System Modeling Language (SysML)

Prof. Dr. Franz Wotawa
Institute of Software Engineering and Artificial Intelligence
wotawa@tugraz.at

Excerpt from the *OMG Systems Modeling Language (OMG SysML™)Tutorial, September, 2009, Authors: Sanford Friedenthal, Alan Moore, and Rick Steiner*

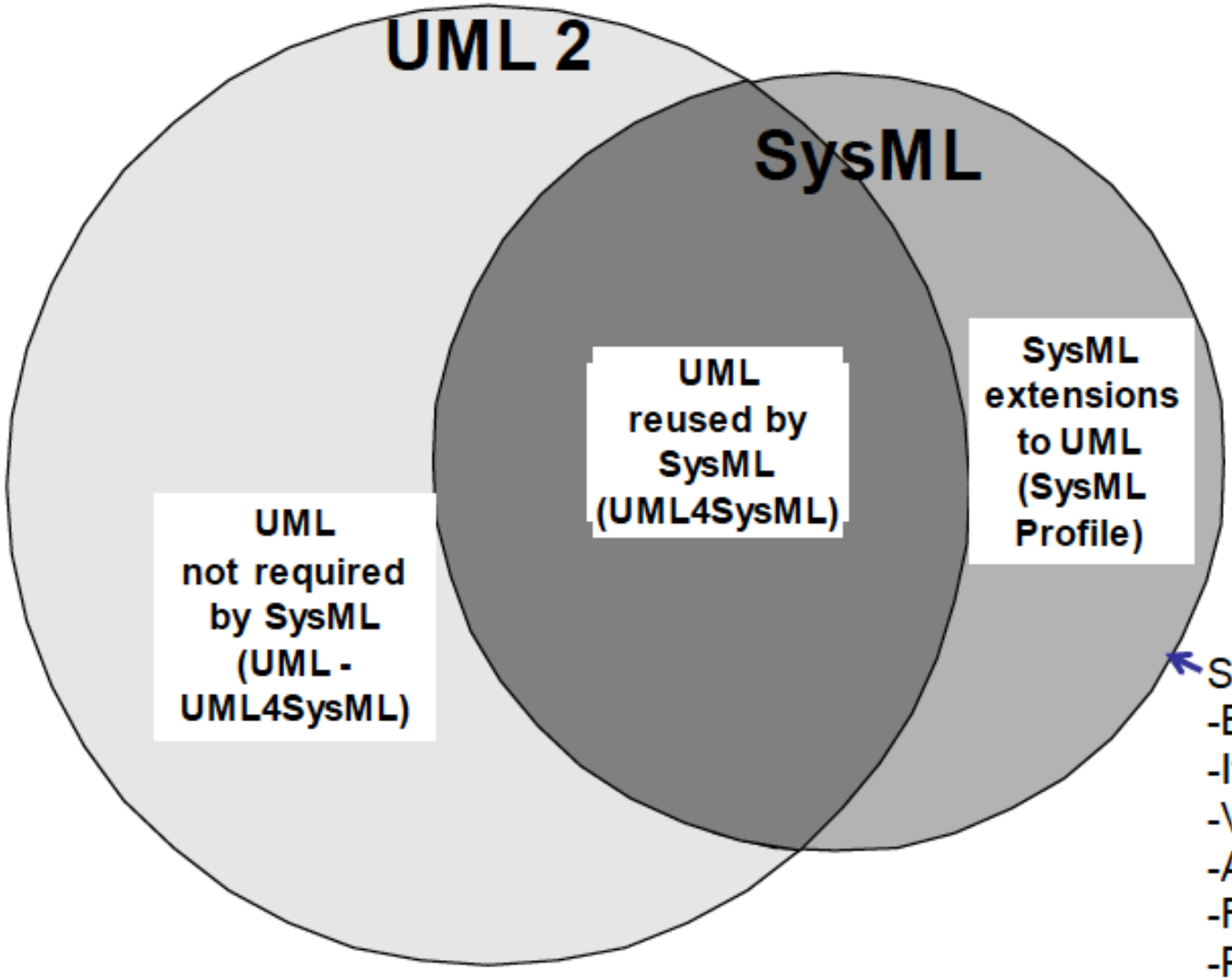
http://www.omgsysml.org/INCOSE-OMGSysML-Tutorial-Final-090901.pdf

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)

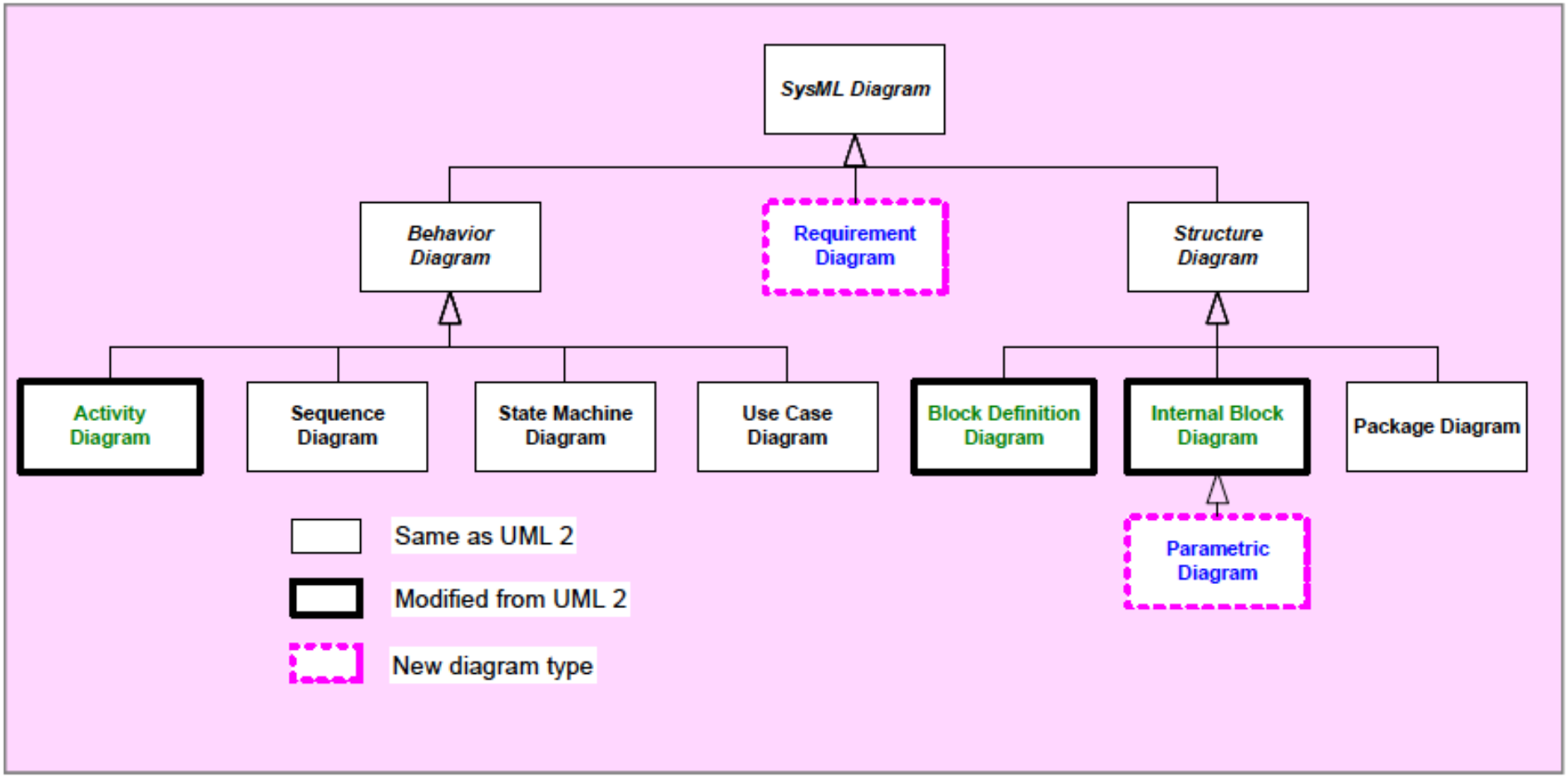
SysML is Critical Enabler for Model Driven SE

Relationship Between SysML and UML



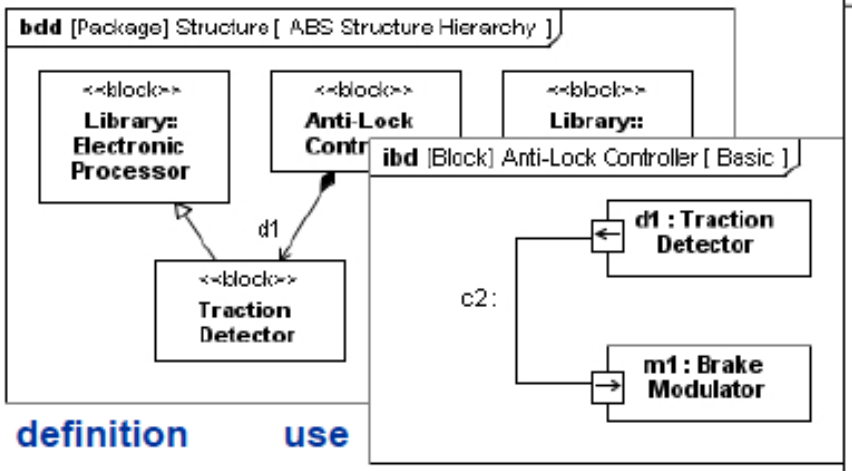
SysML Extensions
-Blocks
-Item flows
-Value properties
-Allocations
-Requirements
-Parametrics
-Continuous flows
-…

SysML Diagram Taxonomy



4 Pillars of SysML – ABS Example

1. Structure

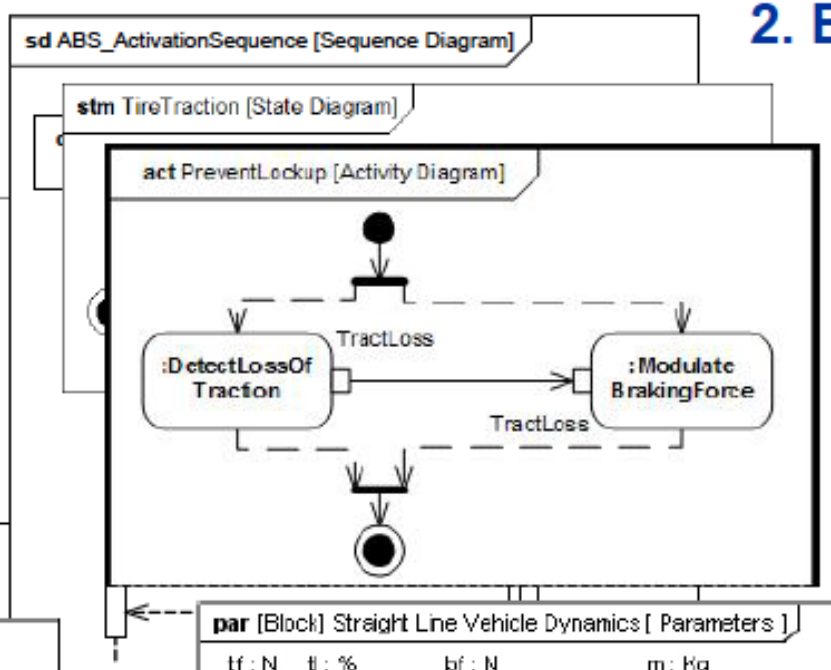


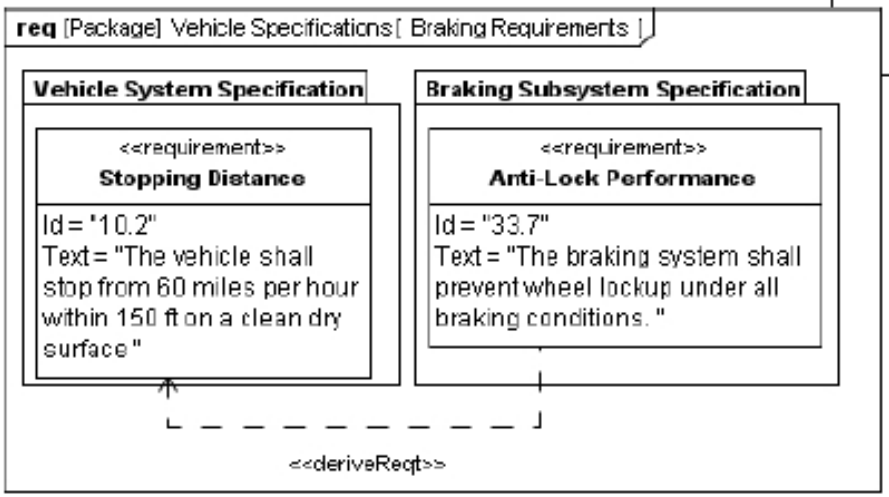
2. Behavior

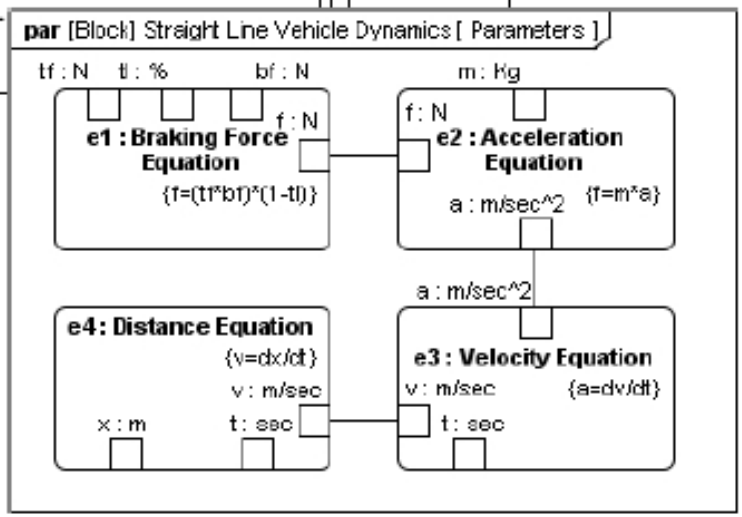
interaction

state machine

activity/ function







3. Requirements

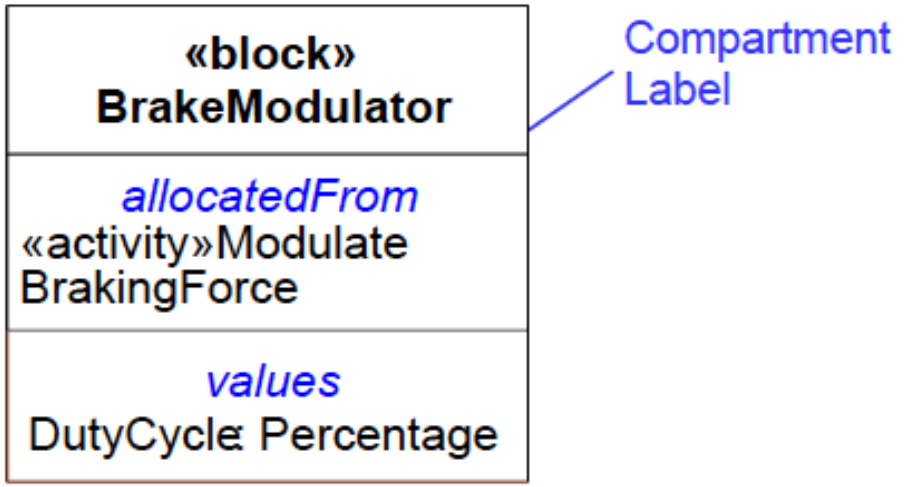
4. Parametrics

SysML Block Diagrams

- Ako UML Class diagrams

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

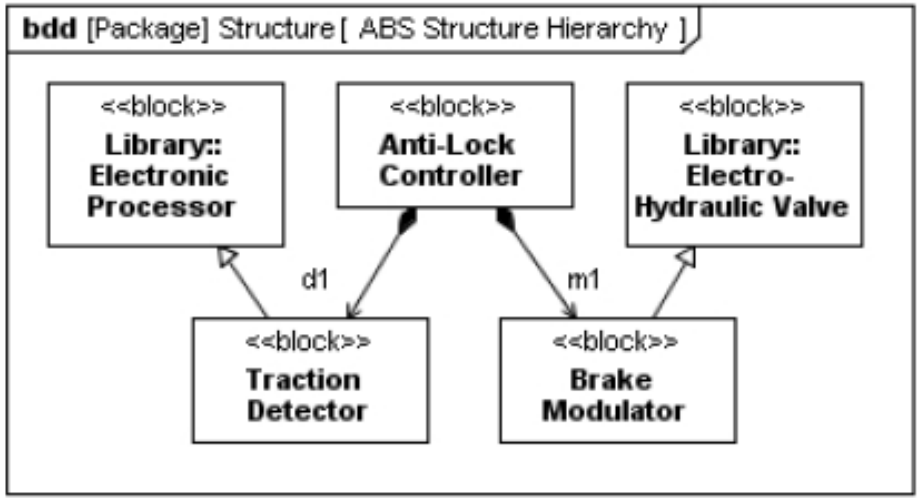
Using Blocks

- 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

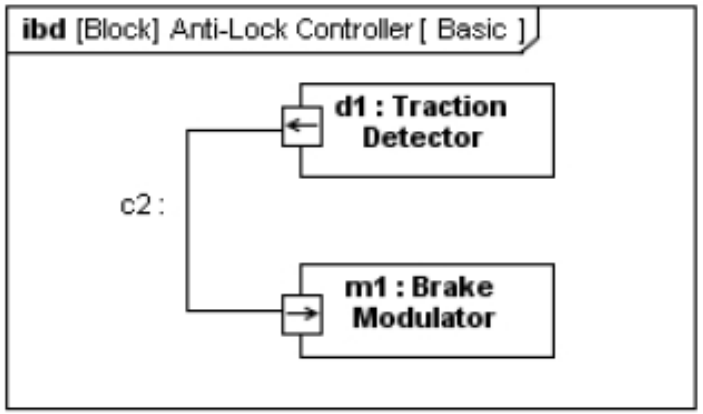
Blocks Used to Specify Hierarchies and Interconnection

Block Definition vs. Usage

Block Definition Diagram



Internal Block Diagram



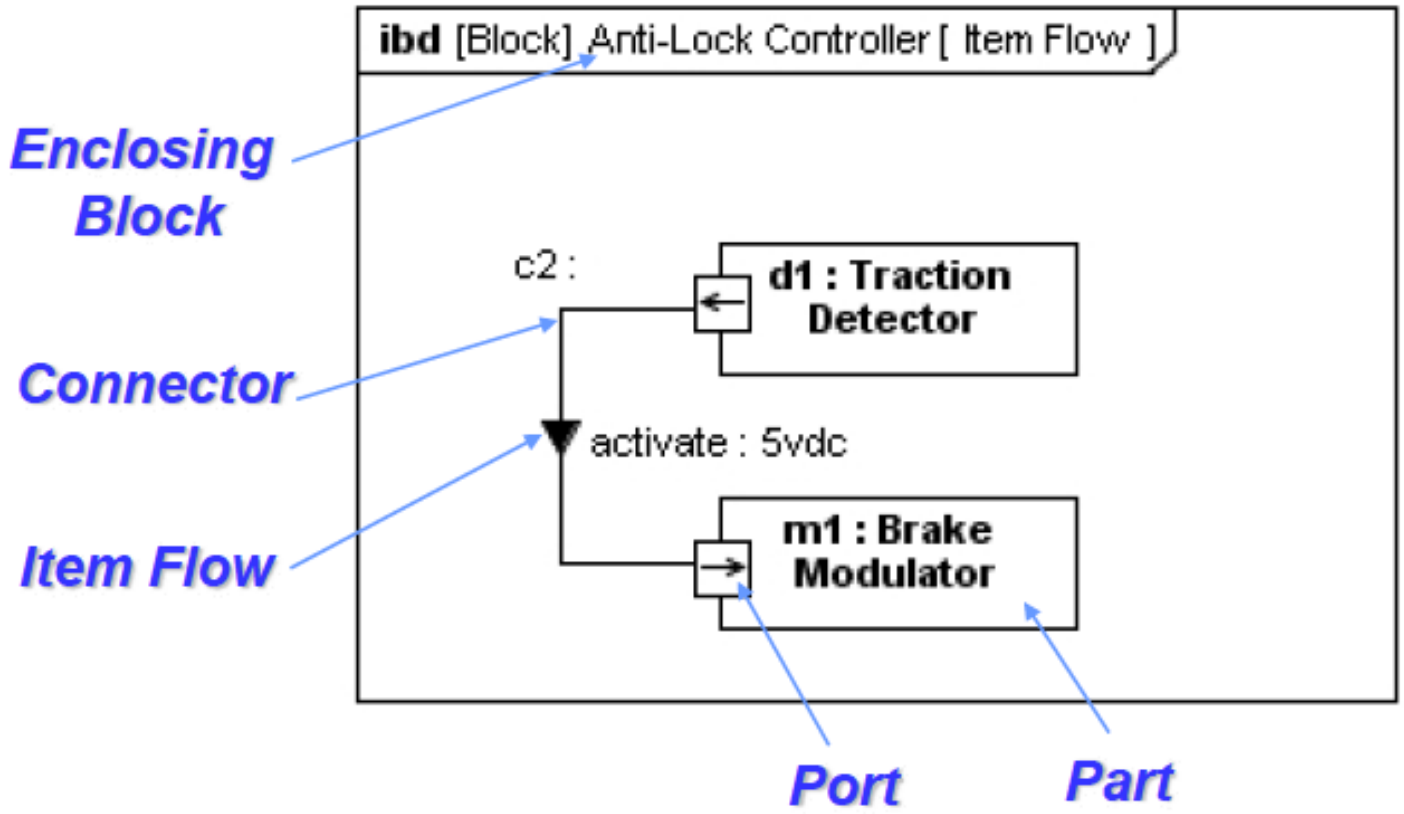
Definition

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

Usage

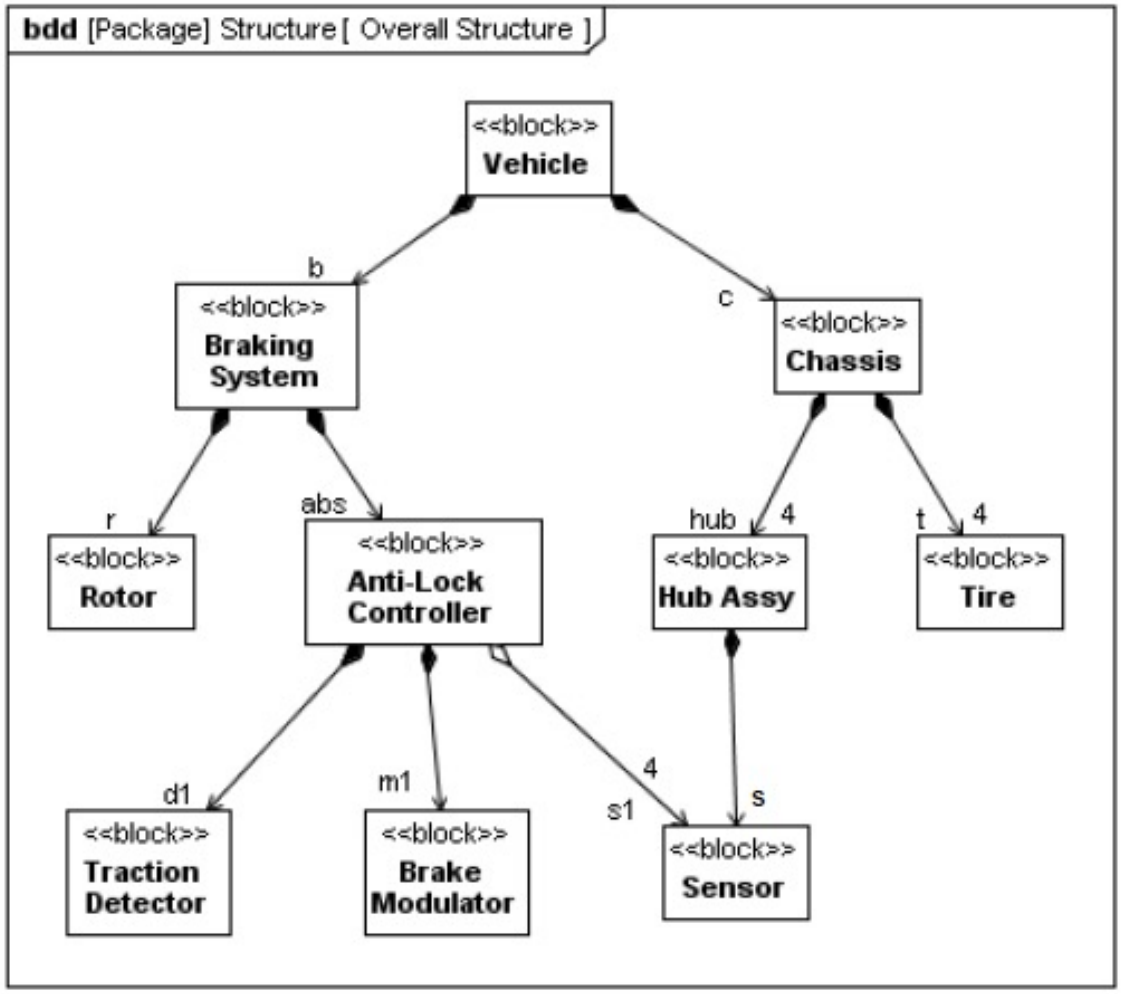
- Part is the usage of a block in the context of a composing block
- Also known as a role

Internal Block Diagram (ibd)
Blocks, Parts, Ports, Connectors & Flows

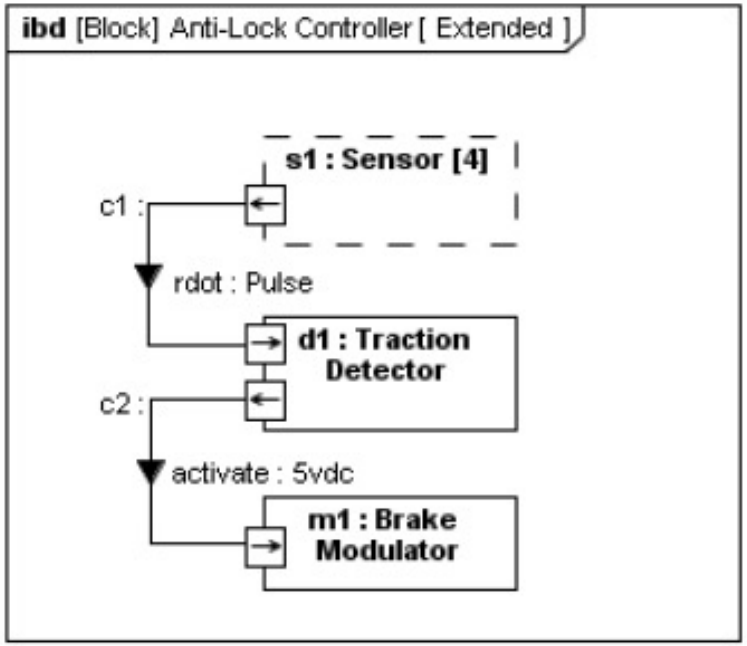


Internal Block Diagram Specifies Interconnection of Parts

Reference Property Explained



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



*Actual name is reference property

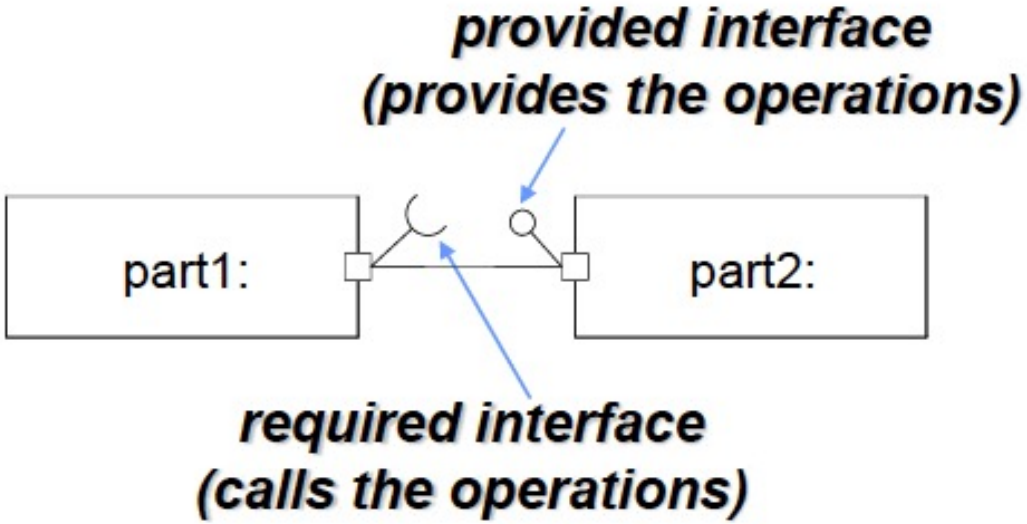
SysML Ports

- 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

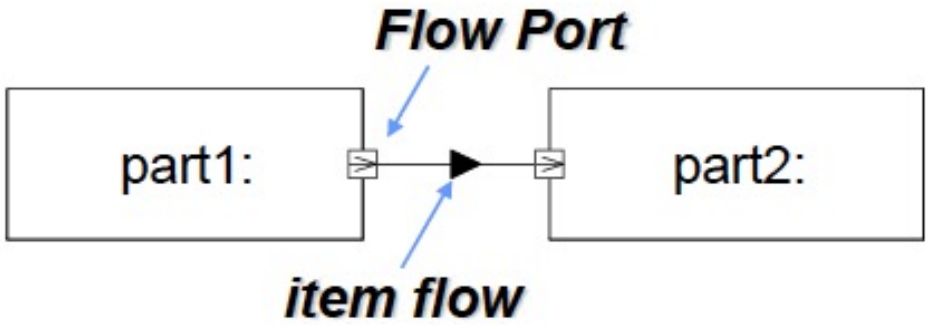
Standard Port and Flow Port Support Different Interface Concepts

Port Notation

Standard Port

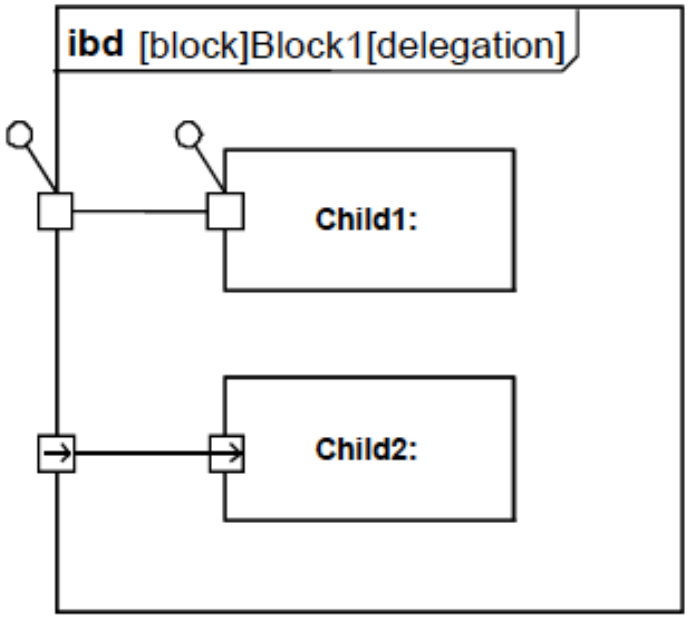


Flow Port



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 nested hierarchy

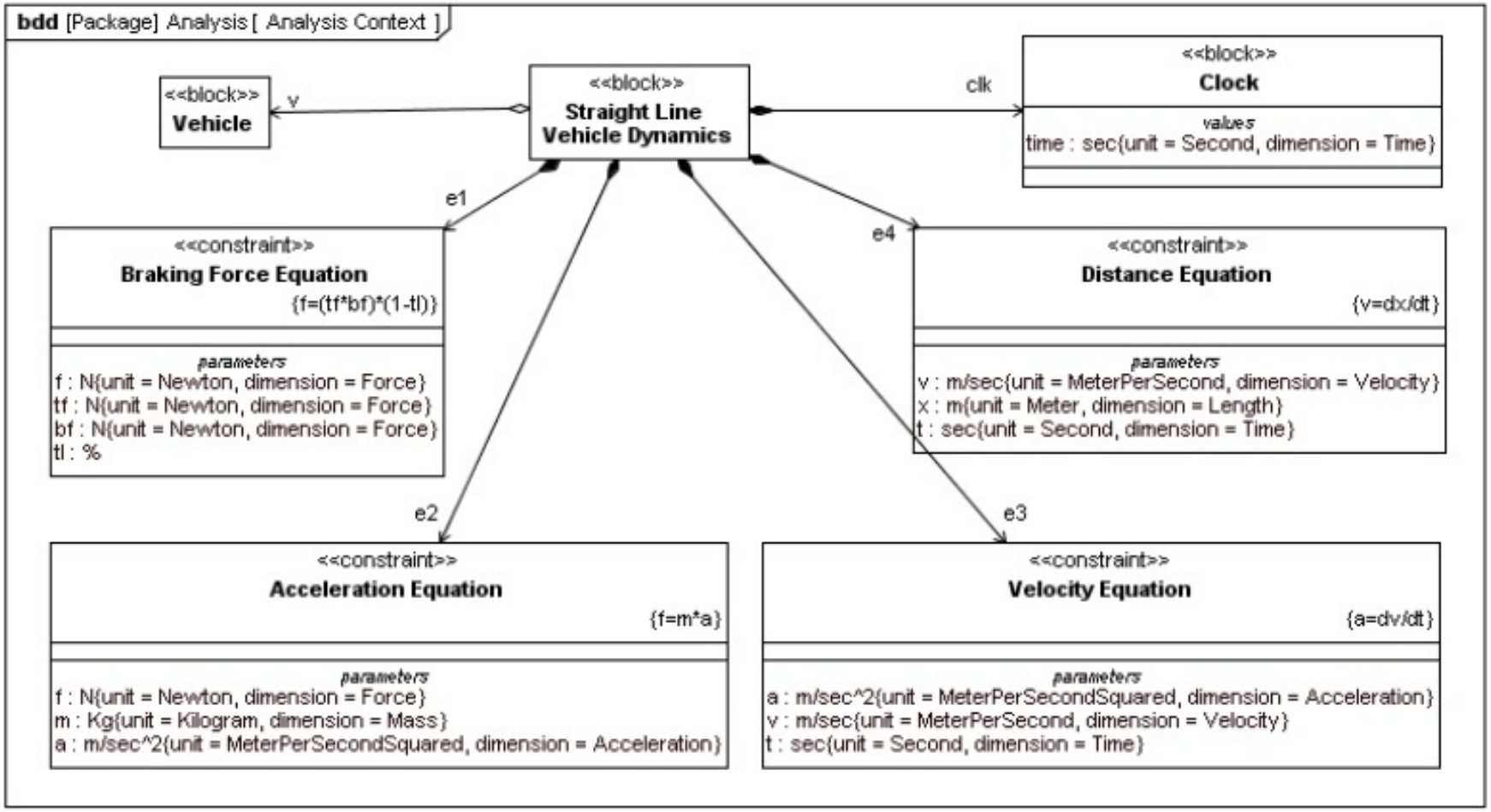


Parametrics

- 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)

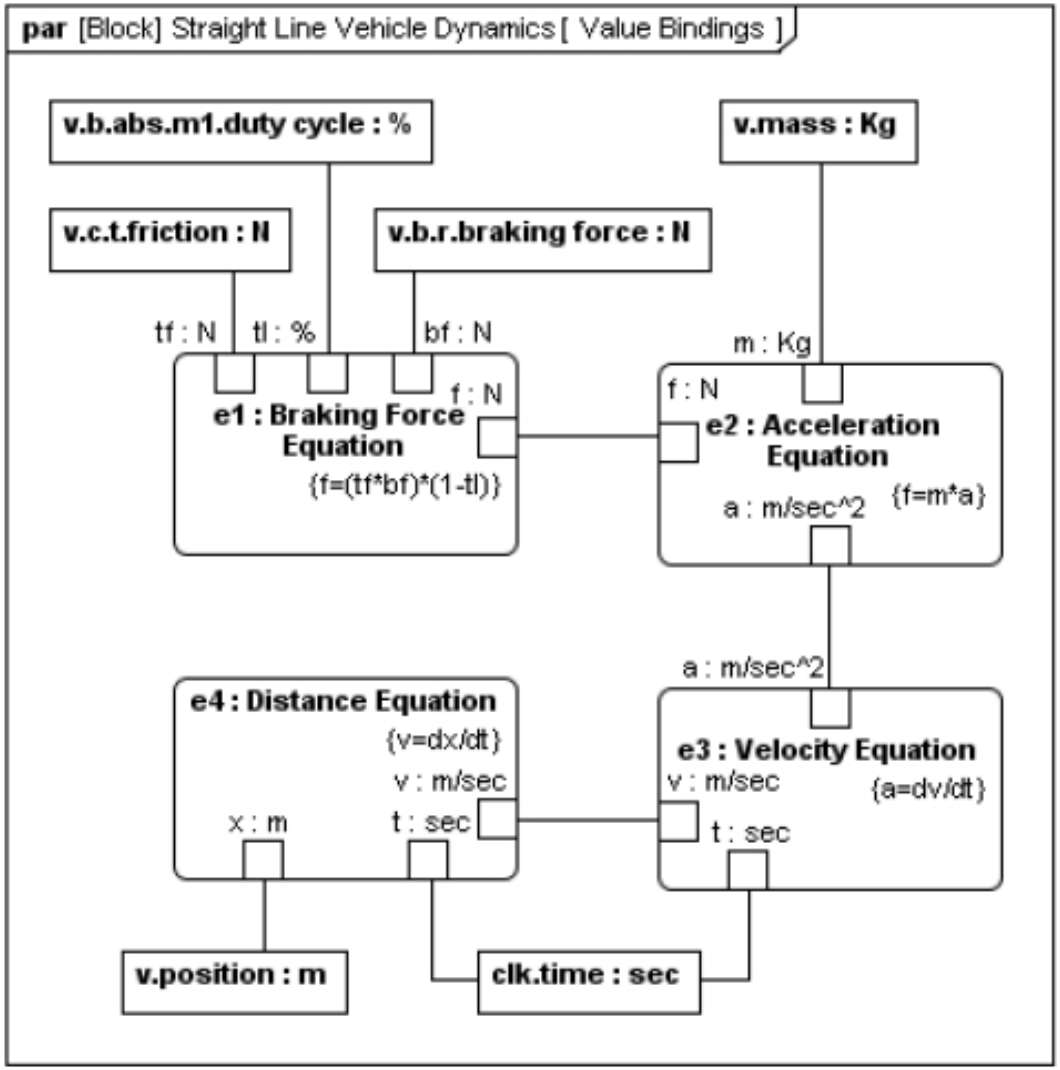
Parametrics Enable Integration of Engineering Analysis with Design Models

Defining Vehicle Dynamics



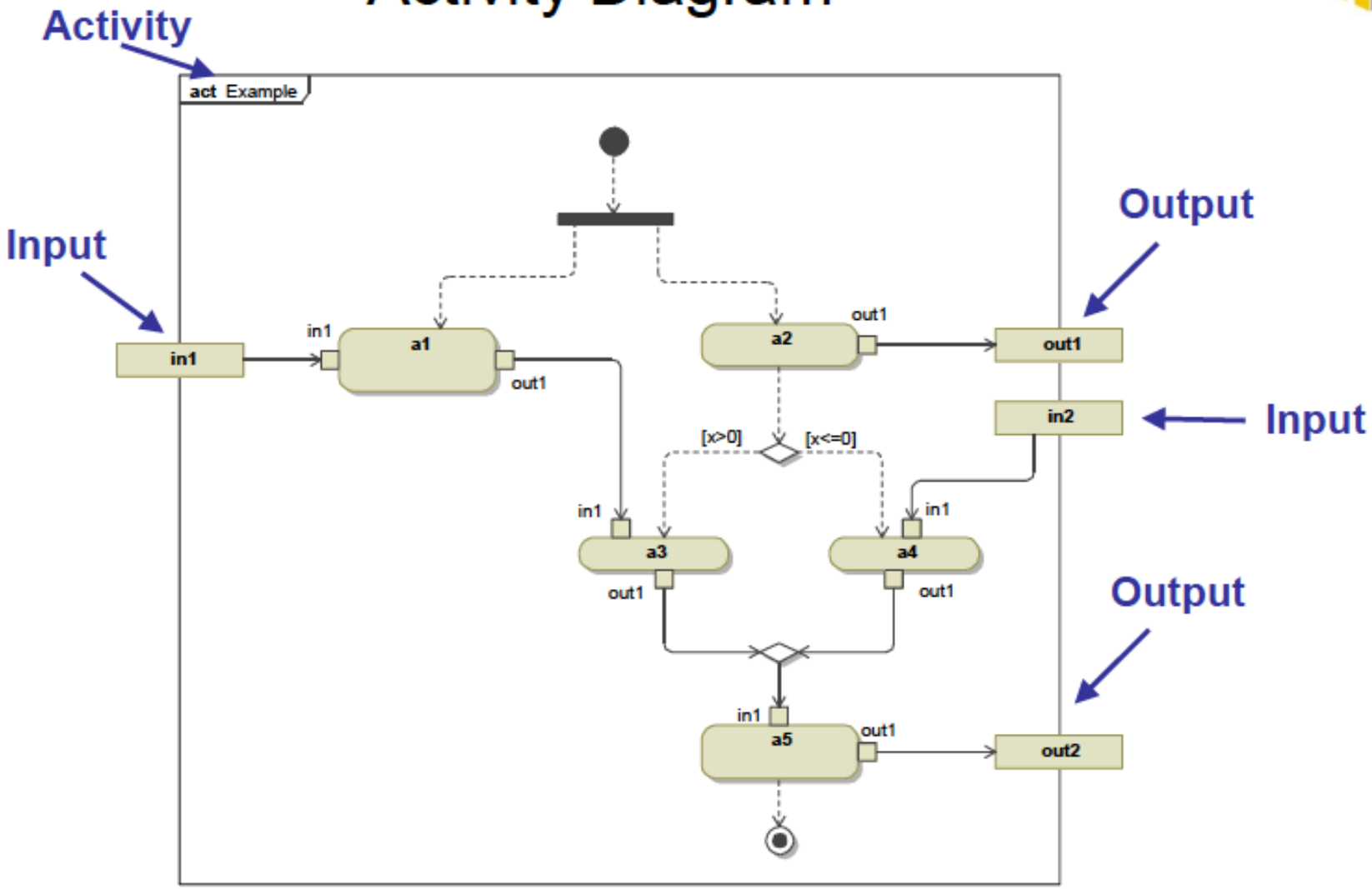
Defining Reusable Equations for Parametrics

Vehicle Dynamics Analysis



Using the Equations in a Parametric Diagram to Constrain Value Properties

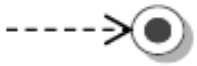
Activity Diagram



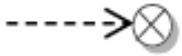
Activity Diagram Specifies Controlled Sequence of Actions

Routing Flows

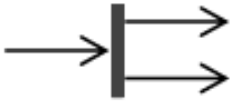
Initial Node – On execution of parent control token placed on outgoing control flows



Activity Final Node – Receipt of a control token terminates parent

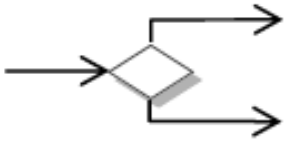


Flow Final Node – Sink for control tokens

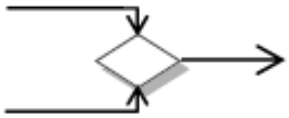


Fork Node – Duplicates input (control or object) tokens from its input flow onto all outgoing flows

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

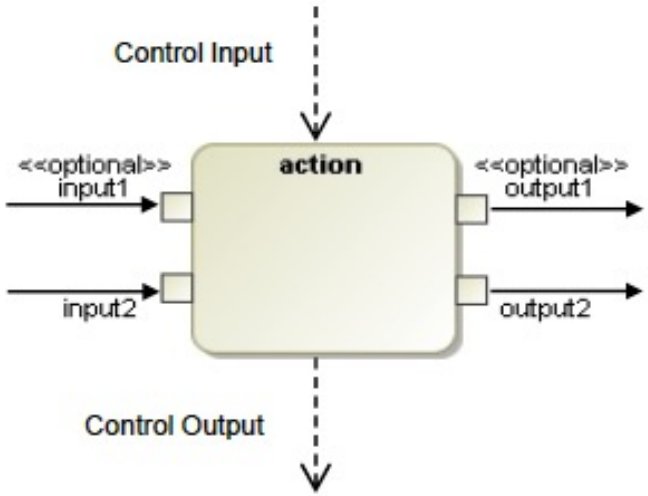


Merge Node – Waits for an input (control or object) token on any input flows and then places it on the outgoing flow

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

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

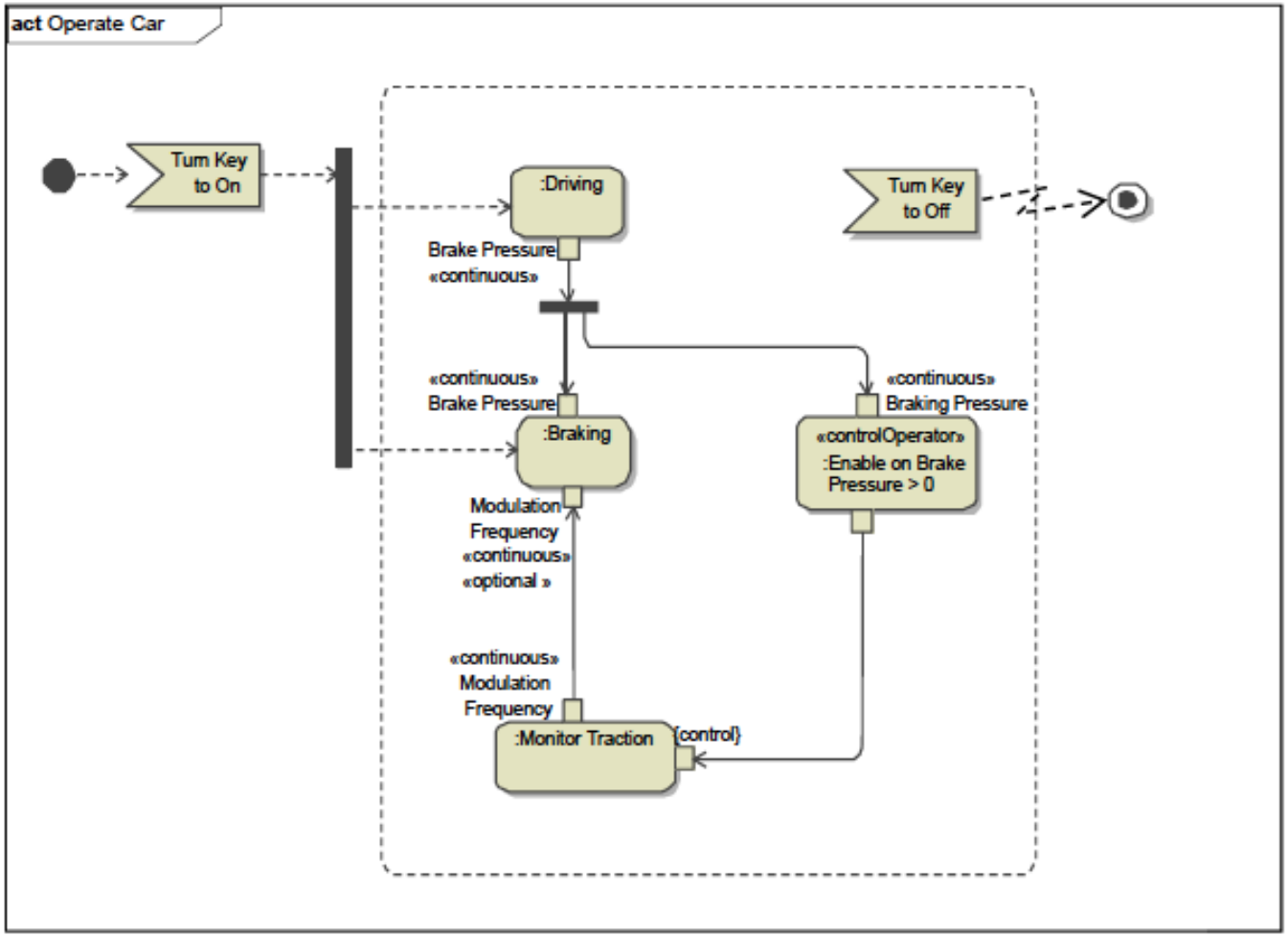
Terminating an action:

- 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

Example – Operate Car



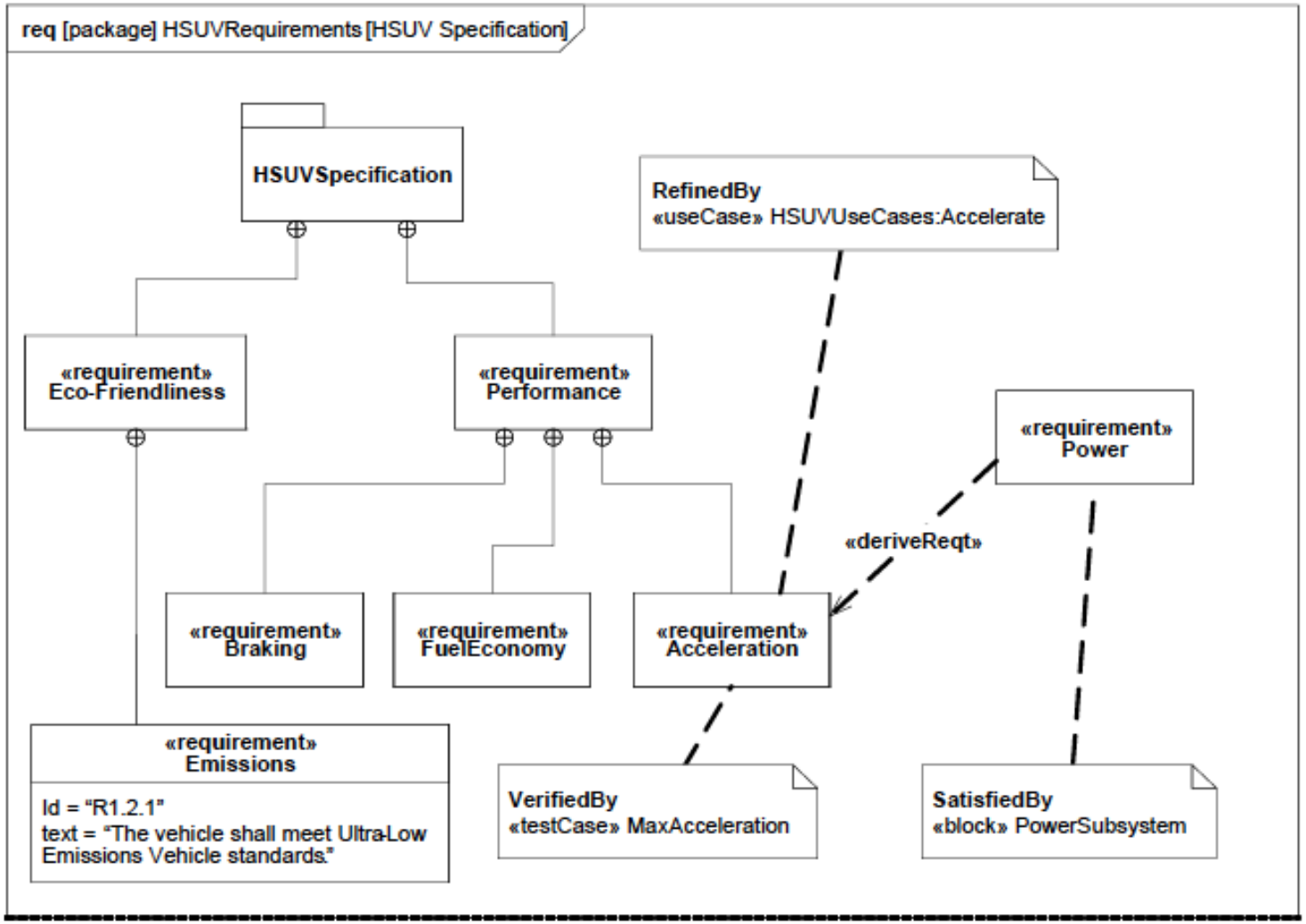
Enabling and Disabling Actions With Control Operators

SysML Requirements

Requirements

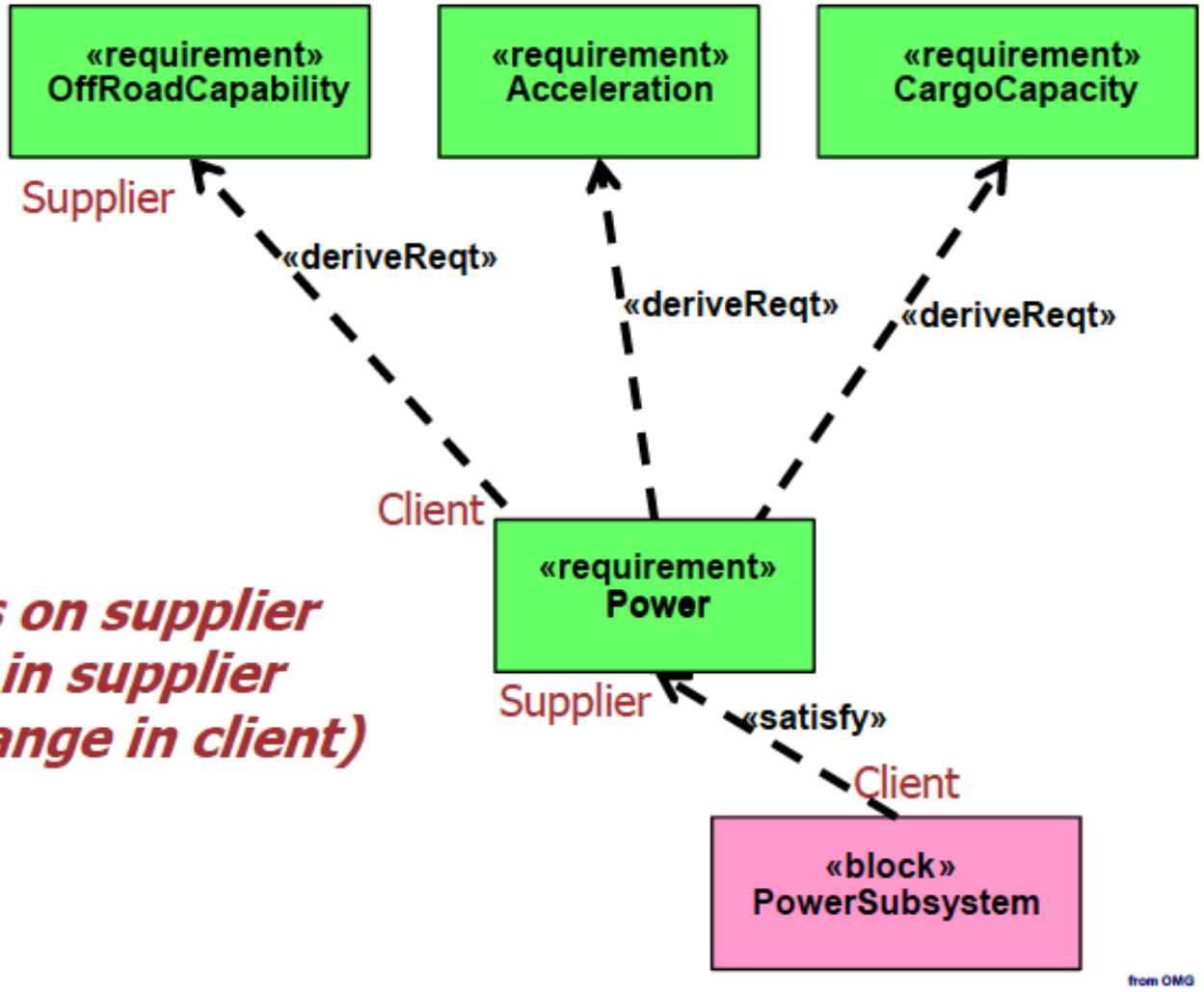
- The «requirement» stereotype represents a text based requirement
  - Includes id and text properties
  - Can add user defined properties such as verification method
  - Can add user defined requirements categories (e.g., functional, interface, performance)
- Requirements hierarchy describes requirements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

Requirements Breakdown



Requirement Relationships Model the Content of a Specification

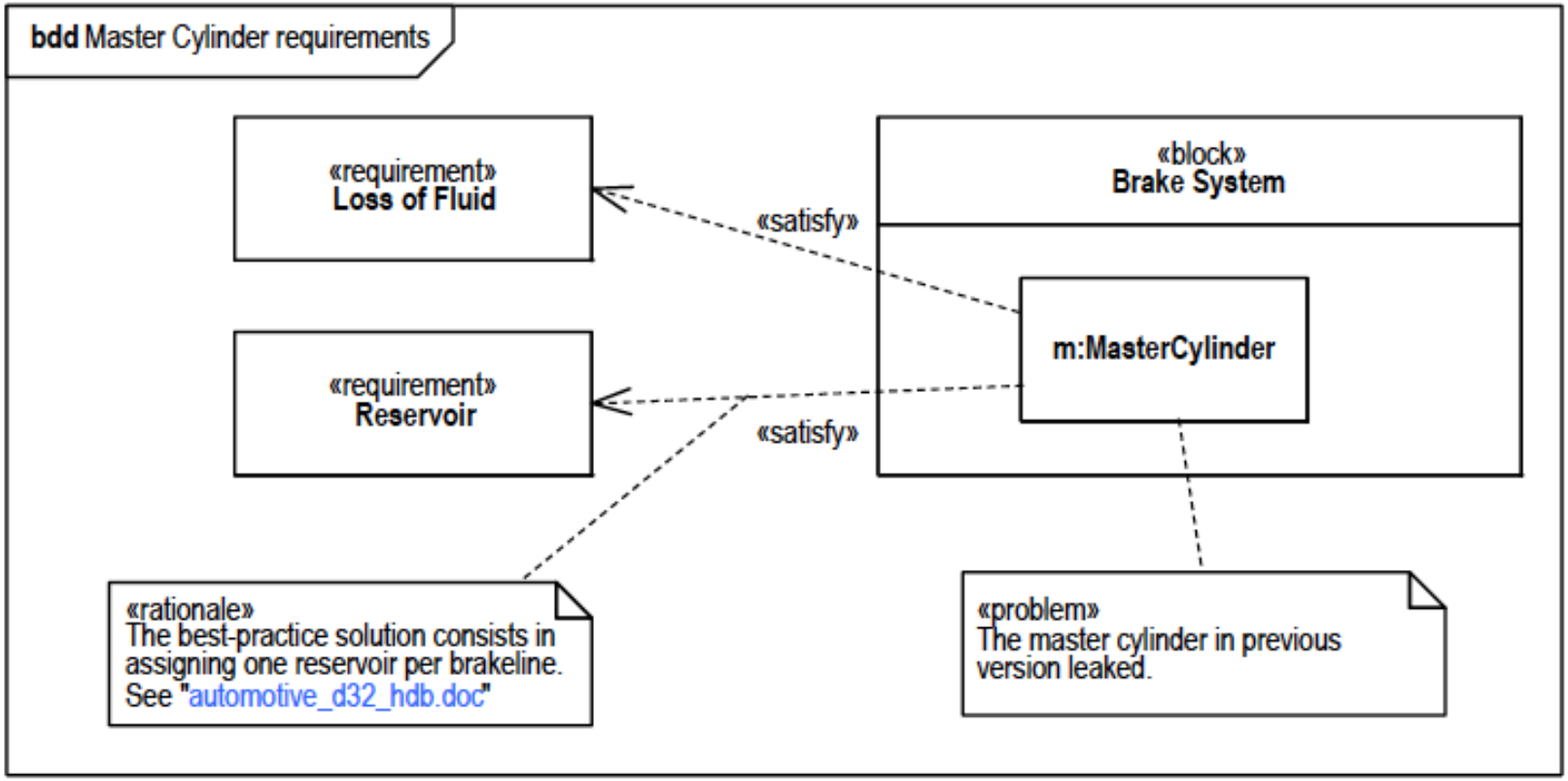
Example of Derive/Satisfy Requirement Dependencies



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

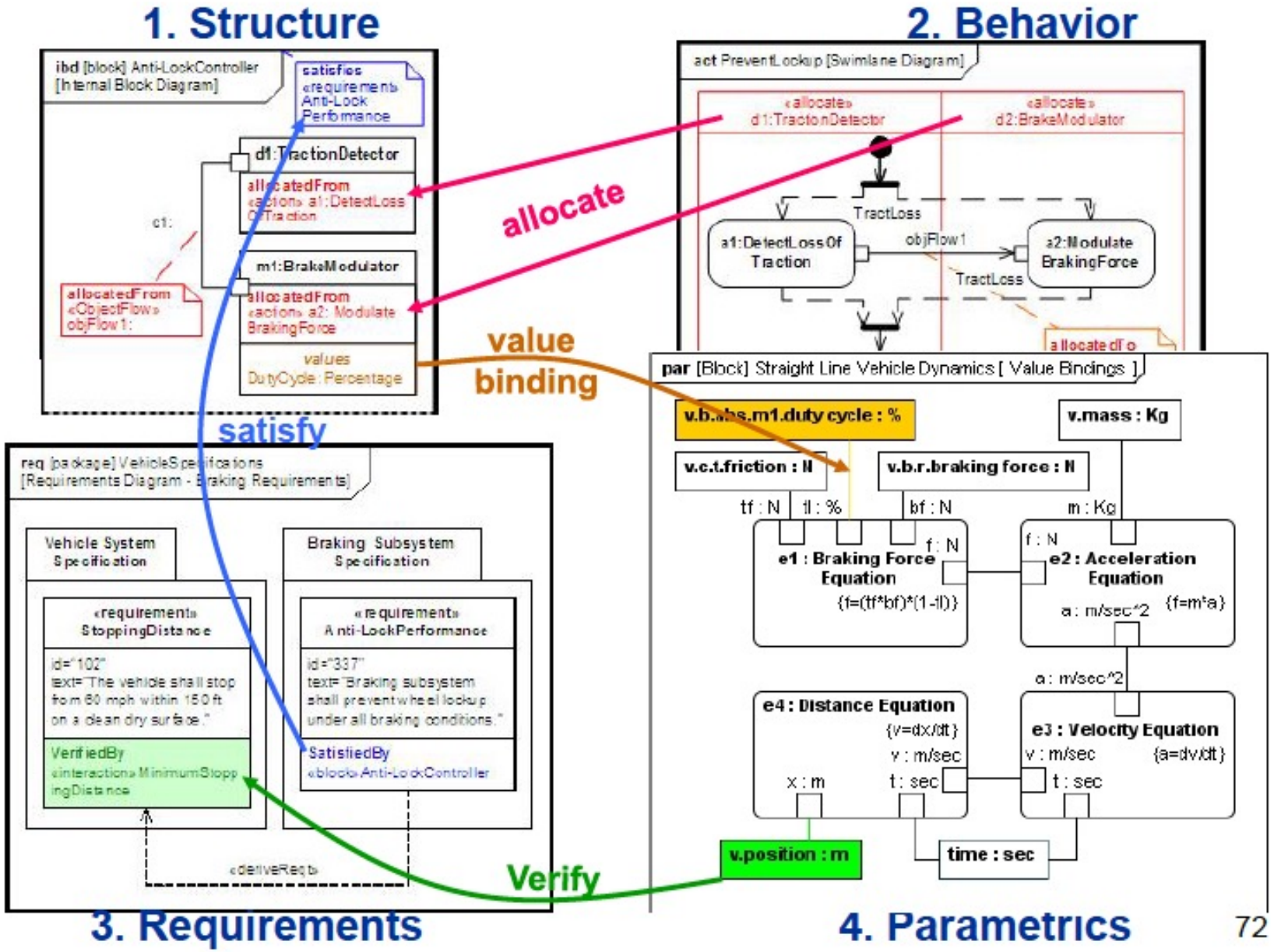
Arrow Direction Opposite Typical Requirements Flow-Down

Problem and Rationale



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

Cross Connecting Model Elements

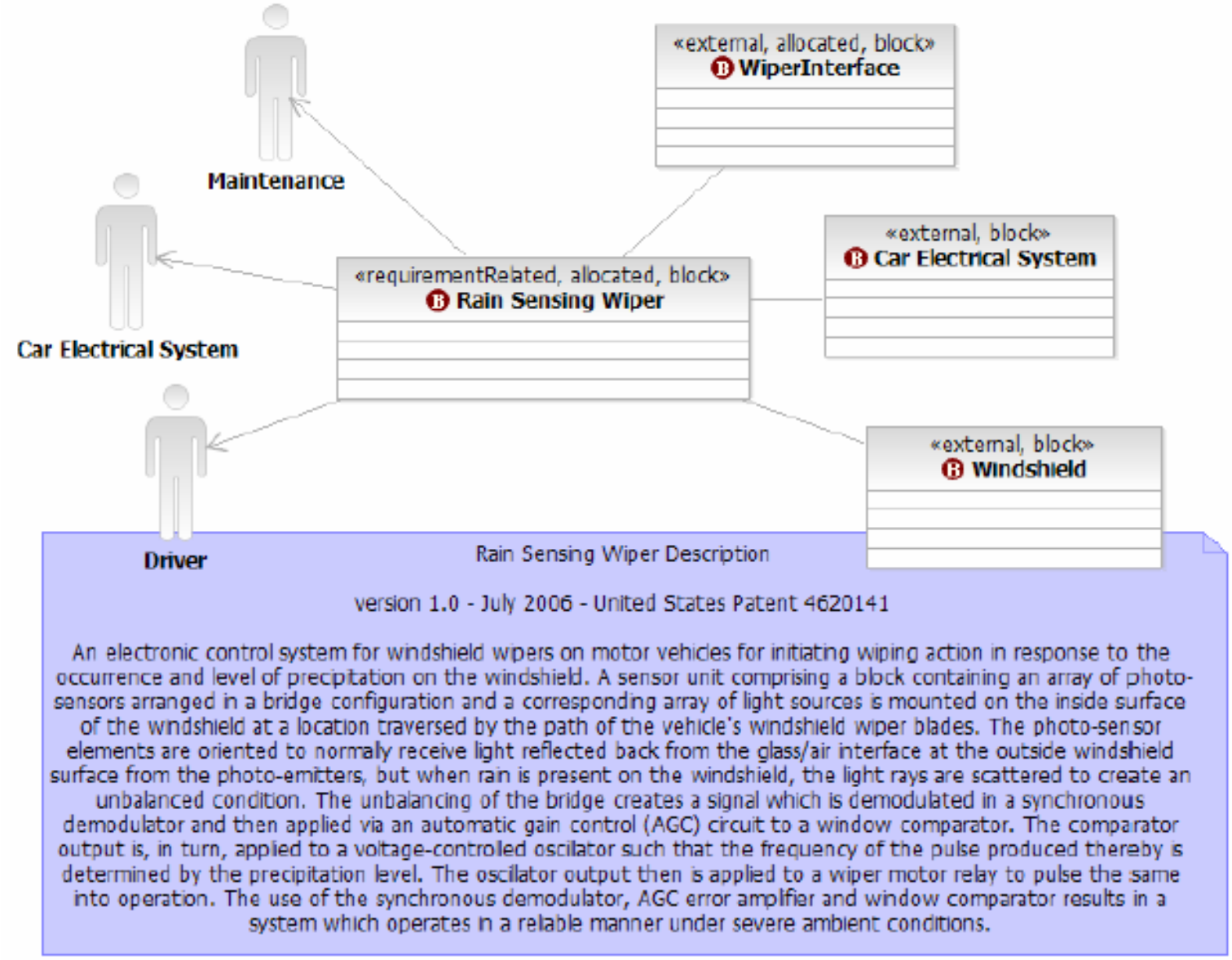


SysML – An Example

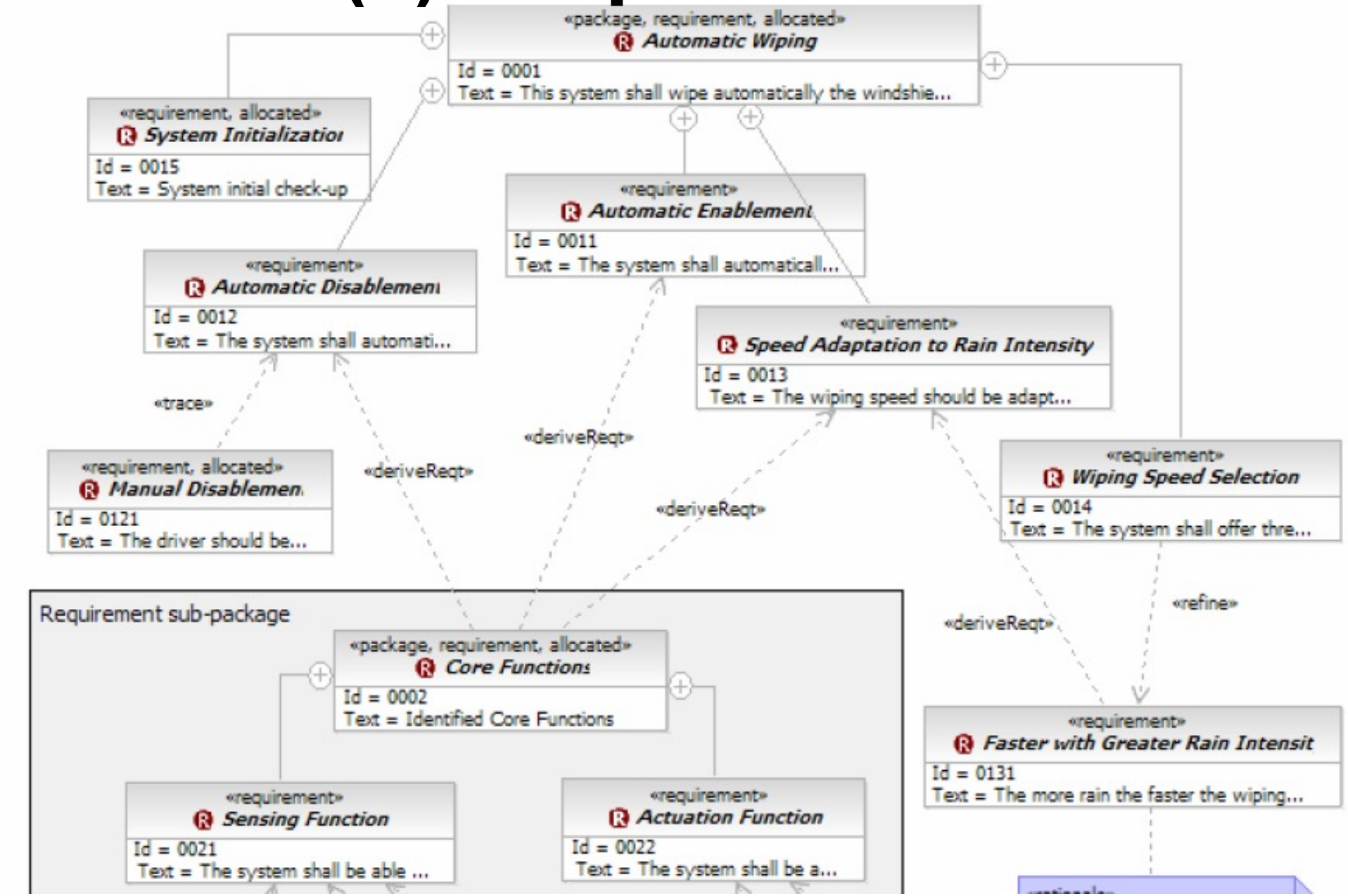
From *Laurent Balmelli, An Overview of the Systems Modeling Language for Products and Systems Development, Journal of Object Technology, Vol. 6, No. 6, July-August, 2007*

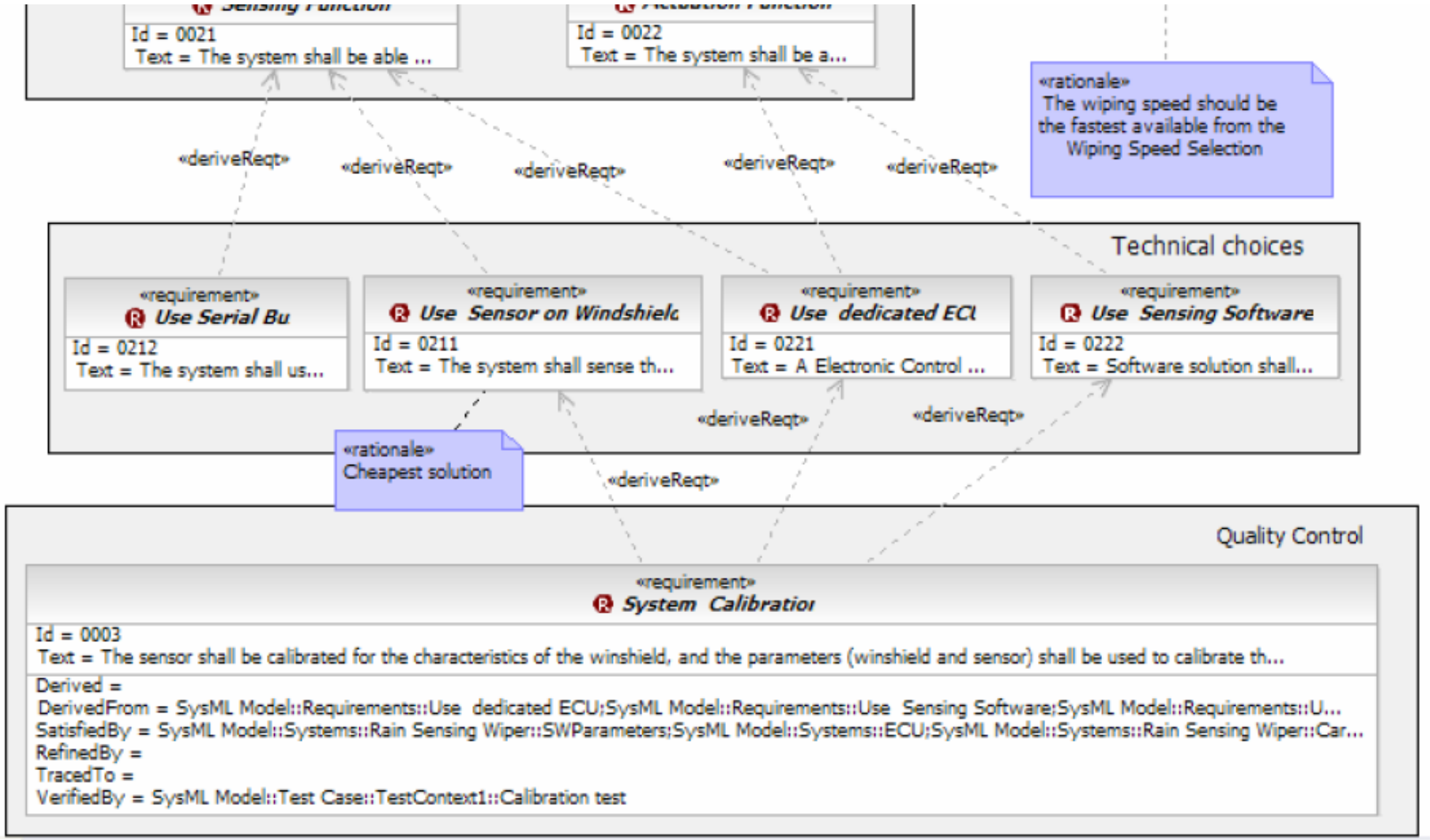
(Focusing on example diagrams)

Rain Sensor Wiper System

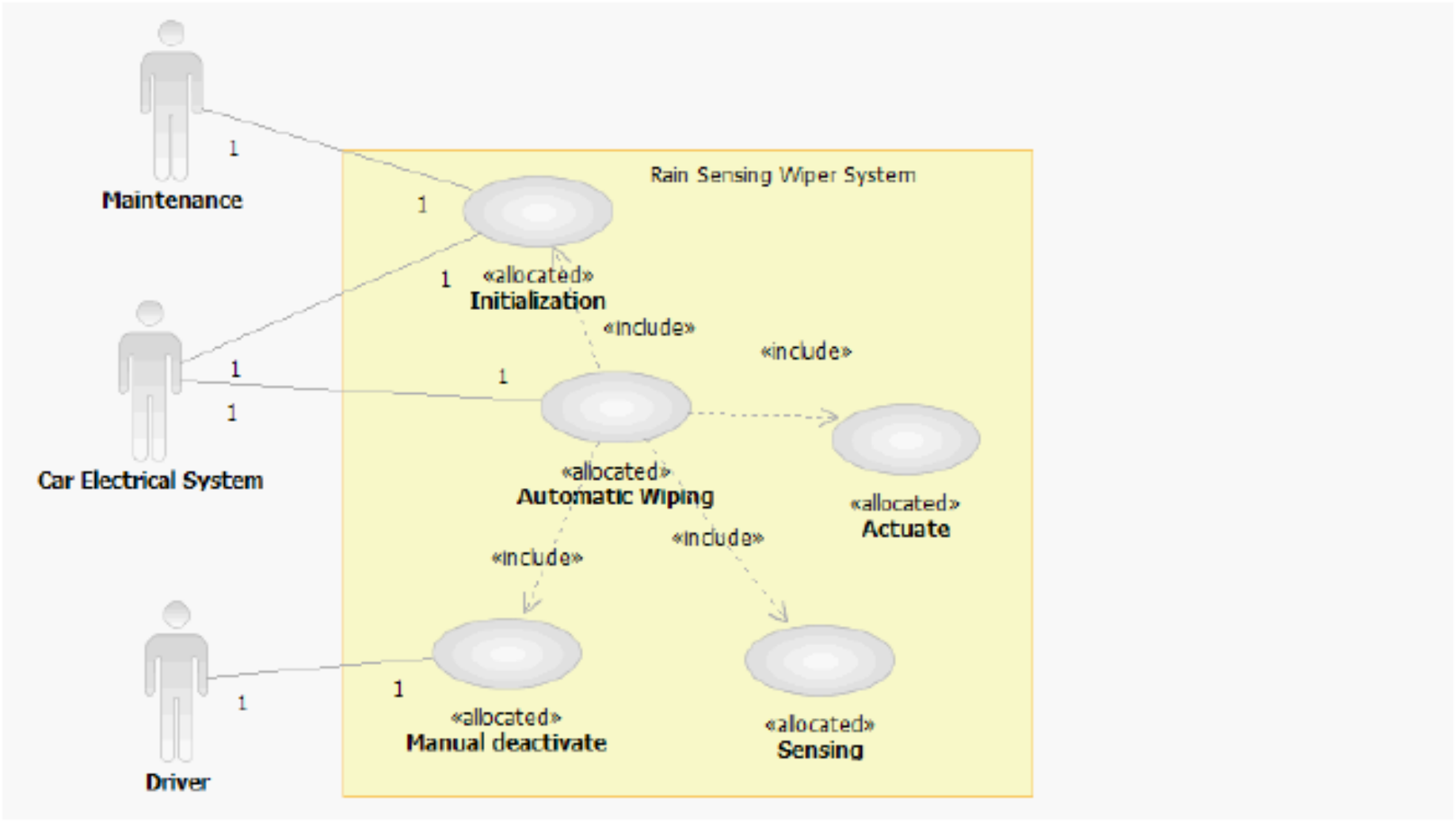


(1) Requirements

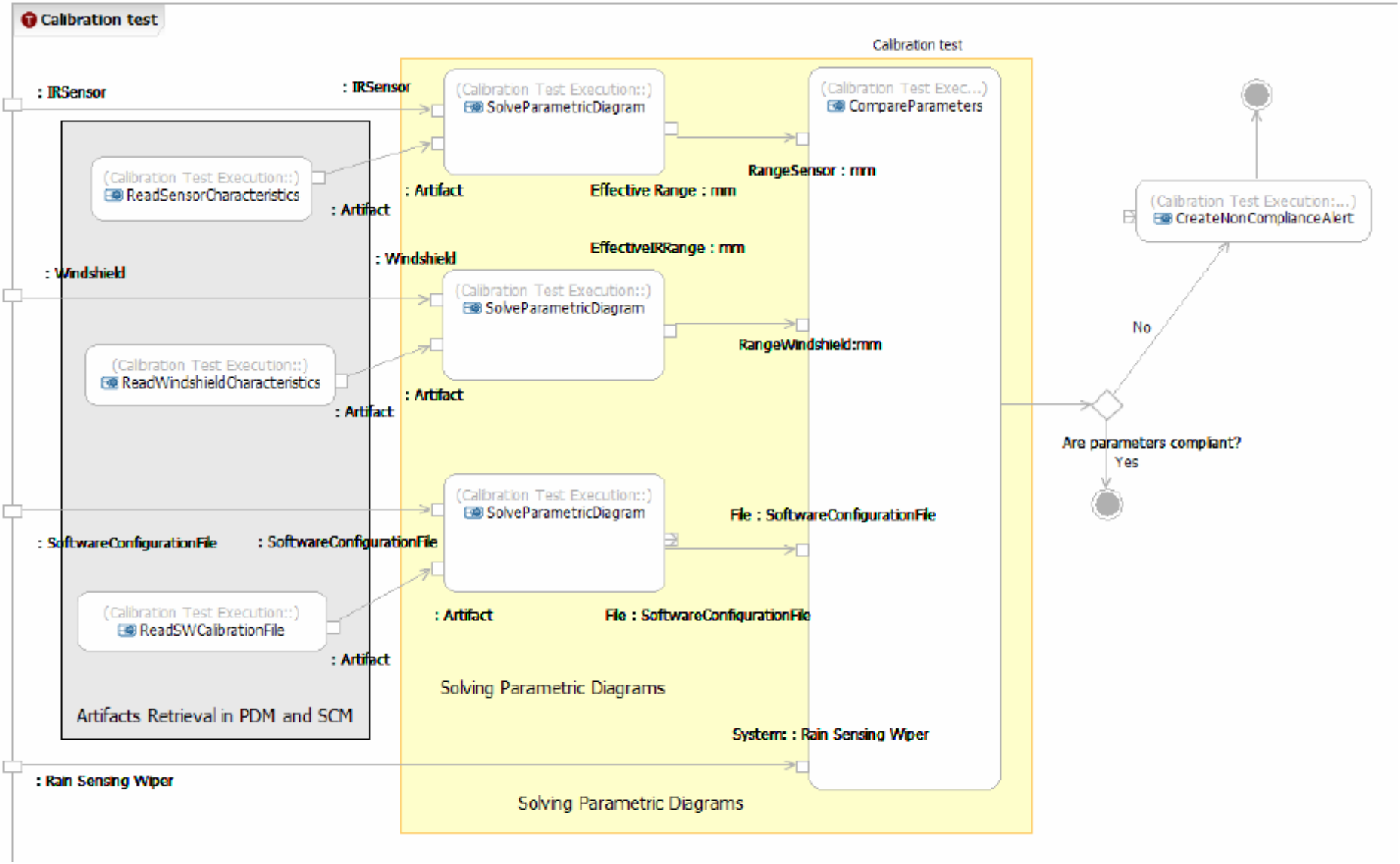


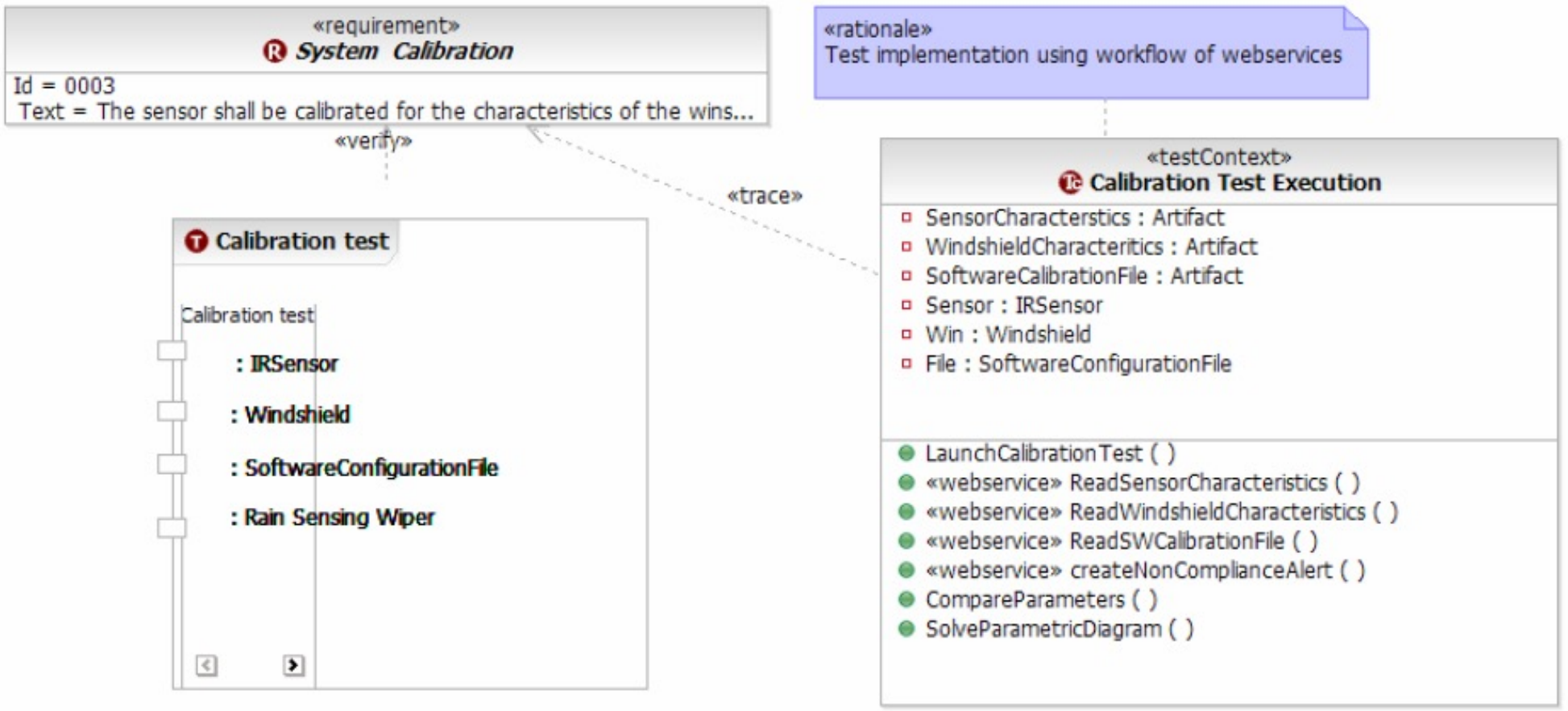

Id = 0021
Text = The system shall be able ...
Id = 0022
Text = The system shall be a...
«rationale»
The wiping speed should be the fastest available from the Wiping Speed Selection
«deriveReqt»
«deriveReqt»
«deriveReqt»
«deriveReqt»
«deriveReqt»
Technical choices
«requirement»
Use Serial Bu
Id = 0212
Text = The system shall us...
«requirement»
Use Sensor on Windshield
Id = 0211
Text = The system shall sense th...
«requirement»
Use dedicated ECU
Id = 0221
Text = A Electronic Control ...
«requirement»
Use Sensing Software
Id = 0222
Text = Software solution shall...
«deriveReqt»
«deriveReqt»
«rationale»
Cheapest solution
«deriveReqt»
Quality Control
«requirement»
System Calibration
Id = 0003
Text = The sensor shall be calibrated for the characteristics of the winshield, and the parameters (winshield and sensor) shall be used to calibrate th...
Derived =
DerivedFrom = SysML Model::Requirements::Use dedicated ECU;SysML Model::Requirements::Use Sensing Software;SysML Model::Requirements::U...
SatisfiedBy = SysML Model::Systems::Rain Sensing Wiper::SWParameters;SysML Model::Systems::ECU;SysML Model::Systems::Rain Sensing Wiper::Car...
RefinedBy =
TracedTo =
VerifiedBy = SysML Model::Test Case::TestContext1::Calibration test

(2) Use Cases


Maintenance
Car Electrical System
Driver
Rain Sensing Wiper System
«allocated»
Initialization
«include»
«include»
«allocated»
Automatic Wiping
«allocated»
Actuate
«include»
«include»
«allocated»
Manual deactivate
«allocated»
Sensing
1

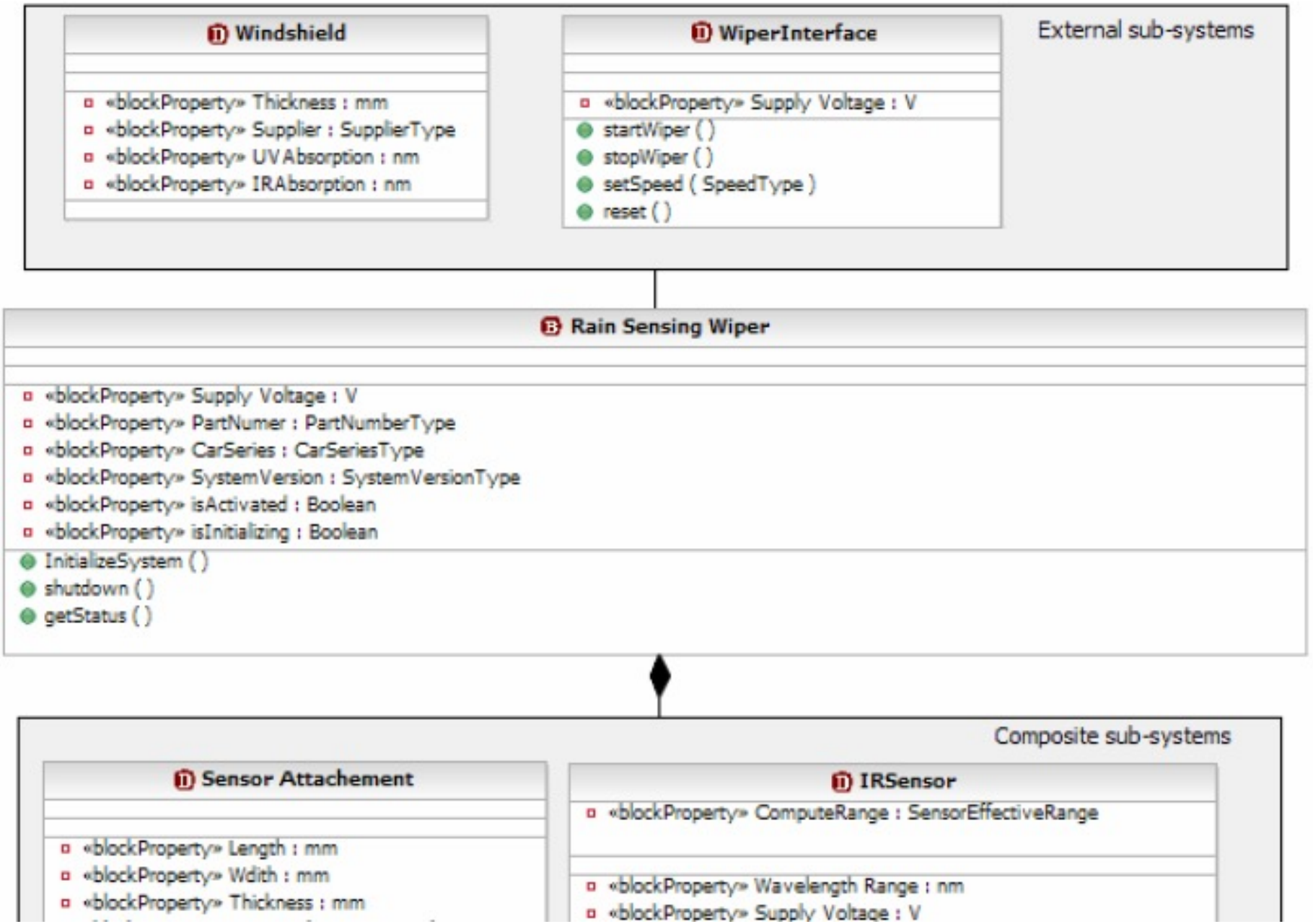
(3) Activity diagrams

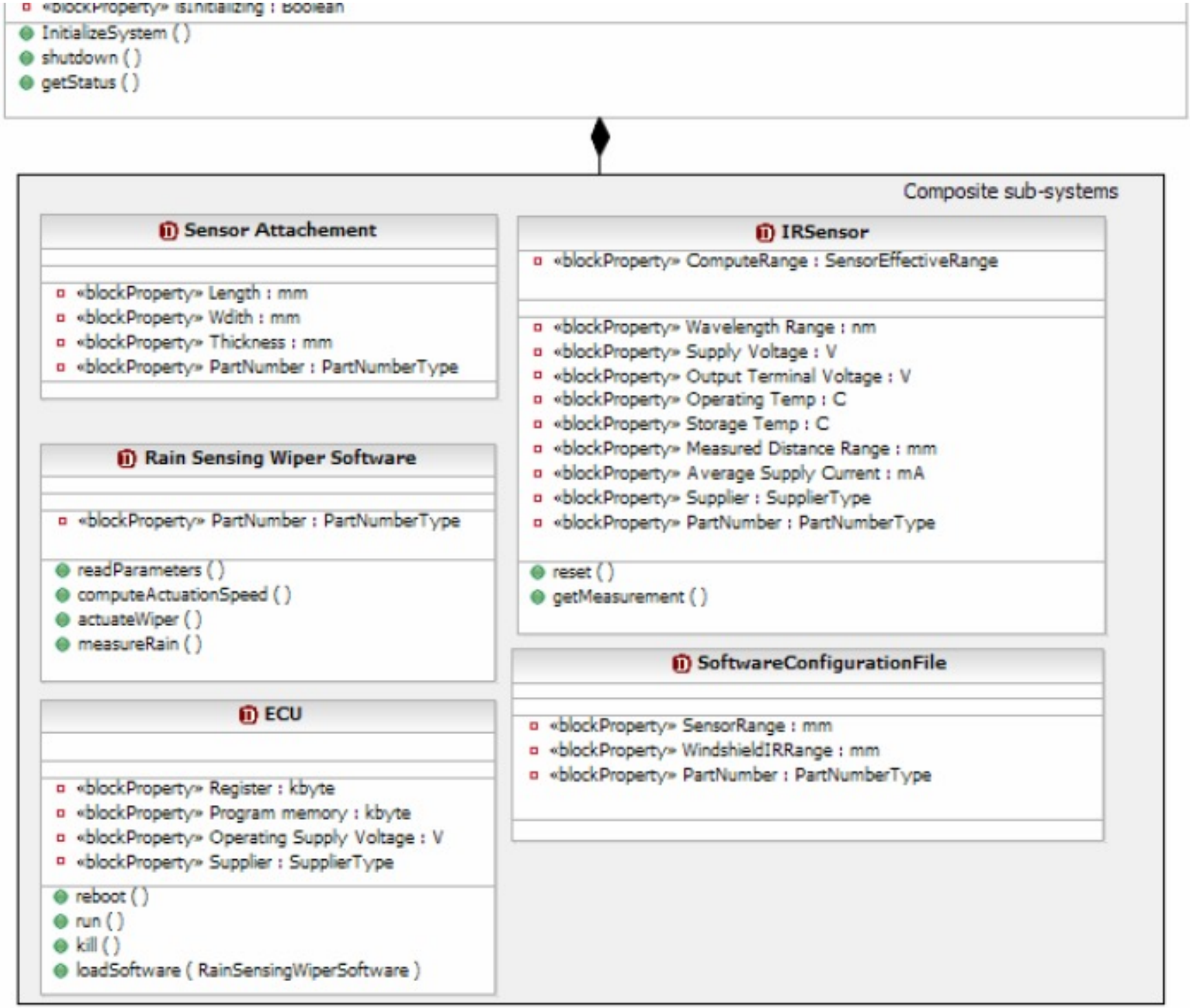




(b) Requirement and test case traceability

Block diagram




«blockProperty» isInitializing : Boolean
InitializeSystem ( )
shutdown ( )
getStatus ( )
Composite sub-systems
Sensor Attachement
«blockProperty» Length : mm
«blockProperty» Width : mm
«blockProperty» Thickness : mm
«blockProperty» PartNumber : PartNumberType
Rain Sensing Wiper Software
«blockProperty» PartNumber : PartNumberType
readParameters ( )
computeActuationSpeed ( )
actuateWiper ( )
measureRain ( )
ECU
«blockProperty» Register : kbyte
«blockProperty» Program memory : kbyte
«blockProperty» Operating Supply Voltage : V
«blockProperty» Supplier : SupplierType
reboot ( )
run ( )
kill ( )
loadSoftware (RainSensingWiperSoftware)
IRSensor
«blockProperty» ComputeRange : SensorEffectiveRange
«blockProperty» Wavelength Range : nm
«blockProperty» Supply Voltage : V
«blockProperty» Output Terminal Voltage : V
«blockProperty» Operating Temp : C
«blockProperty» Storage Temp : C
«blockProperty» Measured Distance Range : mm
«blockProperty» Average Supply Current : mA
«blockProperty» Supplier : SupplierType
«blockProperty» PartNumber : PartNumberType
reset ( )
getMeasurement ( )
SoftwareConfigurationFile
«blockProperty» SensorRange : mm
«blockProperty» WindshieldIRRange : mm
«blockProperty» PartNumber : PartNumberType

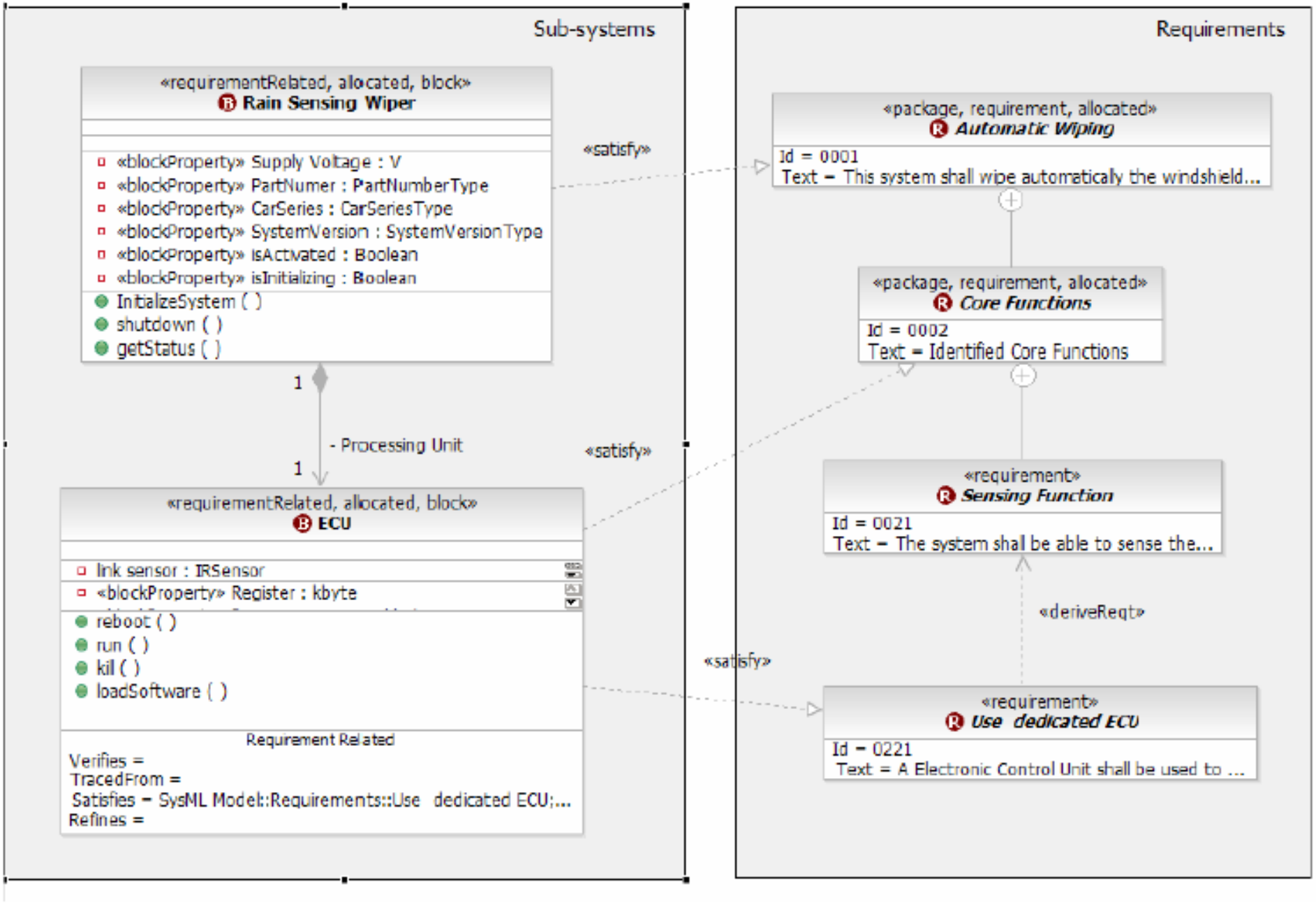


Figure 4 Example of requirement allocation.

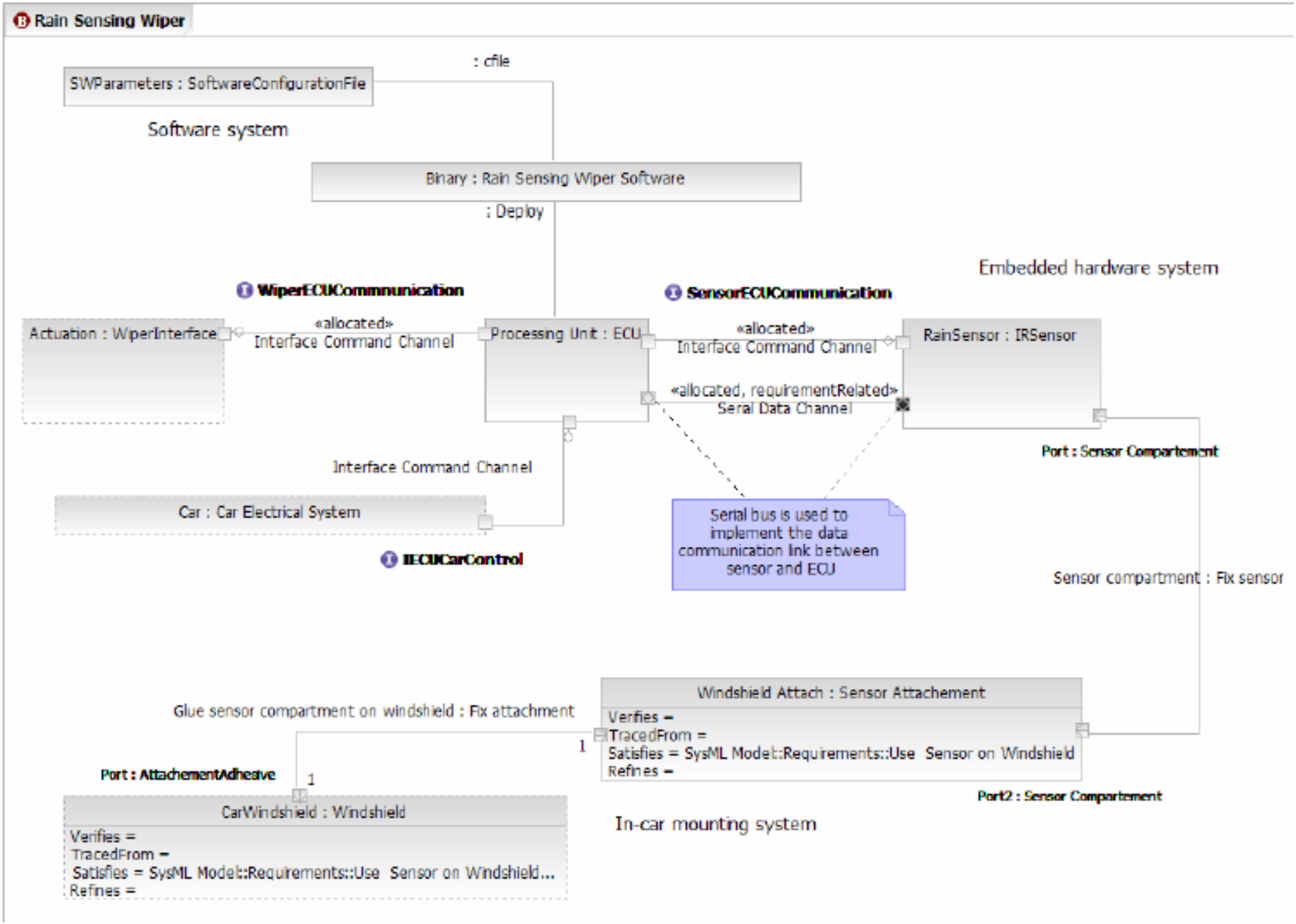


Figure 5 Internal structure of the Rain Sensing Wiper system.

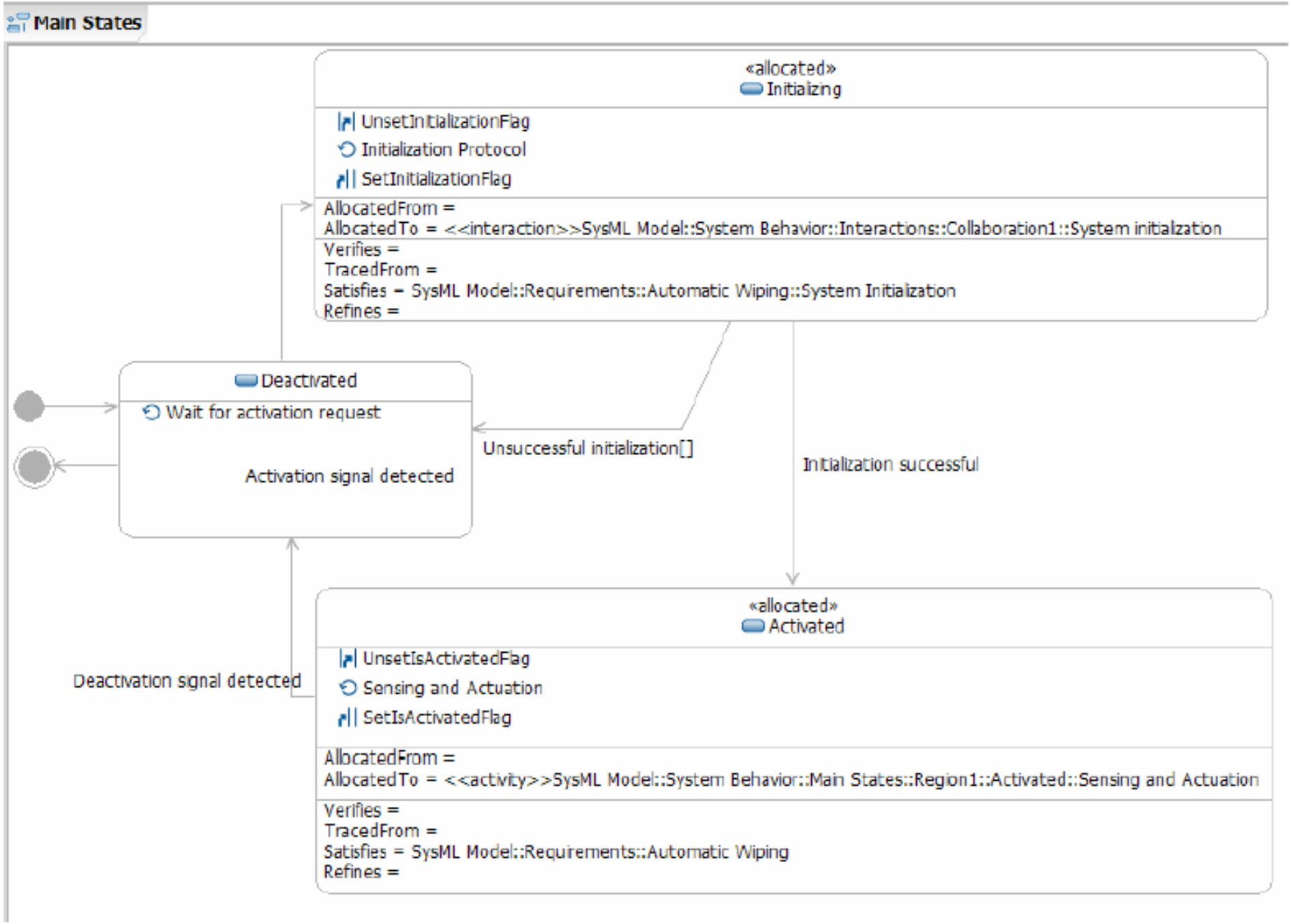


Figure 10 State Machine Diagram for the Rain Sensing Wiper system.

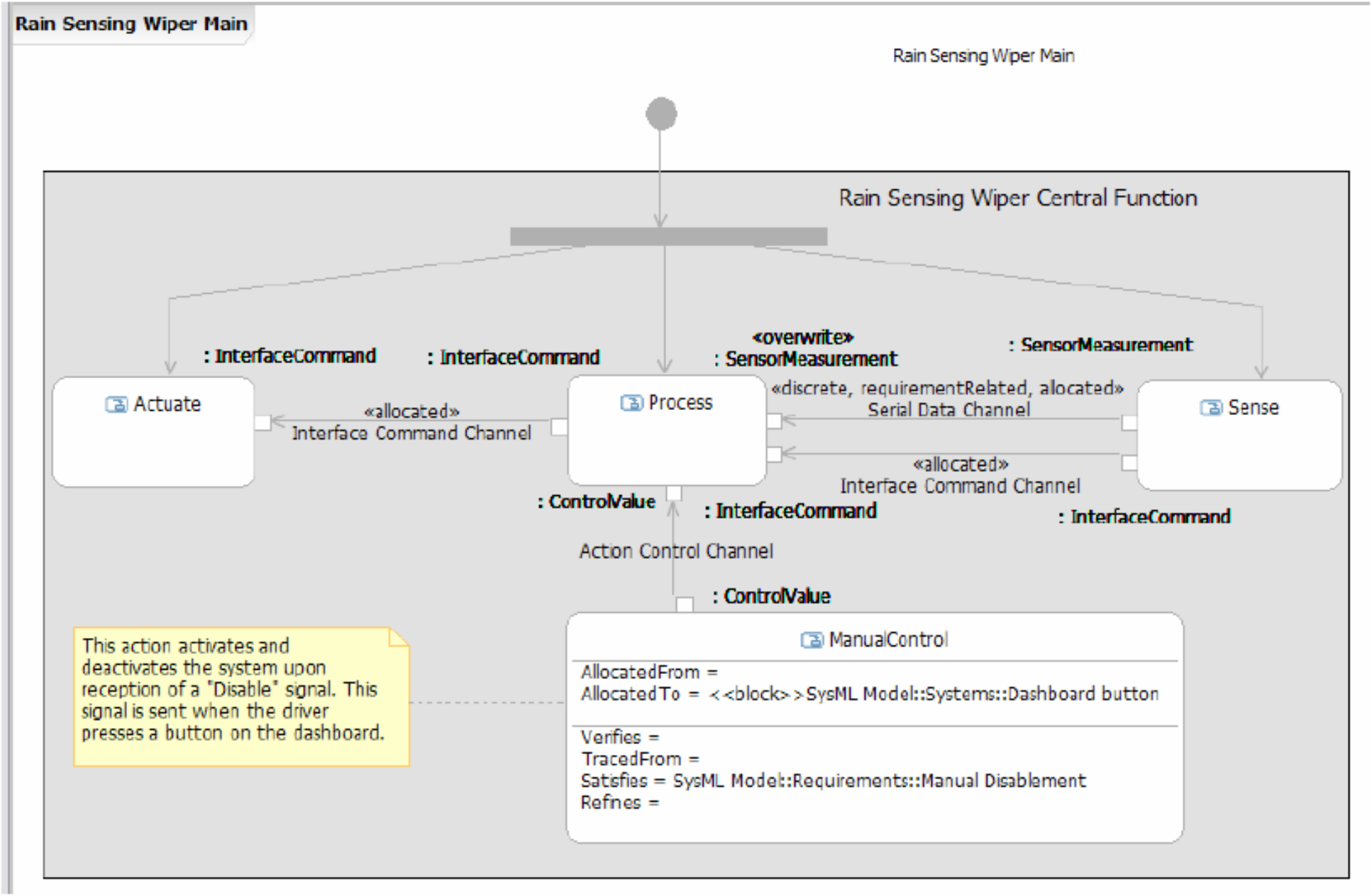


Figure 12 Sensing activity for the Rain Sensing Wiper system.

Conclusion

- SysML is based on UML 2.0
  - Changes AND
  - New diagrams
- SysML is well adapted for system modeling