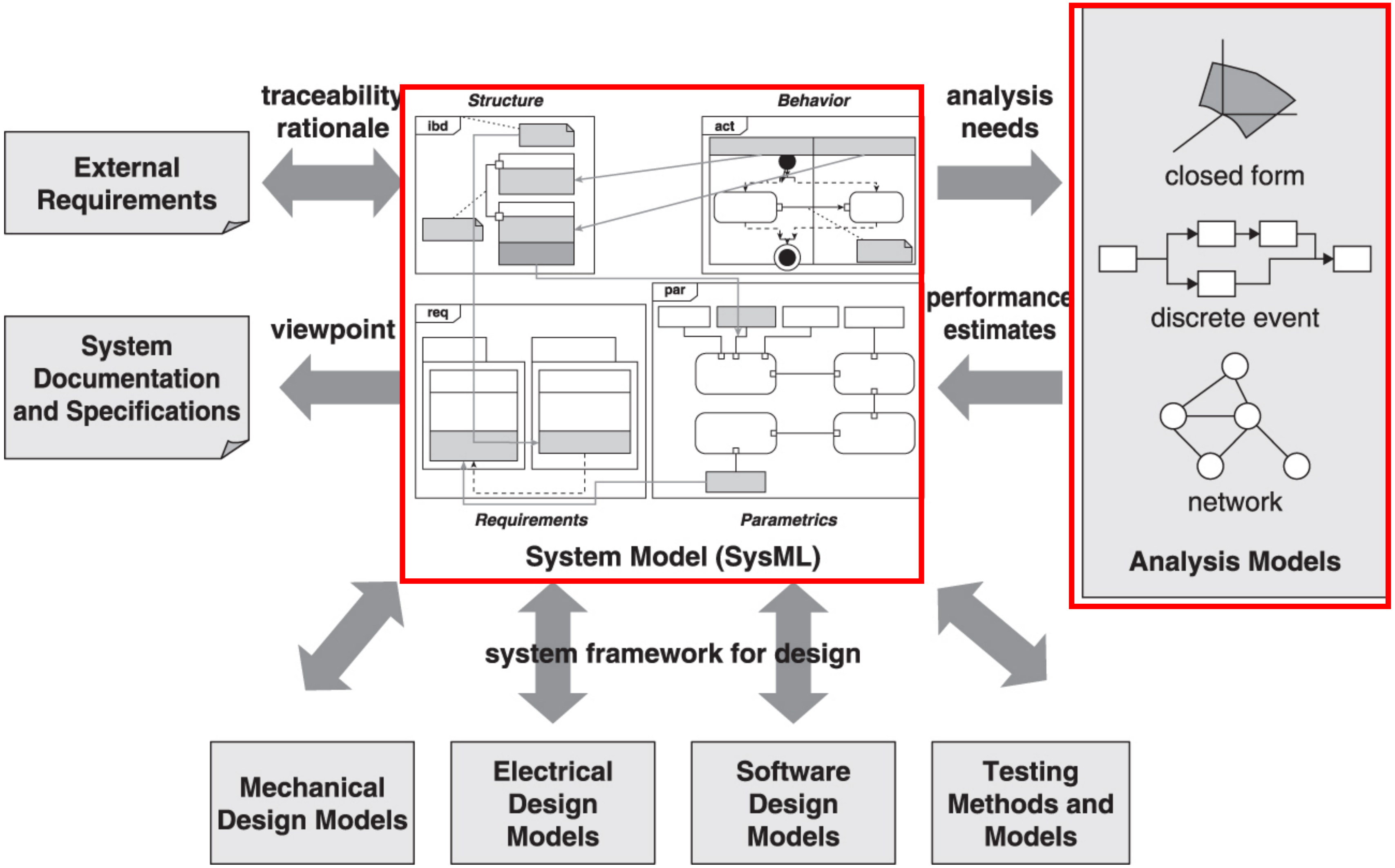
OMG SysPhs: Integrating SysML, Simulink, Modelica and FMI

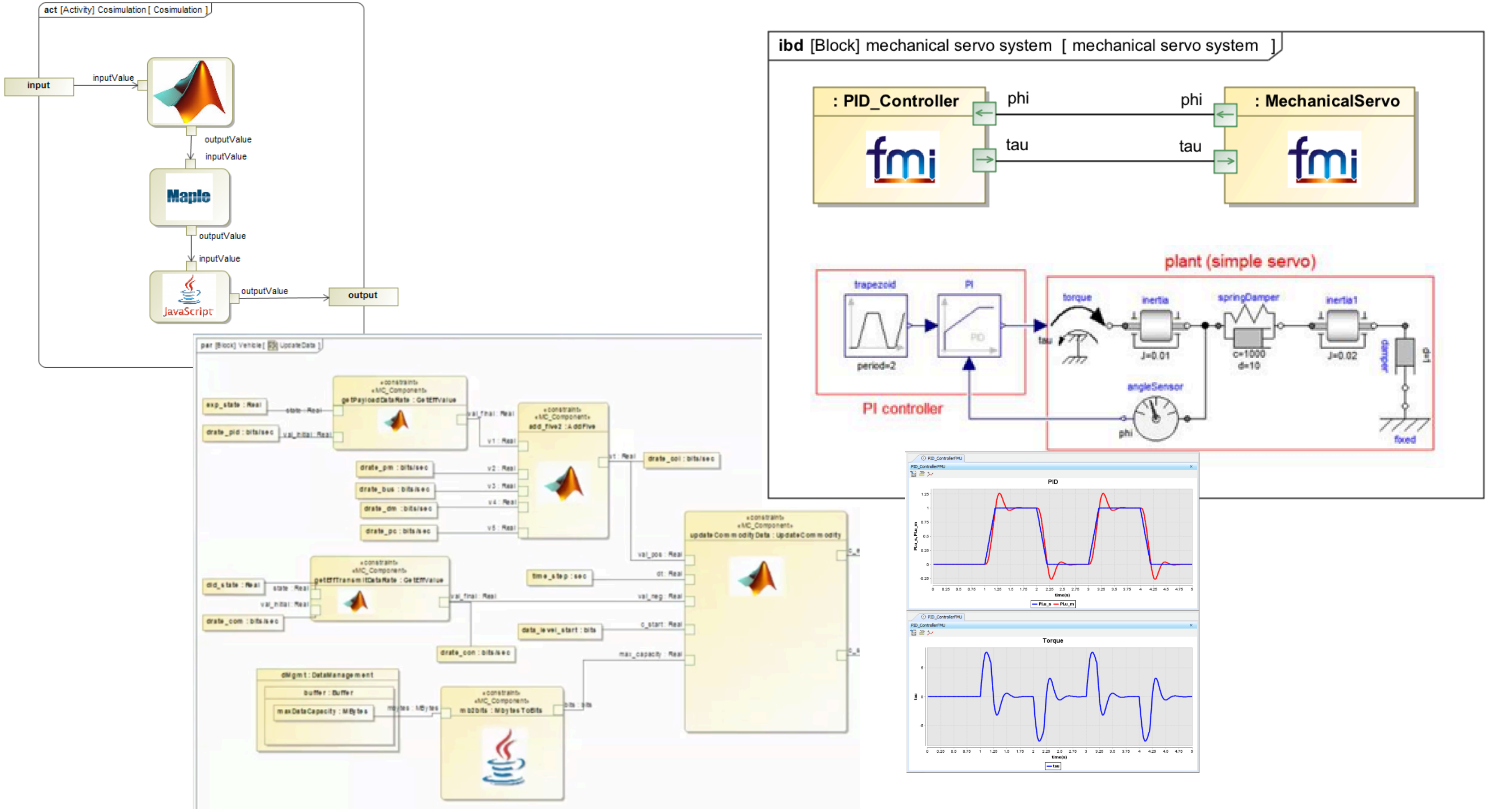
Nerijus Jankevicius
CATIA | No Magic

INCOSE IW, Torrance, Jan 27, 2020

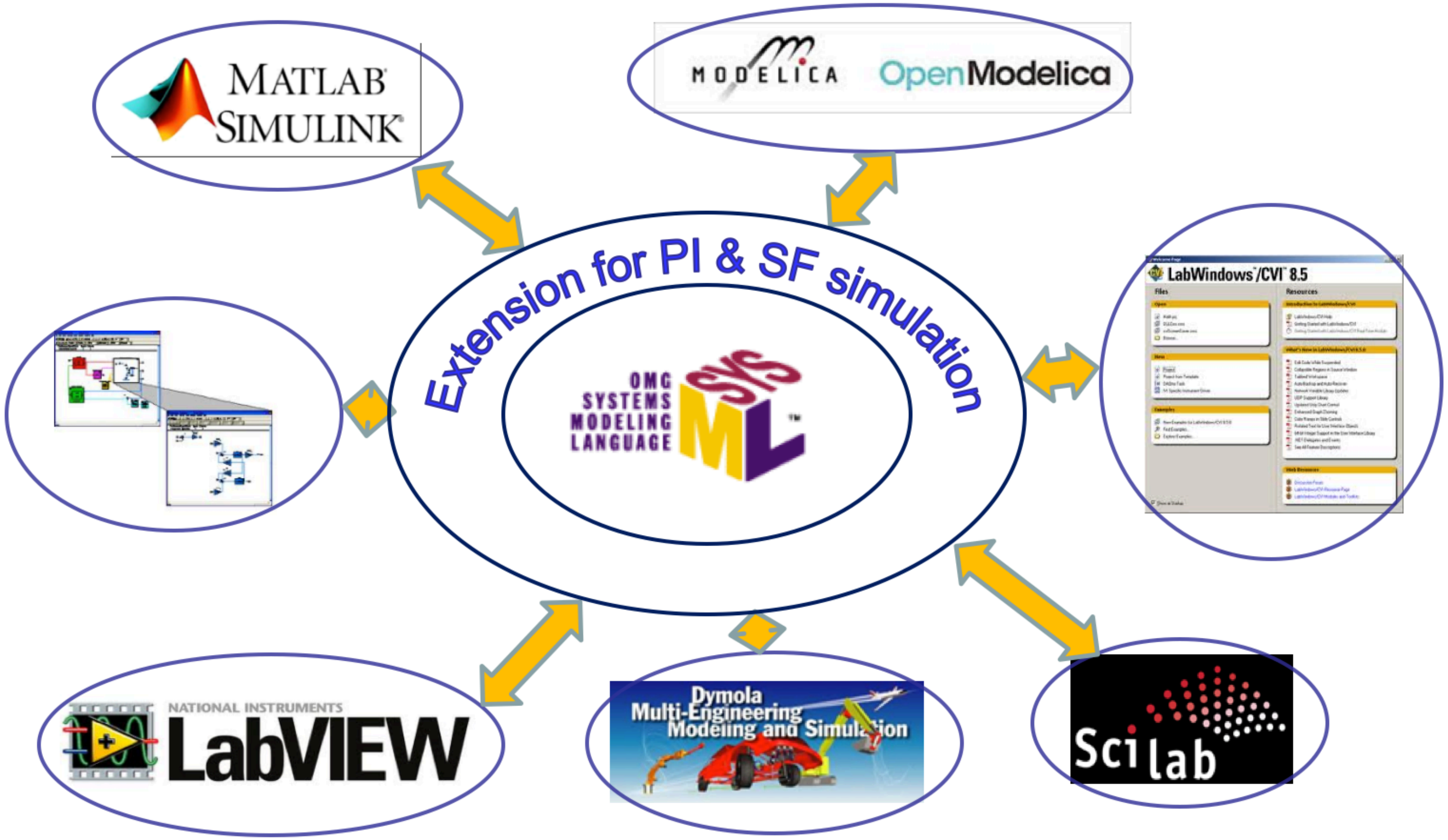
System Model as an Integration Framework



SysML as co-simulation environment



Reduce and standardize mappings



New OMG standard:

SysML Extension for Physical Interaction and Signal Flow Simulation (SysPhS)

Unified Physics

Domain	**Flowing Substance**	**Flow rate**	**Potential to flow**
Electrical	Charge	Current	Voltage
Hydraulic	Volume	Volumetric flow rate	Pressure
Rotational	Angular momentum	Torque	Angular velocity
Translational	Linear momentum	Force	Velocity
Thermal	Entropy	Entropy flow	Temperature

flow rate = amount of substance/time
flow rate * potential = energy / time = power

The Standard : SysPhs

- SysPhS - *https://www.omg.org/spec/SysPhS/1.0*
  - SysML mapping to Simulink and Modelica
  - SysPhS profile
  - SysPhS library

OBJECT MANAGEMENT GROUP®

SysML Extension for Physical Interaction and Signal Flow Simulation

Version 1.0

OMG Document Number: formal/18-05-03

Release Date: June 2018

Simulation profile

7.2 Simulation profile

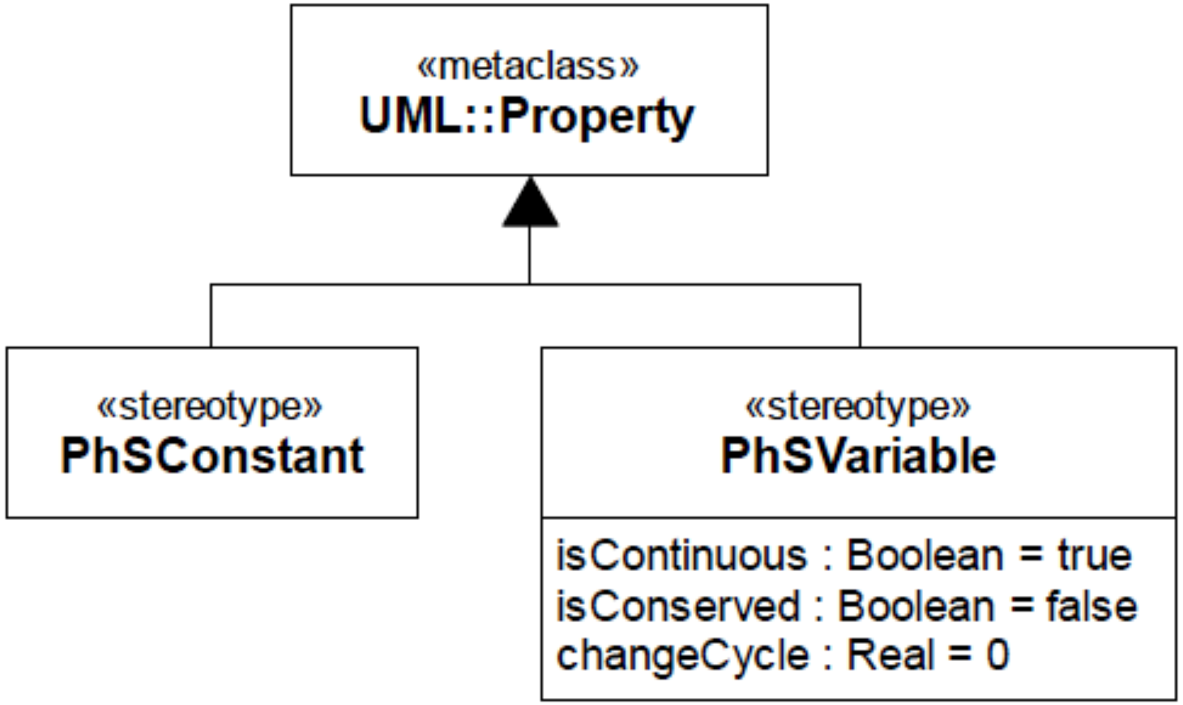


Figure 1: Simulation stereotypes

Modelica vs Simulink

- Modelica
  - Language is better suited for physical modeling (plant)
  - Object oriented approach for modeling physical components (mechanical, electrical, etc.)
  - Causal and A-Causal semantics (equations)
  - Open standard (of the textual language)
  - Multi tool support (although Dymola is dominant)
  - Tool vendor independent

- Simulink
  - Language is well-suited for control algorithms
  - Transformational semantics of signals and signal processing
  - Causal semantics (inputs -> outputs)
  - Well integrated into the “MATLAB universe”
  - Widely used in industry (standard de-facto)
  - Many existing tool integrations
  - Code generation to C/C++/VHDL/Verilog

Platform profile

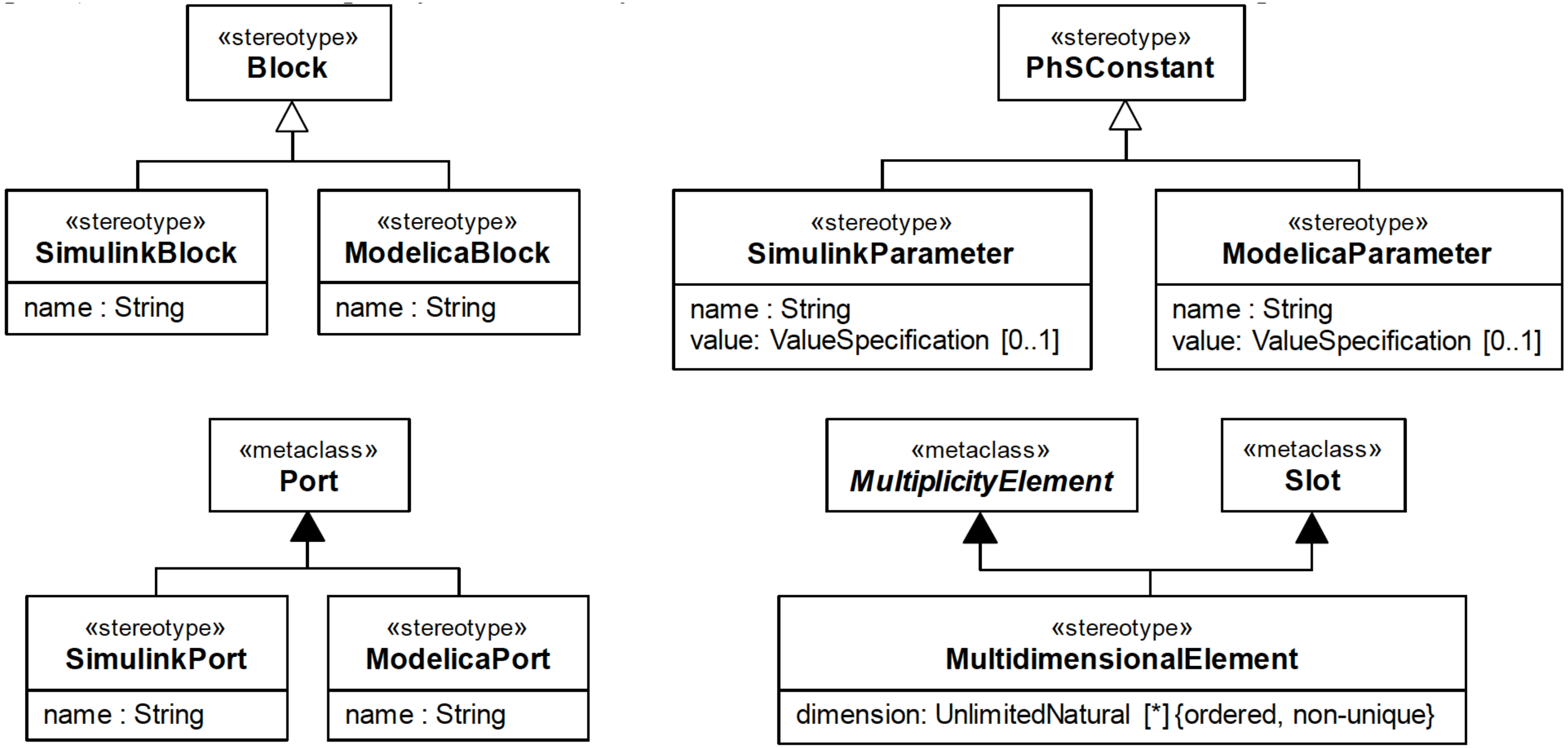


Figure 33: Simulation platform stereotypes

Specification examples

SysML	Modelica	Simulink	Simscape
Port typed by block with an in flow property stereotyped by a non-conserved PhSVariable and typed by Real, Integer, Boolean or one of their specializations (signal flow)	Component typed by an equivalent data type	Inport	Input variable
Port typed by block with an out flow property stereotyped by a non-conserved PhSVariable and typed by Real, Integer, Boolean or one of their specializations (signal flow)	Component typed by an equivalent data type	Outport	Output variable
Port typed by block with an inout flow property typed by block (indirectly) specializing ConservedQuantityKind (physical interaction)	Component typed by connector	Connection port	Node typed by domain
Block (indirectly) specializing ConservedQuantityKind (physical interaction)	Connector	N/A	Domain
PhSVariables on blocks (indirectly) specializing ConservedQuantityKind (physical interaction)	Components of connector	N/A	Variables of domain

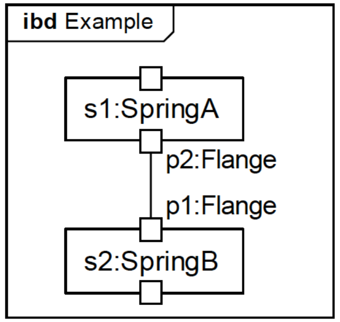


Figure 24: Connectors in SysML

10.8.3 Modelica modeling

SysML connectors correspond to Modelica connect equations, which link components typed by Modelica connectors. This depends on the correspondence between SysML port types and Modelica connectors (see 10.7.8).

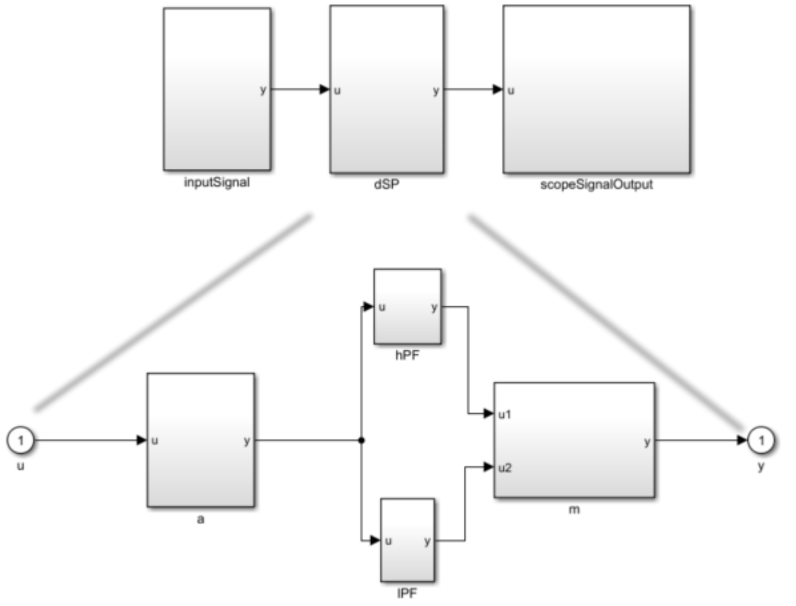
The following Modelica code corresponds to Figure 24. It has a model *Example* with two components *s1* and *s2* of types *SpringA* and *SpringB*, respectively. The models *SpringA* and *SpringB* have two components *p1* and *p2* of type *Flange*, defined similarly to *Spring* in Subclause 10.7.8. *Model* contains a connect equation linking component *p2* of *s1* to component *p1* of *s2*.

```
model Example
  SpringA s1;
  SpringB s2;
equation
  connect(s1.p2, s2.p1);
end Example;
```

The implementation: Cameo Systems Modeler 19.0 SP3

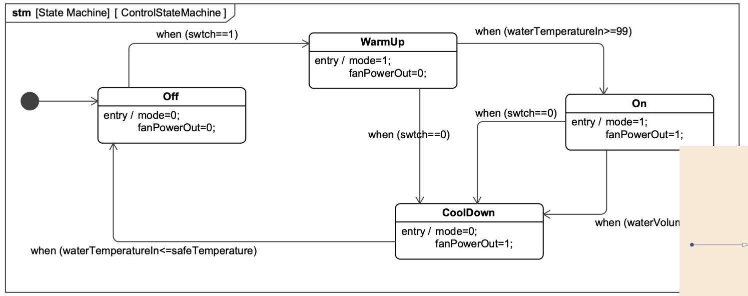
Simulink export

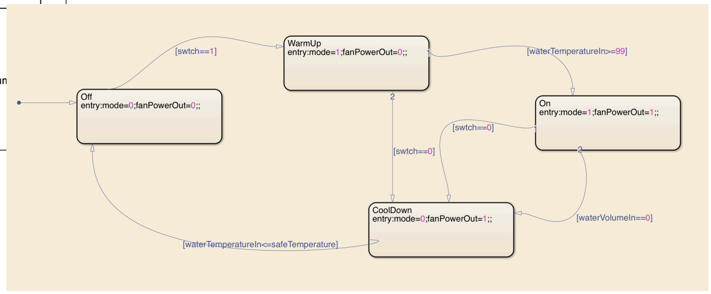
- BDD and IBD -> Simulink blocks
- Statemachines -> Stateflow
- Parametrics -> S-functions or Simscape (acausal)
- Diagram layout
- Black-box and/or full implementation



Modelica export

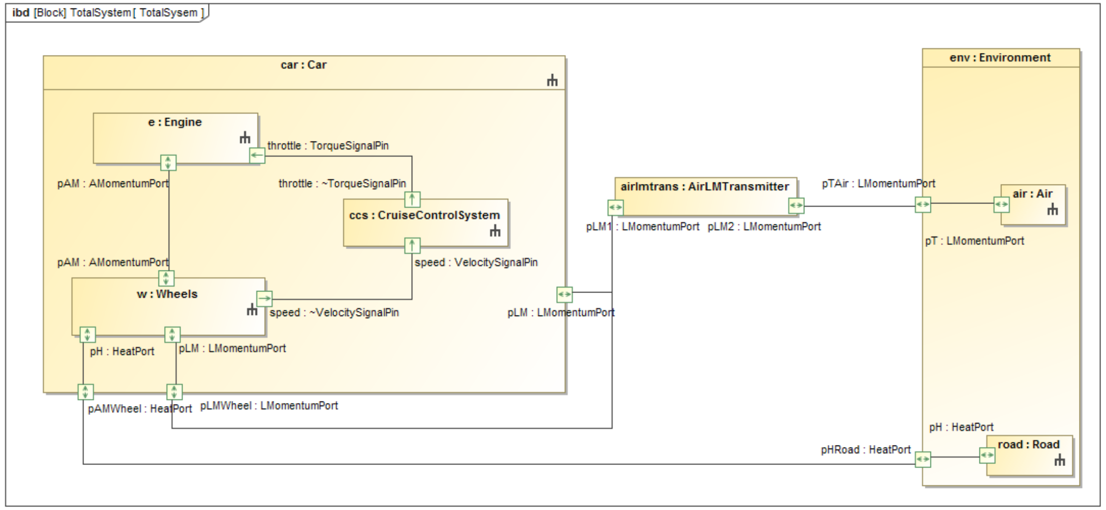
- BDD, IBD, Statemachines, Parametrics
- Variables and parameters
- Time derivatives (der(x))
- Dymola diagram layout annotations
- Standard Modelica connectors
- Units and quantity kinds





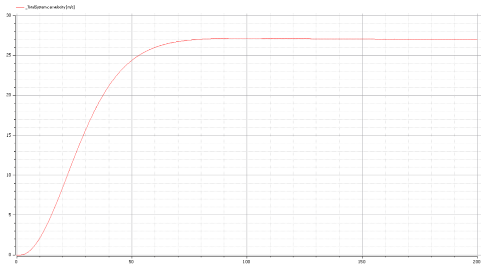
SysML to Simulink/Modelica

System architecture and implementation

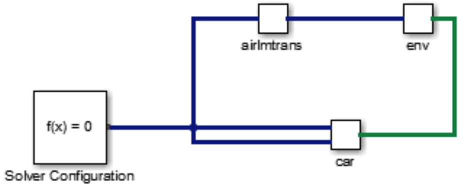


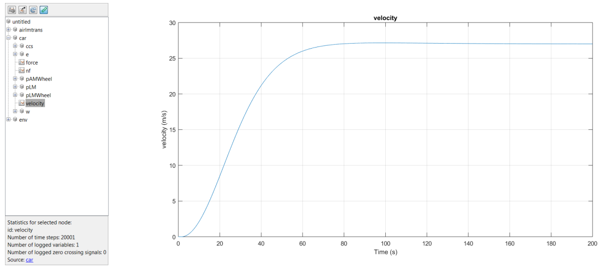
Modelica

```
model TotalSystem
    Car car(g = g, slope = slope, w.velocity.start = 0.0,
w.velocity.fixed = true, e.torque.start = 0.0, e.torque.fixed =
true);
    parameter Real g(start = 9.81, fixed = true);
    parameter Real airdensity(start = 1.2, fixed = true);
    parameter Real slope(start = 0.0, fixed = true);
    Environment env;
    AirLMTransmitter airlmtrans(crossSectionalArea = frontArea,
airdensity = airdensity);
    parameter Surface frontArea(start = 10.0, fixed = true);
equation
    connect(car.pLMWheel, car.pLM);
    connect(env.pHRoad, car.pAMWheel);
    connect(car.pLM, airlmtrans.pLM1);
    connect(airlmtrans.pLM2, env.pTAir);
end TotalSystem;
```



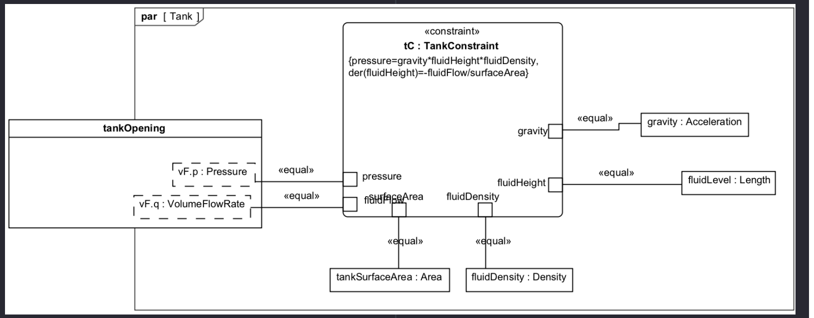
Simulink/Simscape

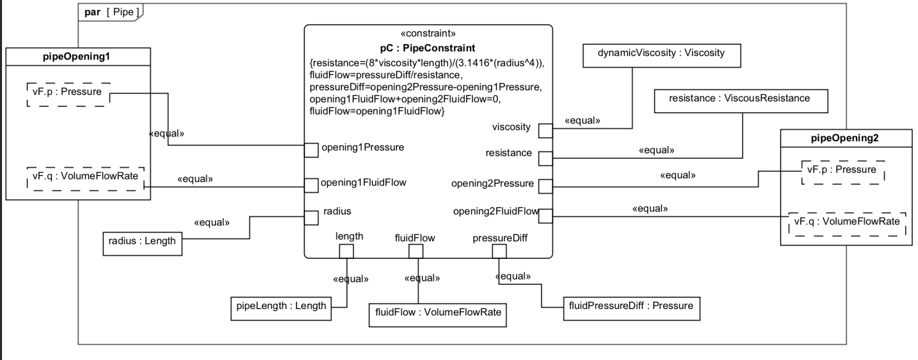


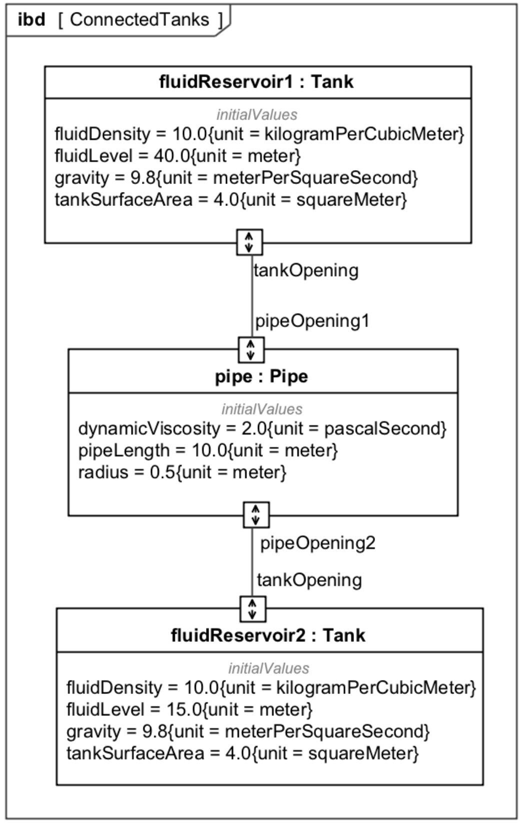


SysML to Modelica example

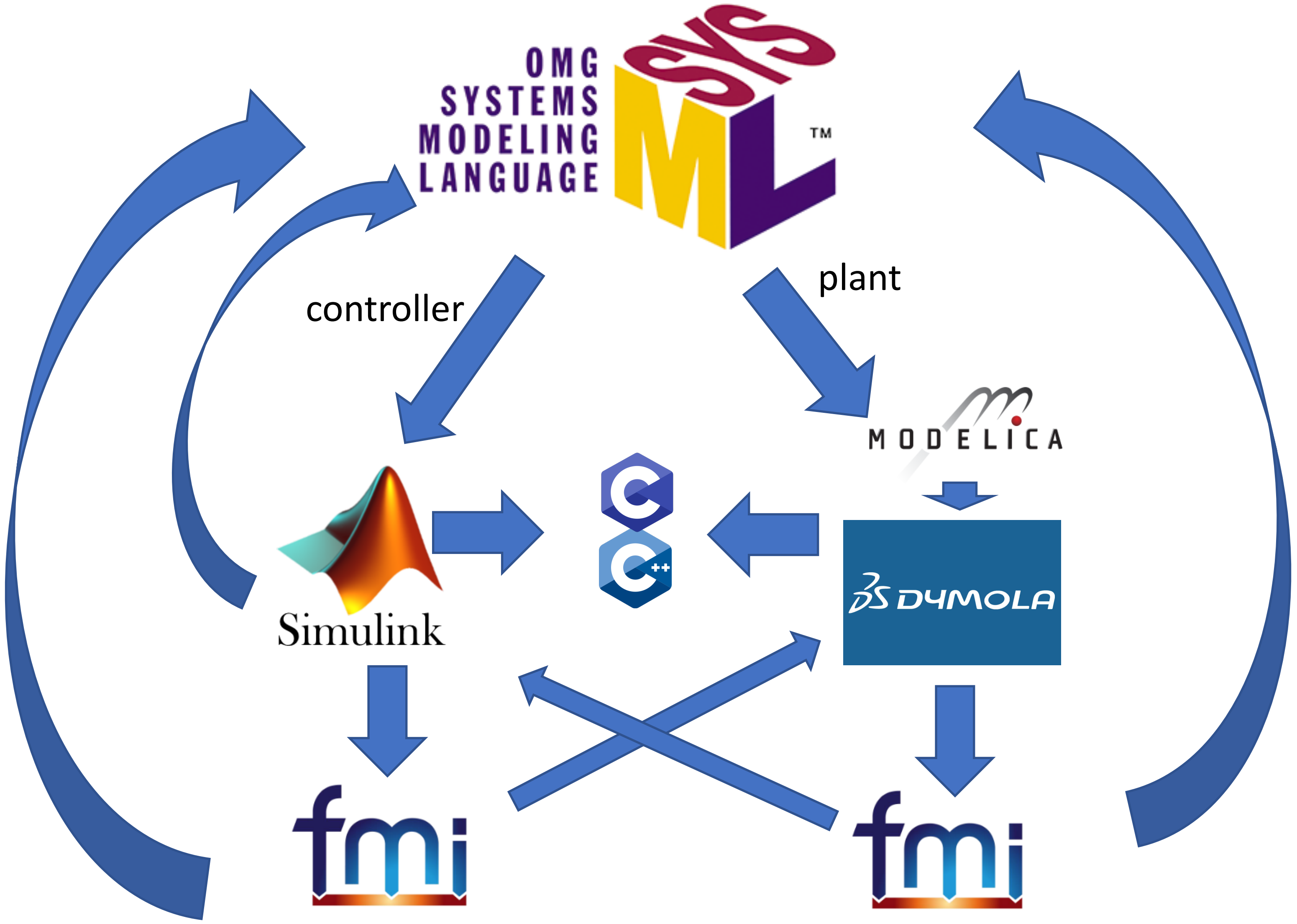
```
model ConnectedTanksModel
  ConnectedTanks _ConnectedTanks;
  model ConnectedTanks
    Pipe pipe(pipeLength.start=10.0,pipeLength.fixed=true,radius.start=0.5,radius.fixed=true,dynamicViscosity.start=2.
    Tank fluidReservoir1(fluidLevel.start=40.0,fluidLevel.fixed=true,gravity.start=9.8,gravity.fixed=true,tankSurfaceAr
    Tank fluidReservoir2(fluidLevel.start=15.0,fluidLevel.fixed=true,gravity.start=9.8,gravity.fixed=true,tankSurfaceAr
  equation
    connect(pipe.pipeOpening1,fluidReservoir1.tankOpening);
    connect(fluidReservoir2.tankOpening,pipe.pipeOpening2);
  end ConnectedTanks;
  connector VolumeFlowElement
    flow VolumeFlowRate q;
    Pressure p;
  end VolumeFlowElement;
  type Pressure=Real(unit="Pa");
  type VolumeFlowRate=Real(unit="m³/s");
  model Tank
    VolumeFlowElement tankOpening;
    parameter Area tankSurfaceArea;
    parameter Acceleration gravity;
    parameter Density fluidDensity;
    Length fluidLevel;
  equation
    tankOpening.p=gravity*fluidLevel*fluidDensity;
    der(fluidLevel)=-tankOpening.q/tankSurfaceArea;
  end Tank;
  type Length=Real(unit="m");
  type Density=Real(unit="kg/m³");
  type Acceleration=Real(unit="m/s²");
  type Area=Real(unit="m²");
  model Pipe
    VolumeFlowElement pipeOpening1;
    VolumeFlowElement pipeOpening2;
    VolumeFlowRate fluidFlow;
    Pressure fluidPressureDiff;
    parameter Length pipeLength;
    parameter Length radius;
    parameter Viscosity dynamicViscosity;
    ViscousResistance resistance;
  equation
    resistance=(8*dynamicViscosity*pipeLength)/(3.1416*(radius^4));
    fluidFlow=fluidPressureDiff/resistance;
    fluidPressureDiff=pipeOpening2.p-pipeOpening1.p;
    pipeOpening1.q+pipeOpening2.q=0;
    fluidFlow=pipeOpening1.q;
  end Pipe;
  type ViscousResistance=Real(unit="N·s/m⁵");
  type Viscosity=Real(unit="Pa·s");
end ConnectedTanksModel;
```







Usecases


OMG
SYSTEMS
MODELING
LANGUAGE
SYS ML
TM
controller
plant
MODELICA
Simulink
DYMOLA
fmi
fmi

IF WE™