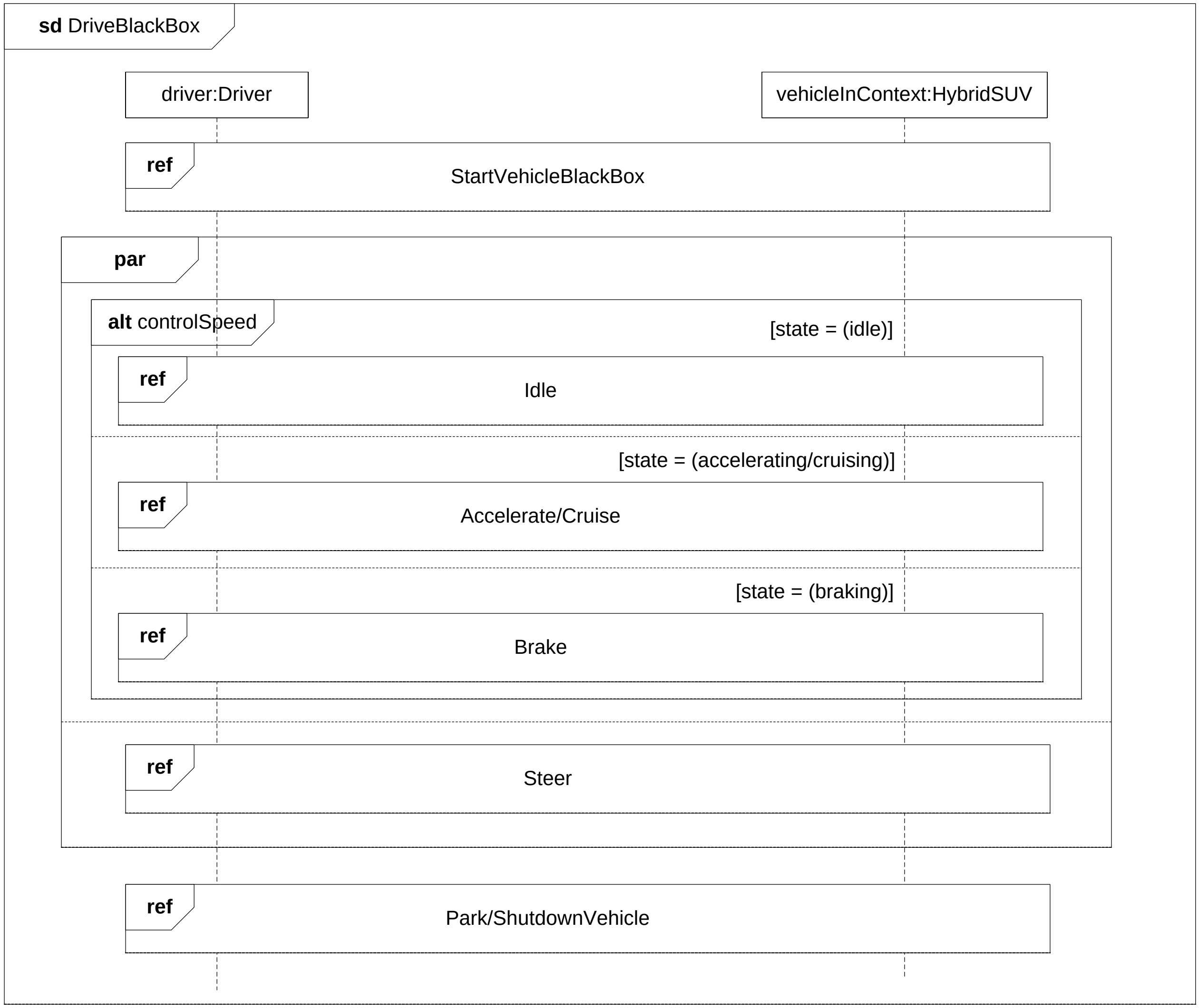
Definition

Use

Interactions

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

Black Box Interaction (Drive)



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

Black Box Sequence (StartVehicle)

Simple Black Box Interaction

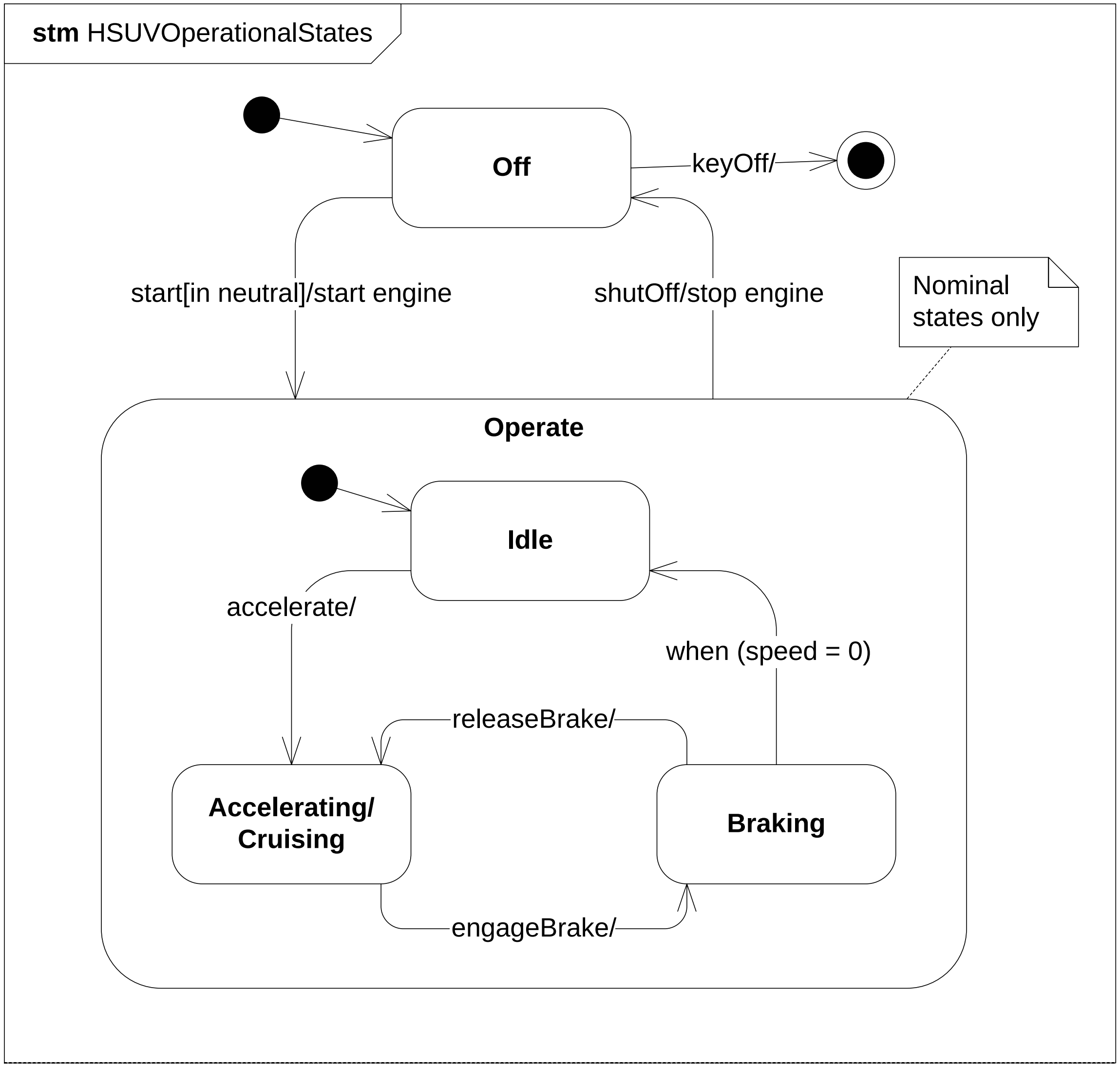
x

driver:Driver

State Machines

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

Operational States (Drive)

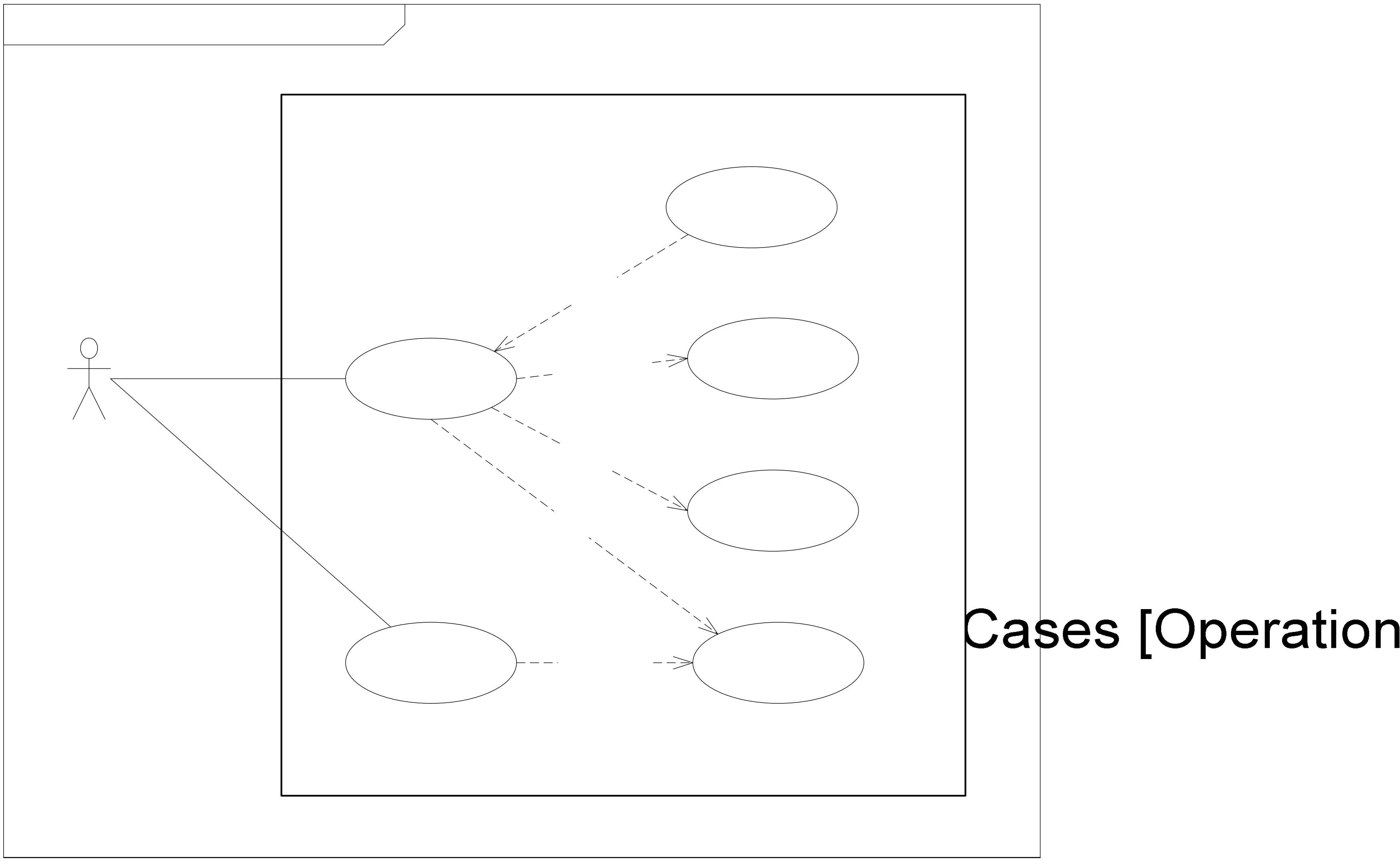


Transition notation: trigger[guard]/action

Use Cases

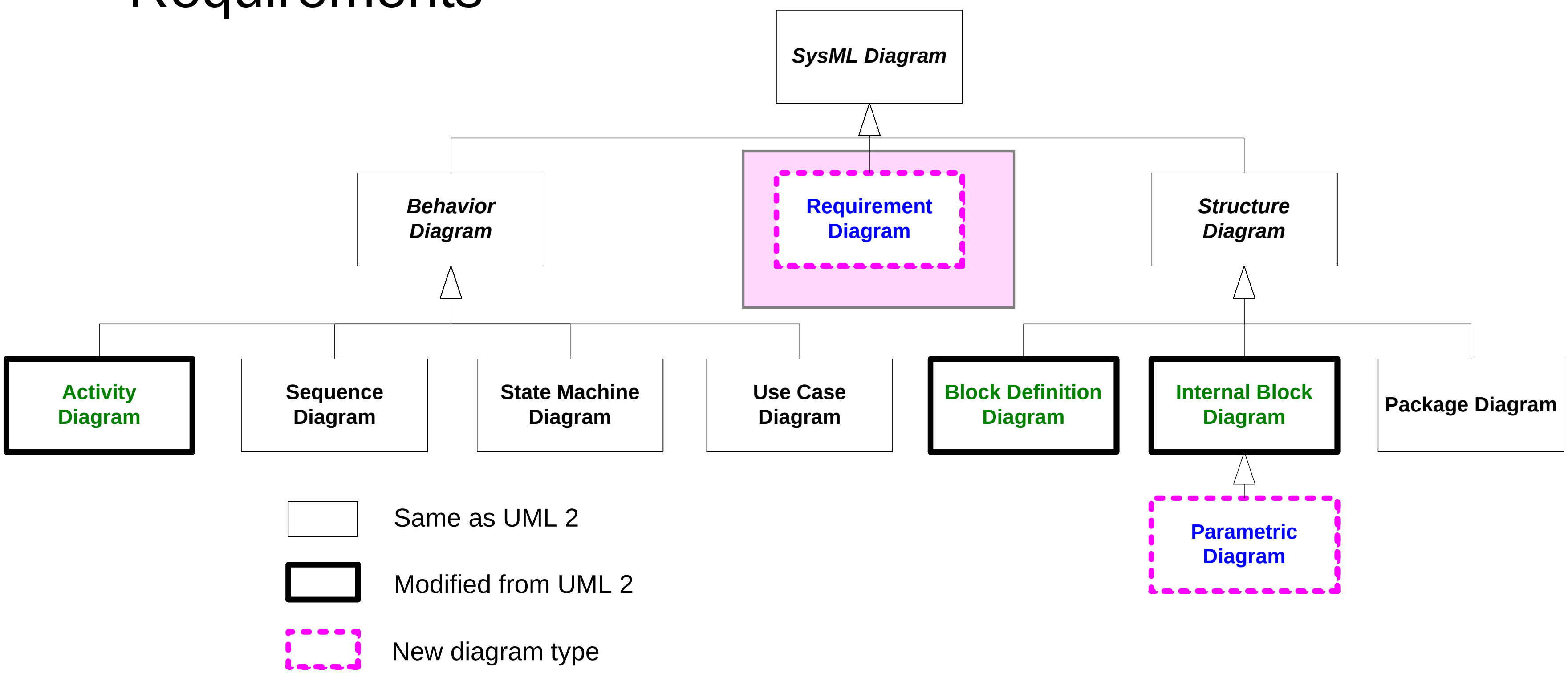
- Provide means for describing basic functionality in terms of usages/goals of the system by actors
- Common functionality can be factored out via include and extend relationships
- Generally elaborated via other behavioral representations to describe detailed scenarios
- No change to UML

Operational Use Cases



Cross-cutting Constructs

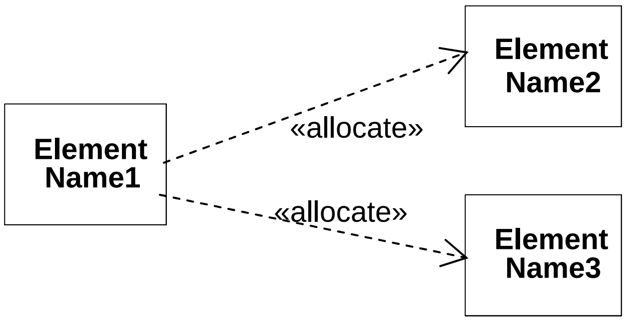
- Allocations
- Requirements



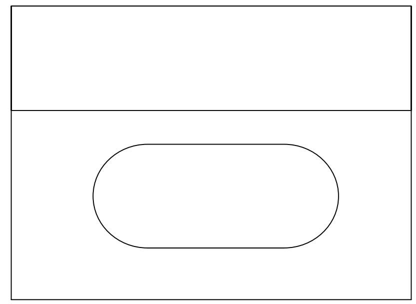
Allocations

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

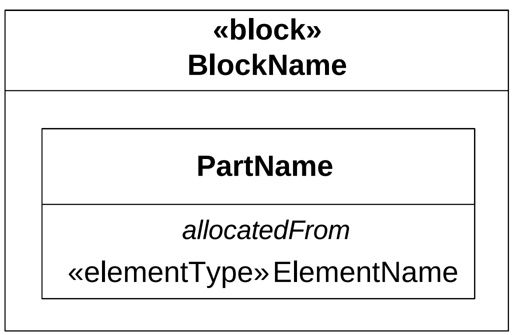
Different Allocation Representations (Tabular Representation Not Shown)



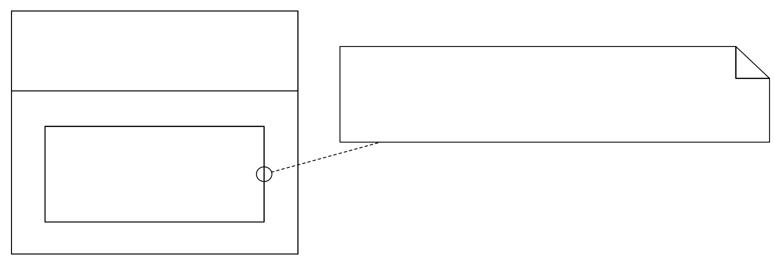
Allocate Relationship



Explicit Allocation of Activity to Swim Lane



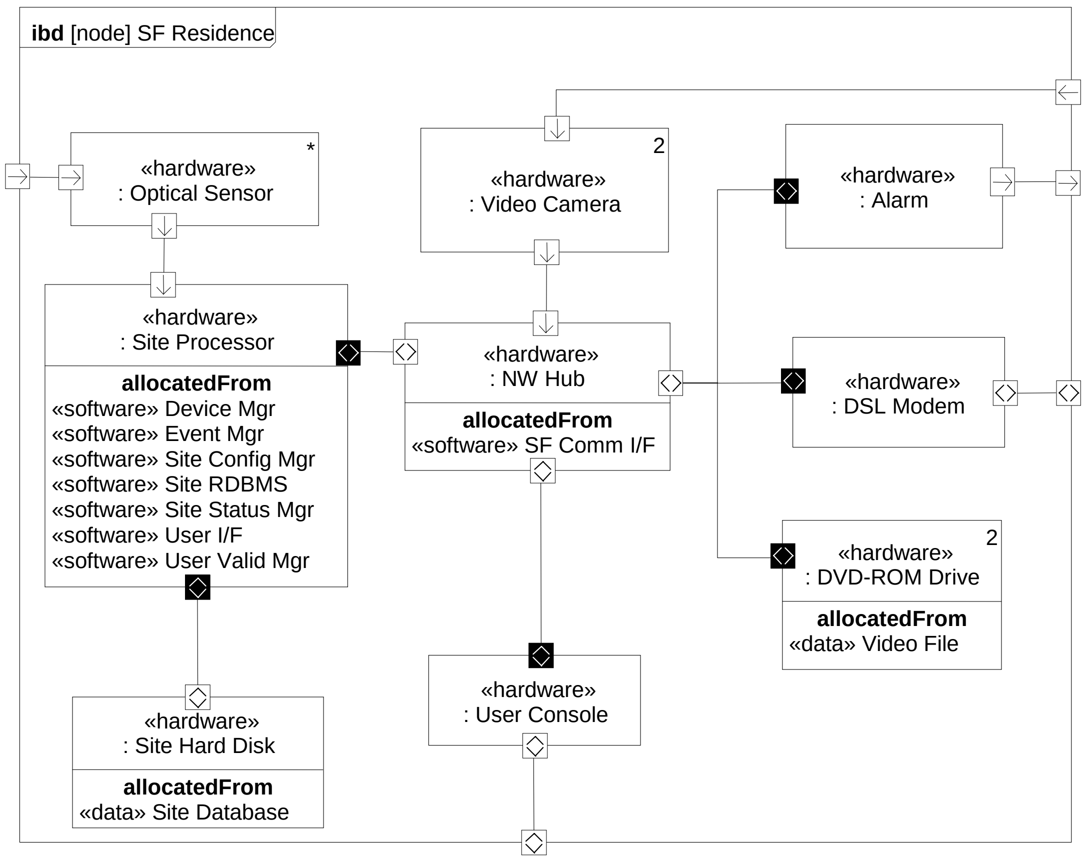
Compartment Notation



Callout Notation

SysML Allocation of SW to HW

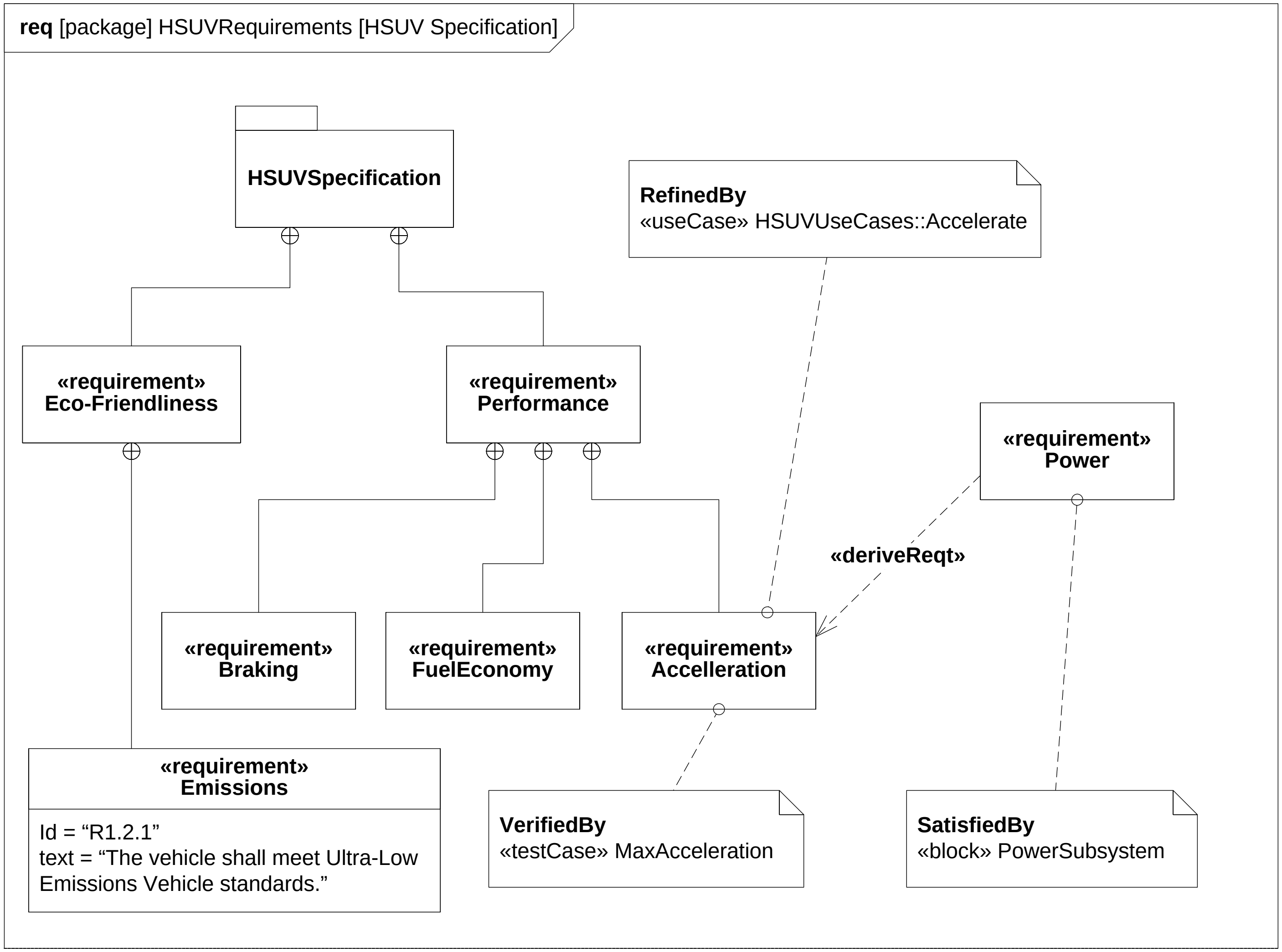
- In UML the deployment diagram is used to deploy artifacts to nodes
- In SysML allocation on ibd and bdd is used to deploy software/data to hardware



Requirements

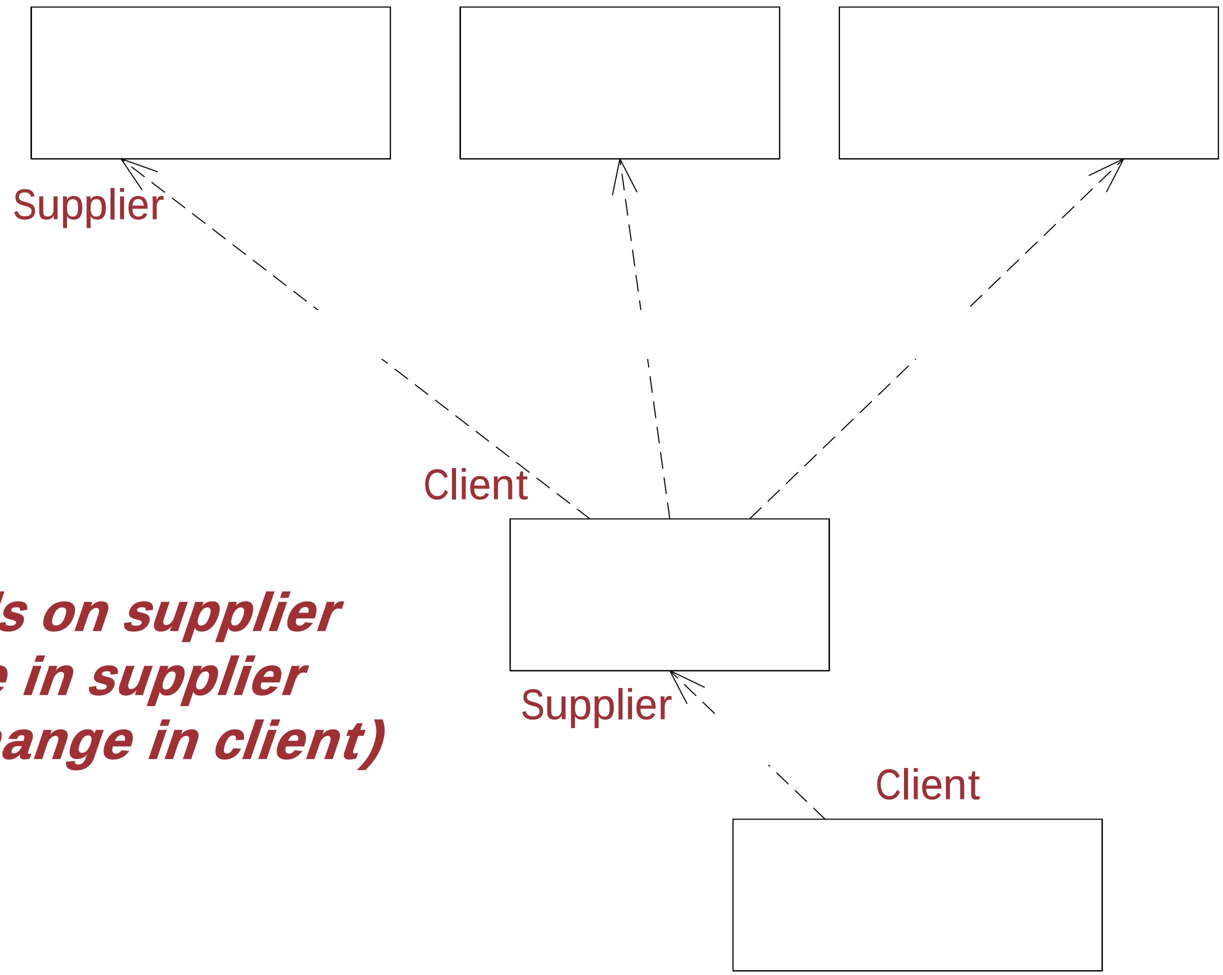
- The «requirement» stereotype represents a text based requirement
  - Includes id and text properties
  - Can add user defined properties such as verification method
  - Can add user defined requirements categories (e.g., functional, interface, performance)
- Requirements hierarchy describes requirements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

Requirements Breakdown



Requirement Relationships Model the Content of a Specification

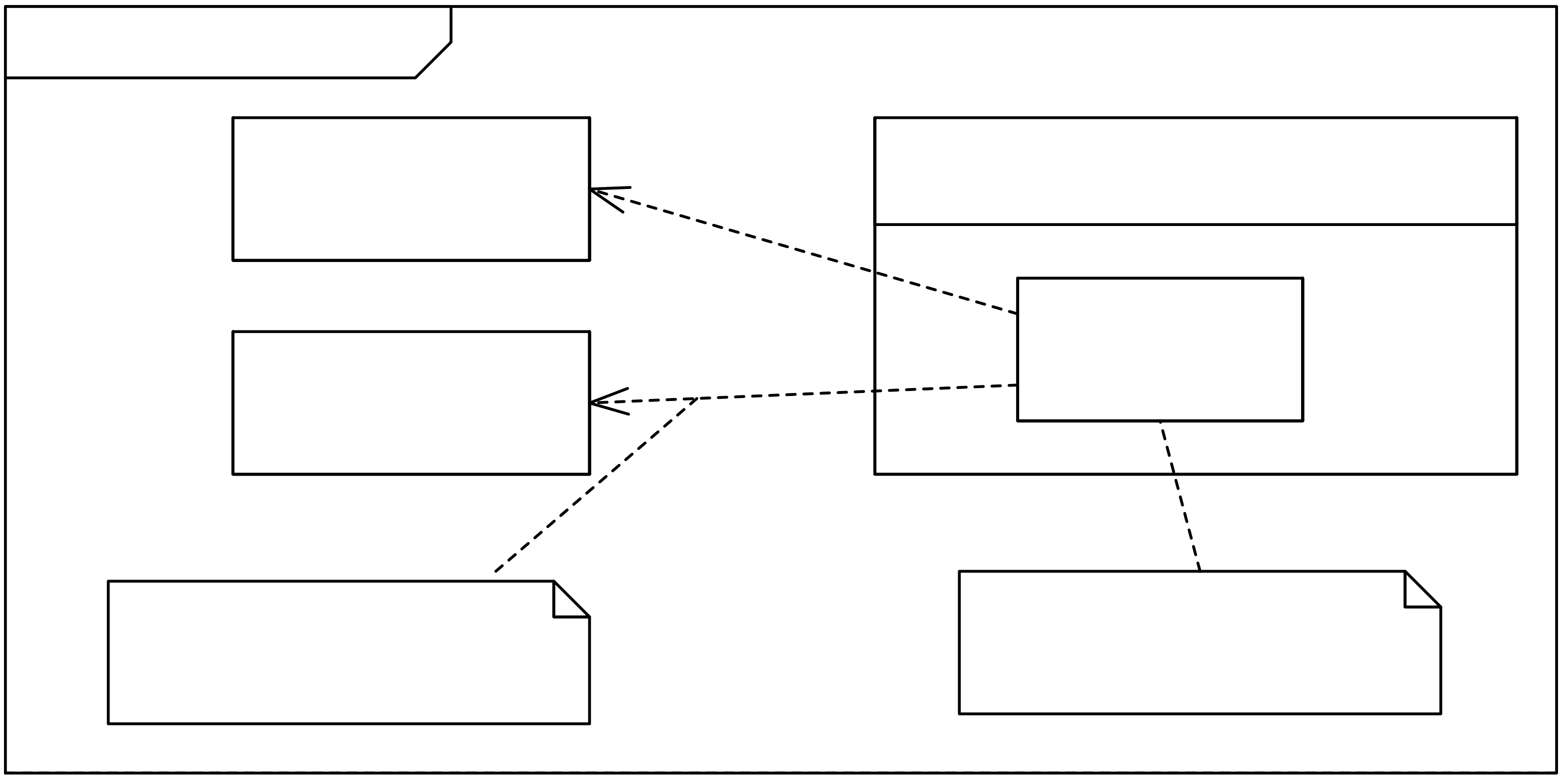
Example of Derive/Satisfy Requirement Dependencies



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

Arrow Direction Opposite Typical Requirements Flow-Down

Problem and Rationale



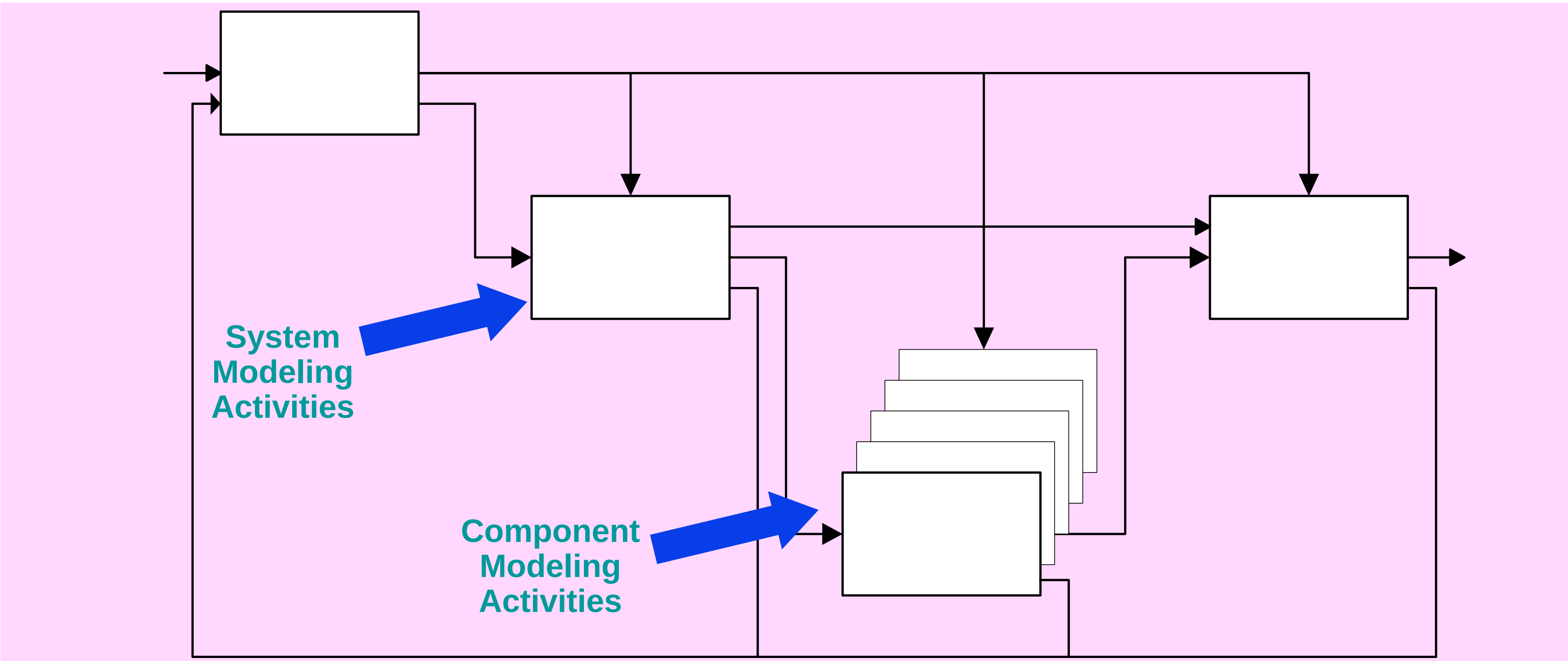
Problem and Rationale can be attached to any Model Element to Capture Issues and Decisions

bdd Master Cylinder requirements

SysML Modeling
as Part of the SE Process

OOSEM – Enhance Security System (ESS) Example

System Development Process



Integrated Product Development (IPD) is essential to improve communications

Stakeholder Reqts

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

Manage System Development

Pla

Systems Modeling Activities - OOSEM

Major SE Development Activities

Analyze Needs

- Mission use cases/scenarios
- Enterprise model

Define System Requirements

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

Define Logical Architecture

- Logical architecture

Synthesize Physical Architecture

- Node diagram
- HW, SW, Data architecture

Optimize & Evaluate Alternatives

- Engr Analysis Models
- Trade studies

Validate & Verify System

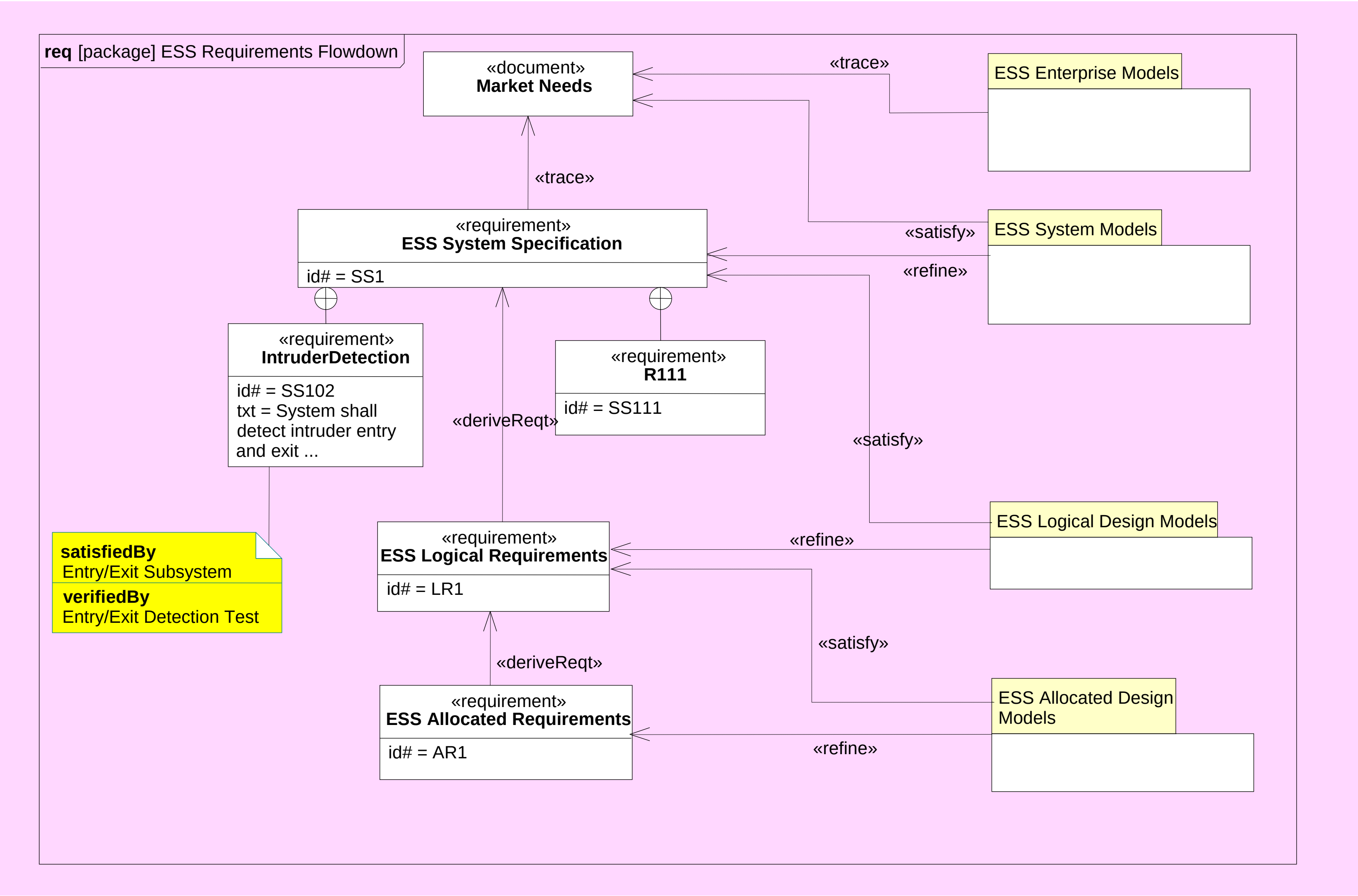
- Test cases/procedures

Common Subactivities

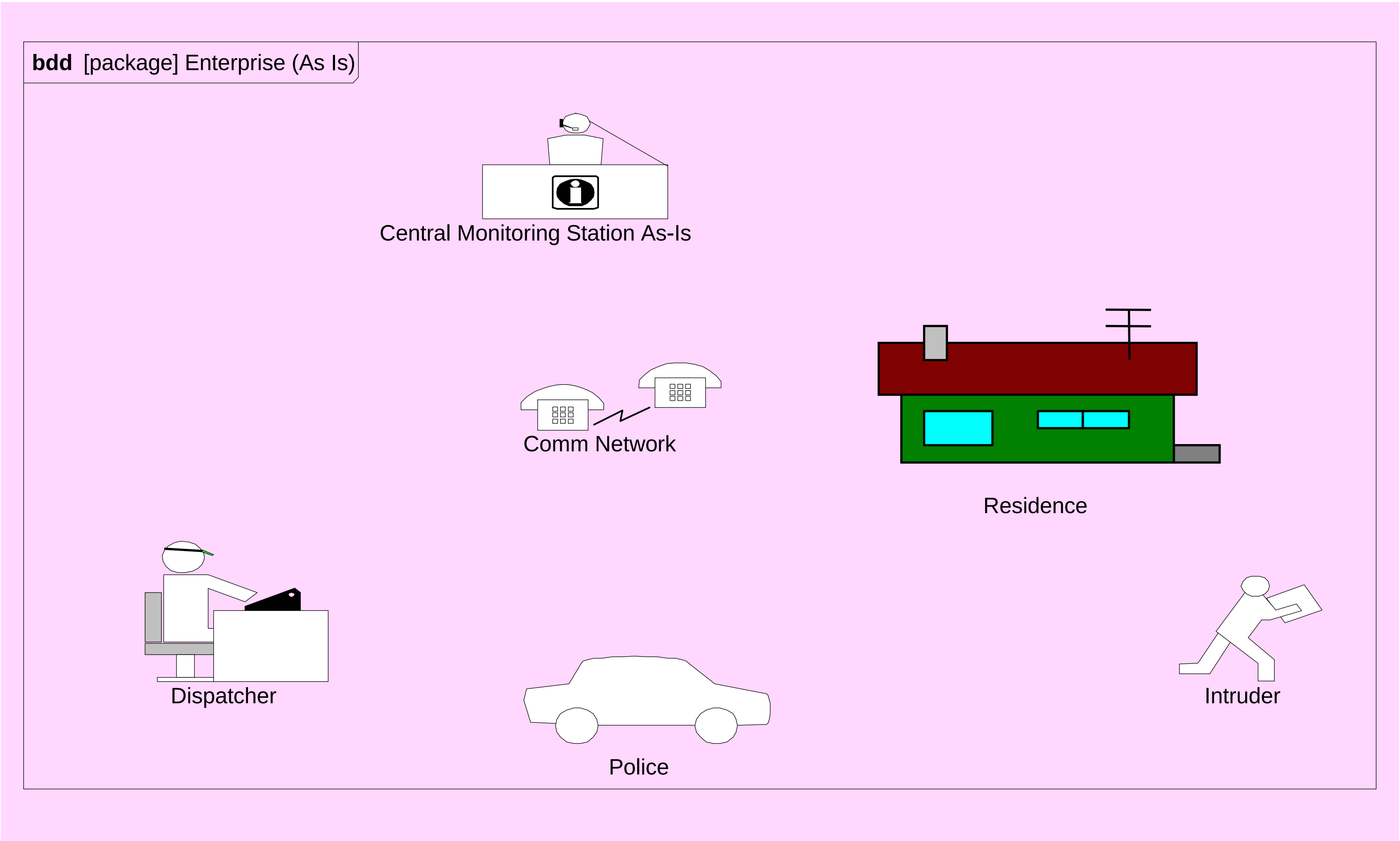
Enhanced Security System Example

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

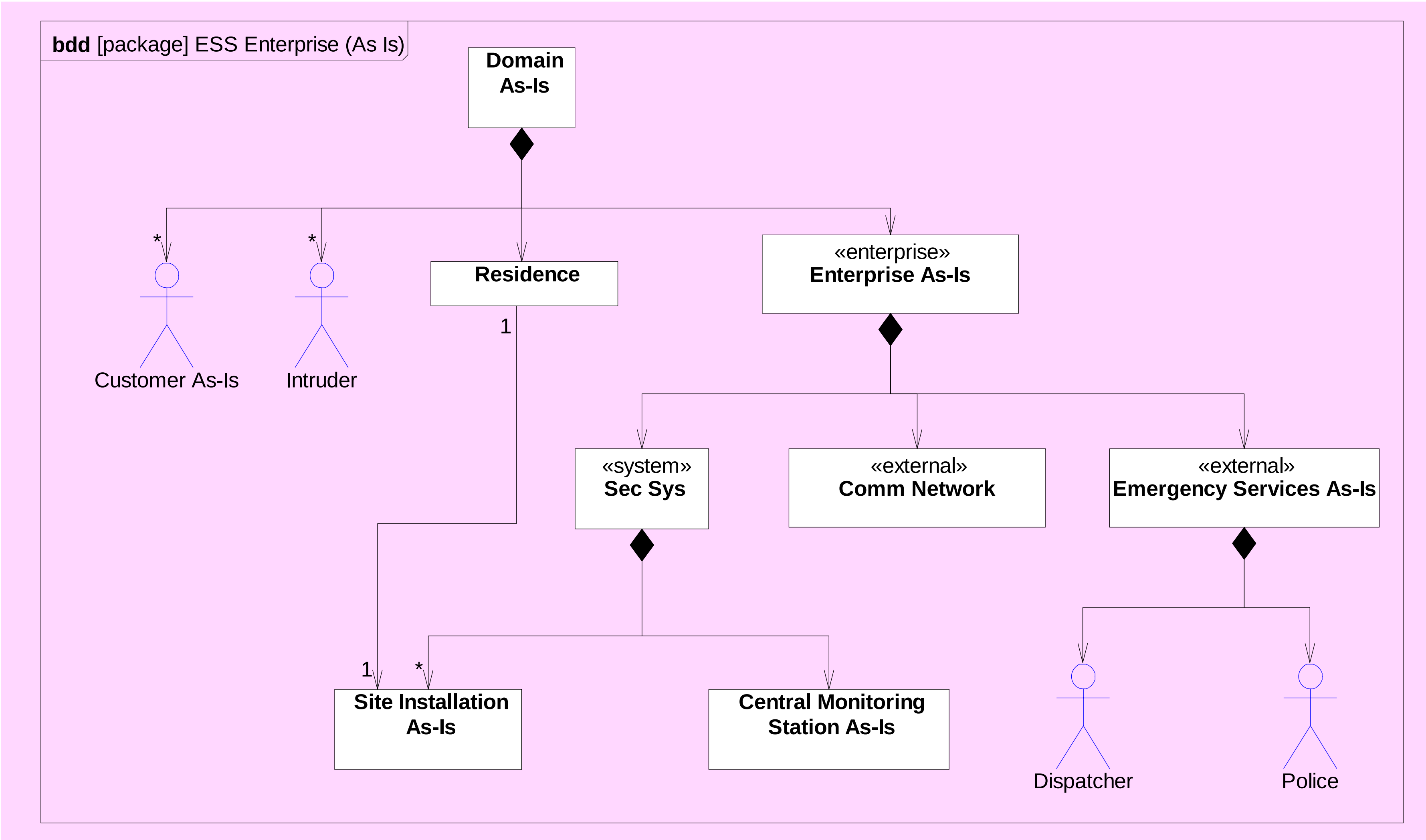
ESS Requirements Flowdown



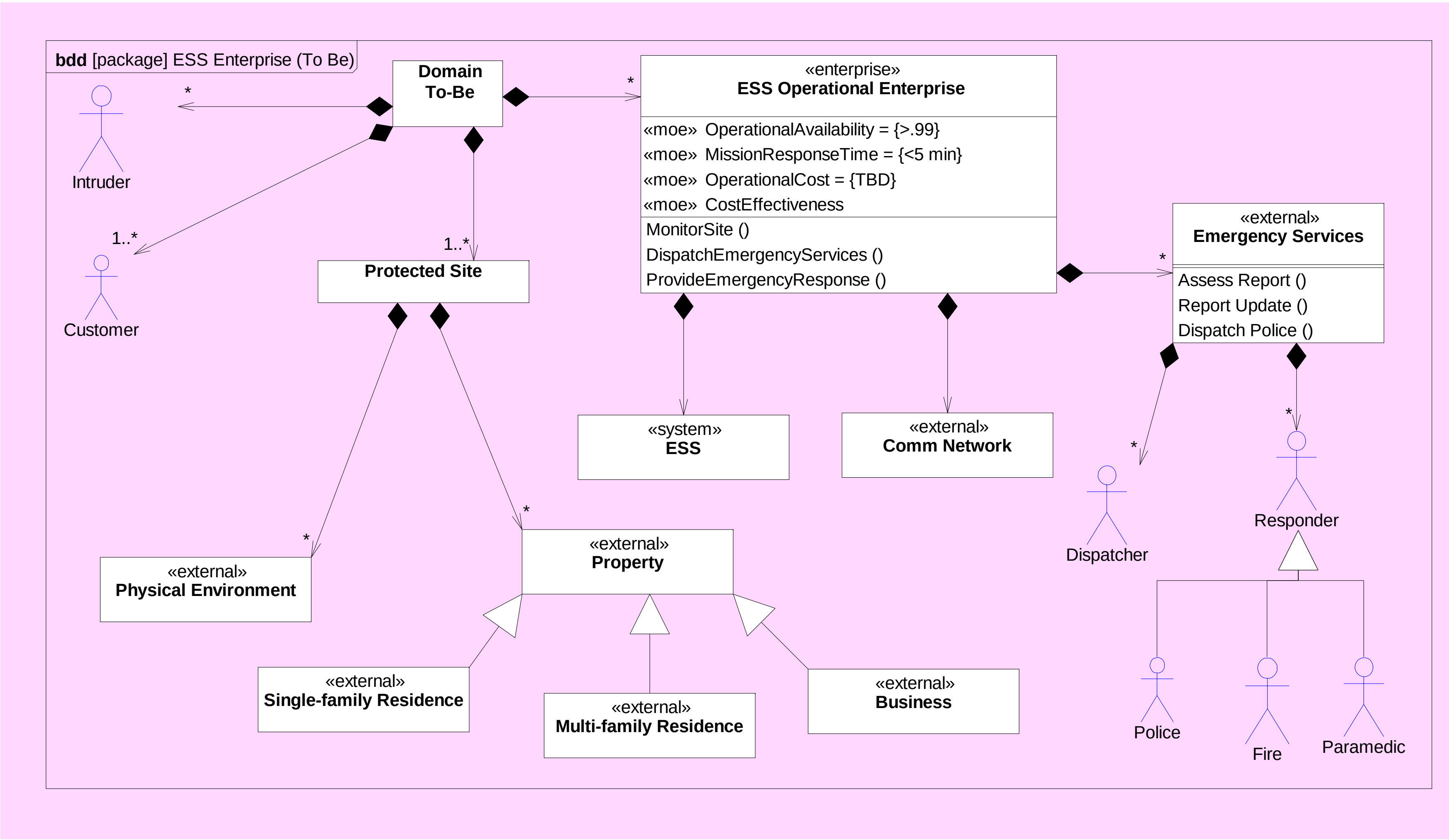
Operational View Depiction



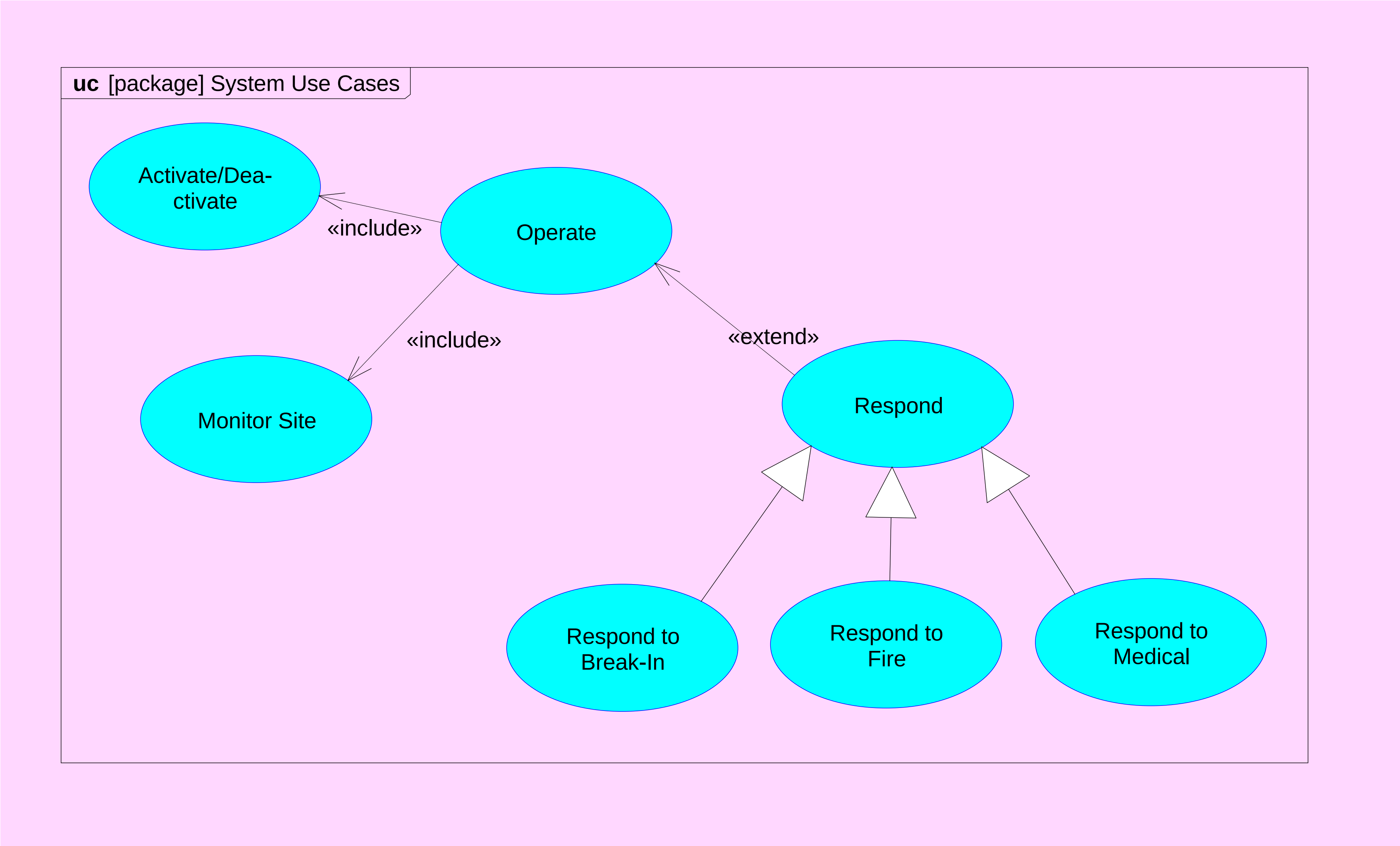
ESS Enterprise As-Is Model



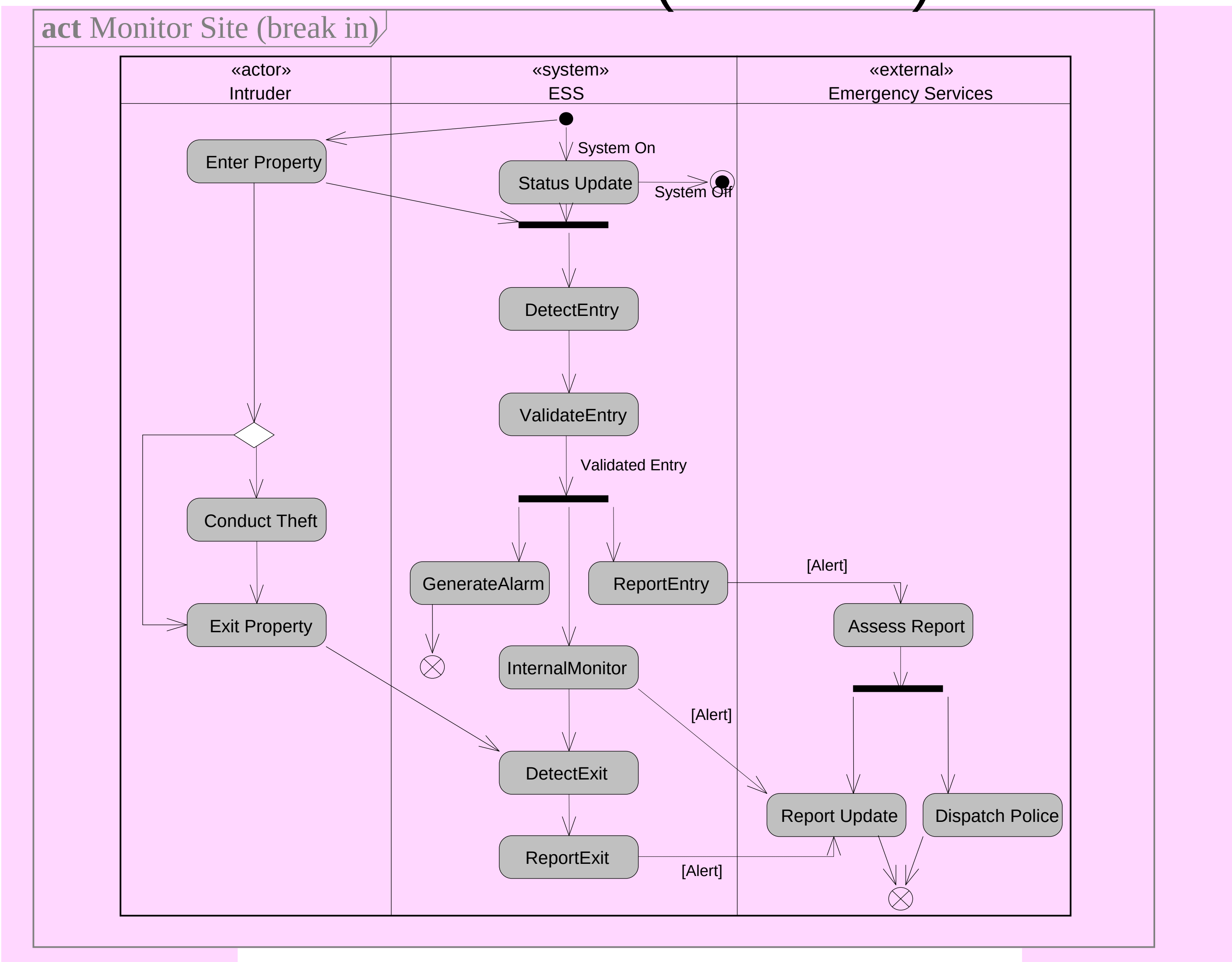
ESS Operational Enterprise To-Be Model



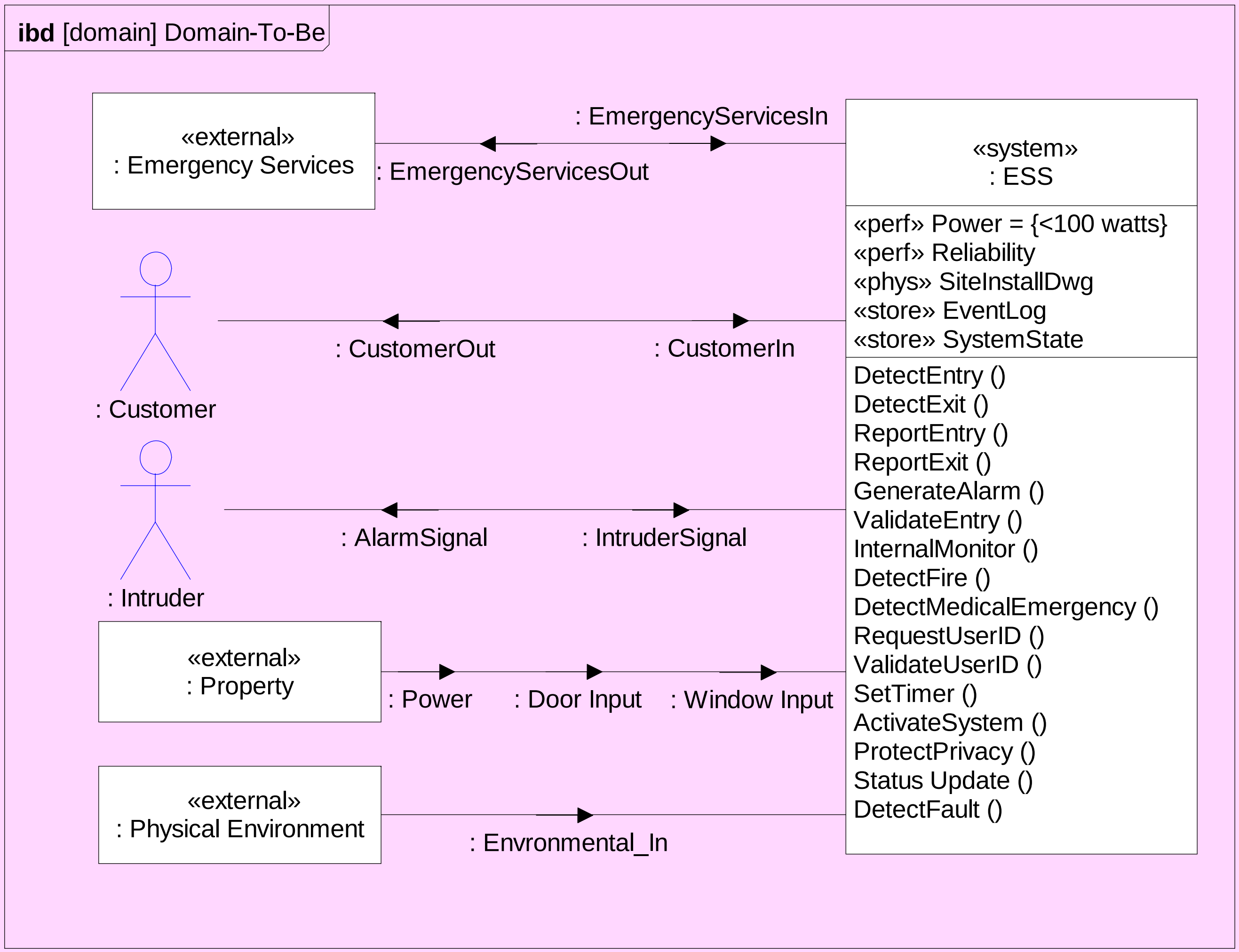
System Use Cases - Operate



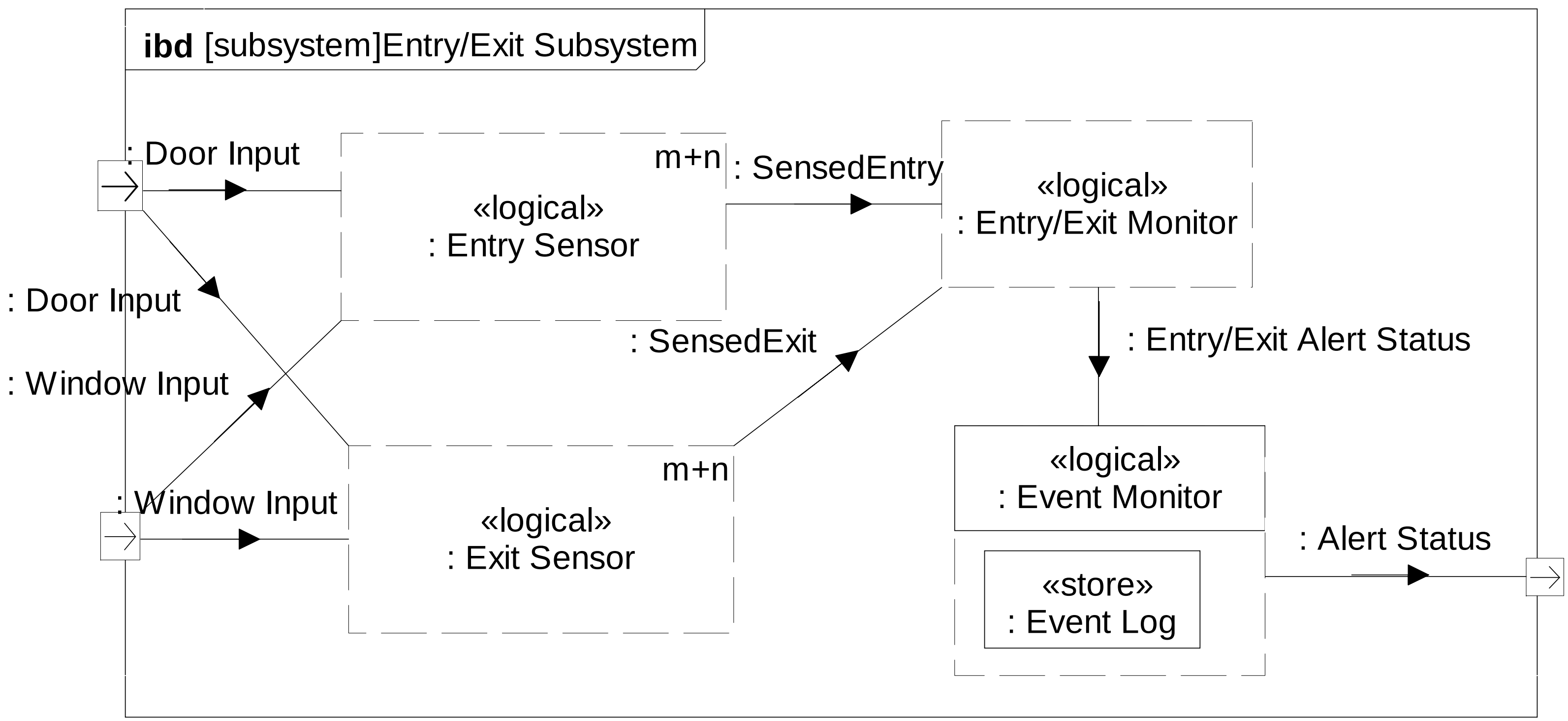
System Scenario: Activity Diagram Monitor Site (Break-In)



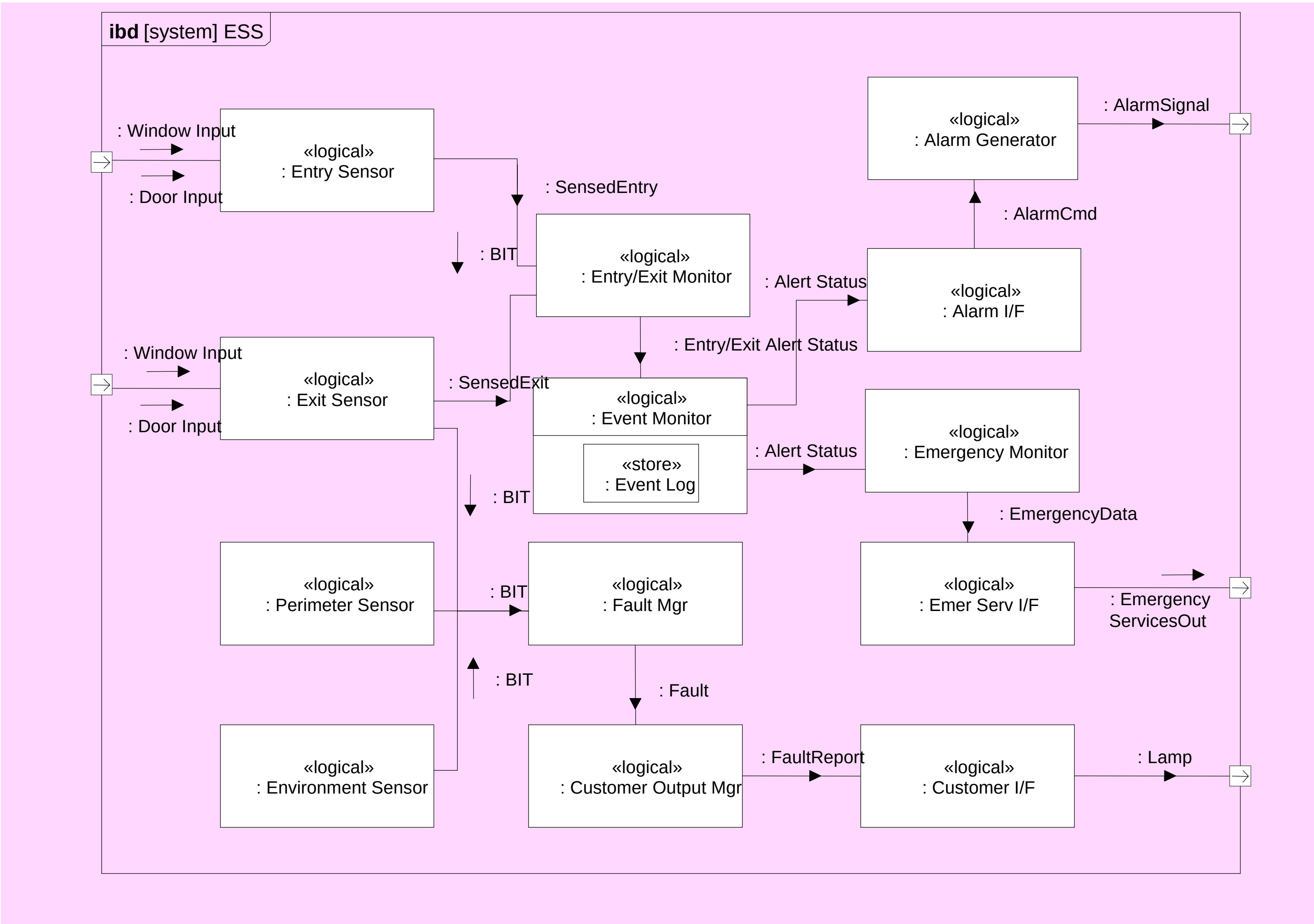
ESS Elaborated Context Diagram



ESS Logical Design – Example Subsystem



ESS Logical Design (Partial)

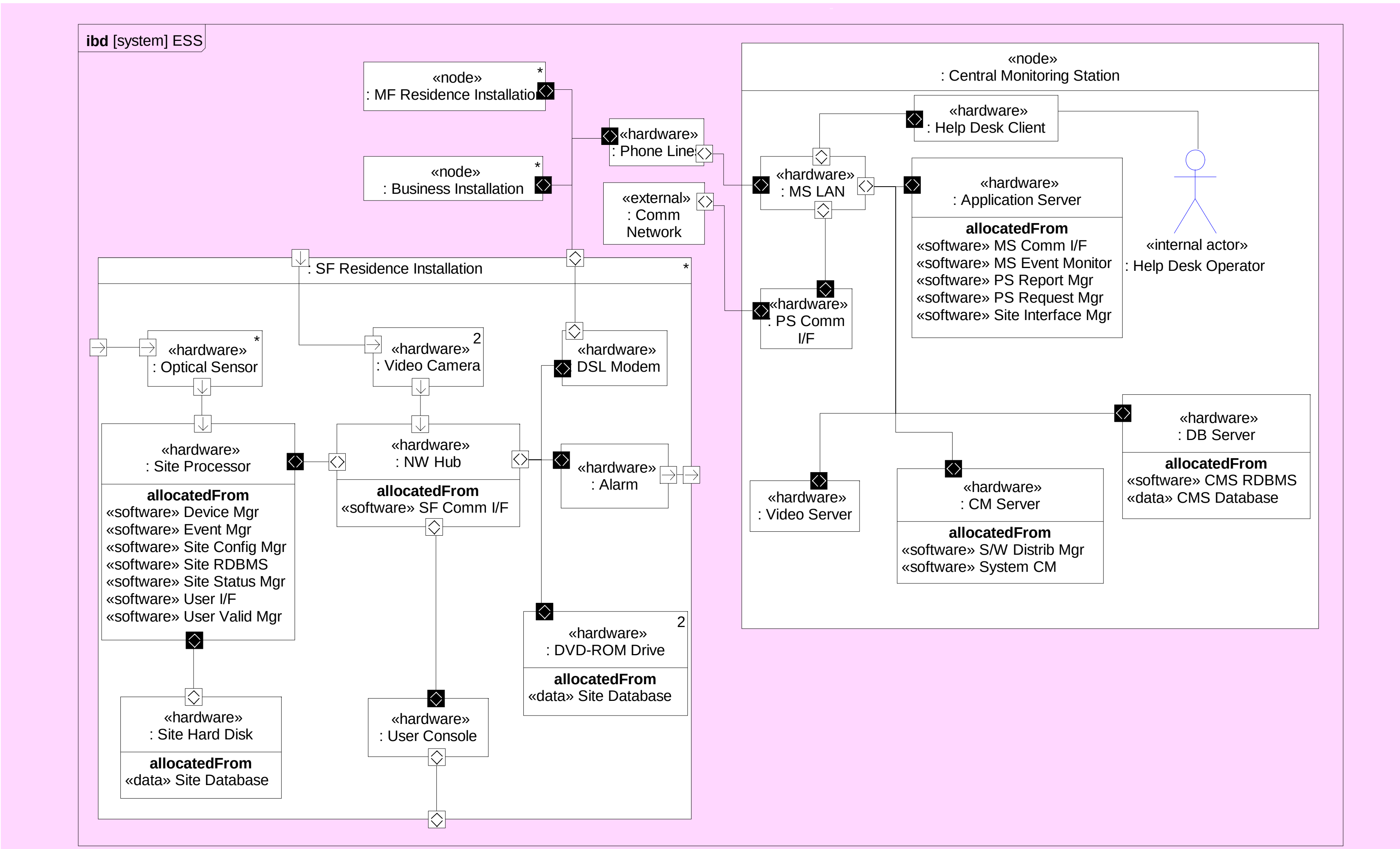


ESS Allocation Table (partial)

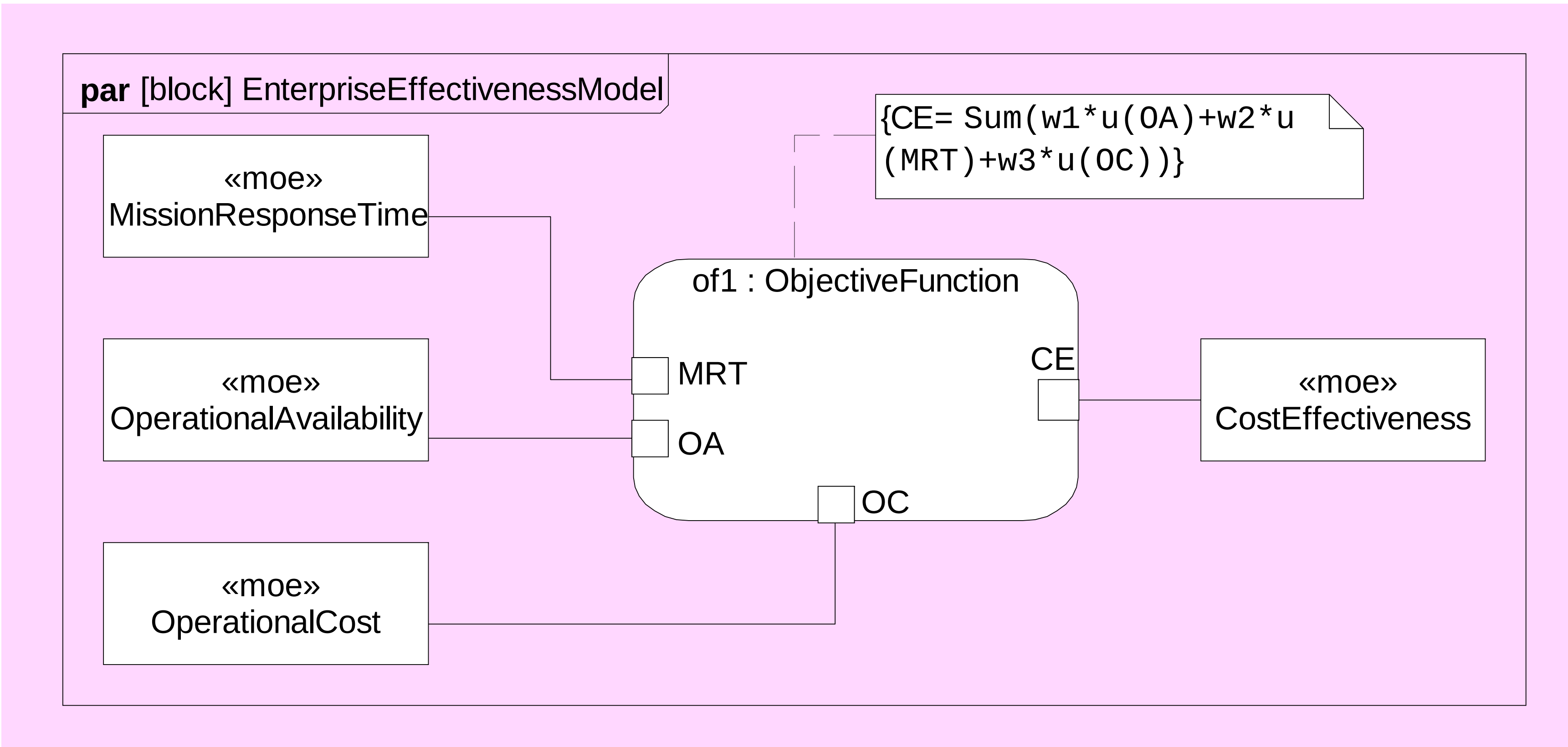
- Allocating Logical Components to HW, SW, Data, and Procedures components

	Type		Logical Components: 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
Physical Components	«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	

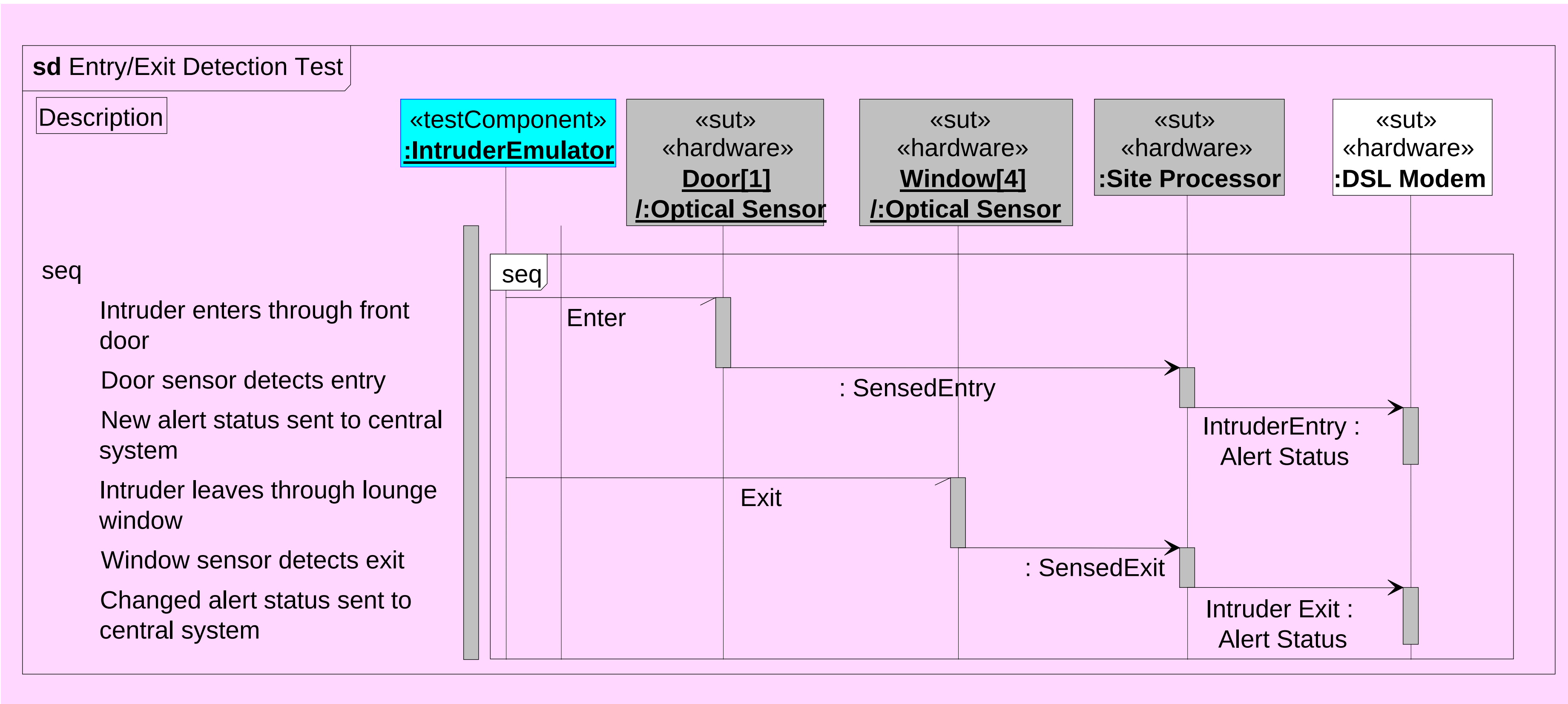
ESS Deployment View



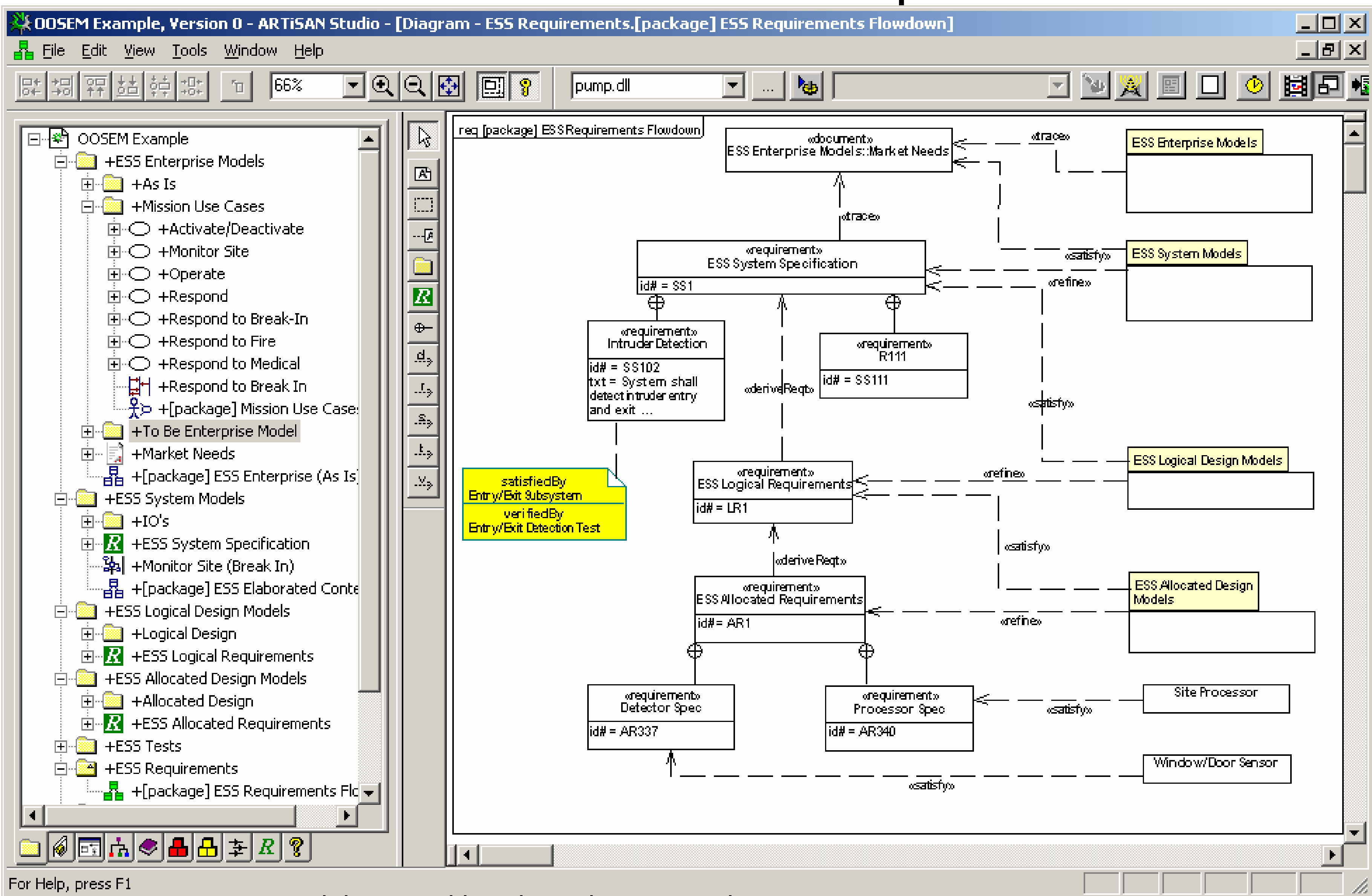
ESS Parametric Diagram To Support Trade-off Analysis



Entry/Exit Test Case

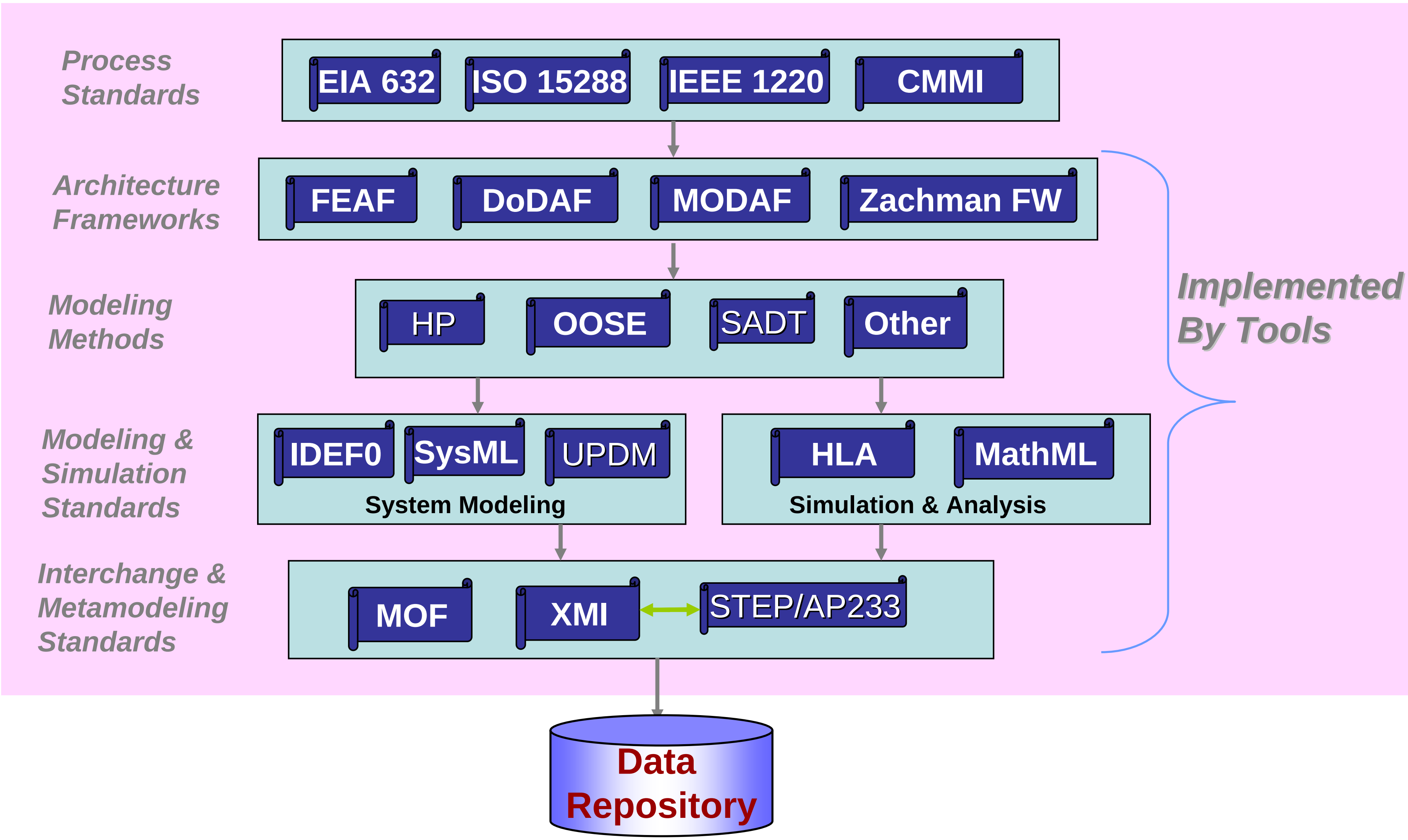


OOSEM Browser View
Artisan Studio™ Example



SysML in a Standards Framework

Systems Engineering Standards Framework (Partial List)



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

Agreement Processes

- 5.2.2 Acquisition Process
- 5.2.3 Supply Process

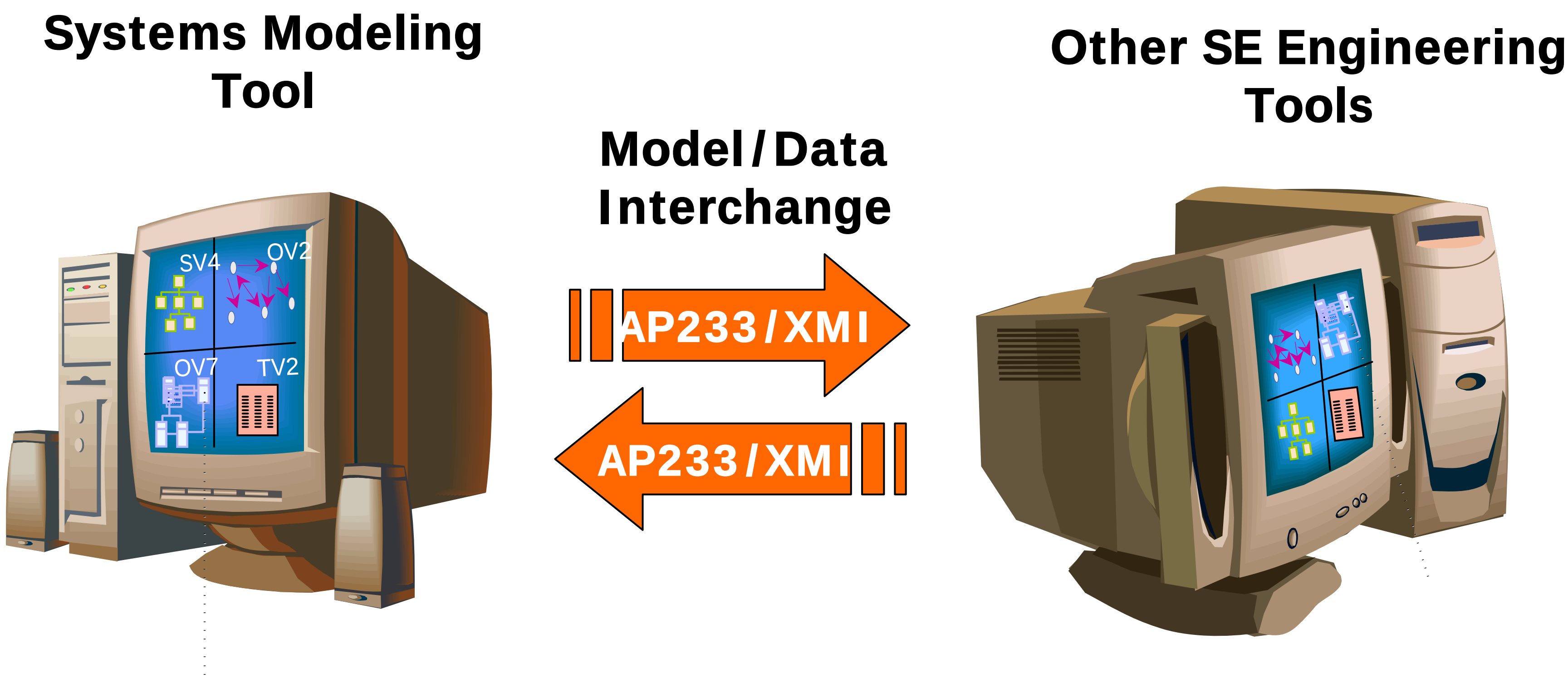
Project Processes

- 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 Reqts Definition Process
- 5.5.3 Reqts 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

Standards-based Tool Integration with SysML



Participating SysML Tool Vendors

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

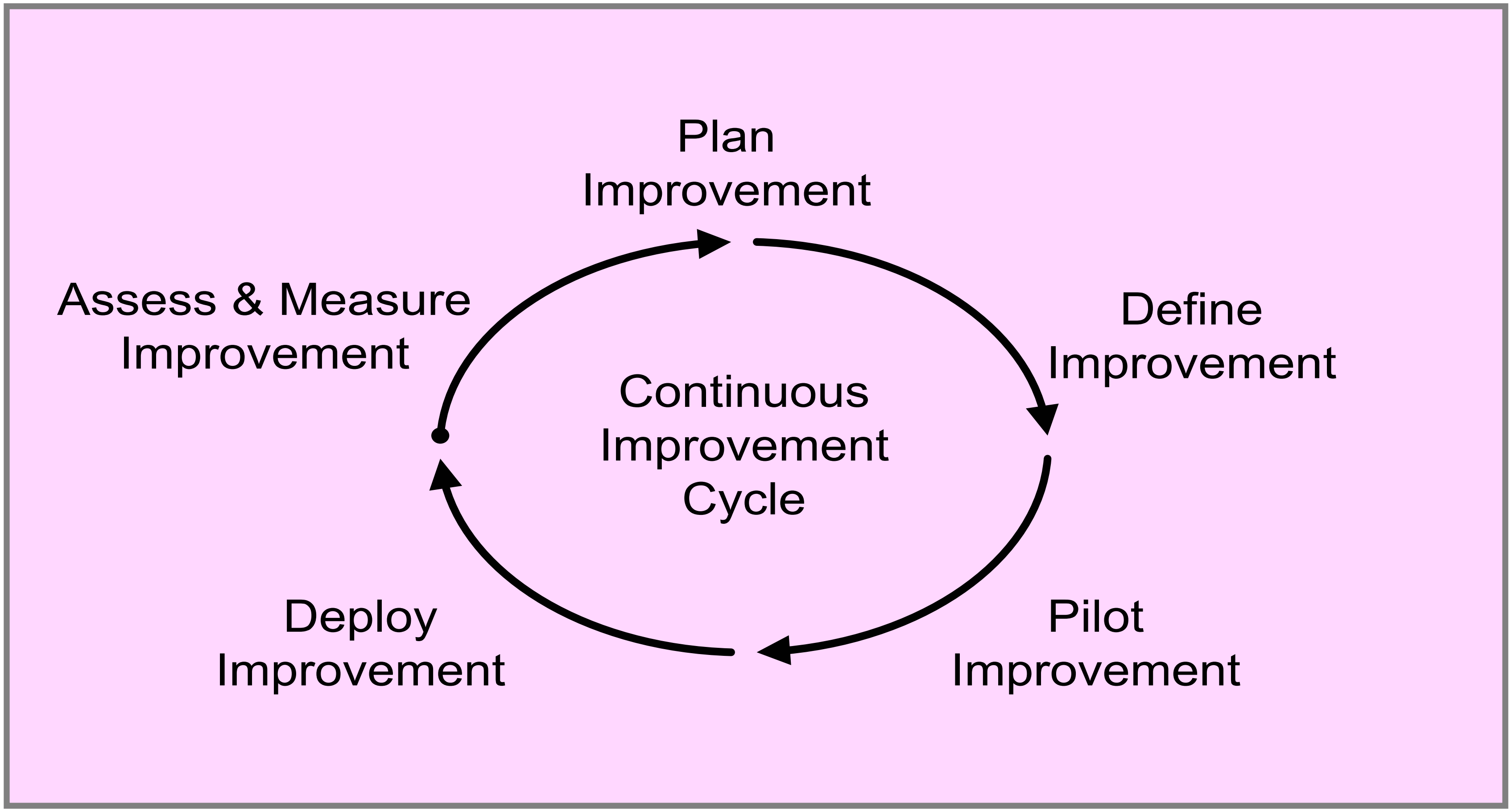
Note: Free Visio SysML Template available at OMG SysML site (http://www.omgsysml.org)

UML Profile for DoDAF/MODAF (UPDM) Standardization

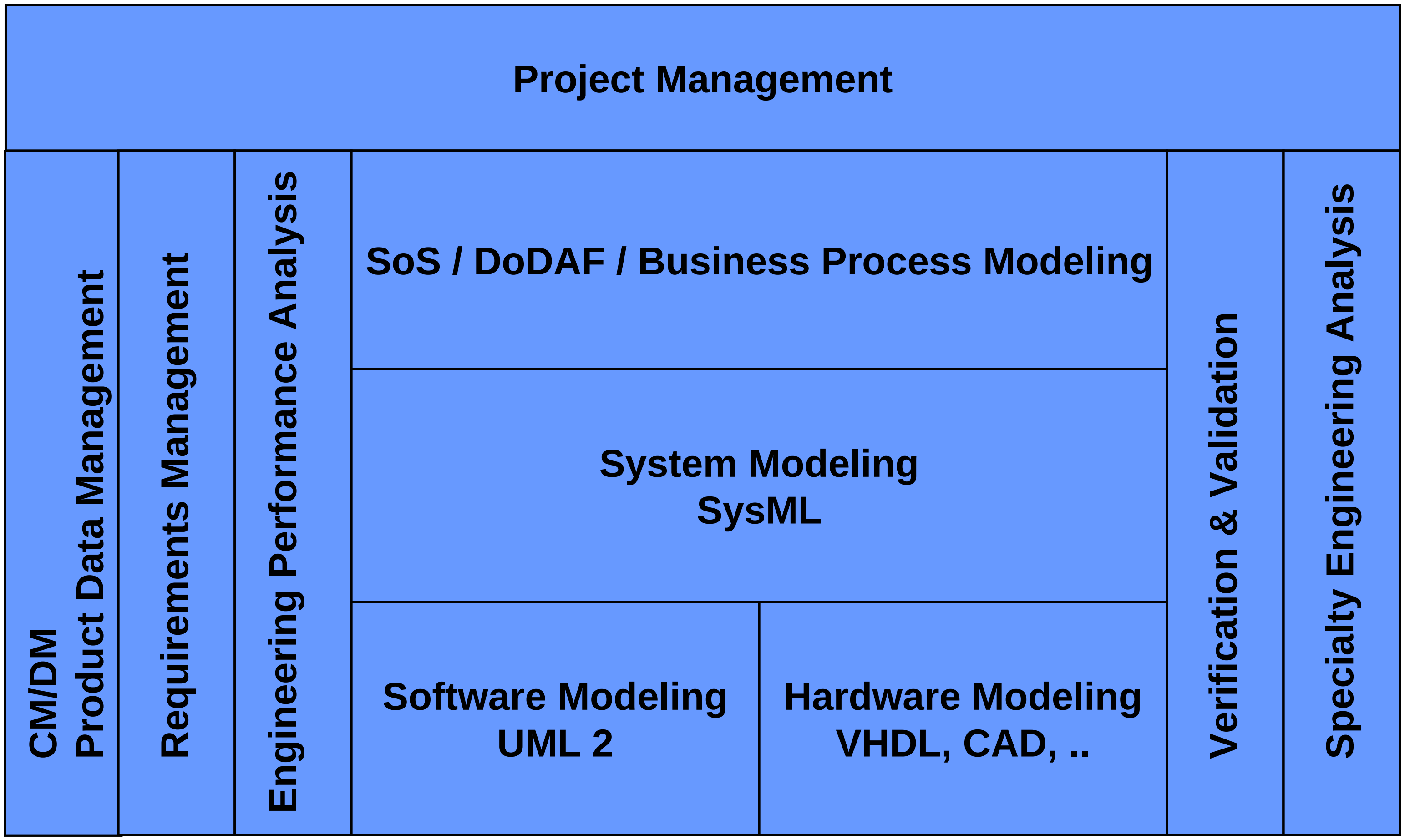
- Current initiative underway to develop standard profile for representing DODAF and MODAF products
  - Requirements for profile issued Sept 05
  - Final submissions expected Dec ‘06
- Multiple vendors and users participating
- Should leverage SysML

Transitioning to SysML

Using Process Improvement To Transition to SysML



Integrated Tool Environment



Summary and Wrap up

Summary

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

References

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