Modeling System Structure and Dynamics with SysML Blocks

Frontiers in Design & Simulation Research 2006
Georgia Institute of Technology
March 16, 2006

Roger Burkhart
Deere & Company

Modeling System Structure and Dynamics with SysML Blocks

Frontiers in Design & Simulation Research 2006
Georgia Institute of Technology
March 16, 2006

Roger Burkhart
Deere & Company

JOHN DEERE

Santa Fe Institute

Swarm Development Group
www.swarm.org

Topics

- **Overview of SysML (UML for Systems Engineering)**
  - **Background, objectives, and scope**
  - **Diagram types and examples**
  - **Status and prospects**
- **SysML Blocks as a foundation for knowledge management across the product development life cycle**
  - **Federated semantic models to cover an ever-expanding scope of cross-domain and cross-process integration**

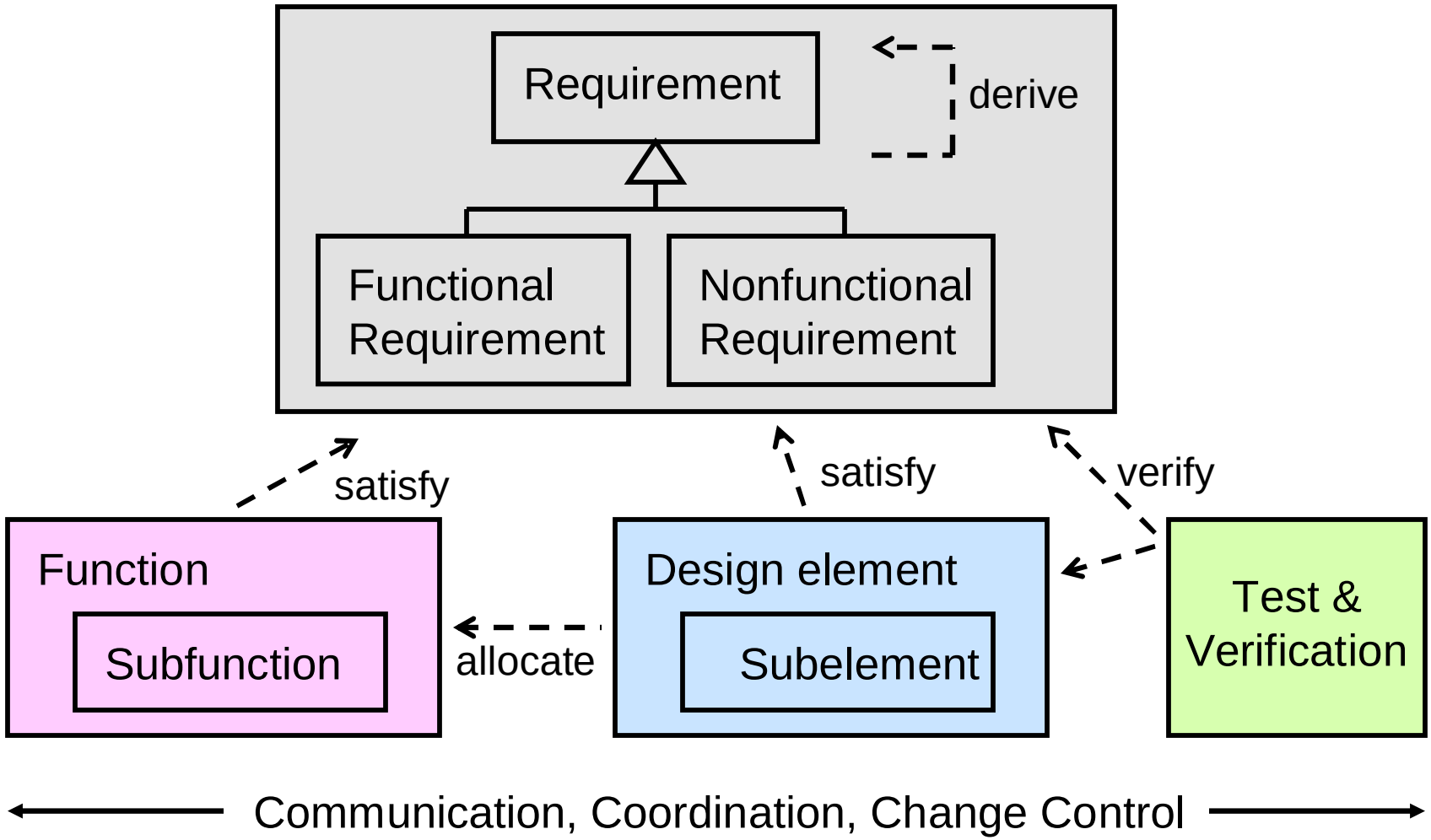
UML for Systems Engineering

- **Joint initiative of International Council on Systems Engineering (INCOSE) and OMG Systems Engineering special-interest group**
- **Provide computer-interpretable representation of products throughout their development lifecycle**
  - **Facilitate communication/collaboration**
    - **across engineering disciplines**
    - **across development tasks and responsibilities**
- **Support systems engineering processes**

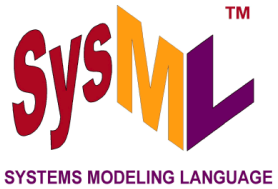
Systems Engineering Processes

- Requirements capture, allocation, traceability
- Conceptual design synthesis
- Optimization and tradeoff analysis
- Virtual validation and verification
- Integration of specialized disciplines
- Transition to downstream processes
  - Detailed design definition
  - Manufacturing & lifetime support

Systems Engineering Lifecycle



SysML Motivation



- **Systems Engineers need a robust language for analyzing, specifying, designing, verifying and validating systems**
- **Many different modeling techniques**
  - **Behavior diagrams, IDEF0, N2 charts, …**
- **General purpose language must:**
  - **satisfy broad set of modeling requirements (behavior, structure, performance, …)**
  - **integrate with other disciplines (SW, HW, ..)**
  - **be scalable**
  - **be adaptable to different SE domains**
  - **be supported by multiple tools**

INCOSE/OMG Joint Initiative

- **OMG Systems Engineering Activity chartered by International Council on Systems Engineering (INCOSE) and Object Management Group (OMG) in July 2001**
  - **create a semantic bridge between ISO 10303-233 standard and ISO/IEC 19501 UML standard (UML 2 extensions in progress…)**
  - **create UML extended modeling language for specifying, designing, and verifying complex systems using profiles, or other extensibility mechanisms.**
  - **provide capability for rigorous transfer of specifications and related information among tools used by systems, software and hardware engineers**
  - **bridge the semantic gap, the professional engineering discipline gap, and the training gap that exists between systems engineering and software engineering**

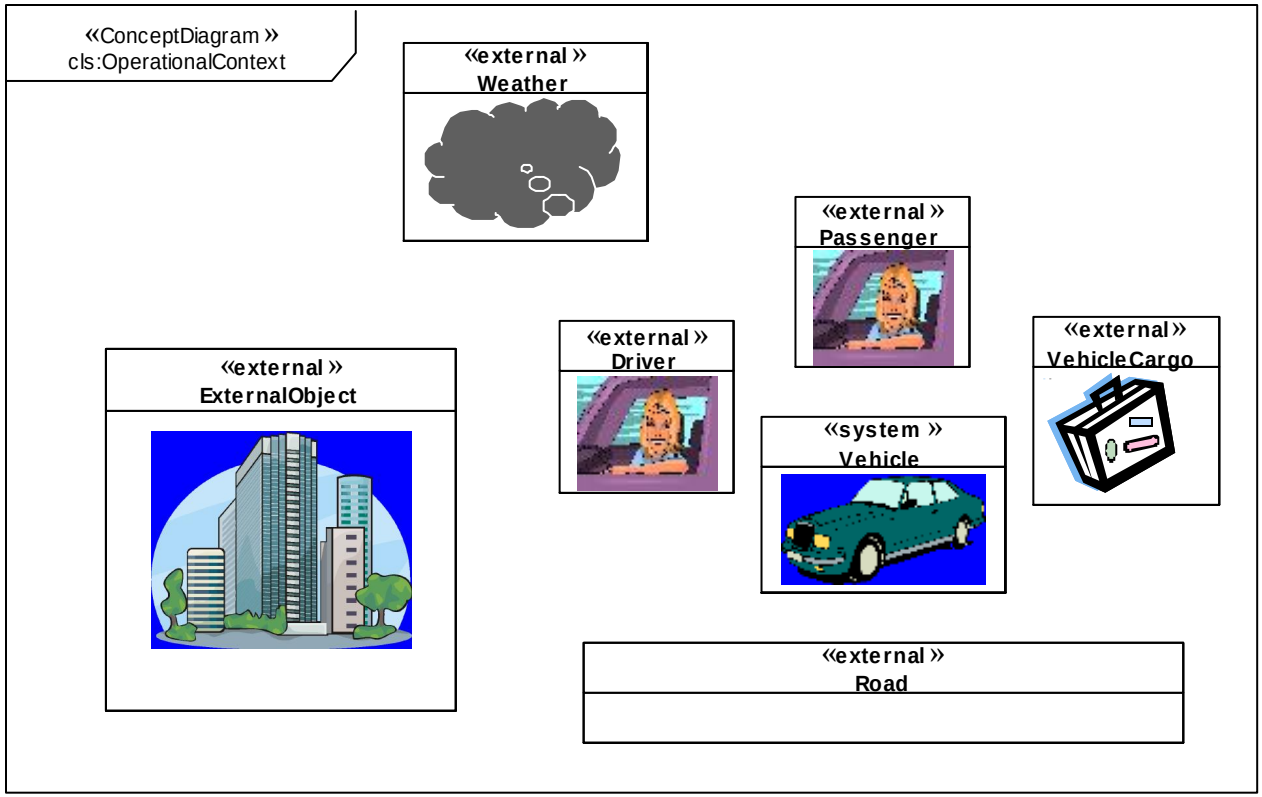
Current SysML Overview Slides

(See separate series of draft SysML Overview slides being developed by SysML Merge Team.)

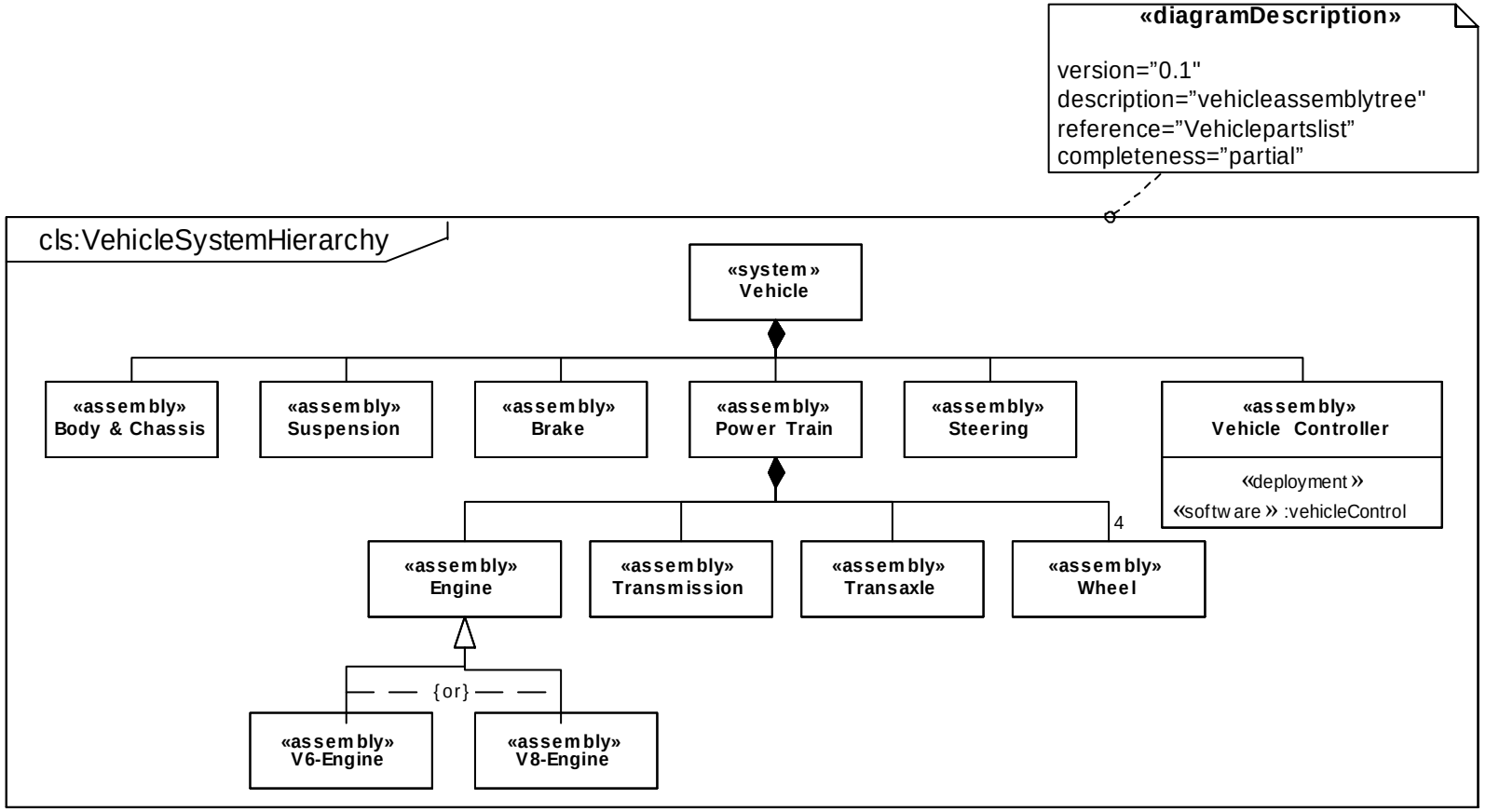
SysML Blocks

- **Built on UML 2 Composite Structure Diagrams originally defined for specification of real-time software components**
- **«block» stereotype provides a common root for user-defined or domain-specific hierarchies of system component types**
  - **Hardware**
  - **Software**
  - **Data**
  - **Procedure**
  - **Facility**
  - **Person**
- **Blocks provide the backbone of the "system hierarchy" or "system of systems" architecture which drives the systems engineering process**

Operational Context



Product Structure Tree



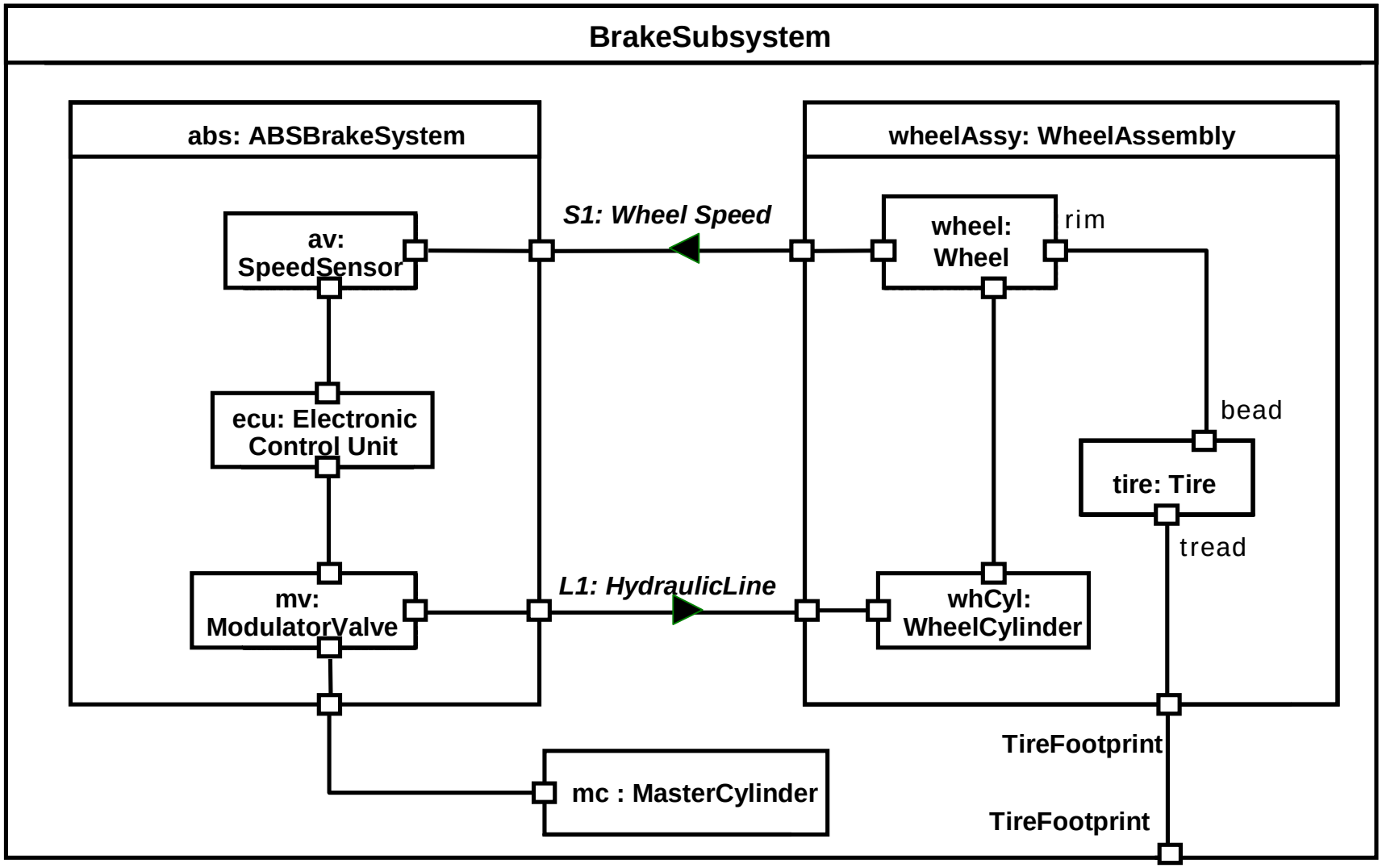
Structure Modeling Foundation

- **Blocks are UML structured classes**
  - **Classes extended with an ability to hold ports, parts, and internal connectors**
- **“Block” captures a module at any level in the system hierarchy.**
  - **Can represent external systems, a system of interest, logical, physical, hardware, software, etc.**
  - **Blocks provide both black-box view (without internal structure) and white-box view (showing internal parts and connectors)**

Parts, Ports, Connectors

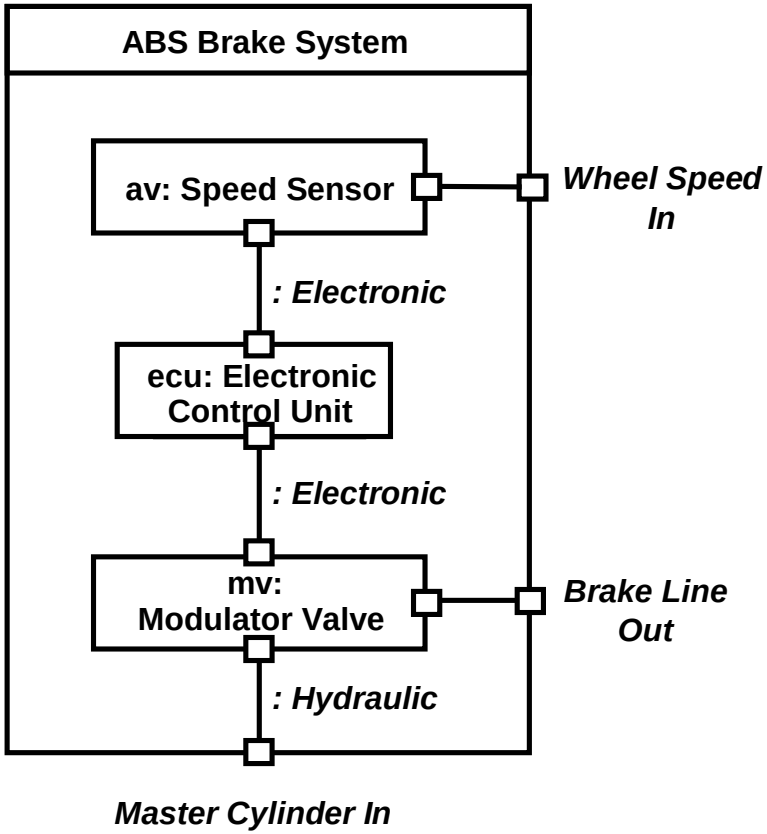
- **Parts are properties that are enclosed by assemblies and typed by classes**
  - **“Part” is defined as internal to assembly vs. ordinary properties that reference other systems/objects**
- **Ports are parts that provide interaction points**
  - **notationally represented as a rectangle on the boundary of a part (same as UML 2, but with option to show name inside)**
- **Connectors bind one part to another**
  - **can connect parts with or without ports**
  - **typed by associations**
  - **structural features of the enclosing class**

Internal Block Diagram - Example

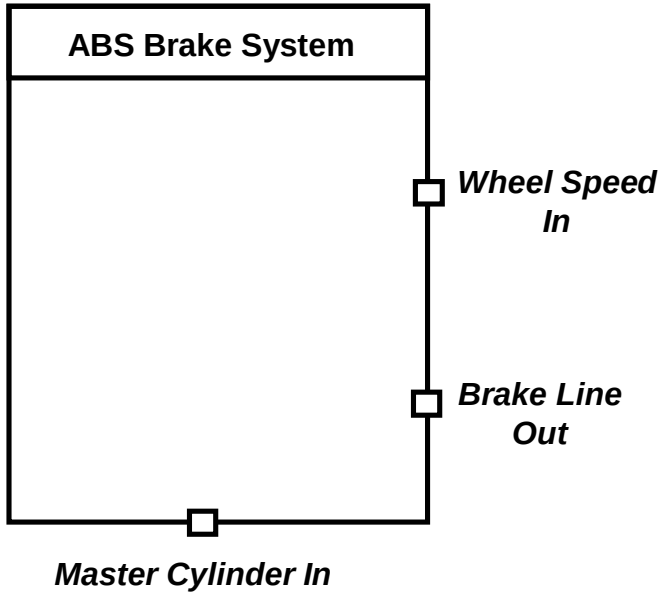


White vs. Black Box Views

White Box



Black Box



Concepts of Structure

- SysML “Block” defines fundamental abstractions of system structure
  - Individuals, whole-part relations, roles, connections
  - Reuse of standard components in larger whole
- Builds on UML class modeling foundation
  - Patterns and multiplicity of part occurrences
  - Built-in support for constraints and rules
  - Currently being mapped to Semantic Web and Common Logic by OMG Ontology Definition Metamodel
- Multiple models of system behavior can be built and expressed on blocks foundation
  - Functional flow, finite-state, procedural, continuous dynamics, discrete-event, ...

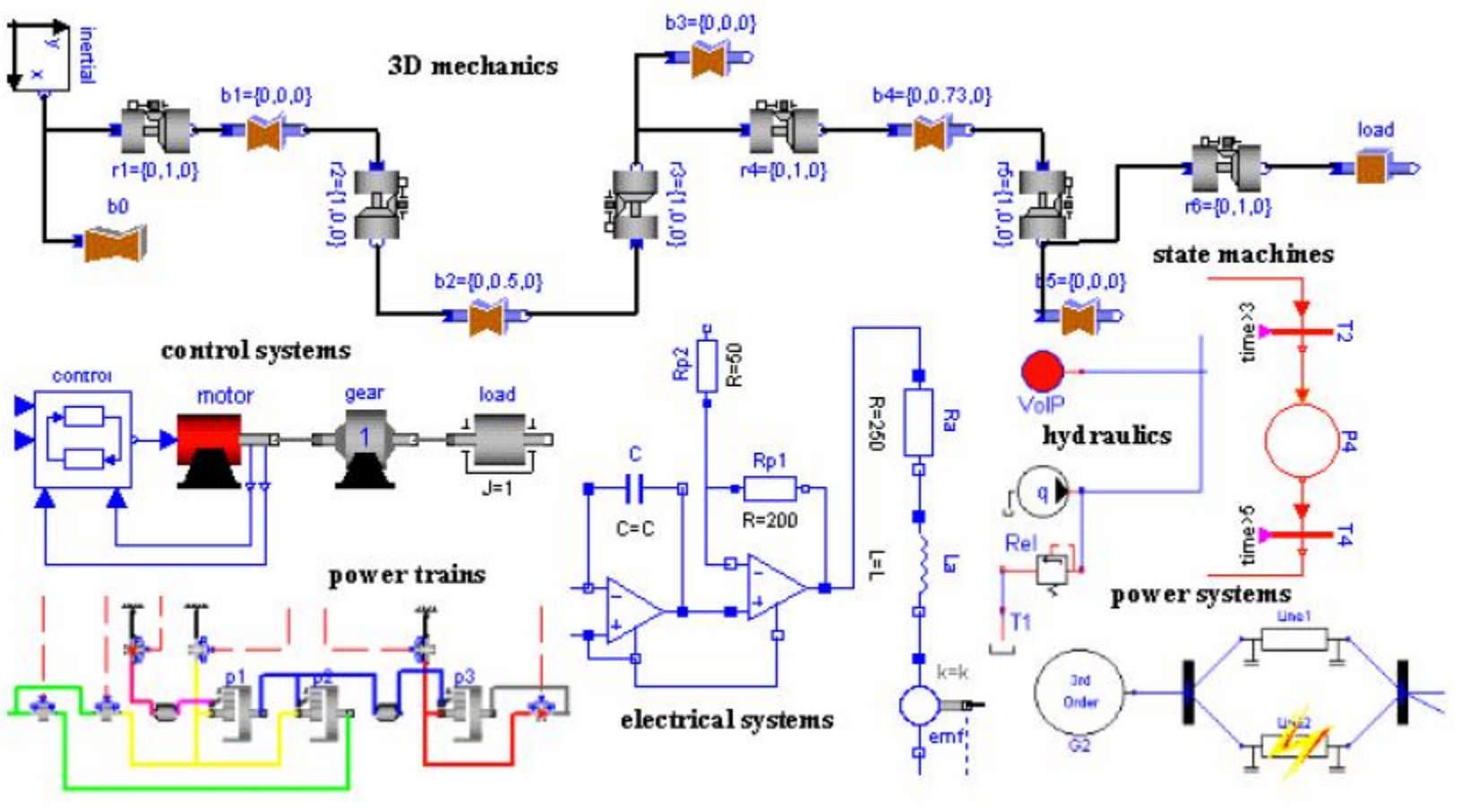
Mapping block structure models from SysML into STEP

- Export UML model in OMG XMI format
  - XML Metadata Interchange (XMI) export from prototype UML 2 tools
- Transform XML file to provide inputs into STEP model
- Define mappings from UML elements to STEP elements from PDM, AP239, and AP233 modules
  - Consulting with Eurostep to define STEP elements that already provide an equivalent to SysML assembly elements
  - Identifying gaps to feed into AP233 structure modules
  - Validating SysML assembly model prior to submission
- Load transformed model into STEP schema translated from EXPRESS to OWL (Web Ontology Language)
  - Extended to show product structure with interfaces

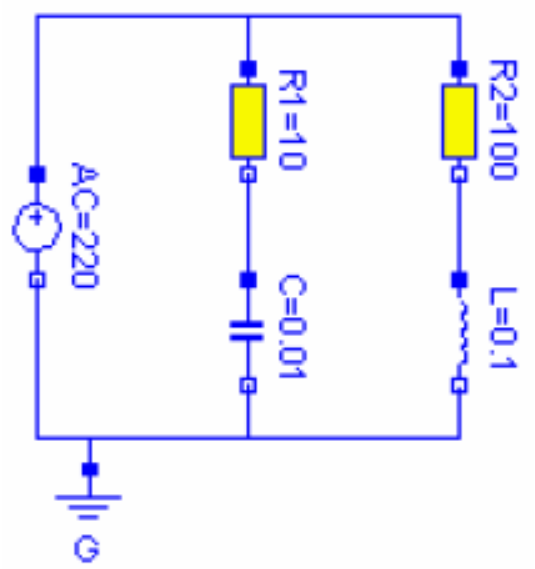
Selected Class of Application

- **Engineering block diagram models with hybrid continuous/discrete behavior**
  - **Widely used with commercial tools across many engineering disciplines**
  - **Examples can highlight added expressibility of SysML**
    - § **Structural variation**
    - § **Specialization/generalization**
    - § **…**
  - **Includes full detail for simulation/execution**
    - § **Tangible verification with completeness check**
    - § **Export to many other tools and mathematical solvers**

Modelica Diagram Examples



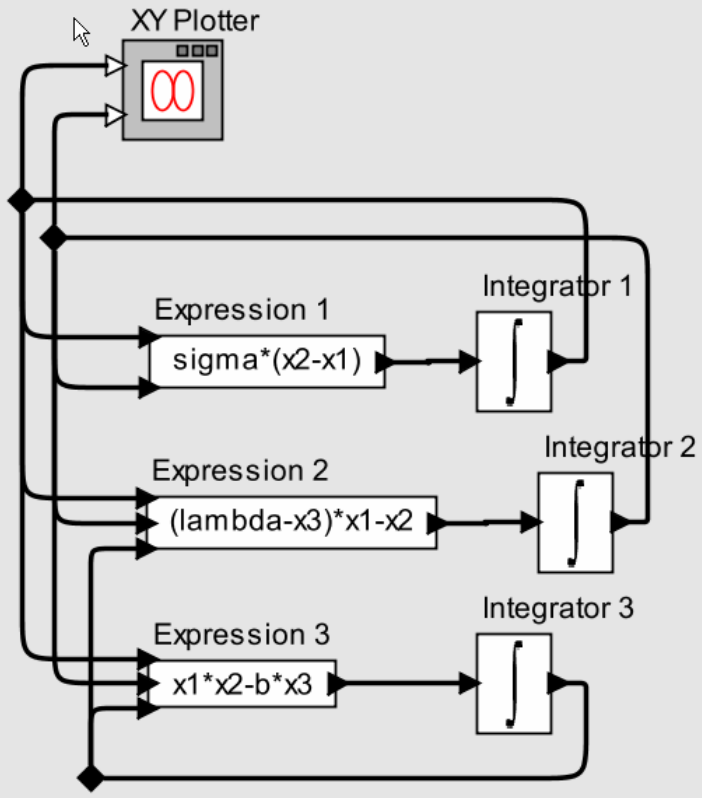
Modelica Language

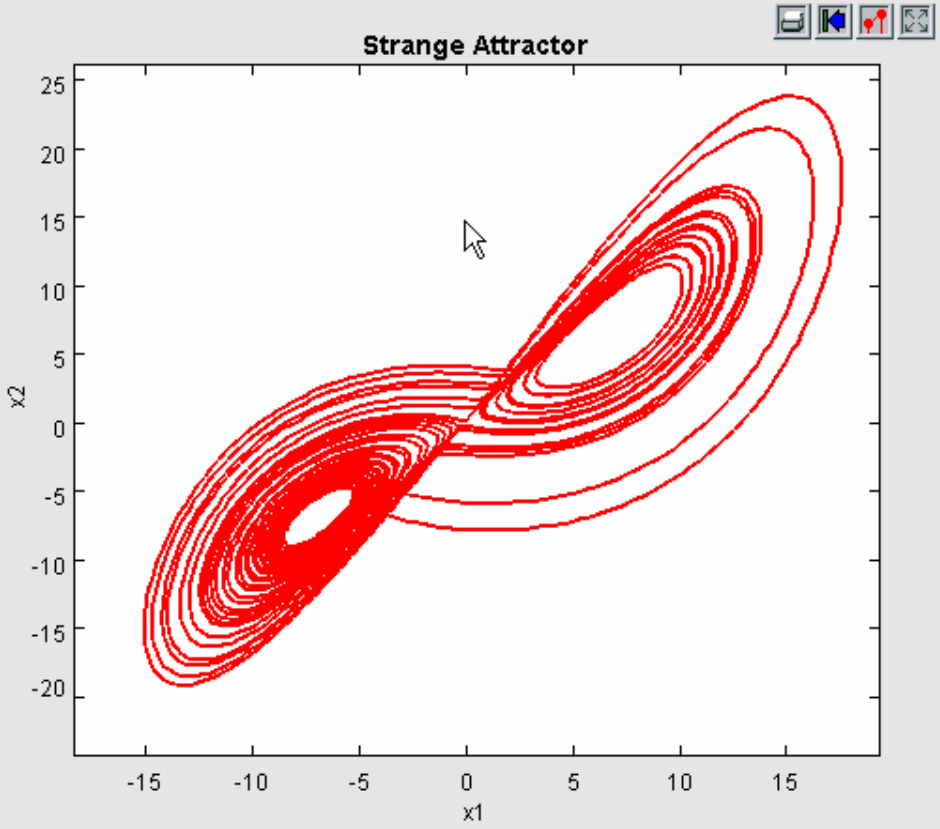



```
model circuit
  Resistor  R1(R=10);
  Capacitor C(C=0.01);
  Resistor  R2(R=100);
  Inductor  L(L=0.1);
  Vsource   AC AC;
  Ground    G;
equation
  connect (AC.p, R1.p); // Capacitor
  connect (R1.n, C.p);
  connect (C.n, AC.n);
  connect (R1.p, R2.p); // Inductor
  connect (R2.n, L.p);
  connect (L.n, C.n);
  connect (AC.n, G.p);  // Ground
end circuit;
```

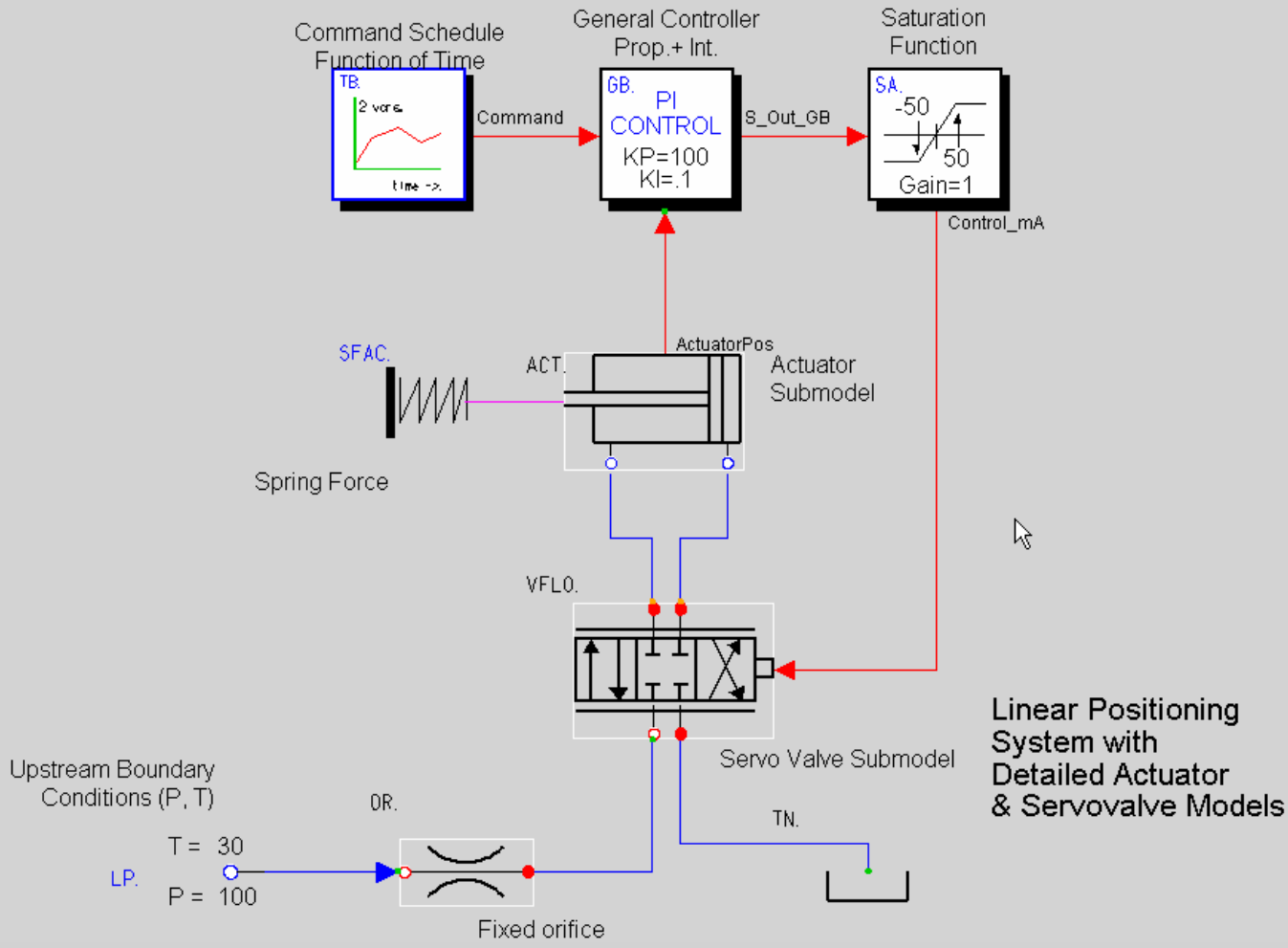
Ptolemy II System

- Free package from U.C. Berkeley EECS Department
- Support for multiple “Models of Computation”
- Built on generic abstract syntax for “actor-oriented design” with hierarchical actors, ports, channels

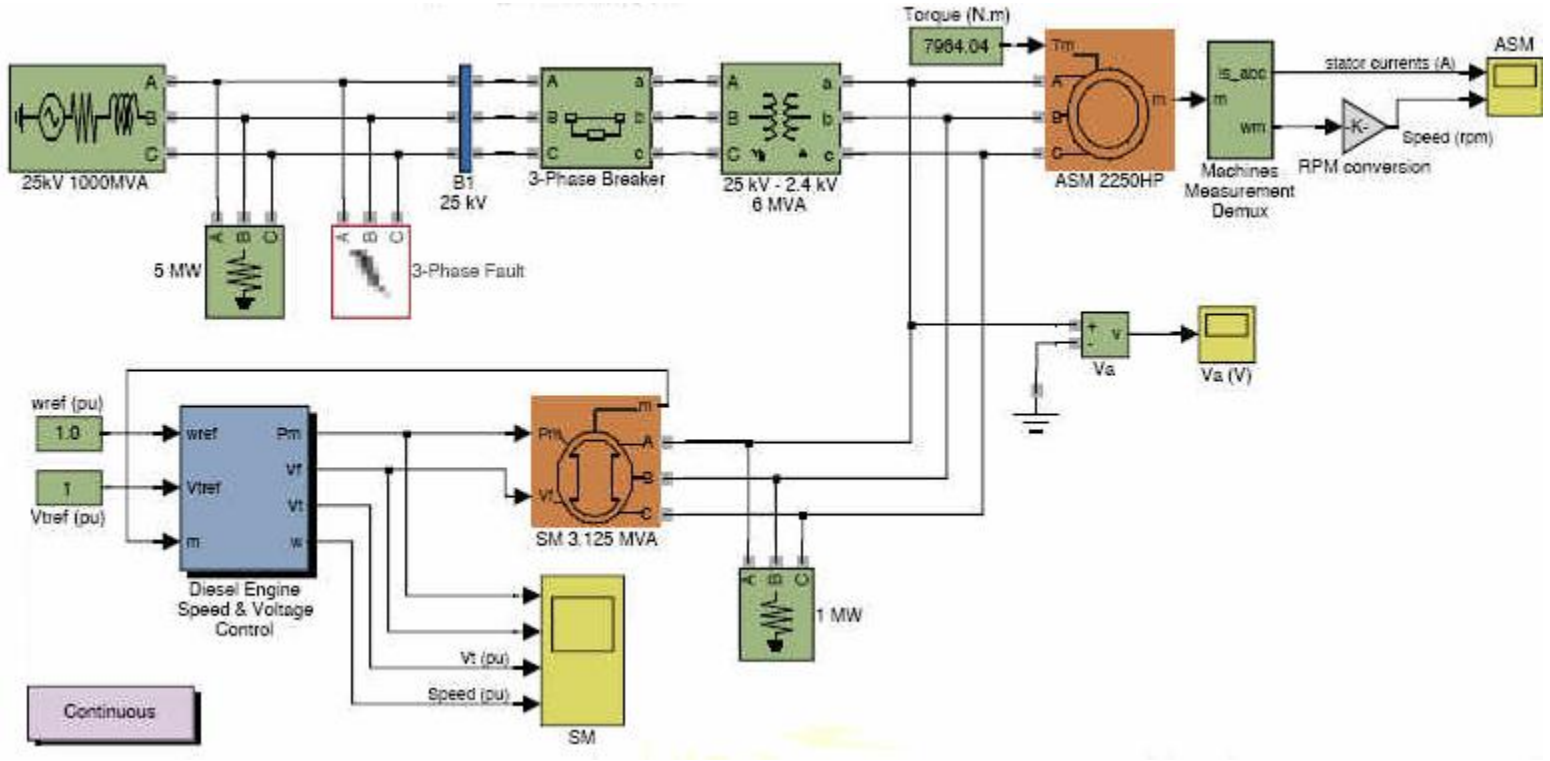




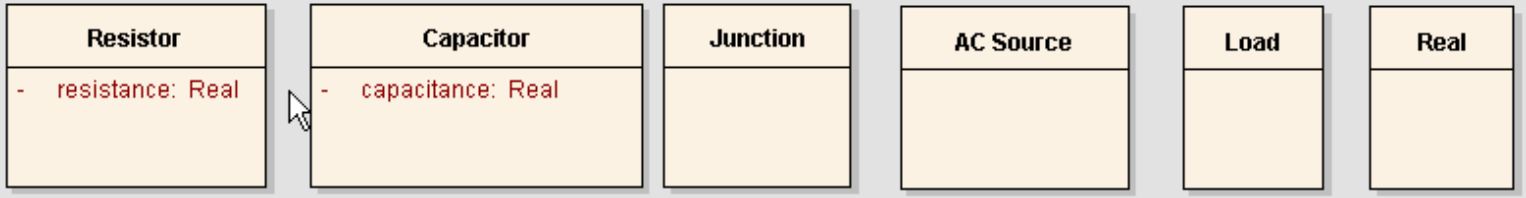
MSC EASY5

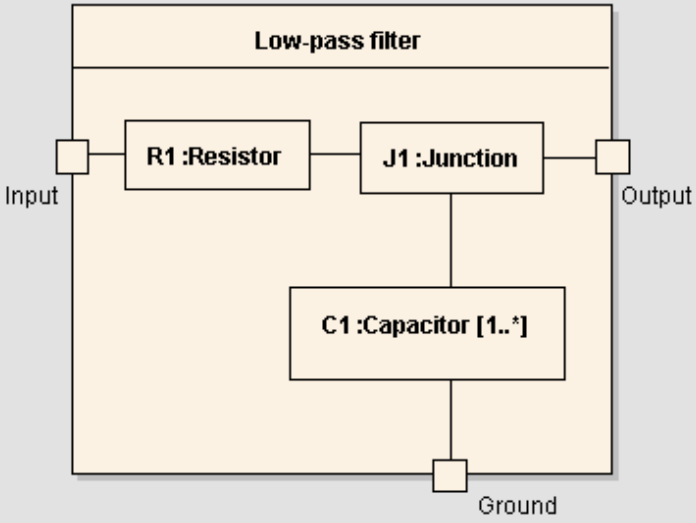

Command Schedule
Function of Time
TB.
2 vars
time ->
Command
General Controller
Prop.+ Int.
GB.
PI
CONTROL
KP=100
KI=.1
S_Out_GB
Saturation
Function
SA.
-50
50
Gain=1
Control_mA
SFAC.
ACT.
ActuatorPos
Actuator
Submodel
Spring Force
VFLO.
Servo Valve Submodel
Upstream Boundary
Conditions (P, T)
T = 30
P = 100
LP.
OR.
TN.
Fixed orifice
Linear Positioning
System with
Detailed Actuator
& Servovalve Models

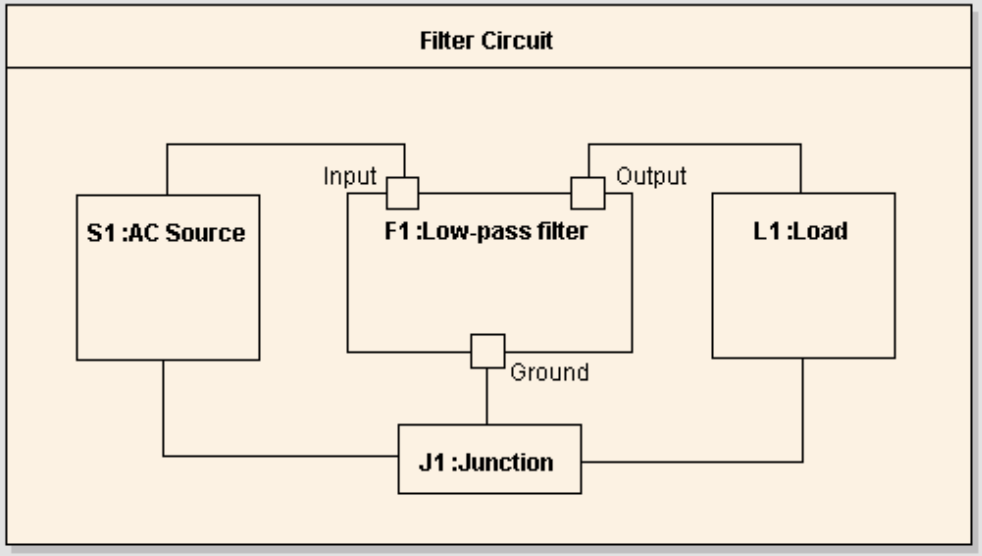
Mathworks Matlab Simulink



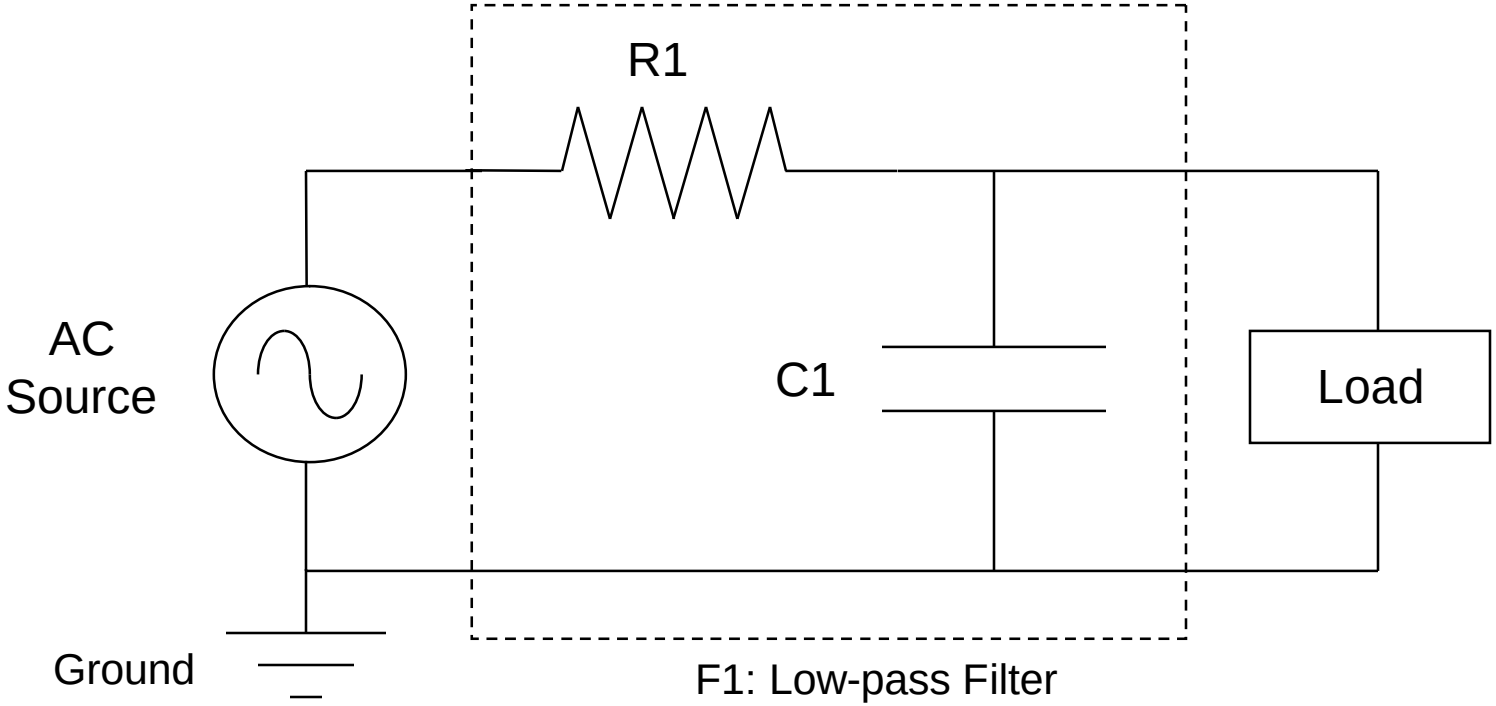
SysML Test Example



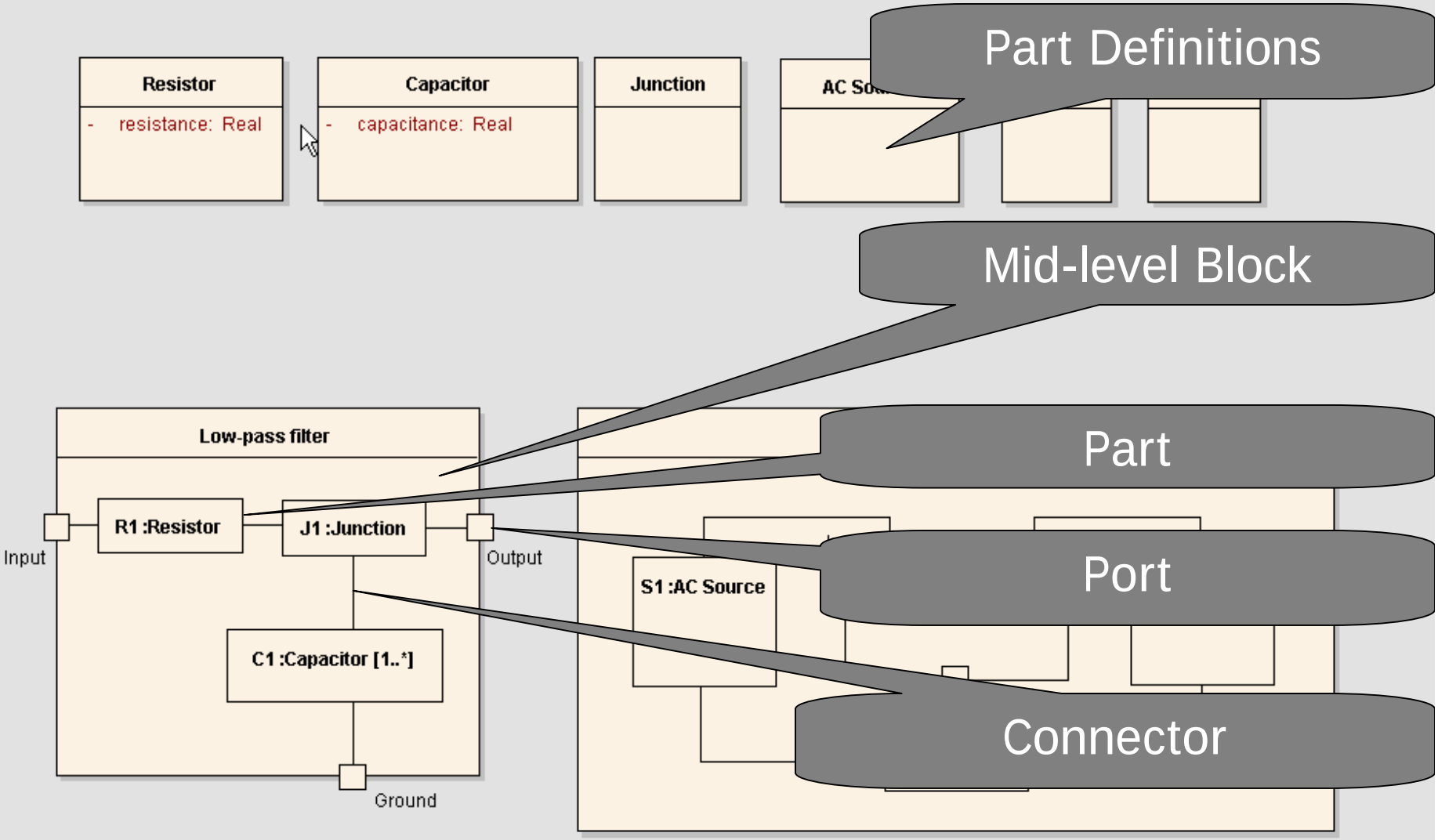




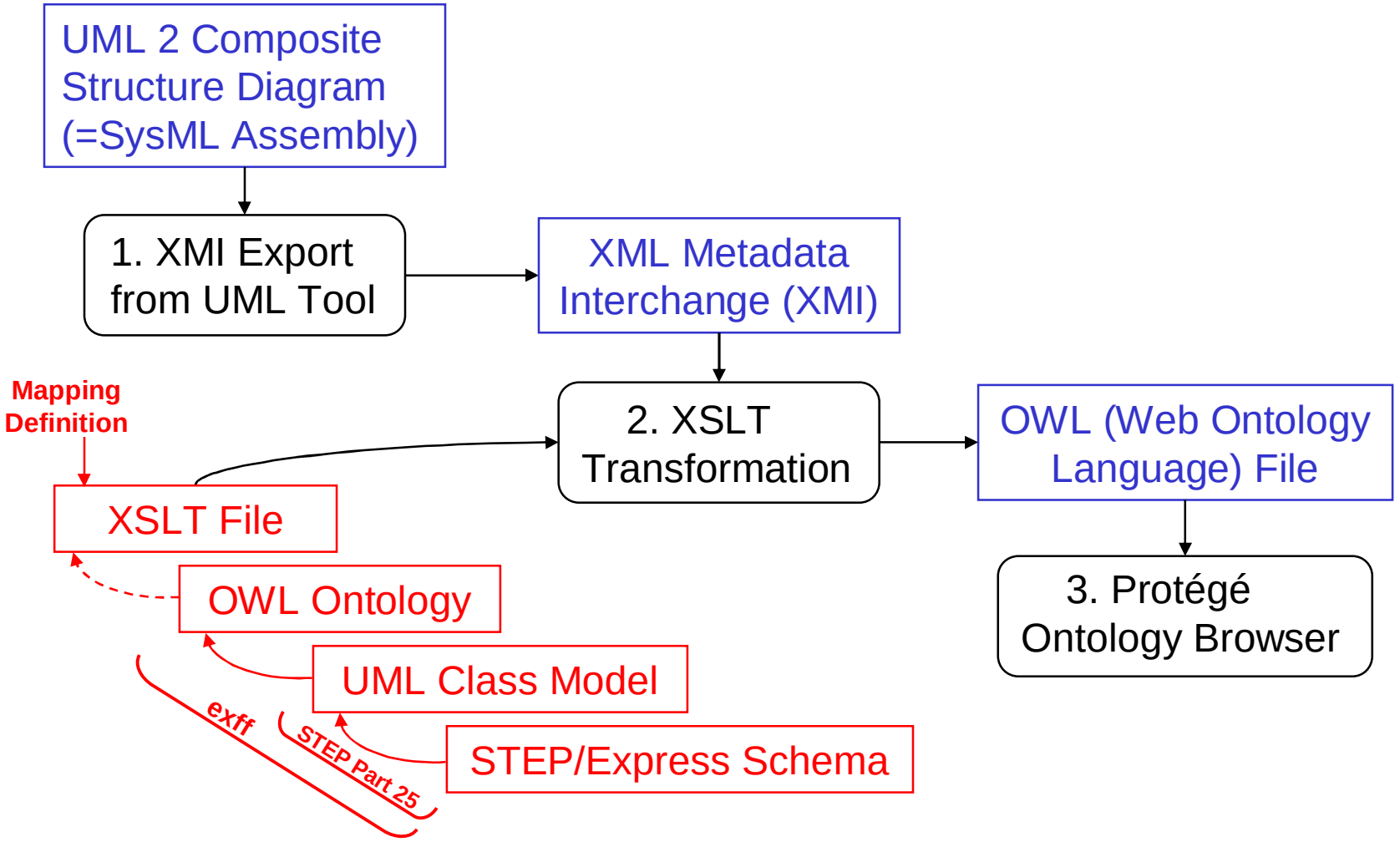
Equivalent Schematic



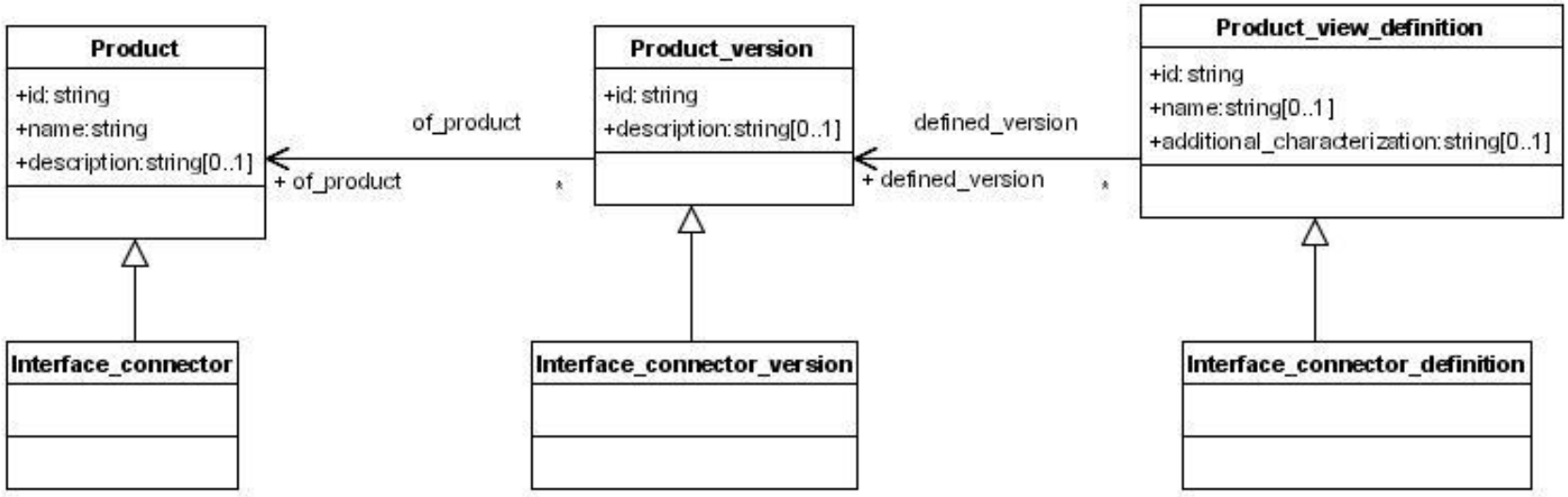
SysML Test Example



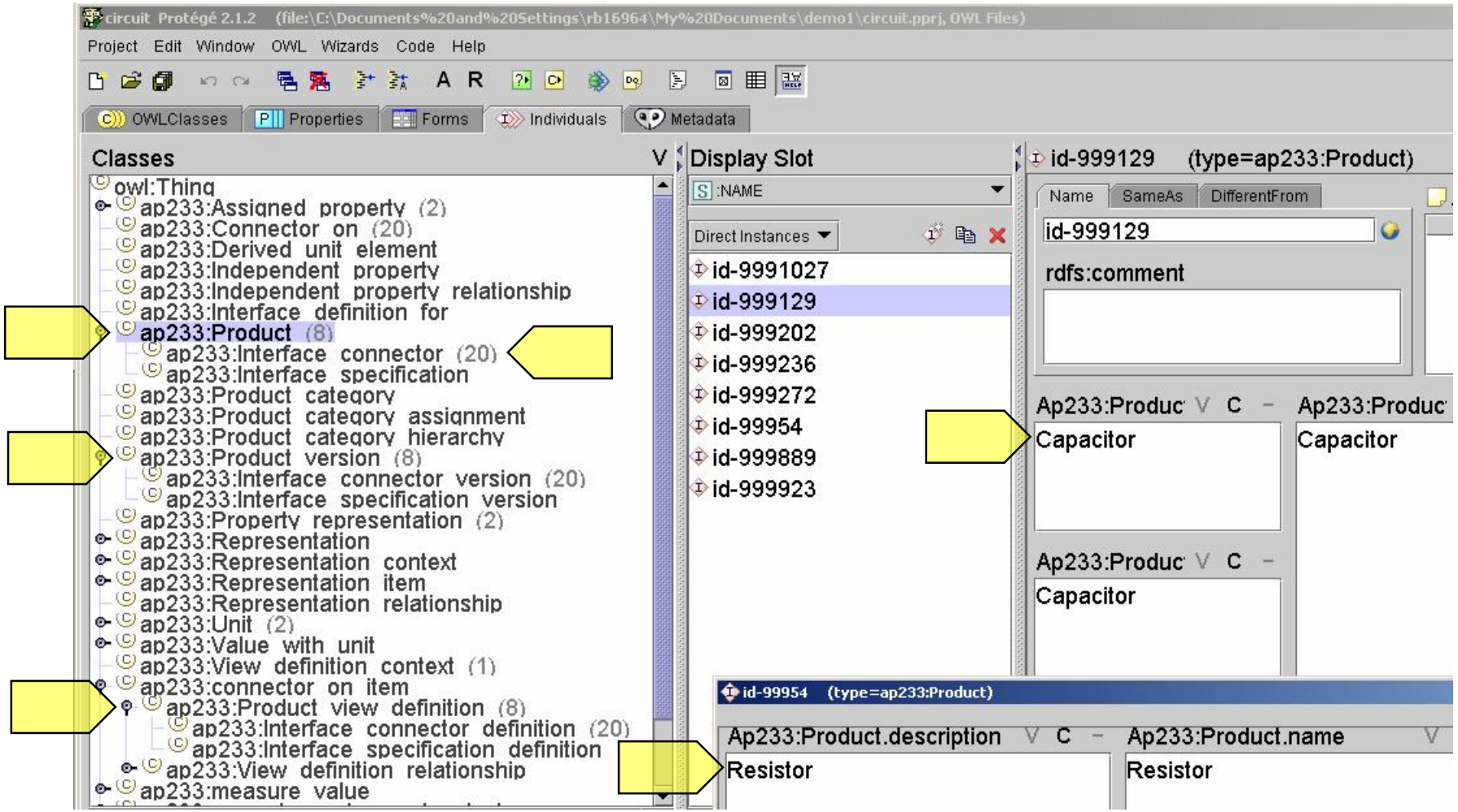
Transformation Flow



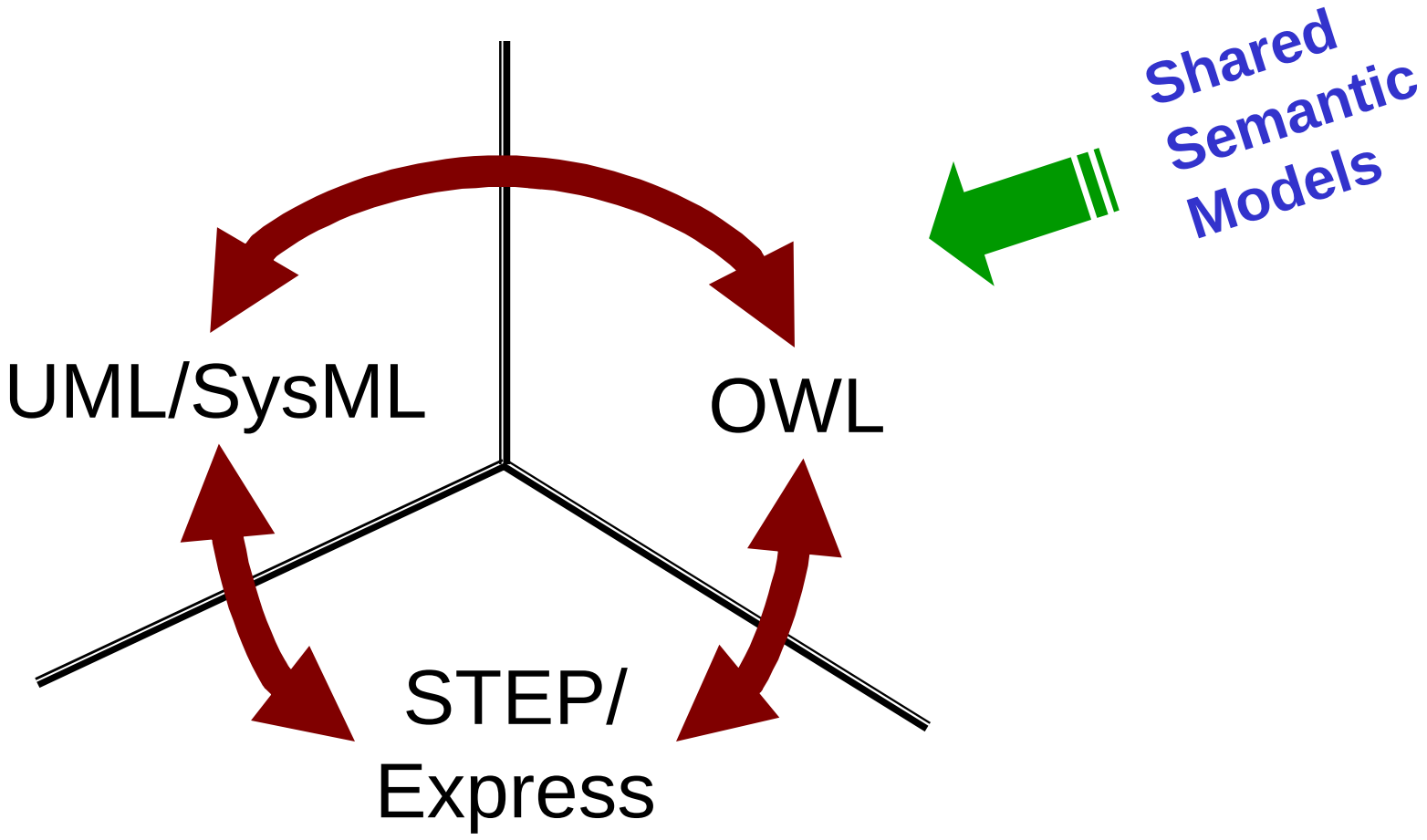
STEP Products and Connectors



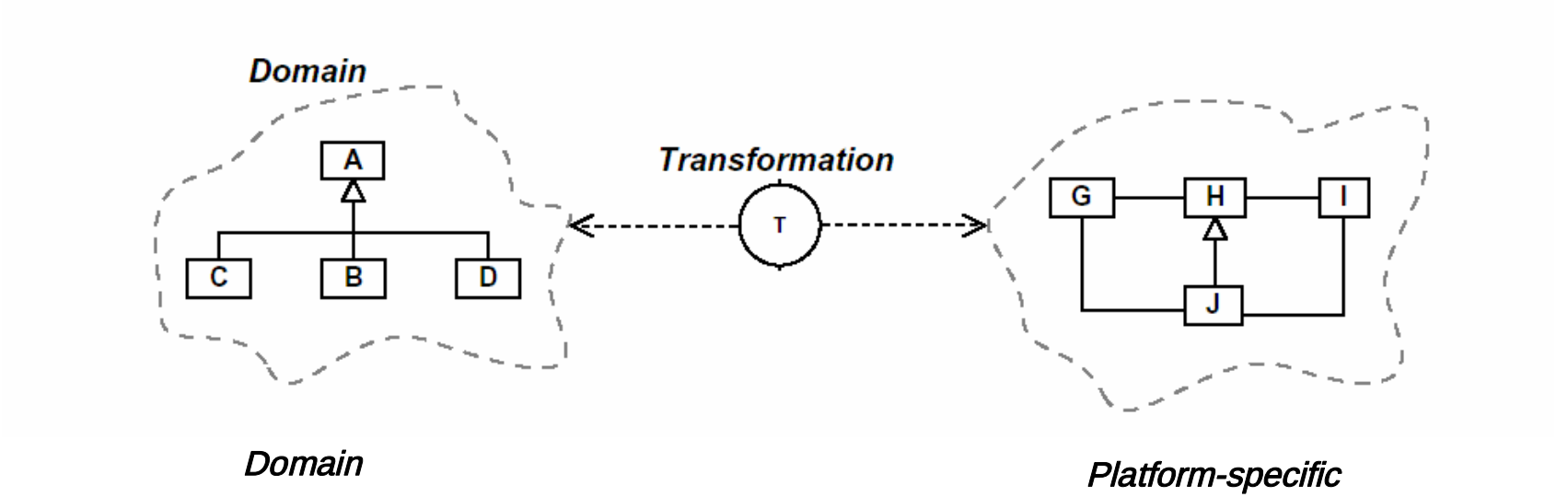
Result of Mapping



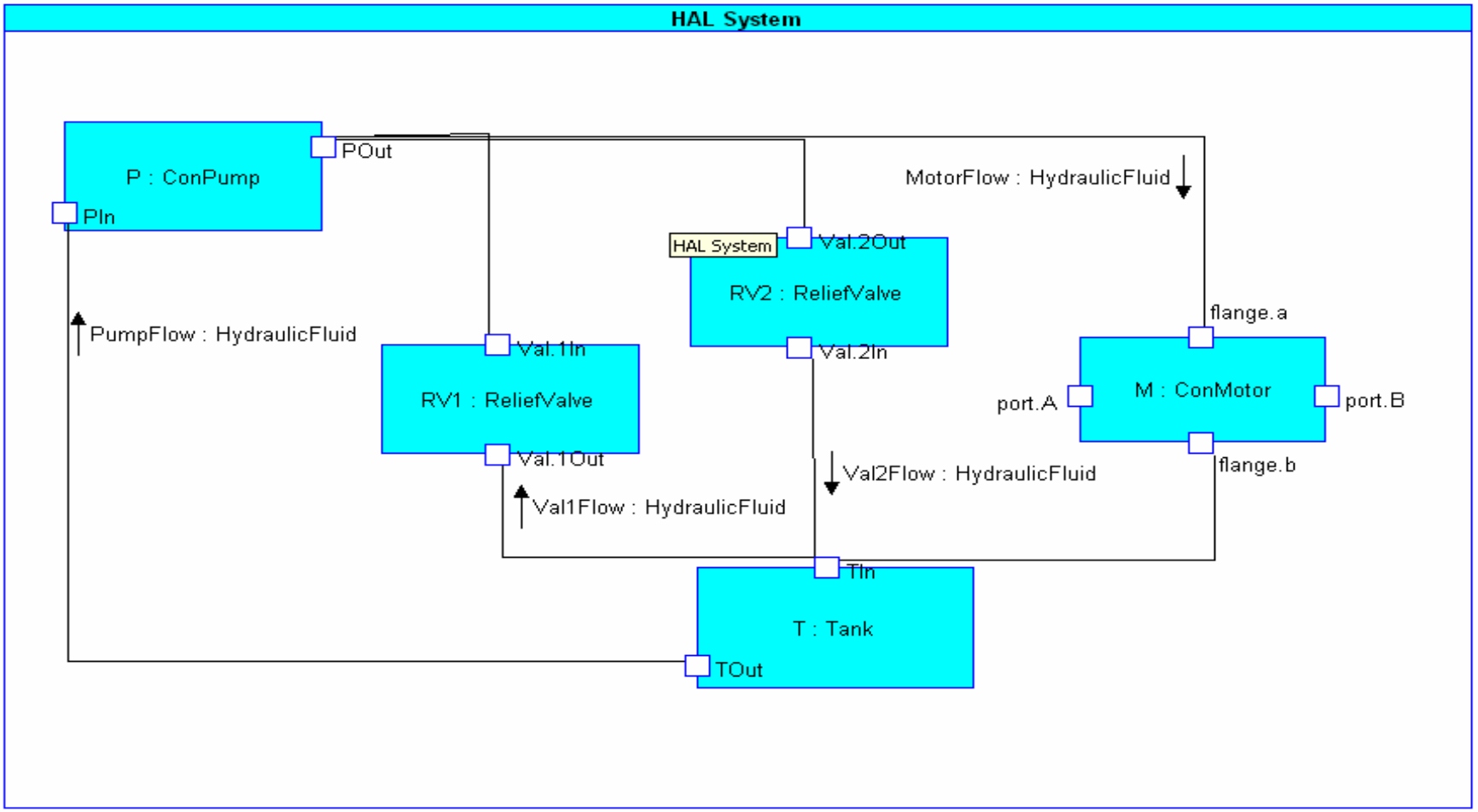
Neutral Modeling Infrastructure



Model-to-Model Transformation



HAL System Structure Diagram



Georgia Institute of Technology

Modeling Continuous Dynamics in SysML

- UML/SysML lacks any native representation for equations of continuous dynamics
- Georgia Tech "Reference Model" approach links system-level models in SysML to library components that carry internal equations
- SysML blocks can also provide a complete native representation of the equations themselves, with or without causality
- SysML can provide a neutral representation for translation between tools, to allow migration and domain-specific choices for solvers, model editors, design and analysis tools

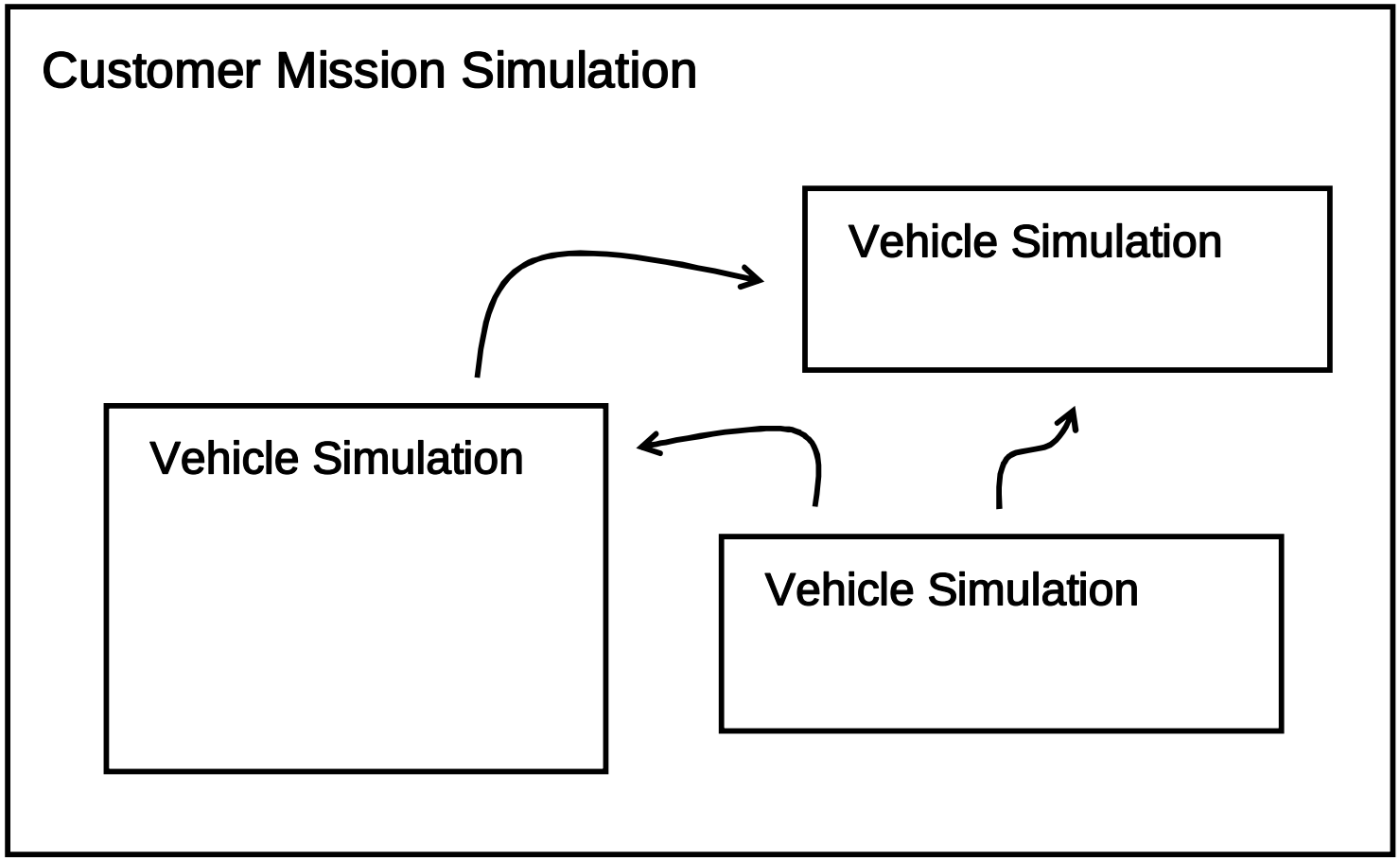
Further Work with Structure Models

- **Multi-agent simulation of complex adaptive systems**
- **Constraint definition and constraint-based programming**
- **A form of mereology as a starting point for core ontology**

System structure models for agents

- **“Systems thinking” is a hallmark of both complex adaptive systems and “system of systems” engineering**
- **Properties and functions at emergent levels is a persistent, common theme**
  - **Many new engineering applications are increasingly recognized as complex adaptive systems**
  - **Optimization criteria force attention to global vs. local levels**
- **Binding of components into system roles is a fundamental abstraction for a “chemistry of composition” by which larger scale systems (including multi-level agents) are built**
  - **Goal of multi-agent simulation**
  - **Architecture for large-scale reuse**
  - **Agent life-cycle model for building the structure of an agent (and its behavior) over its lifetime**

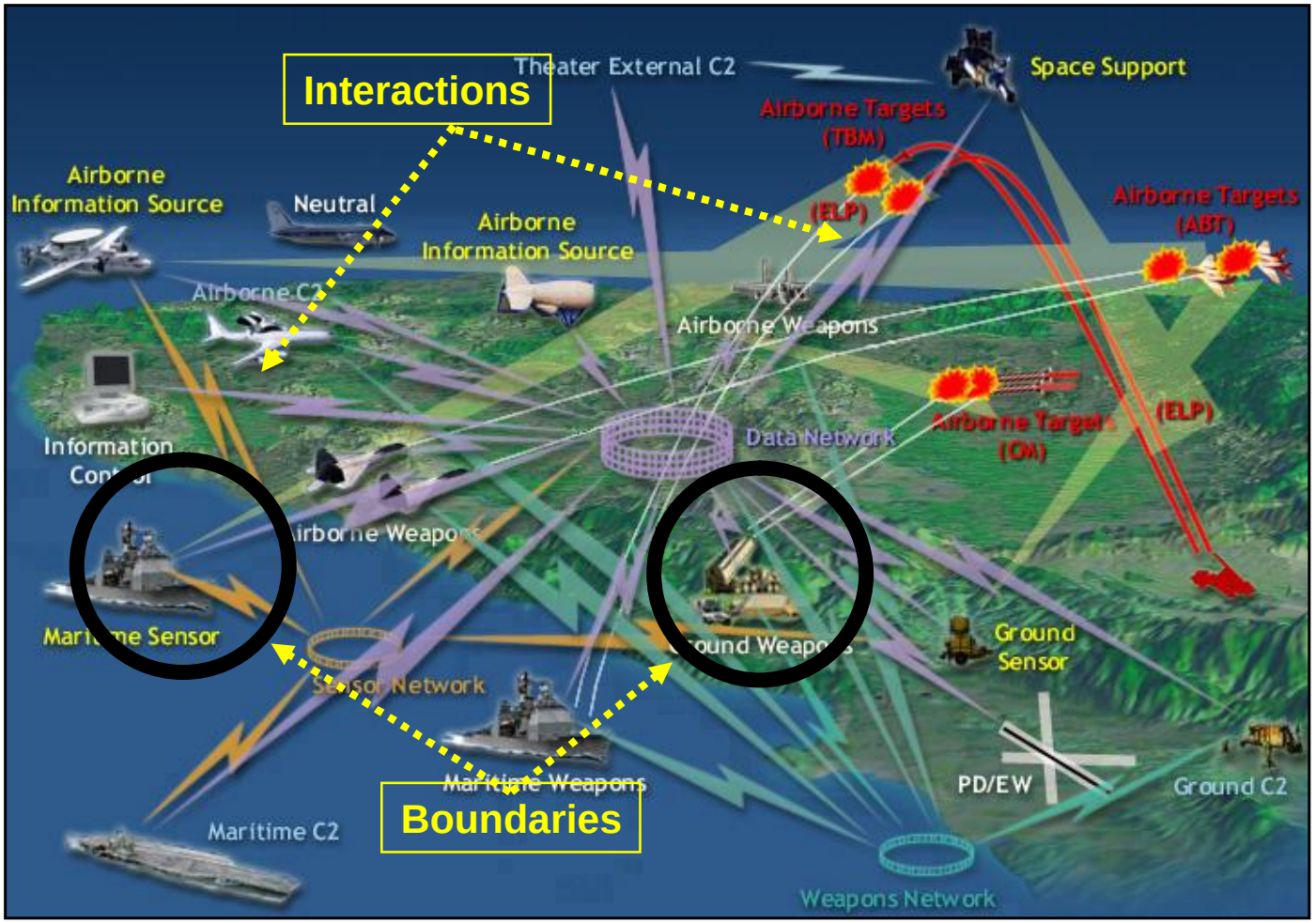
Engineering use of agent models



Multi-level Systems Engineering

- Simulation-based design and optimization of working elements based on field scenarios
- Analysis for robustness, efficiency, adaptability, reusability, …
- Hierarchically nested based on level of interest
  - External customer processes
  - Products and missions
  - Design & internal operations
- Multiple scales in time & space

System-of-Systems

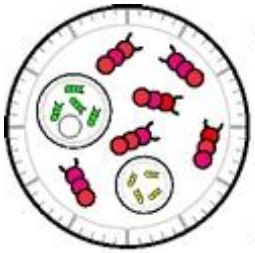


Modeling Needed to Manage System Complexity

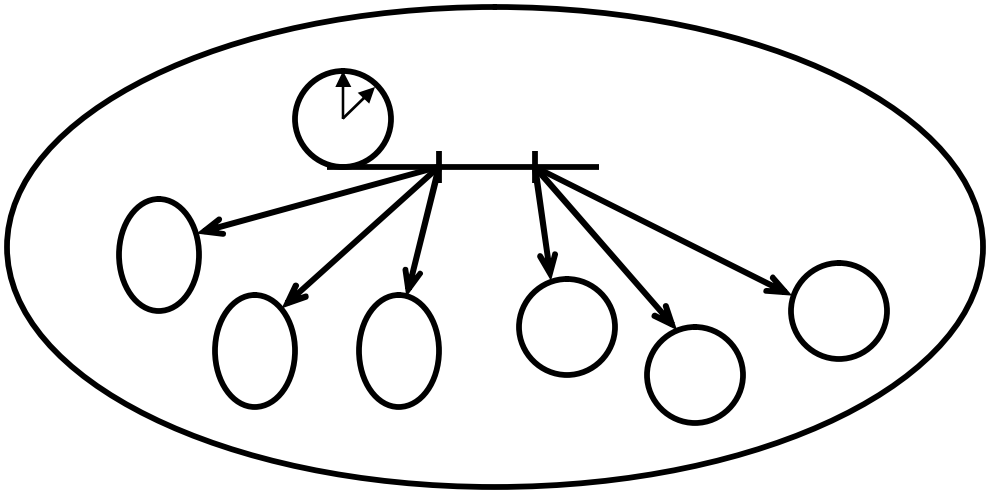
Swarm design goals

- Conceptual framework for agent models
- Programming support for building agent simulations
- Experimenter support for running simulations
- Nucleus for a community of agent modelers

Santa Fe Institute



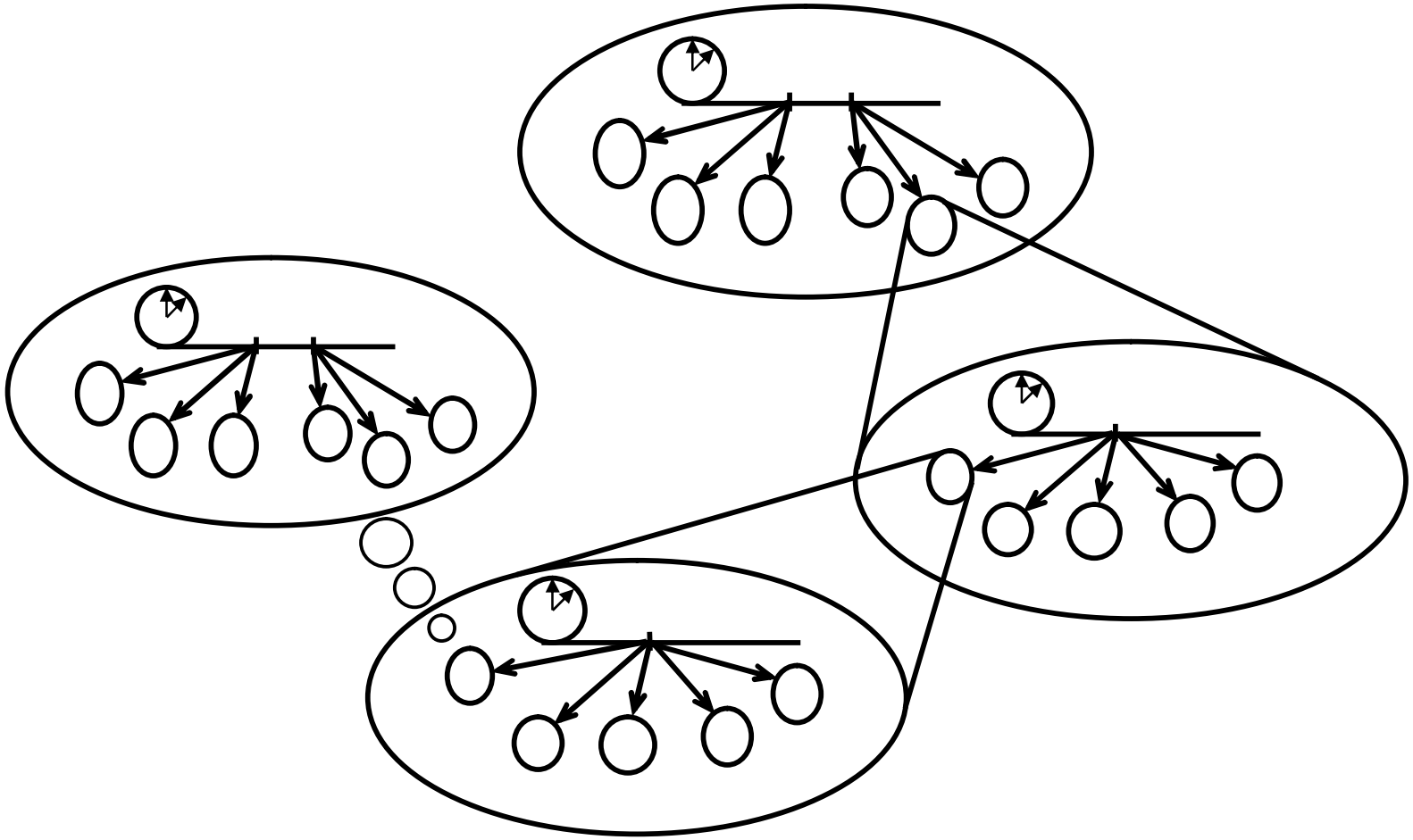
Original Swarm Structure



A swarm is:

- **A collection of objects**
- **A schedule of actions over those agents**

Hierarchical and Reflective Swarms



Swarm conceptual framework

- Agents as objects
- Agent behavior driven by schedules against objects (discrete-event actions)
- Composing behavior by mixing and merging multiple schedules with randomization and partial orders
- Swarms (collections of objects and activity) to express emergent levels

Swarm Demo

Extension for agent life cycles

A swarm is:

- **A collection of objects**
- **A schedule of actions over those agents**
- ⇨ **A schema that controls the development and behavior of the swarm over its entire lifetime**

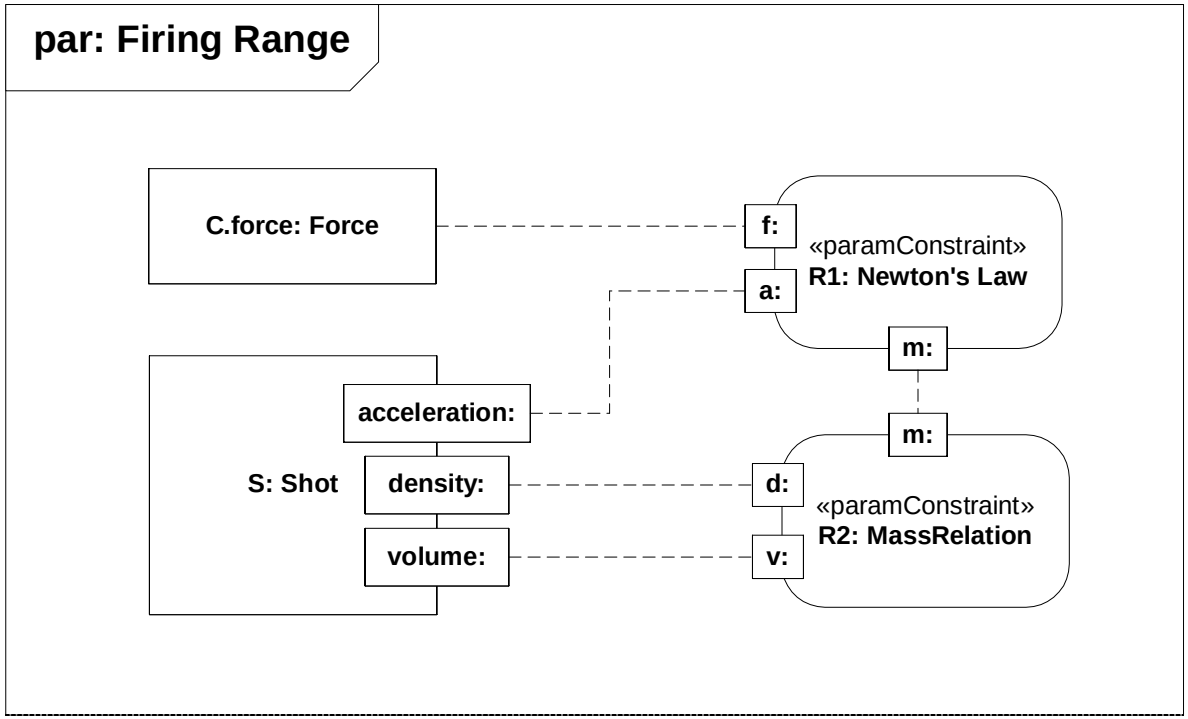
Self-constructing swarms

- Starting from an initial, minimal structure and internal schema, let the swarm itself control the creation of all internal structure and the behavior it enables
- Similar to a process of biological development
- Initial schema serves as an internal “genetic code” that enables agents to share blueprints for component construction and binding, including transfers across independent lifetimes
- Behavior model to express cognition, learning, organization, growth and evolution

Further Work with Structure Models

- **Multi-agent simulation of complex adaptive systems**
- ⇨ **Constraint definition and constraint-based programming**
- **A form of mereology as a starting point for a top-level ontology**

SysML constraint example

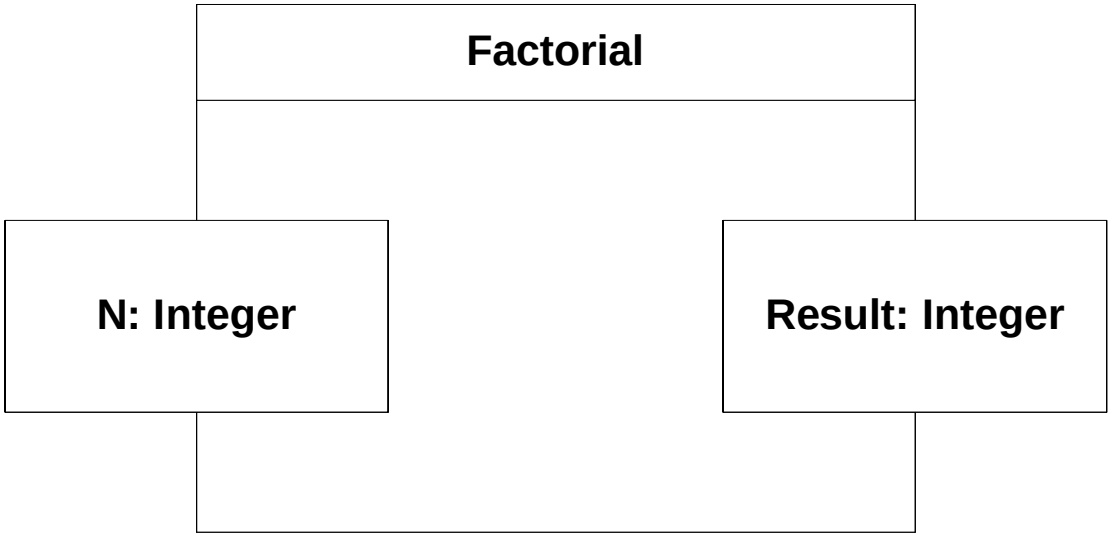


Functional programming example

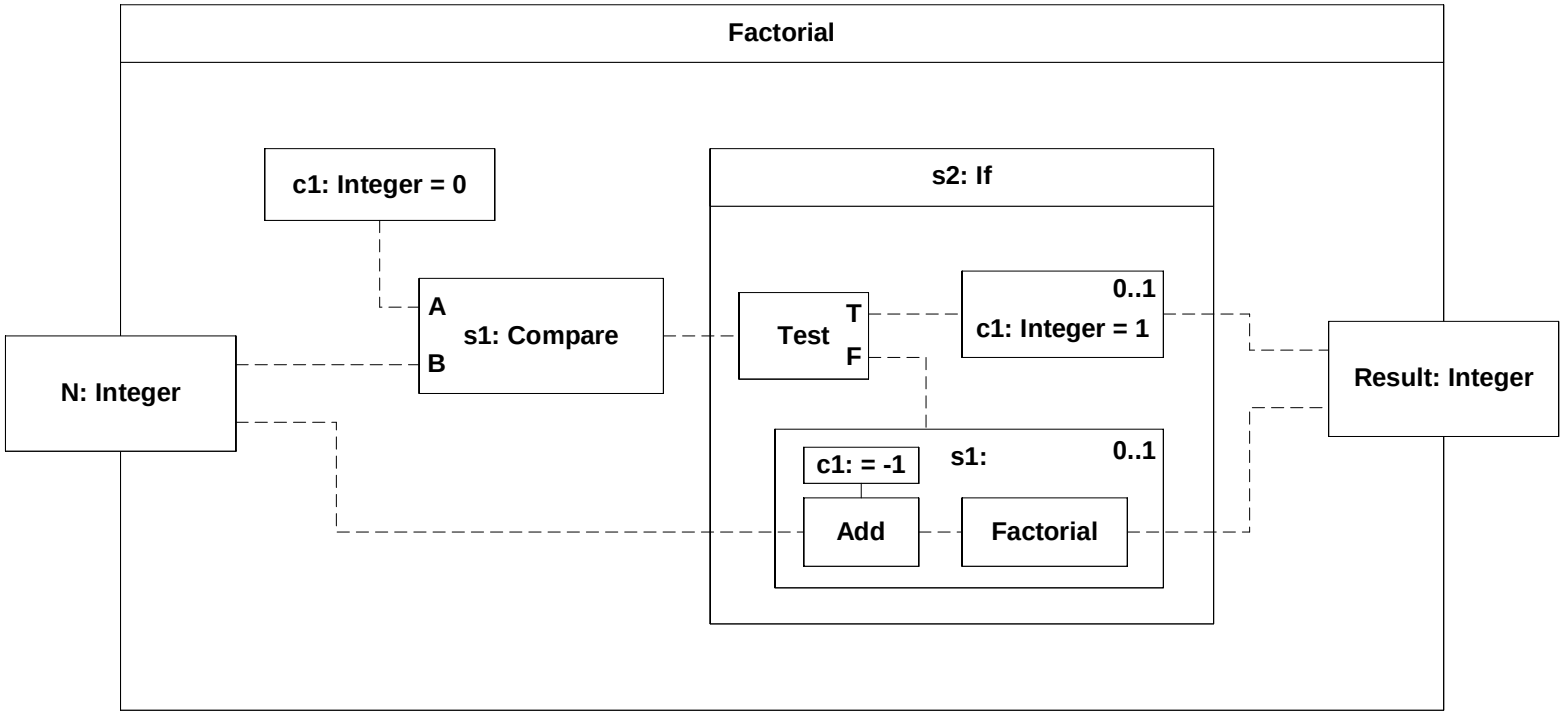
Standard tutorial example, in Oz syntax

```
declare
fun {Fact N}
   if N==0 then 1 else N*{Fact N-1} end
end
```

Black-box view of function



White-box view of function



Further Work with Structure Models

- Multi-agent simulation of complex adaptive systems
- Constraint definition and constraint-based programming
- ⇨ A form of mereology as a starting point for a top-level ontology

UML vs. OWL

4. In UML but not OWL

—

4.2 Complex objects

UML supports various flavors of the part-of relationship between classes…

Composite structures are runtime instances of classes collaborating via connections…

These diagrams extend the capabilities of class diagrams, which do not specify how internal parts are organized within a containing class…

Ports and Connectors model how internal instances are to be organized…

Collaboration provides constructs for modeling roles played by connectors…

—

Although it is recognized that there is a need for facilities to model mereotopological relationships in ontologies, there does not seem to be sufficient agreement on the scope and semantics of existing models for inclusion of specific mereotopological modeling features into the ODM at this stage.

Source: OWL Full and UML 2.0 Compared, Version 2.4, 12 March 2004, ontology/2004-03-01

Further Work with Structure Models

- **Multi-agent simulation of complex adaptive systems**
- **Constraint definition and constraint-based programming**
- **A form of mereology as a starting point for a top-level ontology**

- **General-purpose representation for design structure models (static or dynamic, with or without simulation)**