SysML
The modeling Language for System Engineering

Philippe Desfray - SOFTEAM

Agenda

System engineering

- ***Systems Engineering*** **(SE) is an interdisciplinary field of engineering, that focuses on the development and organization of complex artificial systems. It integrates other disciplines and specialty groups into a team effort, forming a structured development process that proceeds from concept to production to operation and disposal. Systems Engineering considers both the business and the technical needs of all customers, with the goal of providing a quality product that meets the user needs.**
- **A *System* is a set of entities, real or abstract, comprising a whole where each component interacts with or is related to at least one other component and they all serve a common objective. Any object which has no relation with any other element of the system is not part of that system but rather of the system environment. A subsystem then is a set of elements, which is a system itself, and a part of the whole system.**
- **Examples of systems :**
  - Air traffic Management systems
  - Medical Systems
  - Plane
  - Military systems (Naval, ground, Air force, …)

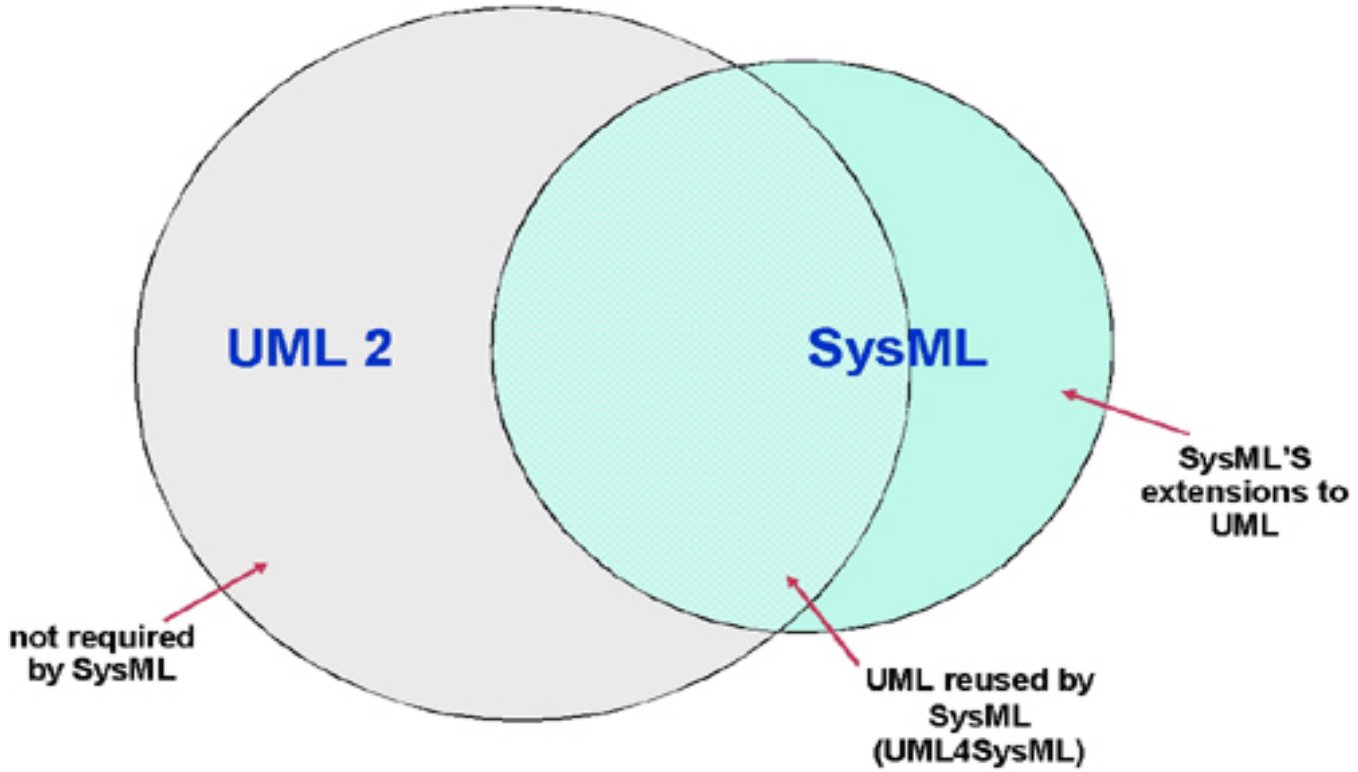
- **Complex**
- **Non software centric**
- **Made of parts constituting a hole**
- **Interdisciplinary, involving mechanics, electricity, software, Human interaction, energy, software radio, hardware, etc.**
- **Large, with a large number of participant in its construction :**
  - Subcontracting issues
  - Integration issues
  - Global consistency issues

- **SysML is a modeling language used for systems engineering applications, supports the specification, analysis, design, verification and validation of a broad range of complex systems.**
- **SysML is an OMG standard, based on UML, adapting UML for:**
  - Not being software centric,
  - Focusing and extending it to the system engineering field
- **RFP (March 2003) developed jointly by the OMG and the INCOSE: Requirements for extending UML to support the needs of the systems engineering community.**
- **The SysML specification (Jul 2006), developed in response to these requirements by tool vendors, end users, academia and government representatives.**

- Standardization of the representation of systems
- Abstraction capacities : from the overview to the detailed description
- Separation of concerns via multiple views of integrated model
- Supports traceability through hierarchical system models
- Facilitates impact analysis of requirements and design changes
- Allows model based support: simulations, model checking, conformities, MDA
- Provides a continuity from system definition down to software development

- **A modeling language**
  - Notation
  - Semantics
- **SysML is NOT a methodology or a tool**
  - Specific usages of SysML are up to organizations or domains

Positioning SysML vs UML2

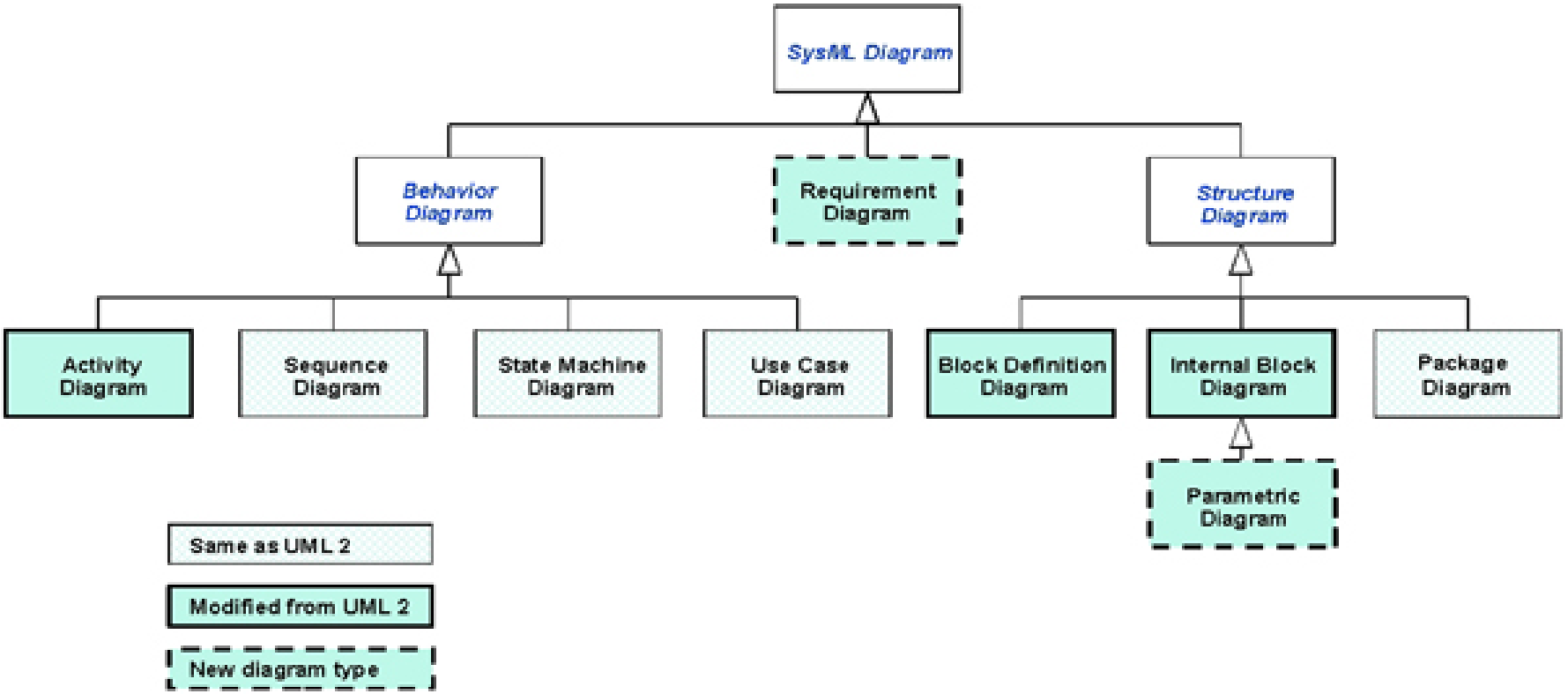


SysML

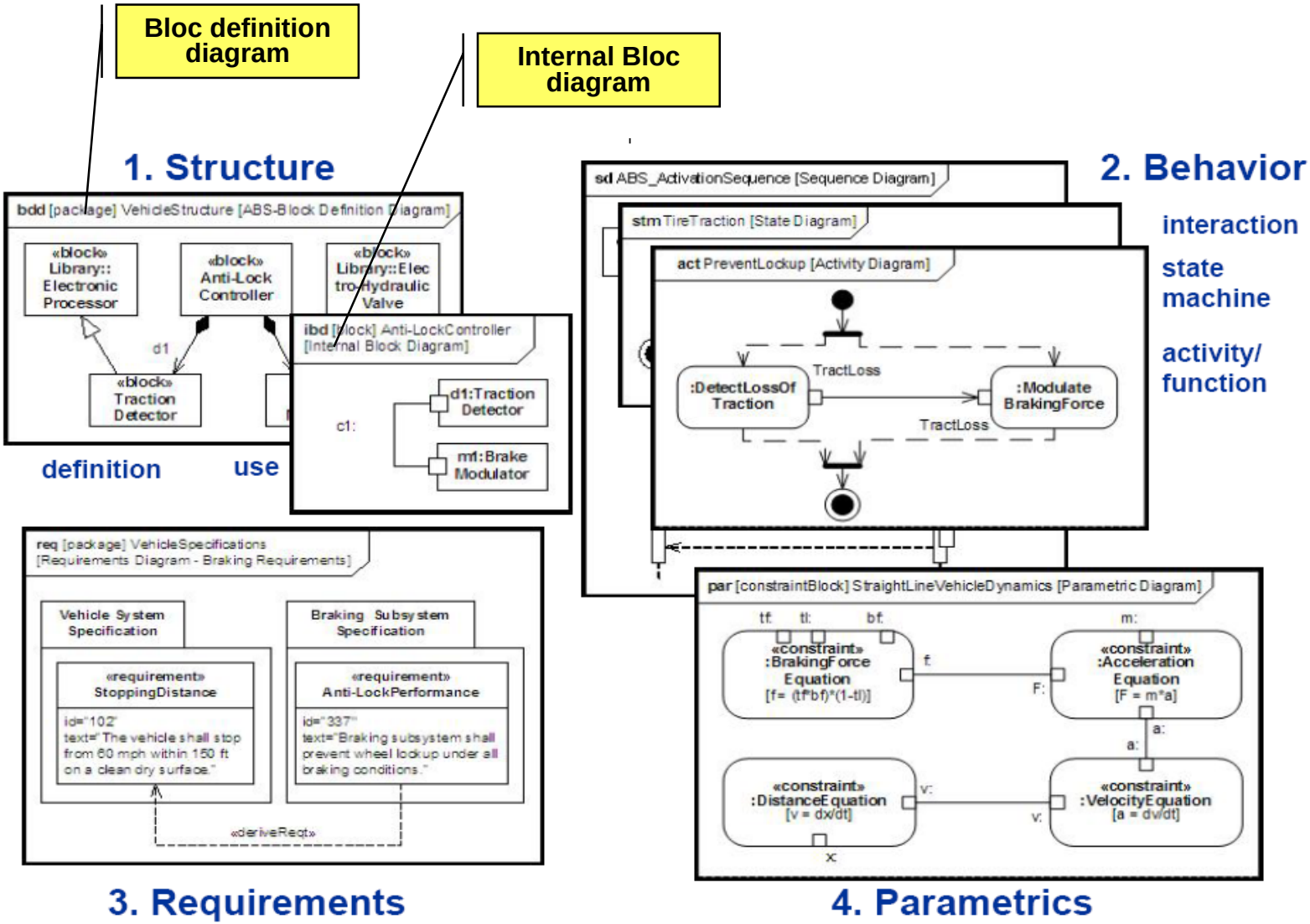
- **uses a simplified version of UML**
- **Reuses parts of UML**
- **Extends & adapt UML using the UML profile mechanism**

- A comprehensive modeling capacity, for dynamic and static aspects
- The assembly description capacity, with structured classes, parts, ports & connector
- An enhanced activity diagram capacity, that covers IDEF0 capacities
- An extension mechanism (profiles) that allows adapting UML for System engineering purposes

The SysML diagrams



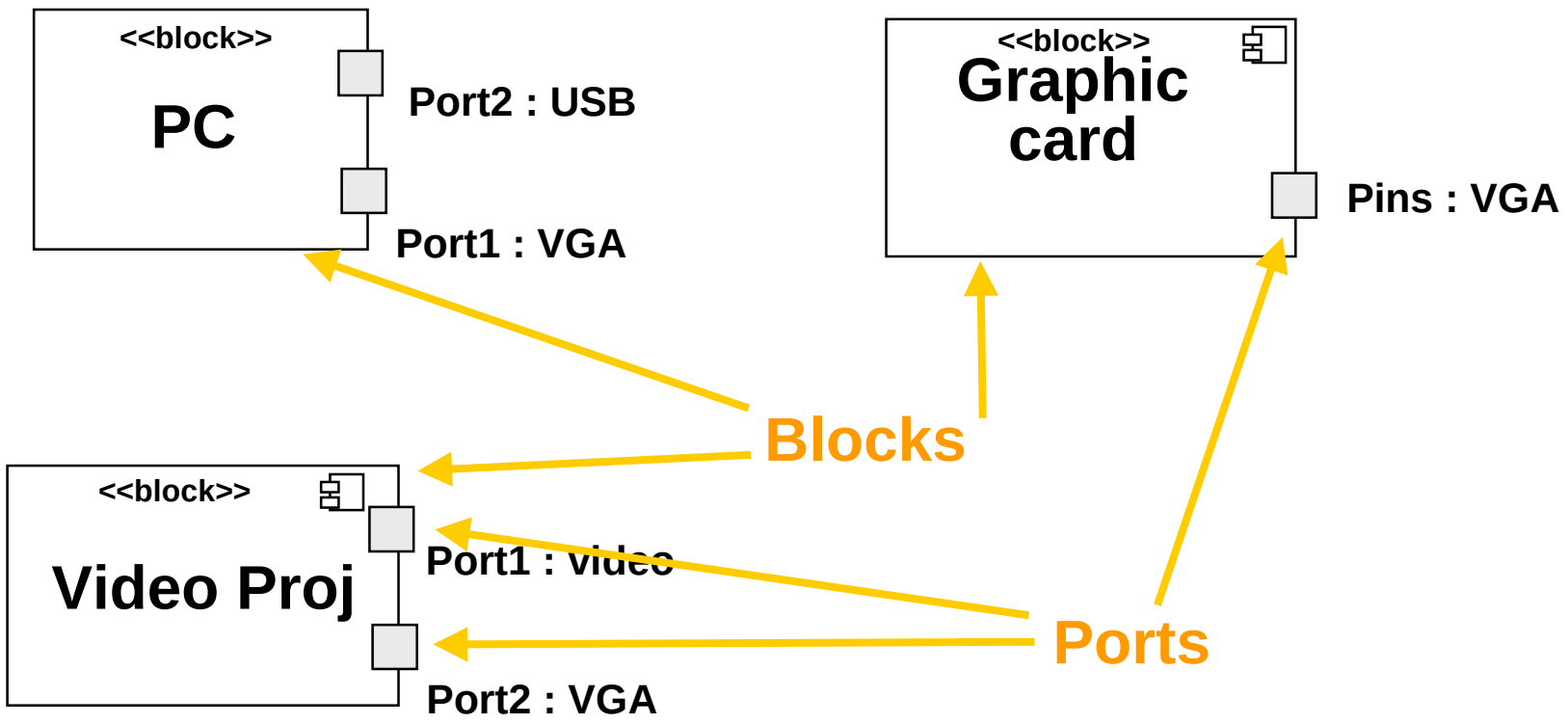
SysML diagrams : first examples



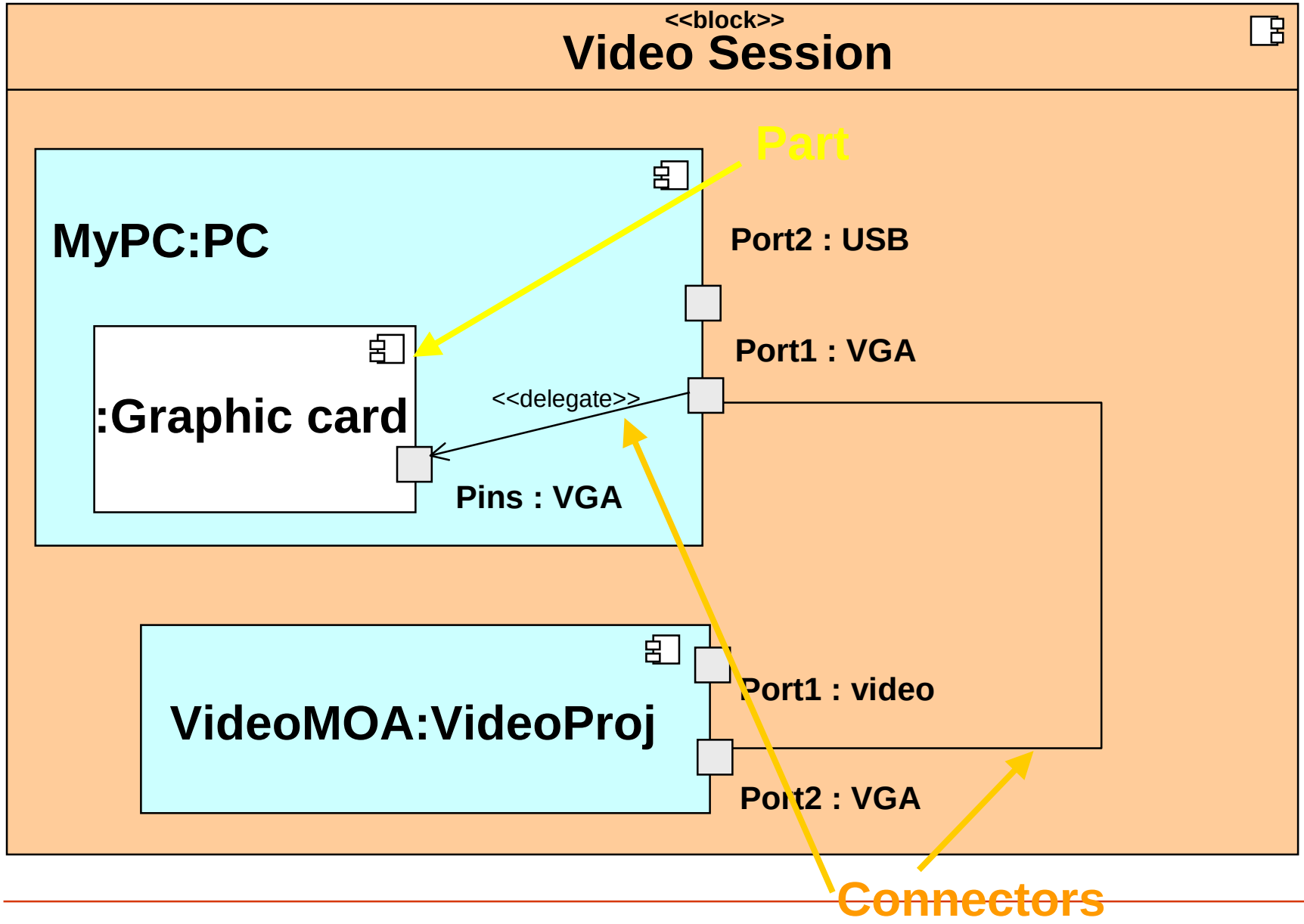
+Package & Use Case diagrams

- **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)**

- Design autonomous component
- Reuse on the shelf components
- Assemble them within different contexts
- Define the assembly points, and the contracts between the components
- Preserve component autonomousity while assembled



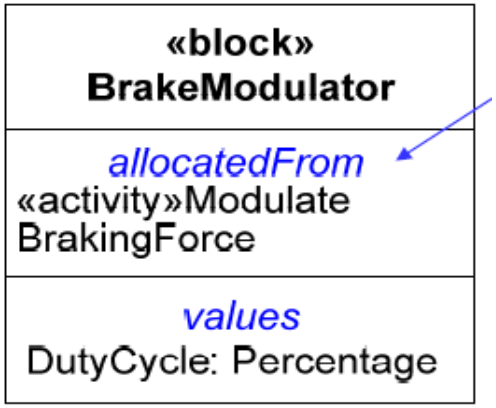
How can a video projection session be modeled by assembling these elements?



Describes 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

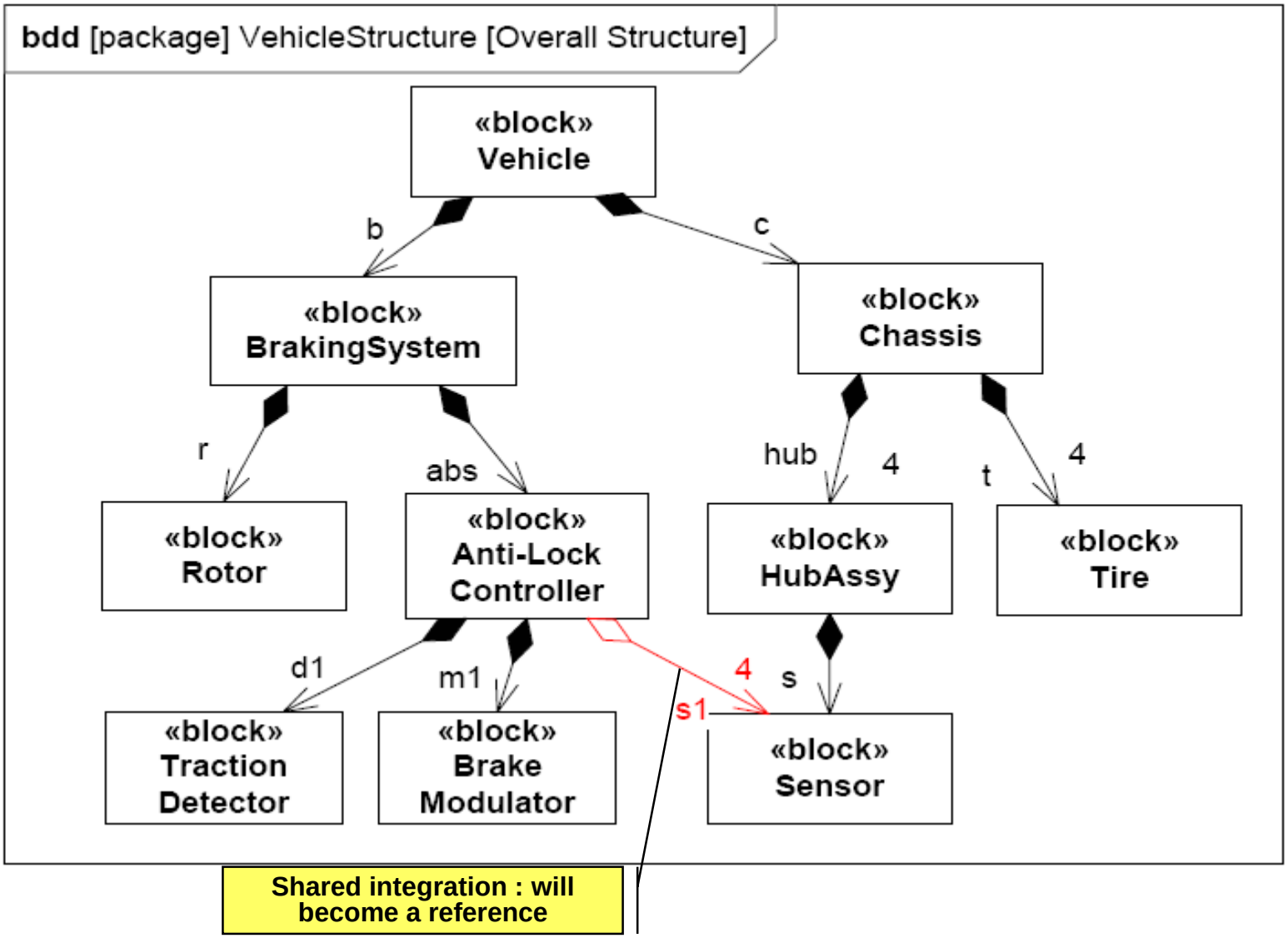


Properties of a block

- **Part property, 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

- Describes the relationship among blocks (e.g., composition, association, classification)
- Specifies the types of blocks that can be reused in different contexts
- Describes the system hierarchy and system/component classifications.

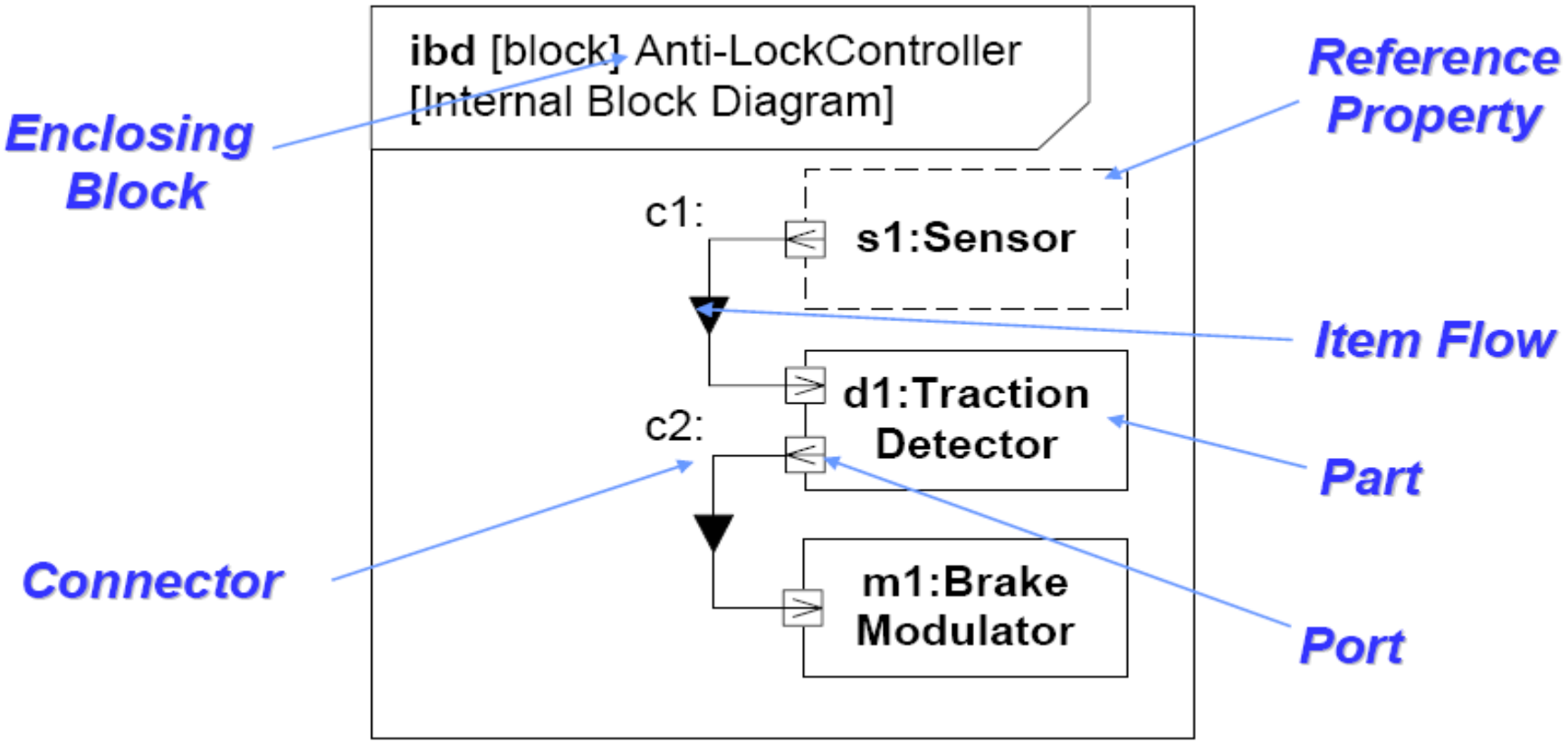
Block definition diagram - Example



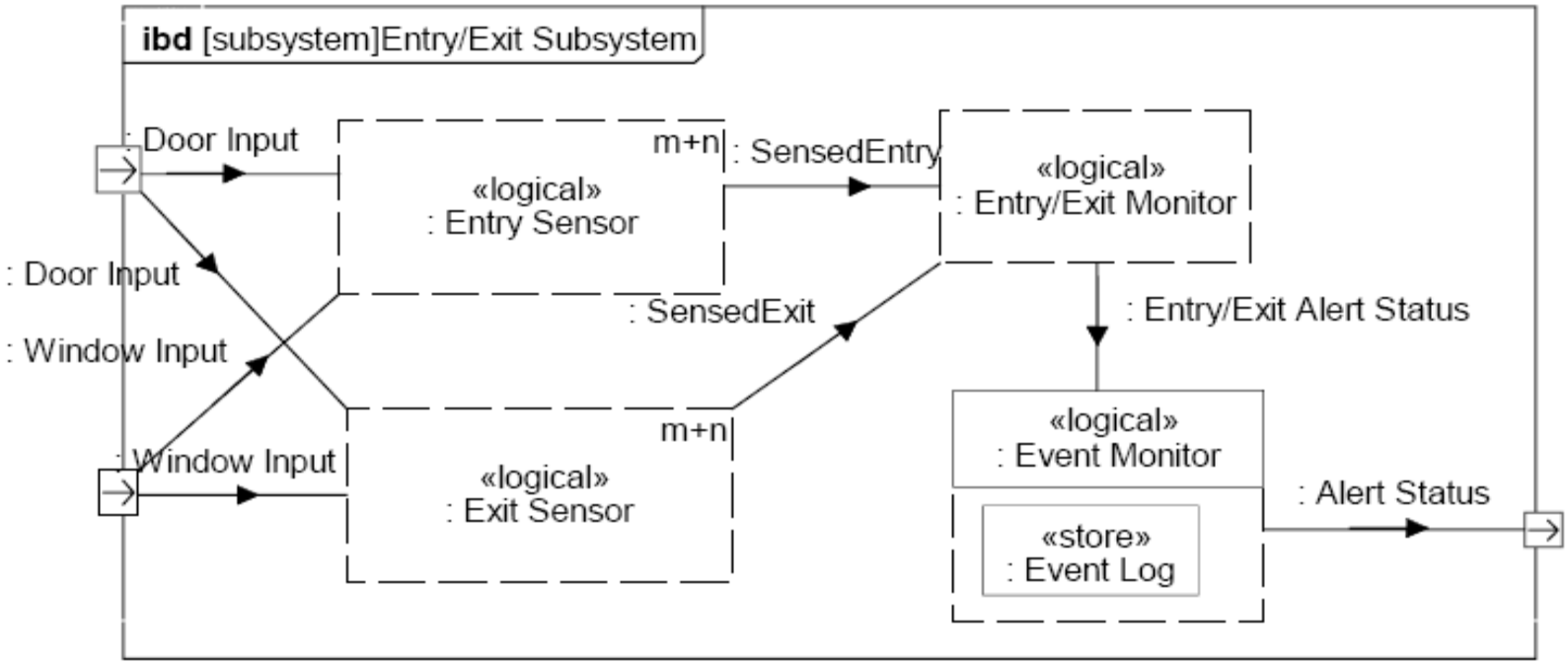
- **Specifies interaction points on blocks and parts**
- **Kinds of ports**
  - Standard (UML) Port
    - Specifies a set of required or provided operations and/or signals
    - Typed by an UML interface
  - Flow Port
    - Specifies what can flow in or out of block/part
    - Typed by a flow specification

- Internal block diagram describes the internal structure of a block in terms of its properties and connectors
- This structure of a block is described in terms of its parts, ports, and connectors.

Internal Block diagram - Example

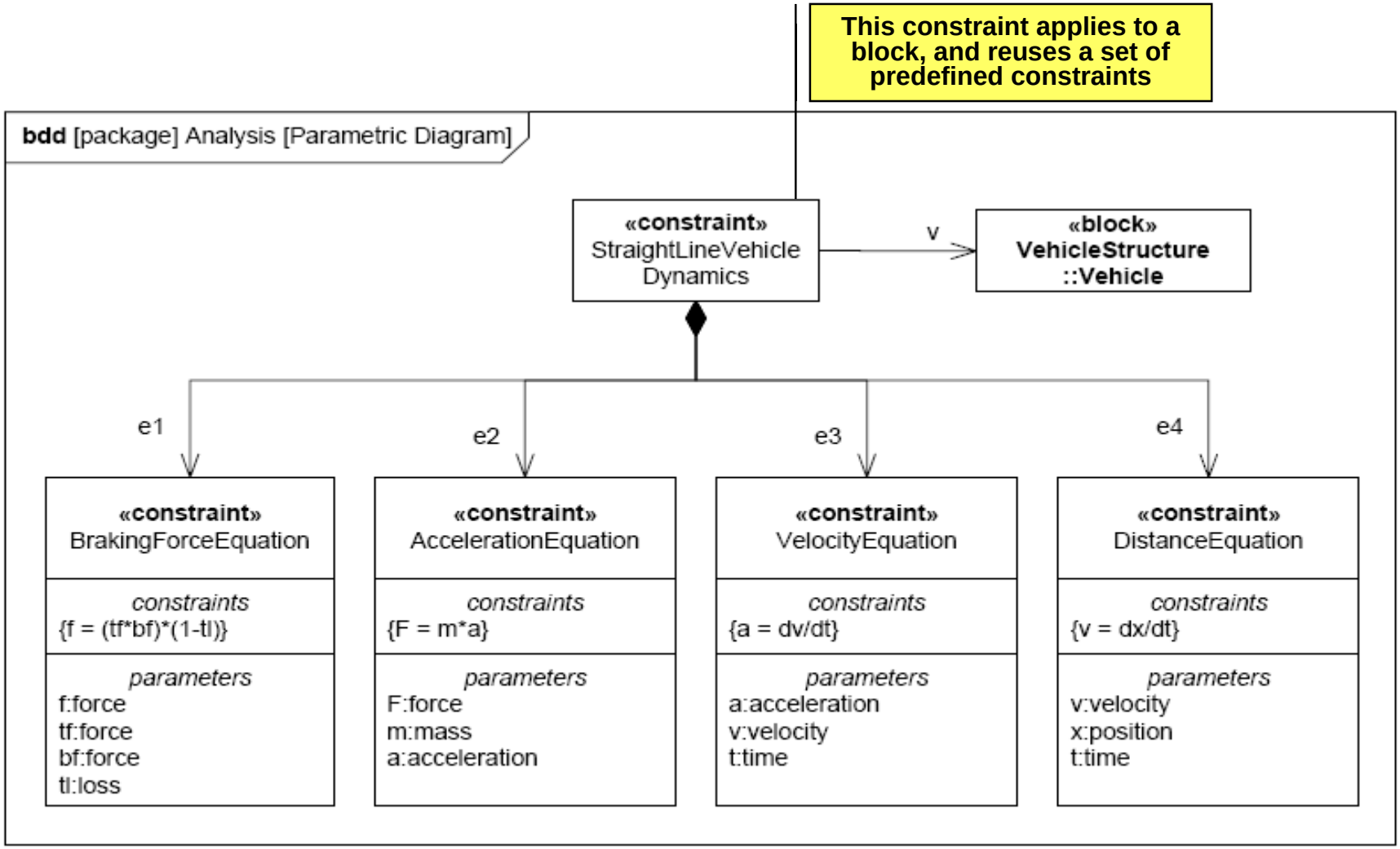


Internal Block diagram : Security system example

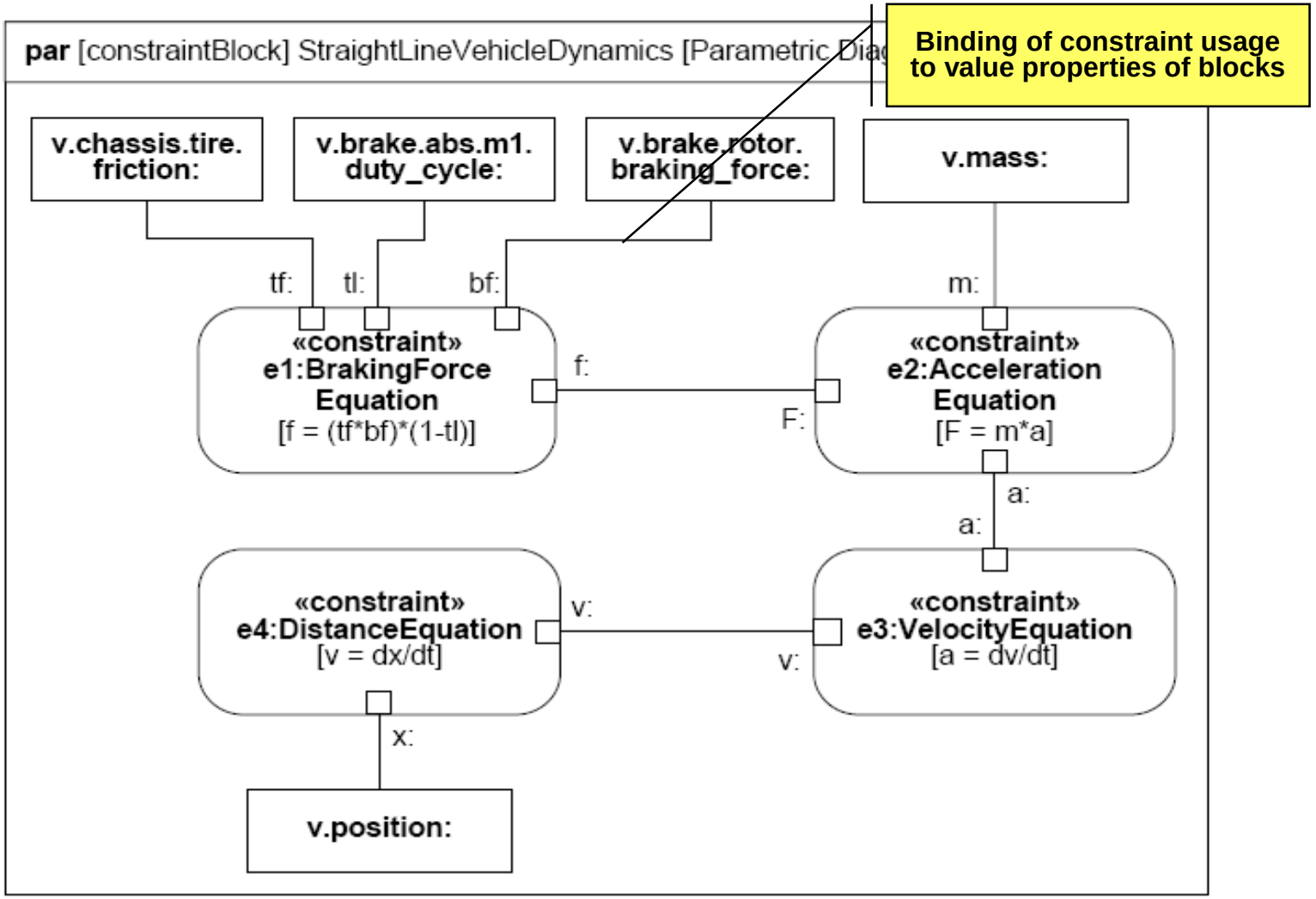


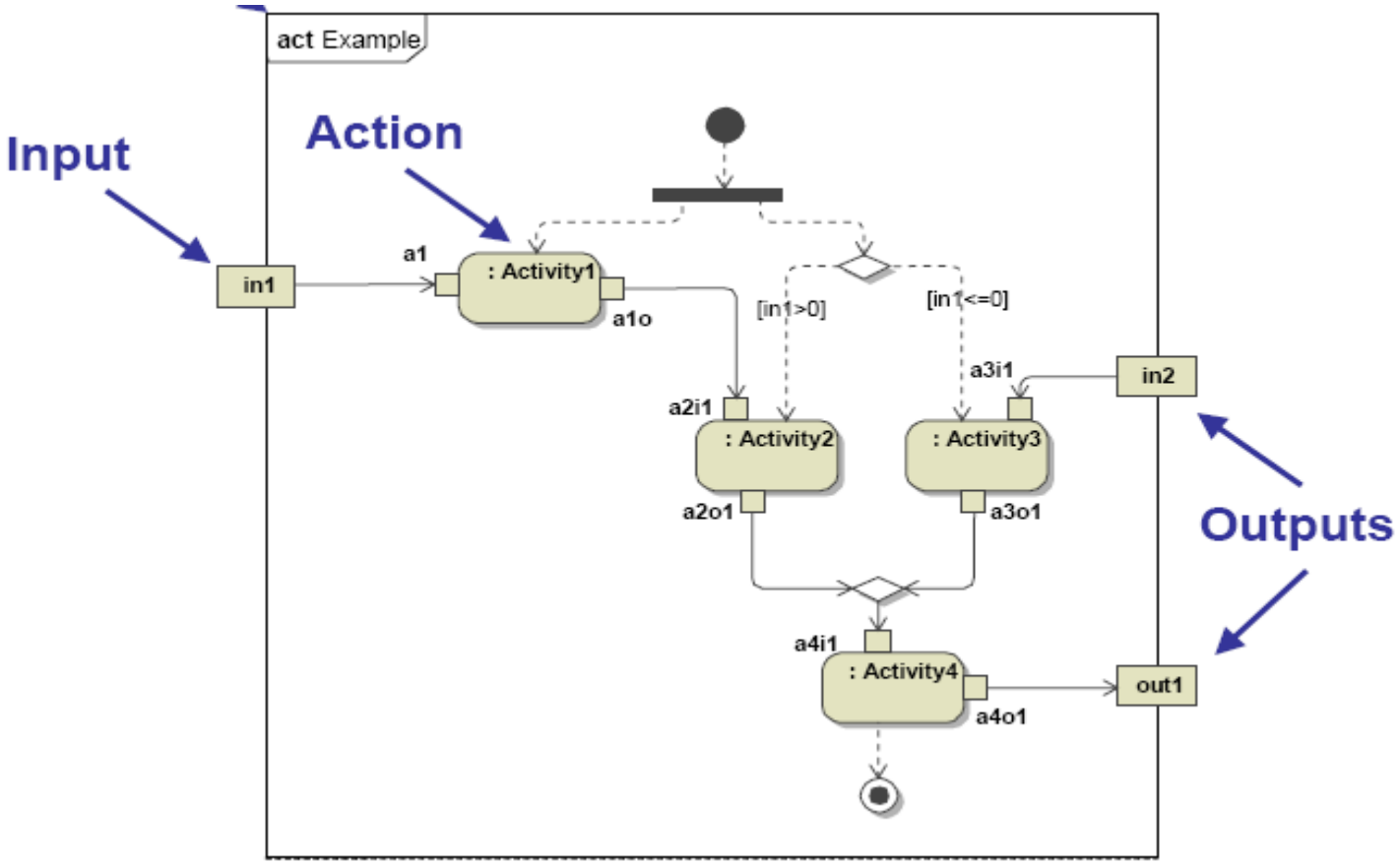
- **Used to express constraints (equations) between value properties**
  - Provides support for engineering analysis (e.g., performance, reliability)
  - Facilitates identification of critical performance properties
- **Constraint block captures equations**
  - Expression language can be formal (e.g., MathML, OCL) or informal
  - analysis tool related to the equation language can compute them
- **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 × a)

Constraint expressions



Parametric diagram

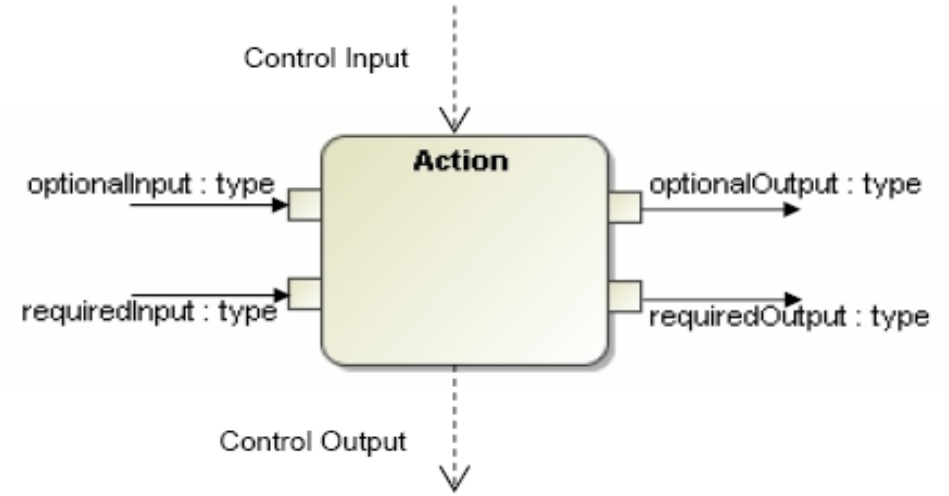




SysML extensions :

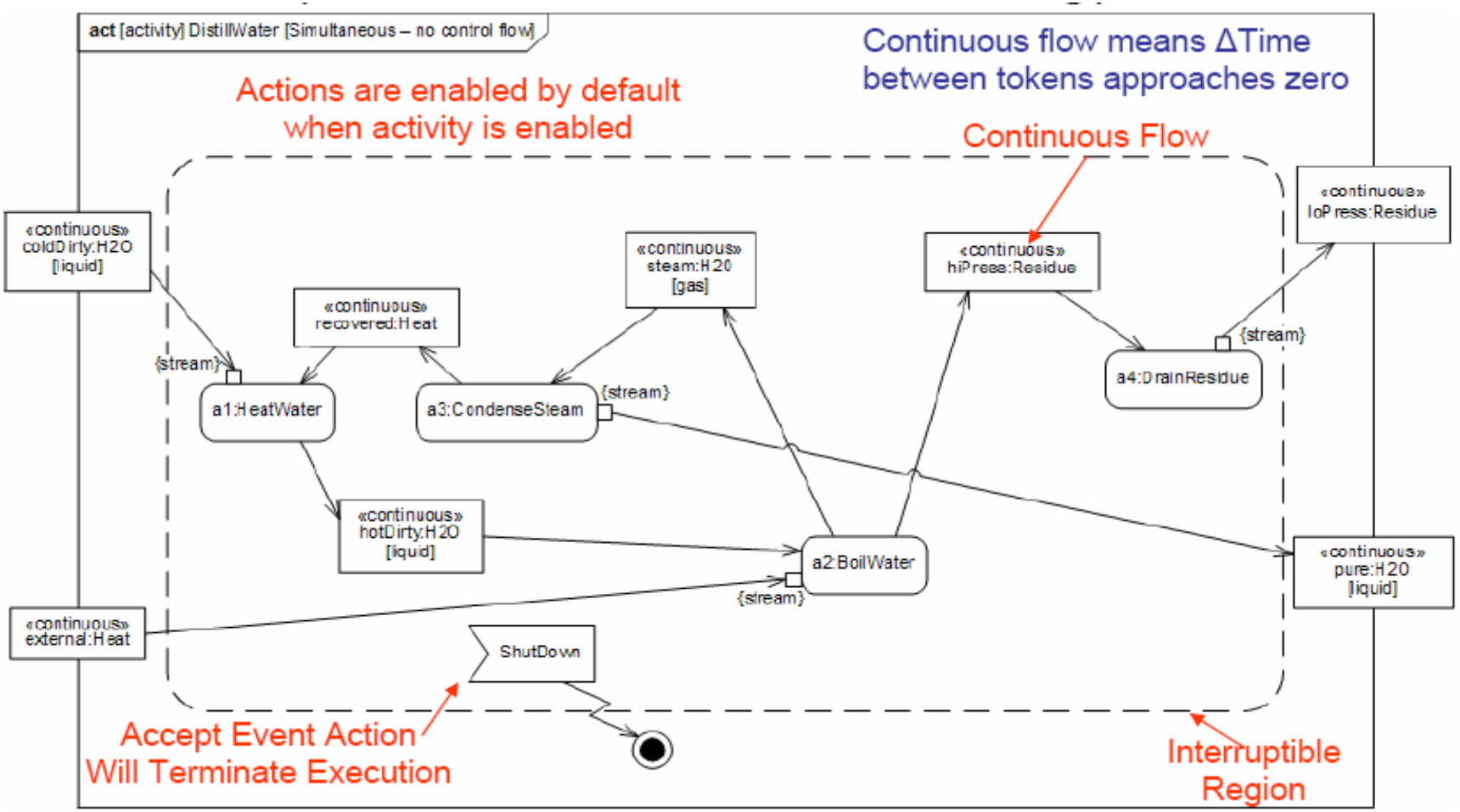
- **support for continuous flow modelling**
- **Notions of control as a kind of flow**
- **Probabilities on edges and parameters**

Actions & Flows : details

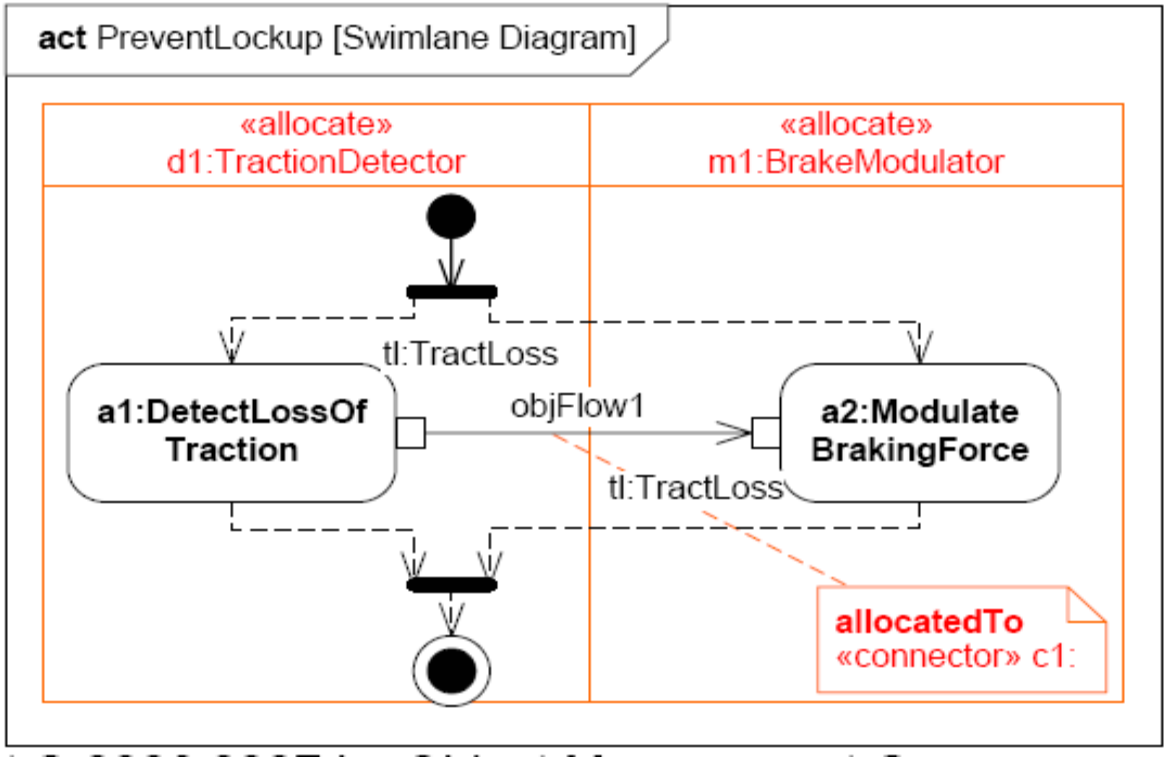


- **Two types of Flow :**
  - Object/Data
  - Control
- **Action execution begins when all tokens are available on control inputs and required input parameters.**
- **Streaming inputs and outputs continues to be consumed & produced while the action is running**

Continuous flows – representing physical processes



Allocation – Using swimlanes

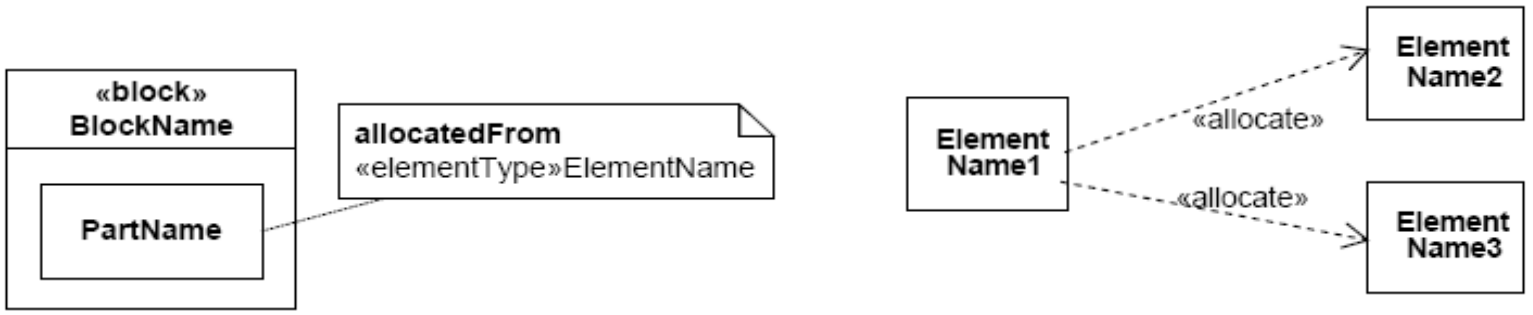


- **Explicit allocation of structure to behavior**

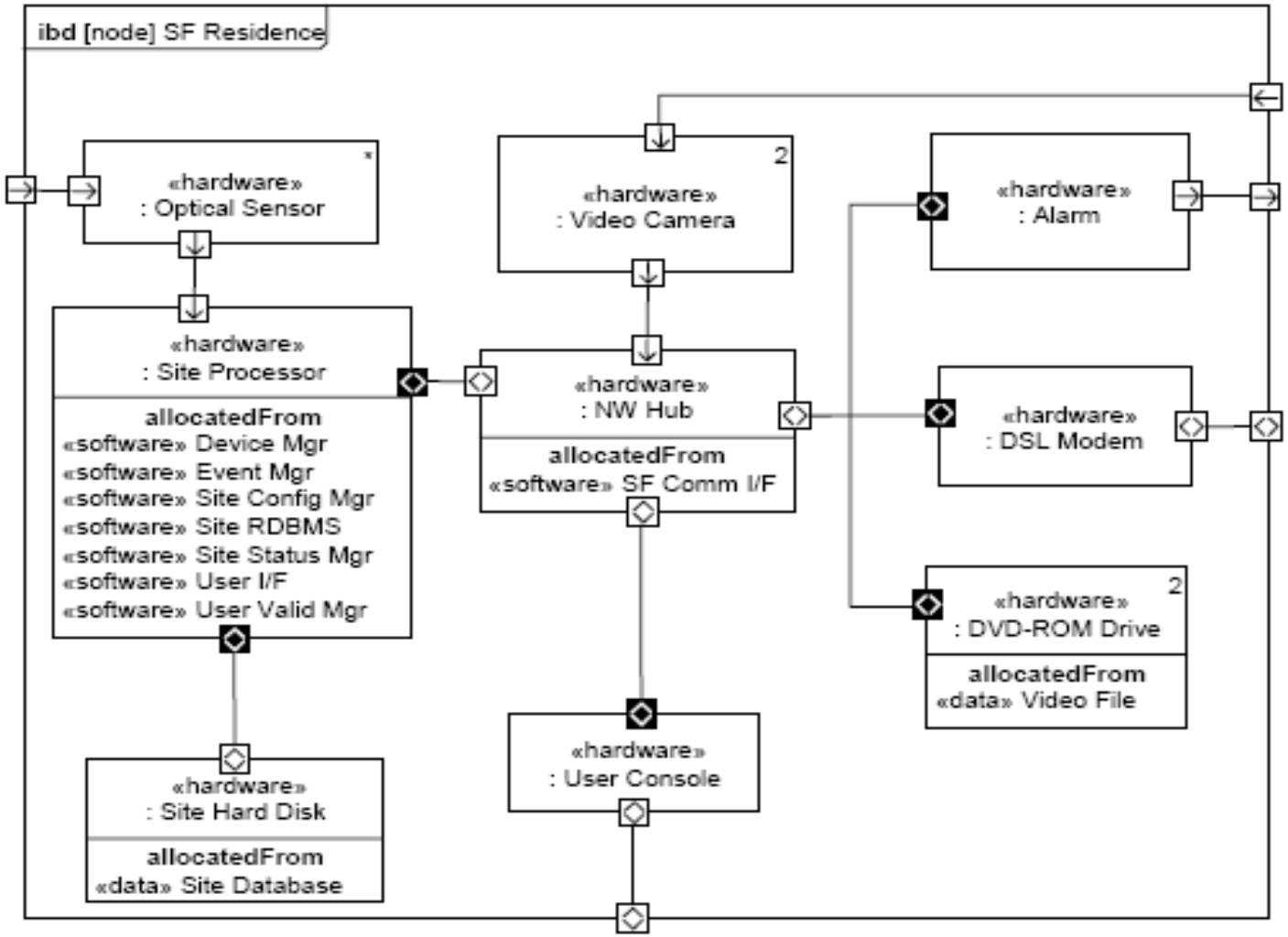
- **Interaction diagrams (sequence)**
  - Timing, interaction overview and communication diagram are excluded
- **State machine diagrams**
- **Use Case diagram**

Allocation

- 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 swimlanes (i.e., activity partitions)
- Both graphical and tabular representations are specified



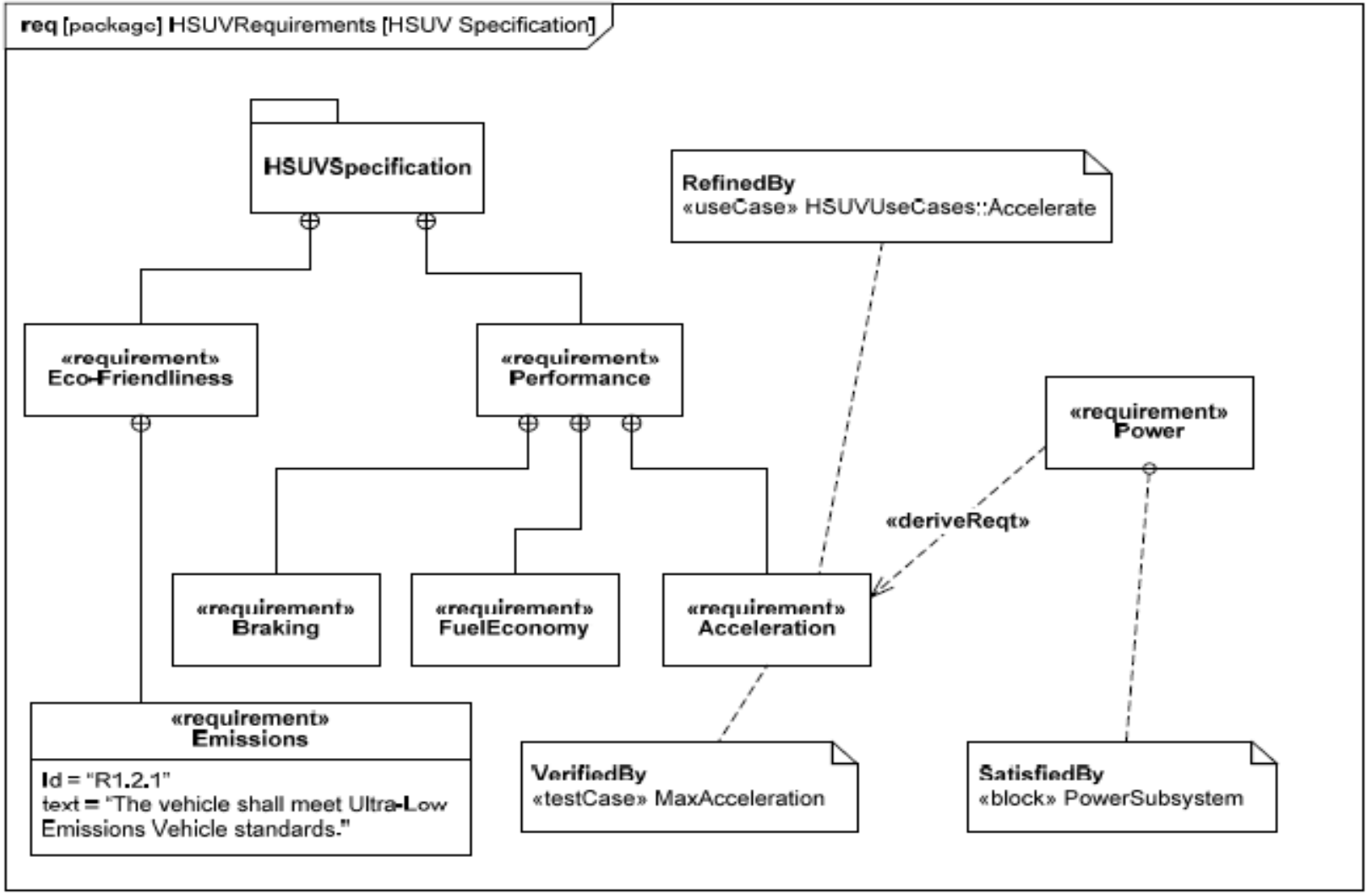
Allocation – Example : software to hardware allocation



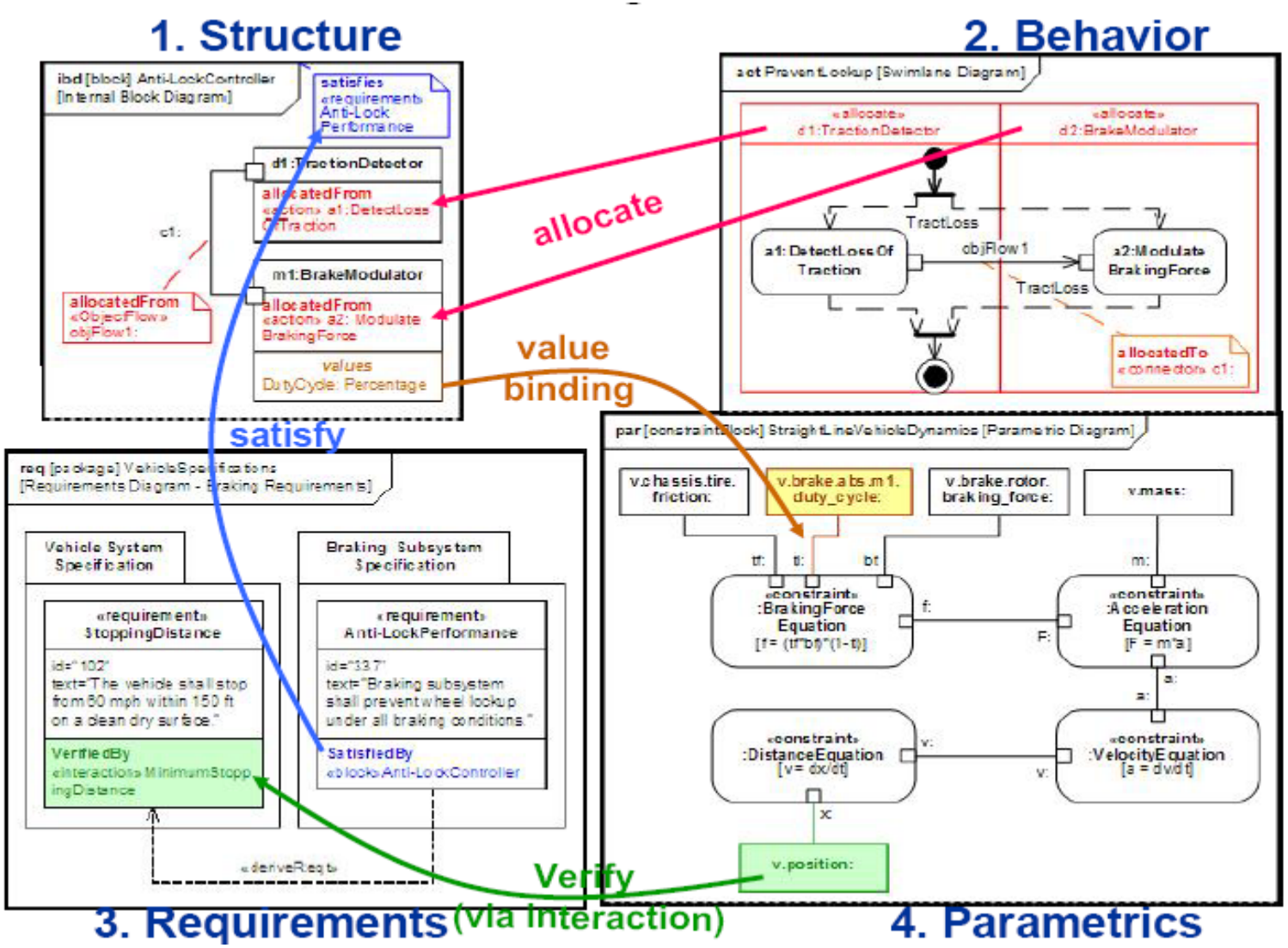
- **Used instead of deployment diagrams (artifacts allocated to nodes)**

- 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 - examples



Connections between te SysML model elements (summary)



- Highly expected standard, quite well accepted
- Still need to prove its generalised acceptance and usage
- Uses many new UML2 features
- Tools are ready to use (extension of UML Case Tools)