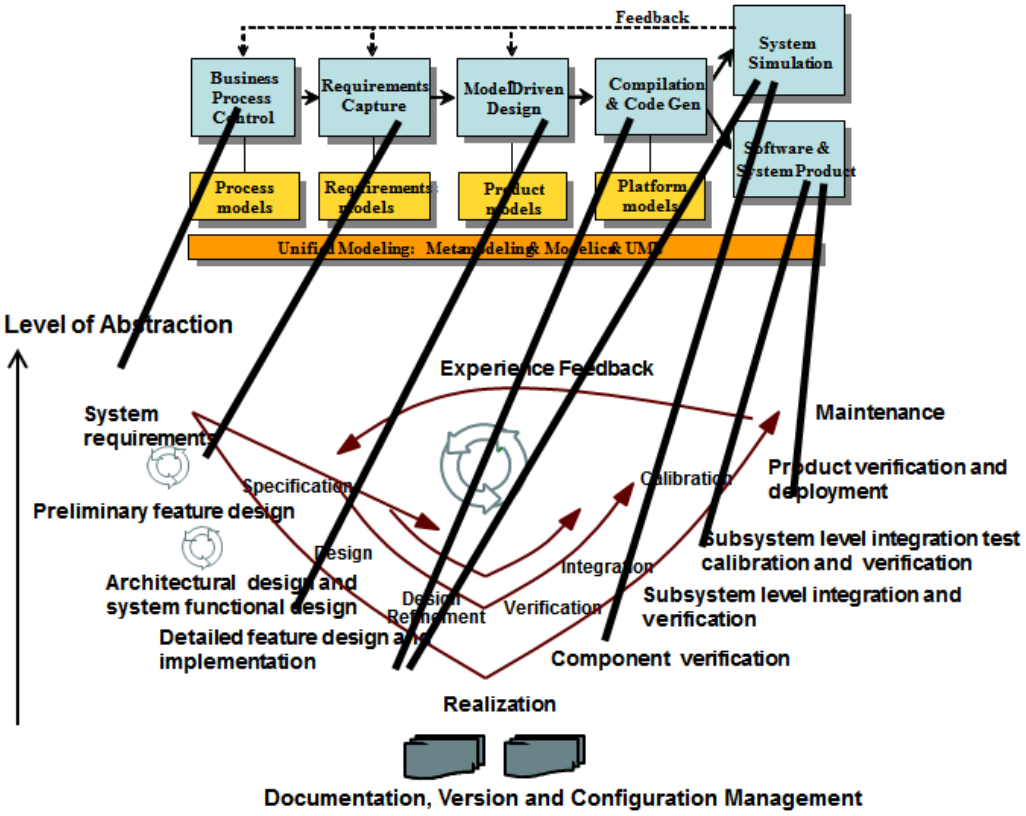
Modeling and Simulation of Cyber-physical Systems including Requirement Formalization using the OpenModelica MBSE Toolkit based on Modelica and partly on UML



Keynote
MESCONF 2016, Munich
October 6, 2016

Peter Fritzson
peter.fritzson@liu.se

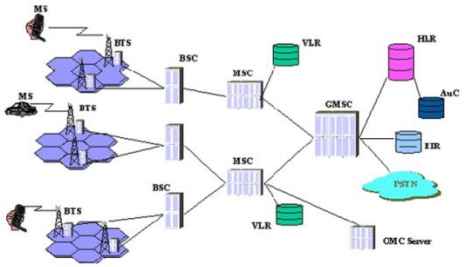
Director of Open Source Modelica Consortium
Vice Chairman of Modelica Association
Professor at Linköping University

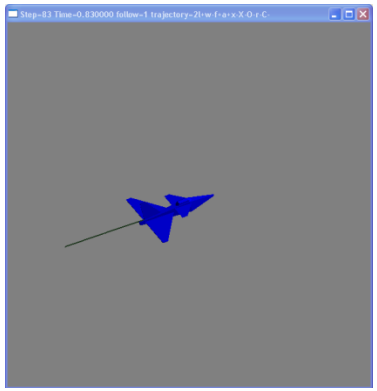
Linköpings universitet

MODELICA

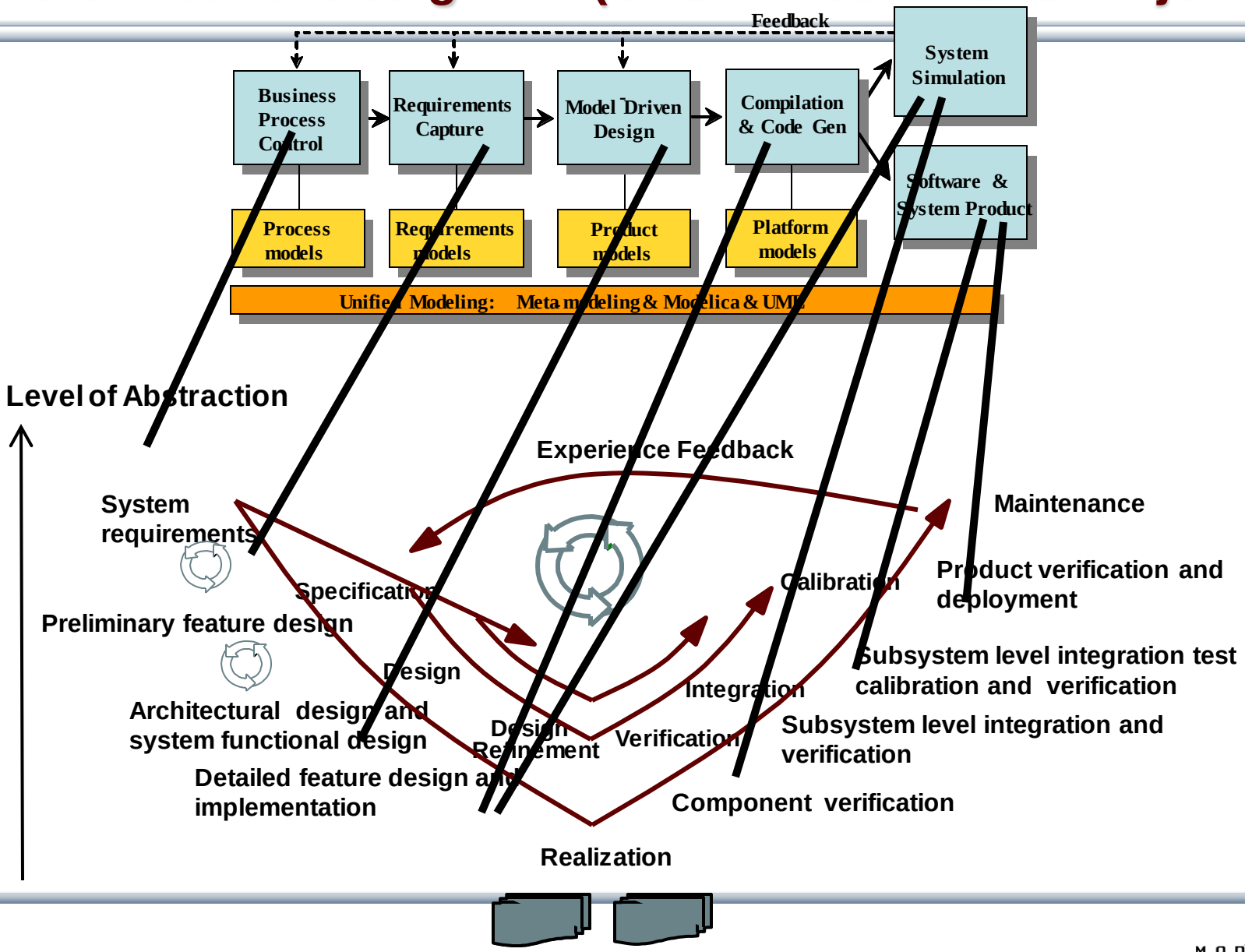
Industrial Challenges for Complex Cyber-Physical System Products of both Software and Hardware

- Increased **Software** Fraction
- **Shorter** Time-to-Market
- Higher demands on effective strategic **decision** making
- **Cyber-Physical** (CPS) – Cyber (software) Physical (hardware) products



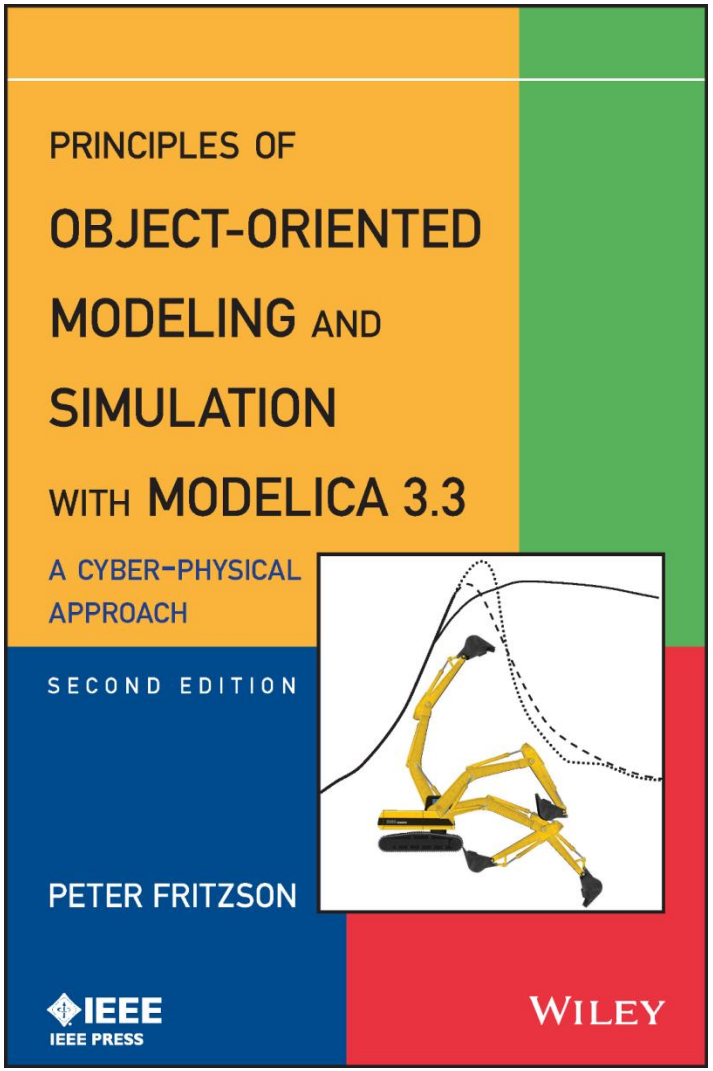


Open Source Model-Based Development Environment Covers Product-Design V – (OPENPROD ITEA2 Project)



MODELICA

Recent Big Modelica Book, 2014 (Warning! Commercial)



Peter Fritzson

Principles of Object Oriented Modeling and Simulation with Modelica 3.3:

A Cyber-Physical Approach

Can be ordered from Wiley or Amazon
Wiley-IEEE Press, 2014, 1250 pages

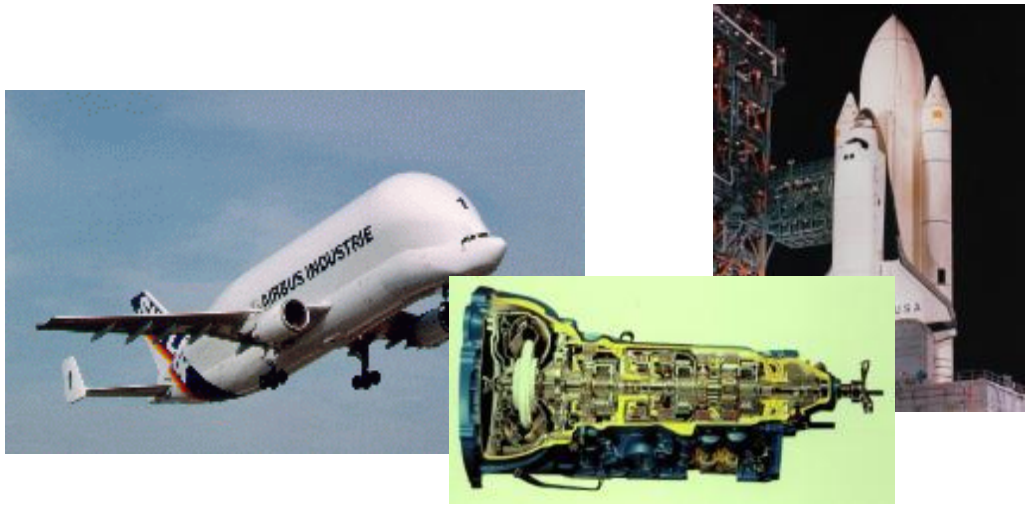
- OpenModelica
  - www.openmodelica.org
- Modelica Association
  - www.modelica.org

Overview of this Talk

- Part I – Introduction to Modelica
- Part II – Introduction to the OpenModelica Open Source MBSE Environment
- Part III – Dynamic verification/testing of formalized requirements vs Models in MBSE
- Part IV – Dynamic debugging of equation-based models
- Part V– New Project OPENCPS

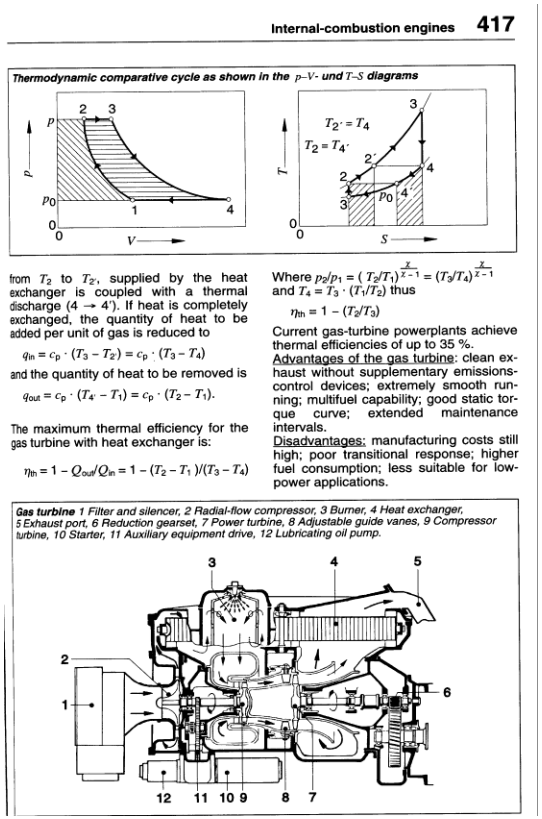
Part I

Introduction to Modelica



Modelica Background: Stored Knowledge

Model knowledge is stored in books and human minds which computers cannot access



Internal-combustion engines 417

Thermodynamic comparative cycle as shown in the *p–V*- und *T–S* diagrams

from T_2 to $T_{2'}$, supplied by the heat exchanger is coupled with a thermal discharge (4 → 4'). If heat is completely exchanged, the quantity of heat to be added per unit of gas is reduced to

$$q_{in} = c_p \cdot (T_3 - T_{2'}) = c_p \cdot (T_3 - T_4)$$

and the quantity of heat to be removed is

$$q_{out} = c_p \cdot (T_{4'} - T_1) = c_p \cdot (T_2 - T_1).$$

The maximum thermal efficiency for the gas turbine with heat exchanger is:

$$\eta_{th} = 1 - Q_{out}/Q_{in} = 1 - (T_2 - T_1)/(T_3 - T_4)$$

Where $p_2/p_1 = (T_2/T_1)^{\frac{\chi}{\chi-1}} = (T_3/T_4)^{\frac{\chi}{\chi-1}}$ and $T_4 = T_3 \cdot (T_1/T_2)$ thus

$$\eta_{th} = 1 - (T_2/T_3)$$

Current gas-turbine powerplants achieve thermal efficiencies of up to 35 %.

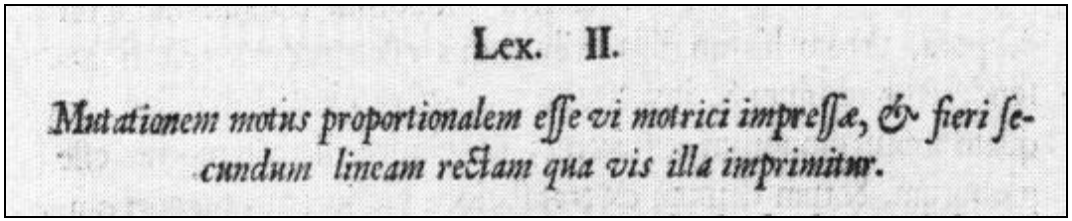
Advantages of the gas turbine: clean exhaust without supplementary emissions-control devices; extremely smooth running; multifuel capability; good static torque curve; extended maintenance intervals.

Disadvantages: manufacturing costs still high; poor transitional response; higher fuel consumption; less suitable for low-power applications.

Gas turbine *1 Filter and silencer, 2 Radial-flow compressor, 3 Burner, 4 Heat exchanger, 5 Exhaust port, 6 Reduction gearset, 7 Power turbine, 8 Adjustable guide vanes, 9 Compressor turbine, 10 Starter, 11 Auxiliary equipment drive, 12 Lubricating oil pump.*

"The change of motion is proportional to the motive force impressed"

– Newton



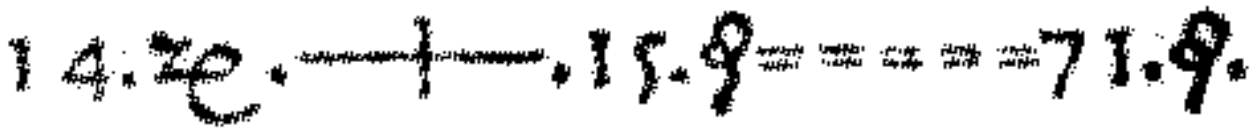
Lex. II.

Mutationem motus proportionalem esse vi motrici impressæ, & fieri secundum lineam rectam qua vis illa imprimitur.

MODELICA

Modelica Background: The Form – Equations

- Equations were used in the third millennium B.C.
- Equality sign was introduced by Robert Recorde in 1557



Newton still wrote text (Principia, vol. 1, 1686)

"*The change of motion is proportional to the motive force impressed*"

CSSL (1967) introduced a special form of "equation":

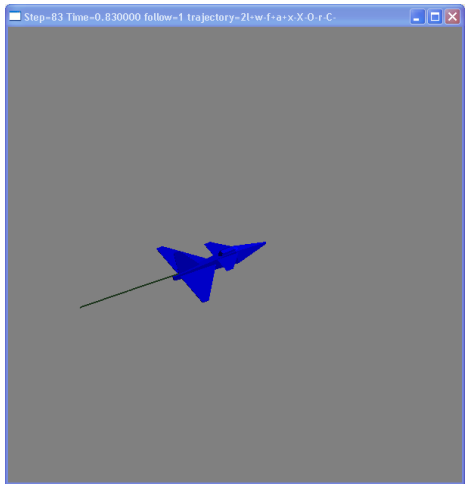
```
variable = expression
v = INTEG(F)/m
```

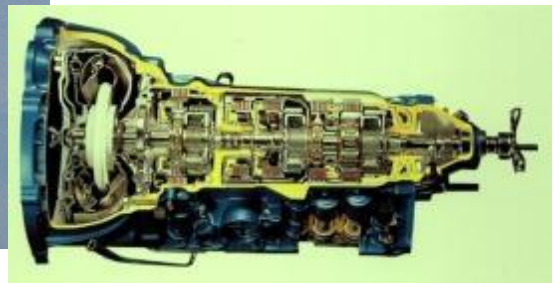
Programming languages usually do not allow equations!

What is Modelica?

A language for modeling of complex cyber-physical systems

- Robotics
- Automotive
- Aircrafts
- Satellites
- Power plants
- Systems biology

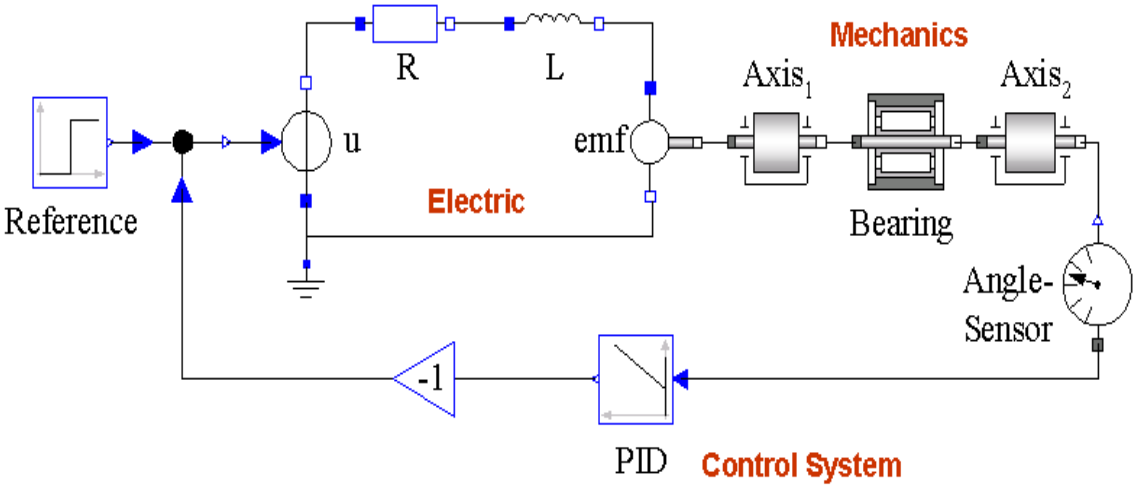




MODELICA

What is Modelica?

A language for modeling of complex cyber-physical systems



Primary designed for **simulation**, but there are also other usages of models, e.g. optimization.

What is Modelica?

A language for modeling of complex cyber-physical systems

i.e., Modelica is **not** a tool

Free, open language specification:

Available at: www.modelica.org

Developed and standardized by Modelica Association

There exist several free and commercial tools, for example:

- **OpenModelica from OSMC**
- Dymola from Dassault systems
- Wolfram System Modeler fr Wolfram MathCore
- SimulationX from ITI
- MapleSim from MapleSoft
- AMESIM from LMS
- JModelica.org from Modelon
- MWORKS from Tongyang Sw & Control
- IDA Simulation Env, from Equa
- ESI Group Modeling tool, ESI Group

Modelica – The Next Generation Modeling Language

Declarative language

Equations and mathematical functions allow acausal modeling, high level specification, increased correctness

Multi-domain modeling

Combine electrical, mechanical, thermodynamic, hydraulic, biological, control, event, real-time, etc...

Everything is a class

Strongly typed object-oriented language with a general class concept, Java & MATLAB-like syntax

Visual component programming

Hierarchical system architecture capabilities

Efficient, non-proprietary

Efficiency comparable to C; advanced equation compilation, e.g. 300 000 equations, ~150 000 lines on standard PC

Modelica Acausal Modeling

What is *acausal* modeling/design?

Why does it increase *reuse*?

> The acausality makes Modelica library classes *more reusable* than traditional classes containing assignment statements where the input-output causality is fixed.

Example: a resistor *equation*:

R*i = v;

can be used in three ways:

i := v/R;

v := R*i;

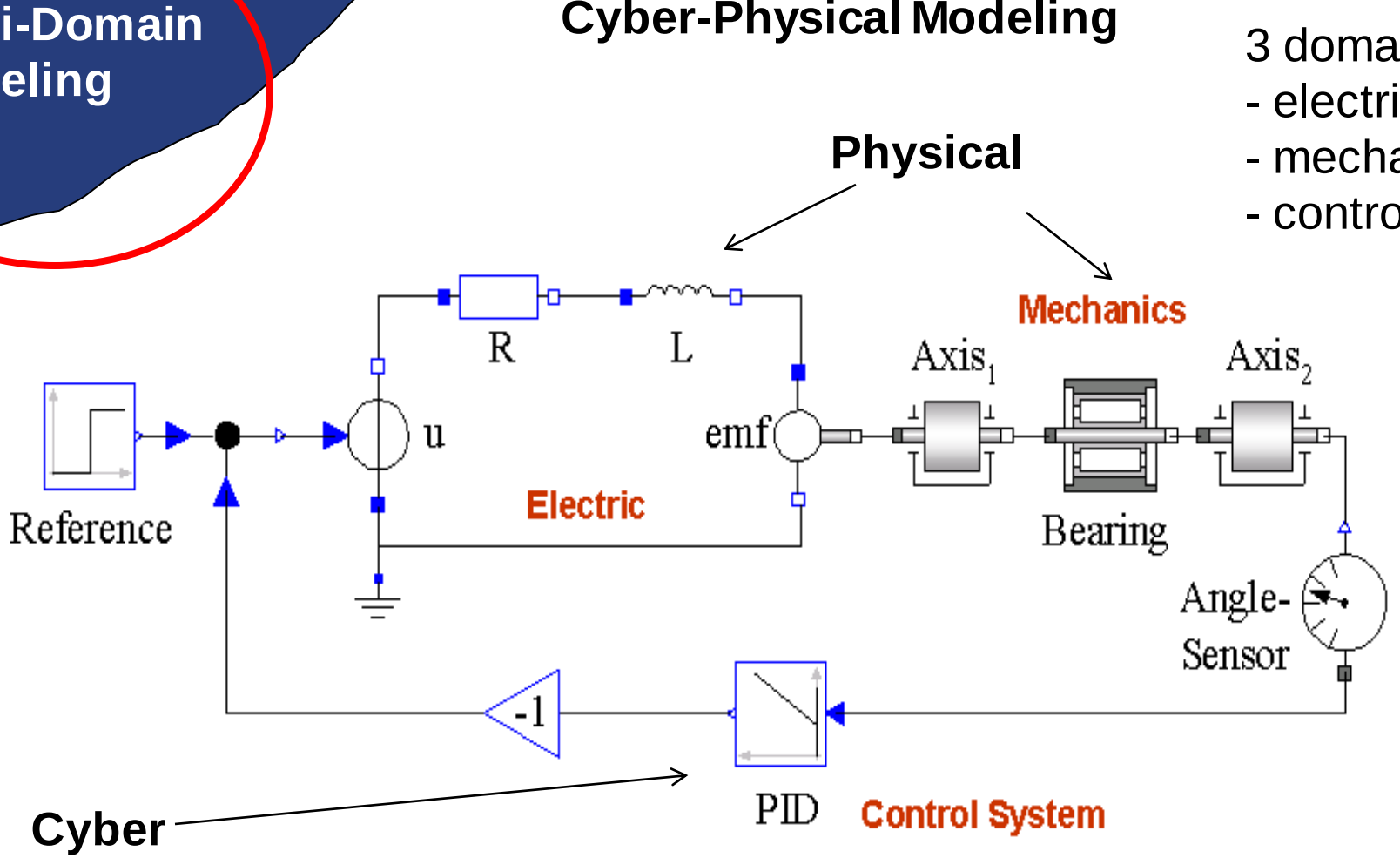
R := v/i;

What is Special about Modelica?

- Multi-Domain Modeling
- Visual acausal hierarchical component modeling
- Typed declarative equation-based textual language
- Hybrid modeling and simulation

What is Special about Modelica?

Multi-Domain Modeling



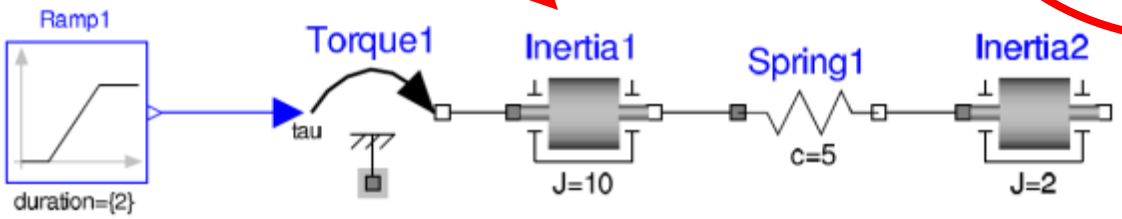
What is Special about Modelica?

Multi-Domain Modeling

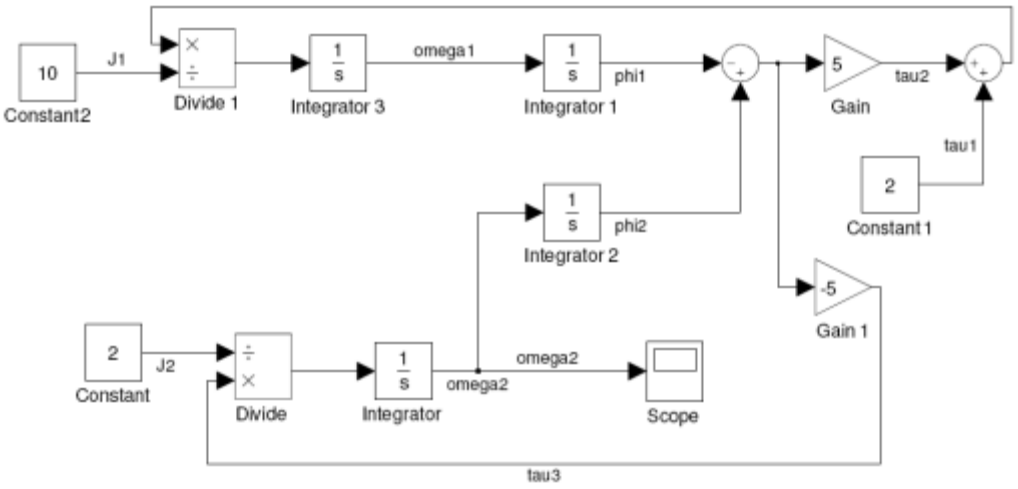
Visual Acausal Hierarchical Component Modeling

Keeps the physical structure

Acausal model (Modelica)



Causal block-based model (Simulink)



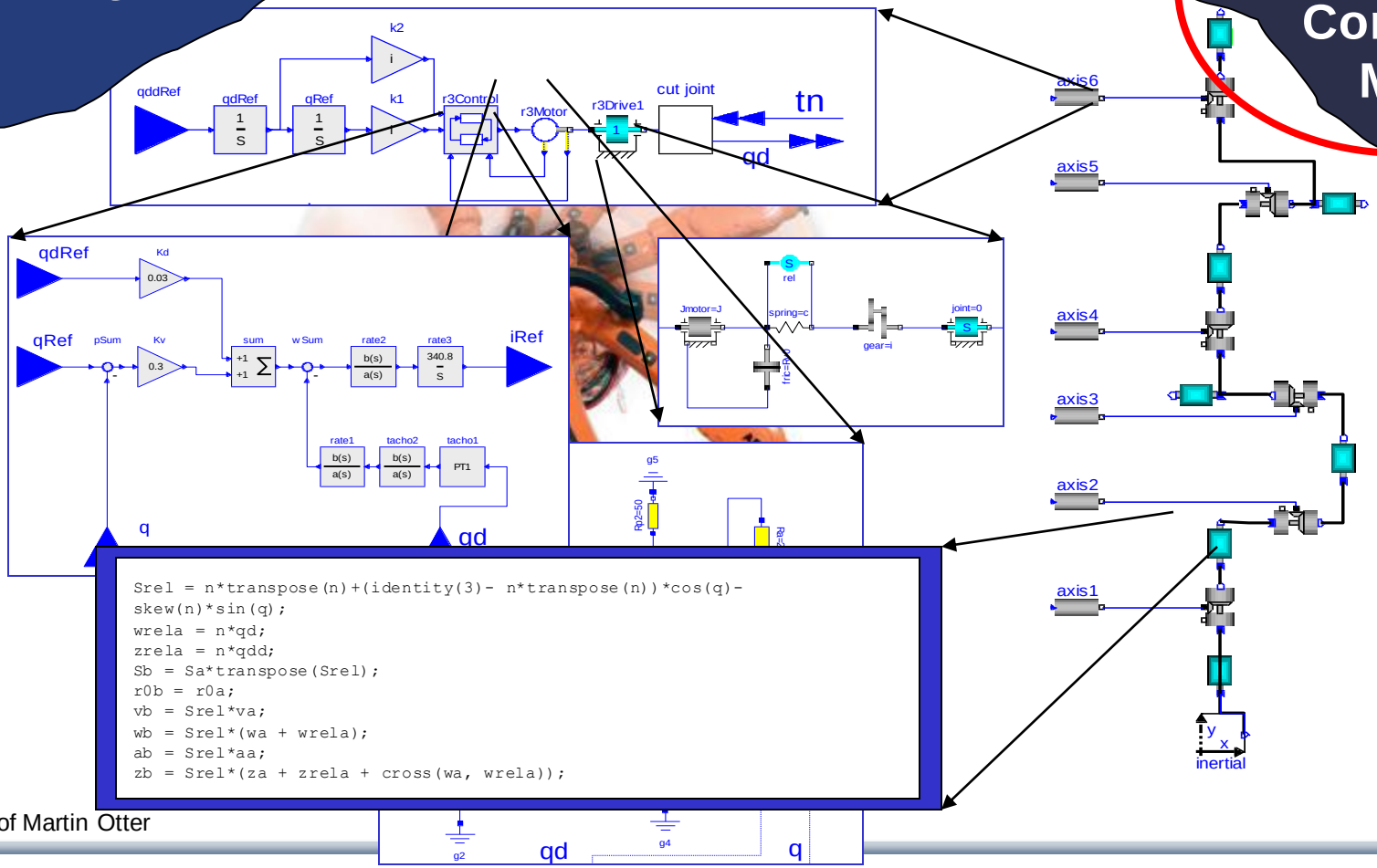
MODELICA

What is Special about Modelica?

Multi-Domain Modeling

Hierarchical system modeling

Visual Acausal Hierarchical Component Modeling



Courtesy of Martin Otter

MODELICA

What is Special about Modelica?

Multi-Domain Modeling

Visual Acausal Hierarchical Component Modeling

A textual *class-based* language
OO primary used for as a structuring concept

Behaviour described declaratively using

- Differential algebraic equations (DAE) (continuous-time)
- Event triggers (discrete-time)

Typed Declarative Equation-based Textual Language

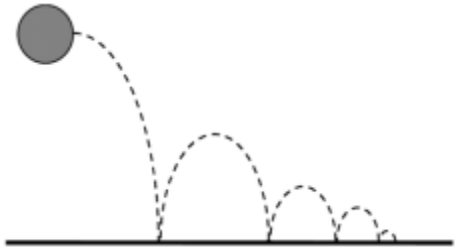
Variable declarations

```
class VanDerPol  "Van der Pol oscillator model"
  Real x(start = 1)  "Descriptive string for x";
  Real y(start = 1)  "y coordinate";
  parameter Real lambda = 0.3;
equation
  der(x) = y;
  der(y) = -x + lambda*(1 - x*x)*y;
end VanDerPol;
```

Differential equations

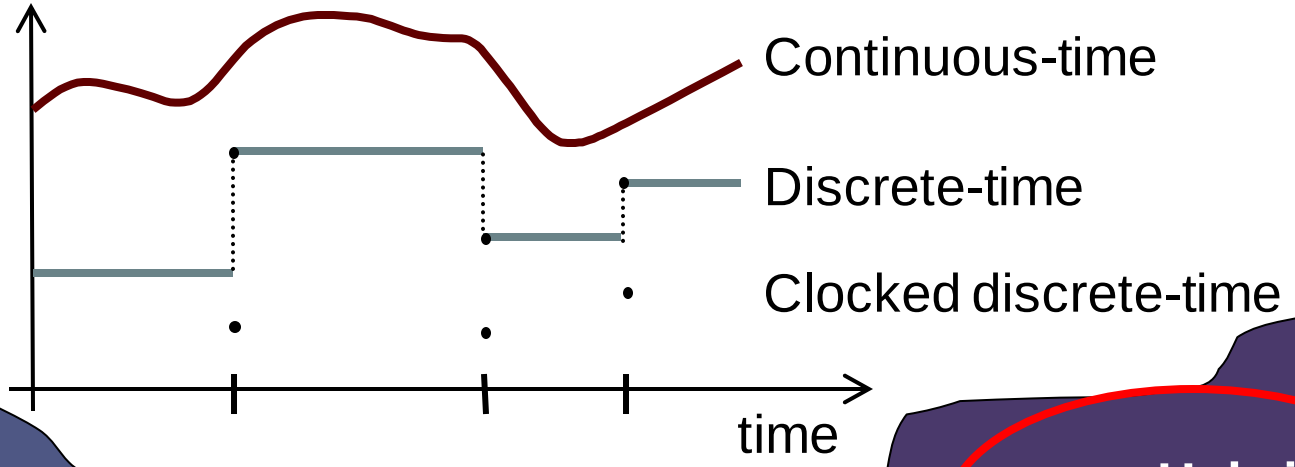
What is Special about Modelica?

Multi-Domain Modeling



Visual Acausal Component Modeling

Hybrid modeling =
continuous-time + discrete-time modeling

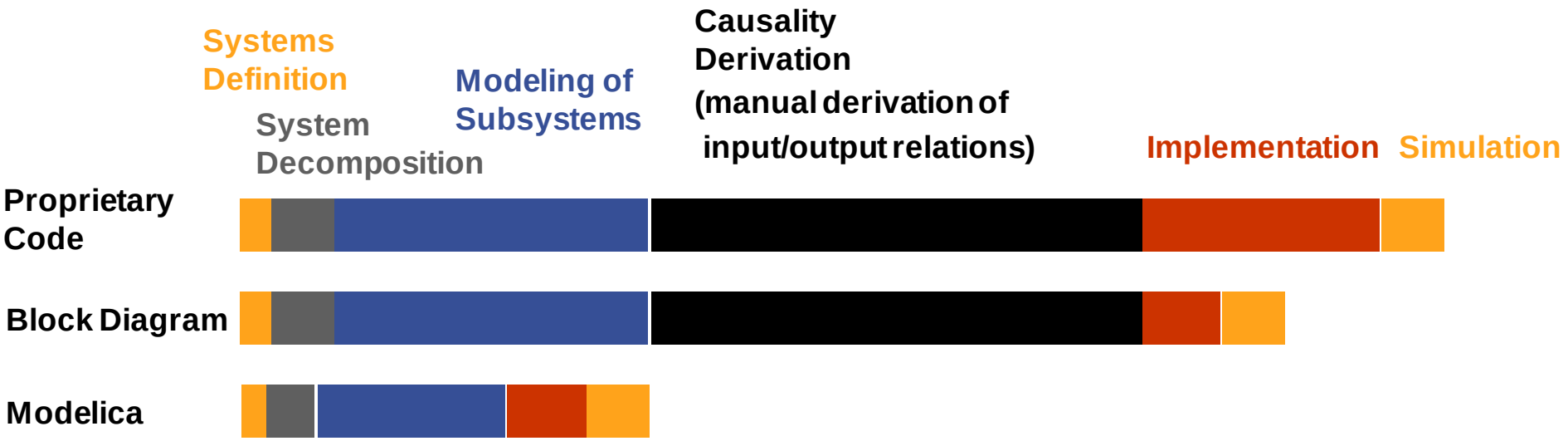


Typed
Declarative
Equation-based
Textual Language

Hybrid Modeling

Modelica – Faster Development, Lower Maintenance than with Traditional Tools

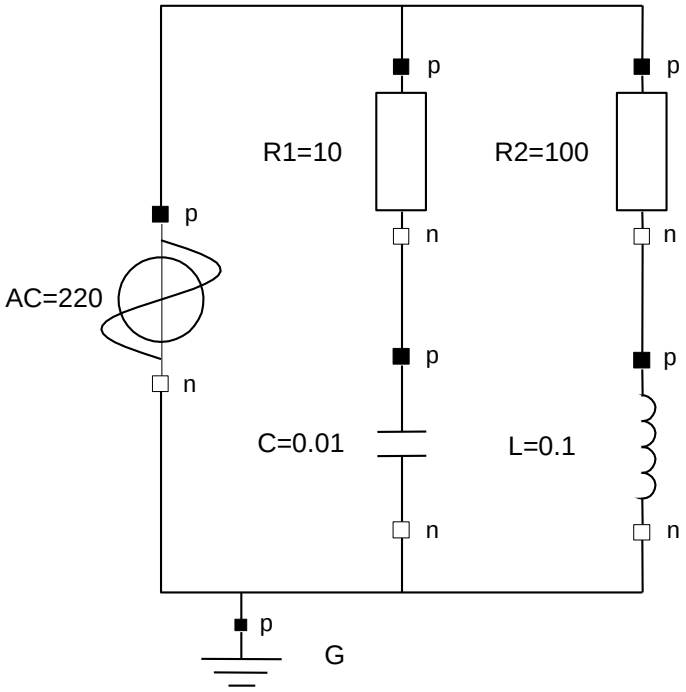
Block Diagram (e.g. Simulink, …) or
Proprietary Code (e.g. Ada, Fortran, C,…)
vs Modelica



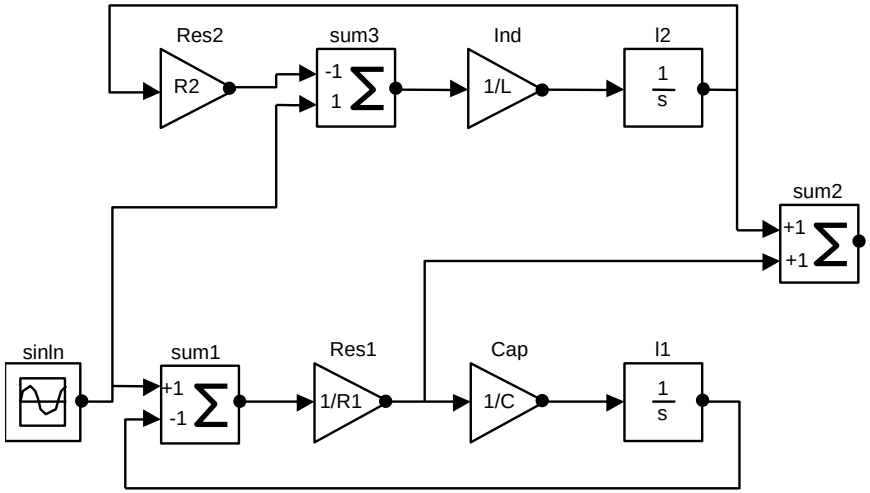
Modelica vs Simulink Block Oriented Modeling Simple Electrical Model

Modelica:
Physical model – easy to understand

Keeps the physical structure

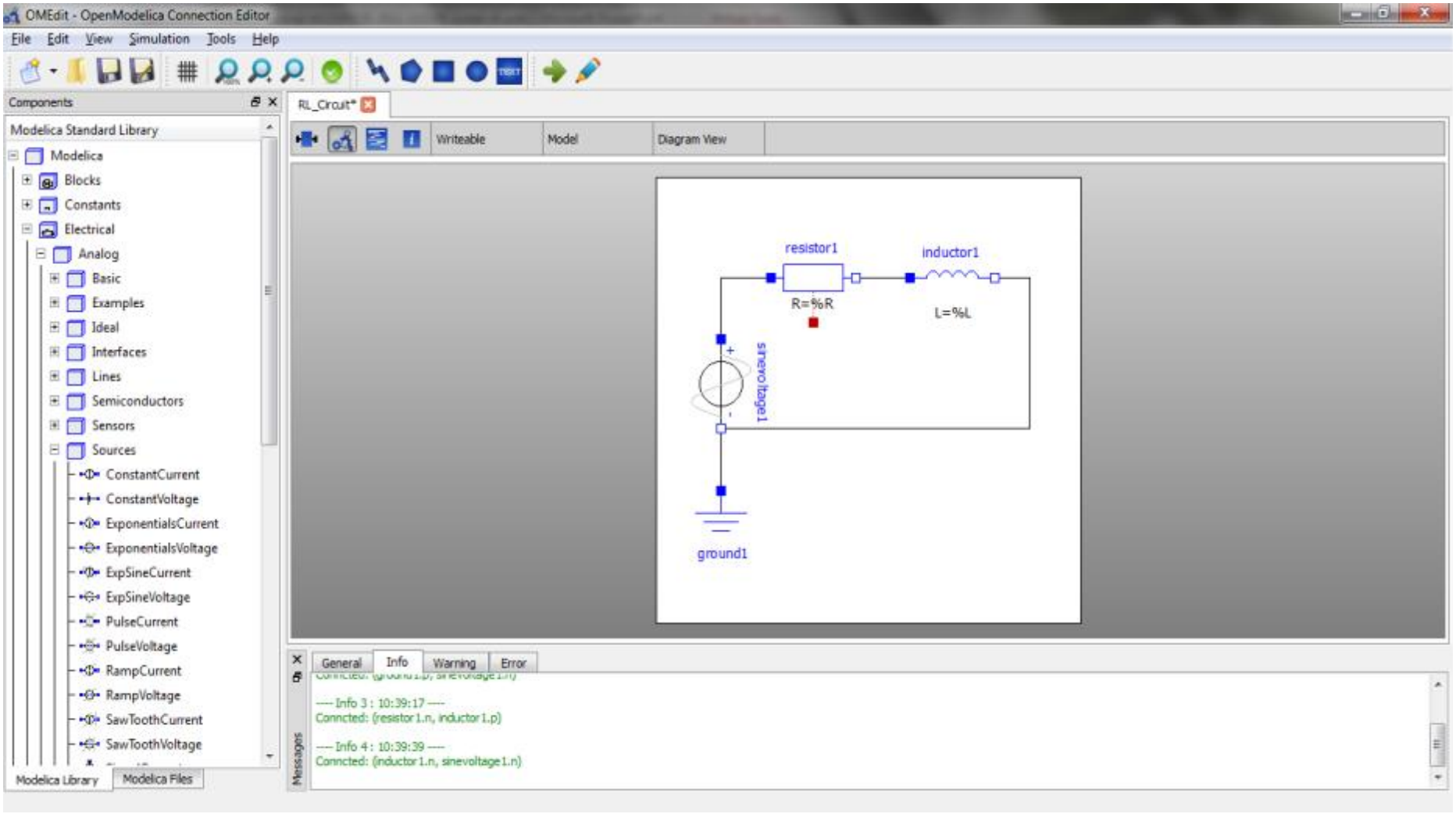


Simulink:
Signal-flow model – hard to understand



MODELICA

Graphical Modeling - Using Drag and Drop Composition

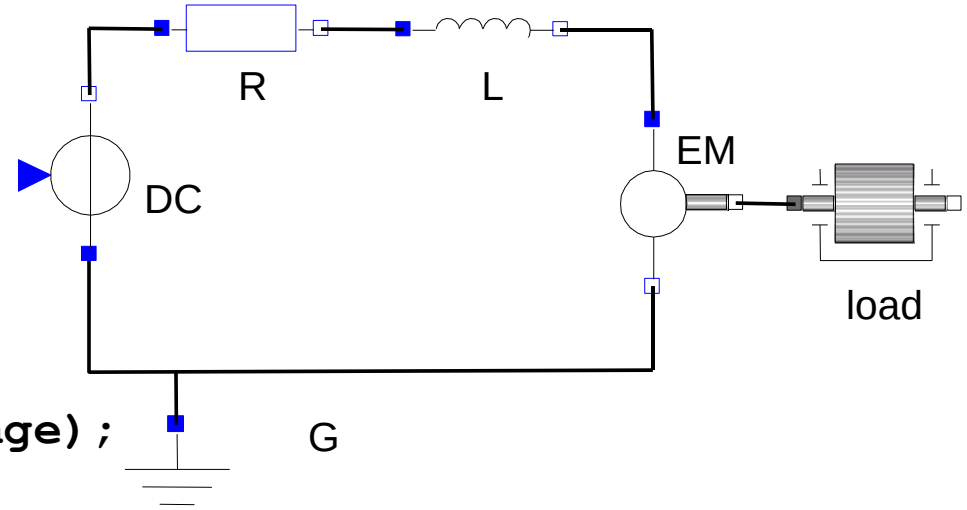


MODELICA

Multi-Domain (Electro-Mechanical) Modelica Model

- A DC motor can be thought of as an electrical circuit which also contains an electromechanical component

```
model DCMotor
   Resistor R(R=100);
   Inductor L(L=100);
   VsourceDC DC(f=10);
   Ground G;
   ElectroMechanicalElement EM(k=10,J=10, b=2);
   Inertia load;
equation
   connect(DC.p,R.n);
   connect(R.p,L.n);
   connect(L.p, EM.n);
   connect(EM.p, DC.n);
   connect(DC.n,G.p);
   connect(EM.flange,load.flange);
end DCMotor
```



Corresponding DCMotor Model Equations

The following equations are automatically derived from the Modelica model:

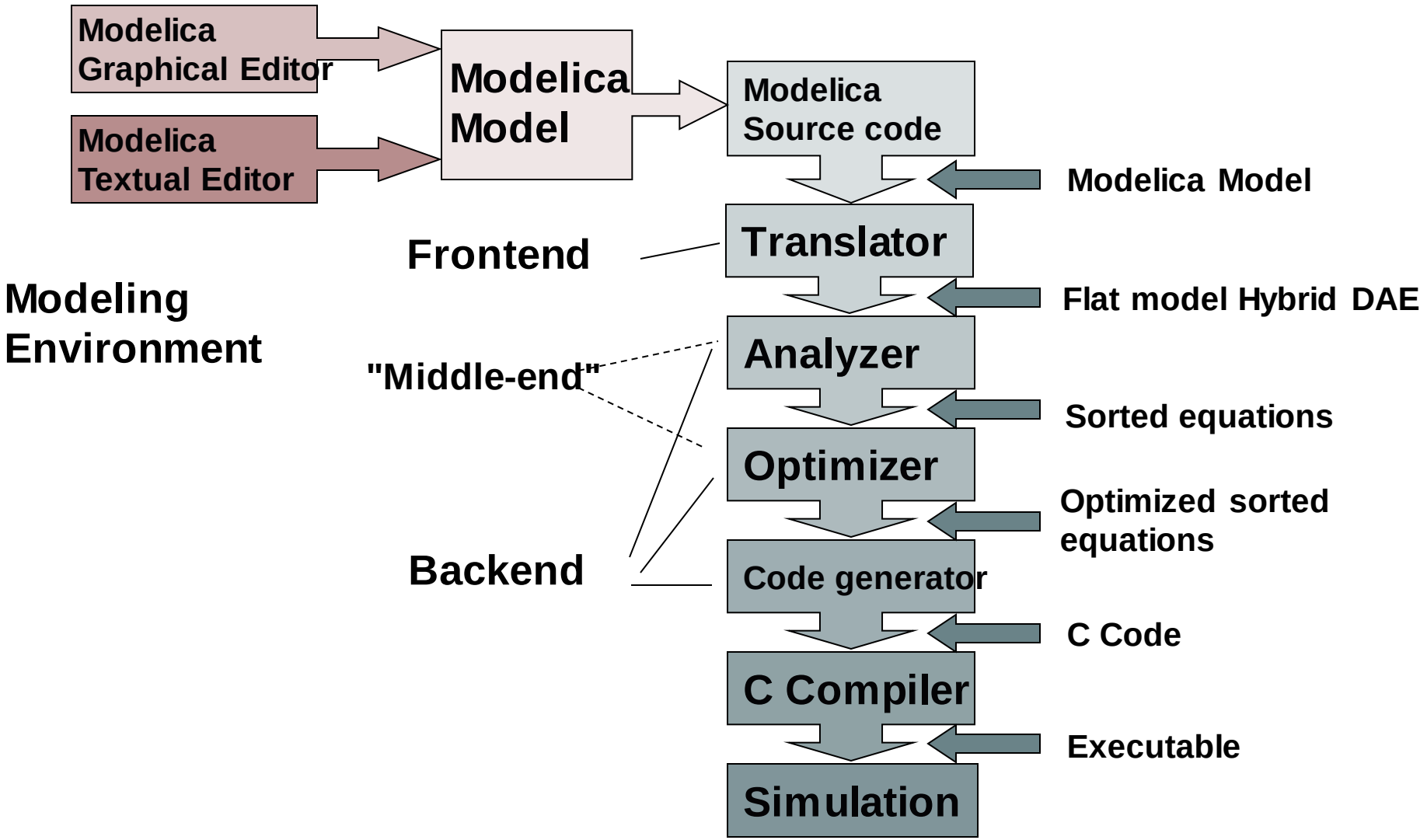
```
0 == DC.p.i + R.n.i
DC.p.v == R.n.v

0 == R.p.i + L.n.i
R.p.v == L.n.v

0 == L.p.i + EM.n.i
L.p.v == EM.n.v

0 == EM.p.i + DC.n.i
EM.p.v == DC.n.v

0 == DC.n.i + G.p.i
DC.n.v == G.p.v

EM.u == EM.p.v - EM.n.v
0 == EM.p.i + EM.n.i
EM.i == EM.p.i
EM.u == EM.k * EM.ω
EM.i == EM.M / EM.k
EM.J * EM.ω == EM.M - EM.b * EM.ω

DC.u == DC.p.v - DC.n.v
0 == DC.p.i + DC.n.i
DC.i == DC.p.i
DC.u == DC.Amp * Sin[2 π DC.f * t]

R.u == R.p.v - R.n.v
0 == R.p.i + R.n.i
R.i == R.p.i
R.u == R.R * R.i

L.u == L.p.v - L.n.v
0 == L.p.i + L.n.i
L.i == L.p.i
L.u == L.L * L.i'
```

(`load` component not included)

Automatic transformation to ODE or DAE for simulation:

$$\frac{dx}{dt} == f[x, u, t] \qquad g\left[\frac{dx}{dt}, x, u, t\right] == 0$$

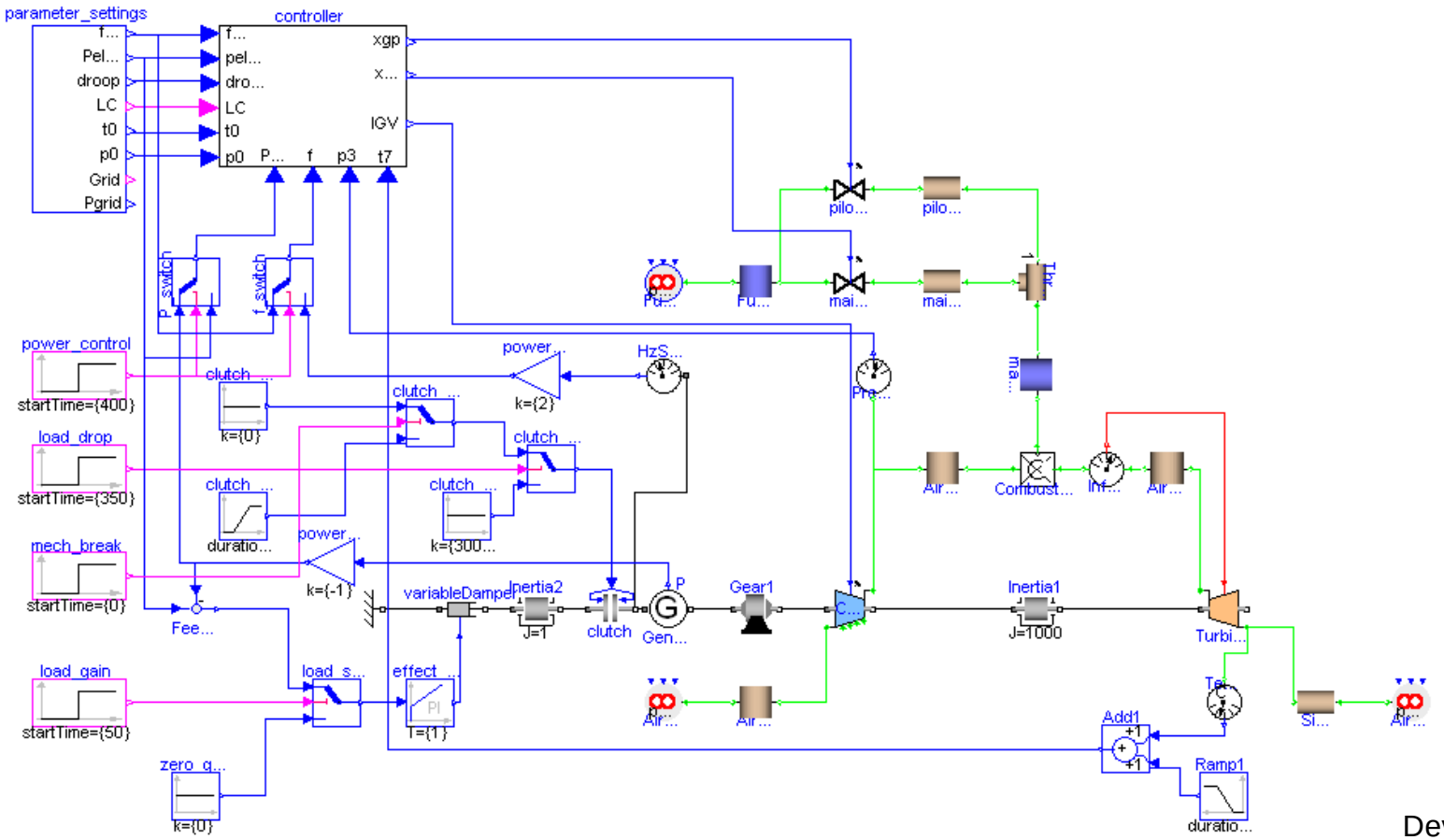
Model Translation Process to Hybrid DAE to Code



Brief Modelica History

- First Modelica design group meeting in fall 1996
  - International group of people with expert knowledge in both language design and physical modeling
  - Industry and academia
- Modelica Versions
  - 1.0 released September 1997
  - 2.0 released March 2002
  - 2.2 released March 2005
  - 3.0 released September 2007
  - 3.1 released May 2009
  - 3.2 released March 2010
  - 3.3 released May 2012
  - 3.2 rev 2 released November 2013
  - 3.3 rev 1 released July 2014
- Modelica Association established 2000 in Linköping
  - Open, non-profit organization

Modelica in Power Generation
GTX Gas Turbine Power Cutoff Mechanism



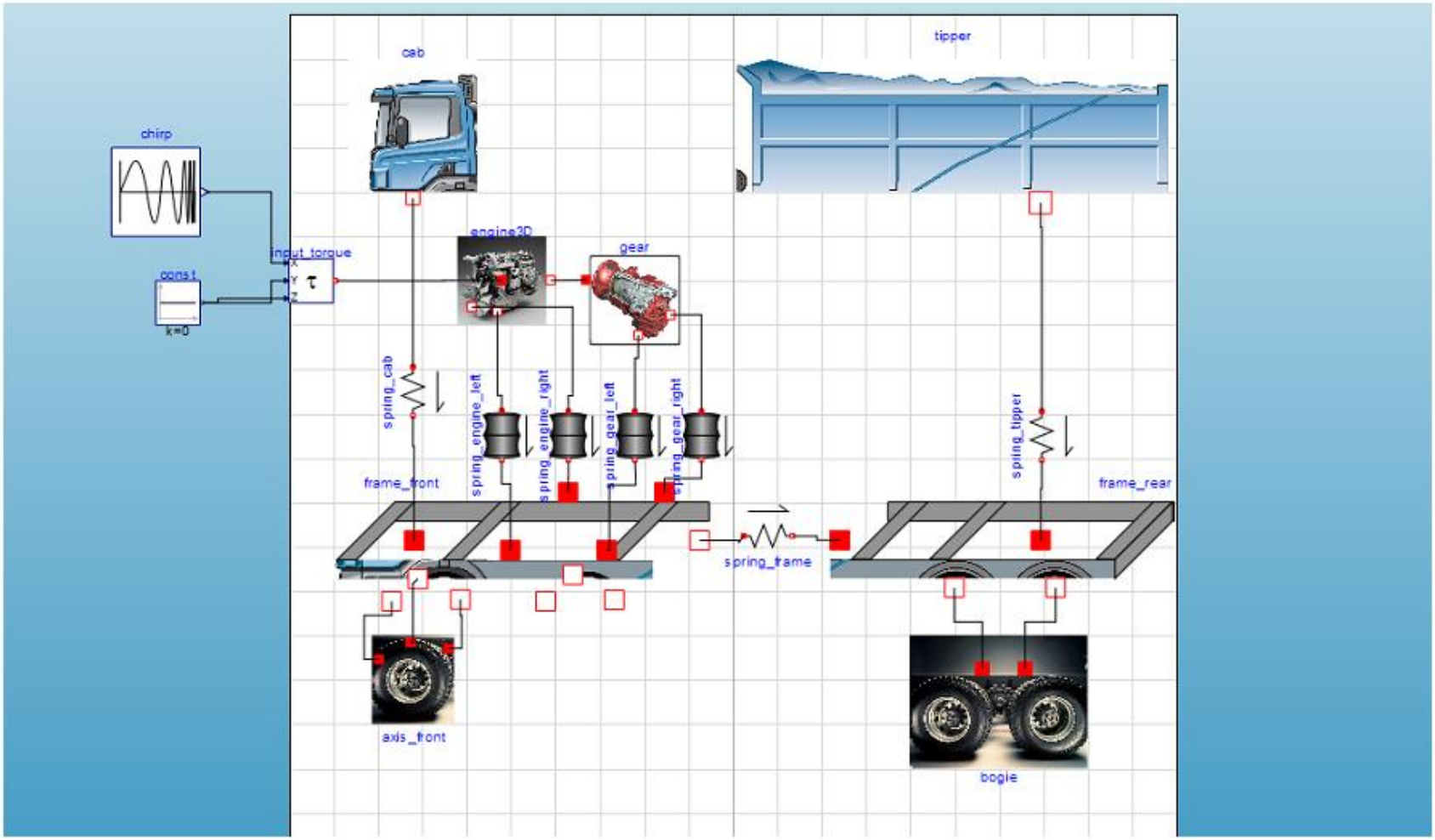
Developed by MathCore for Siemens

Courtesy of Siemens Industrial Turbomachinery AB

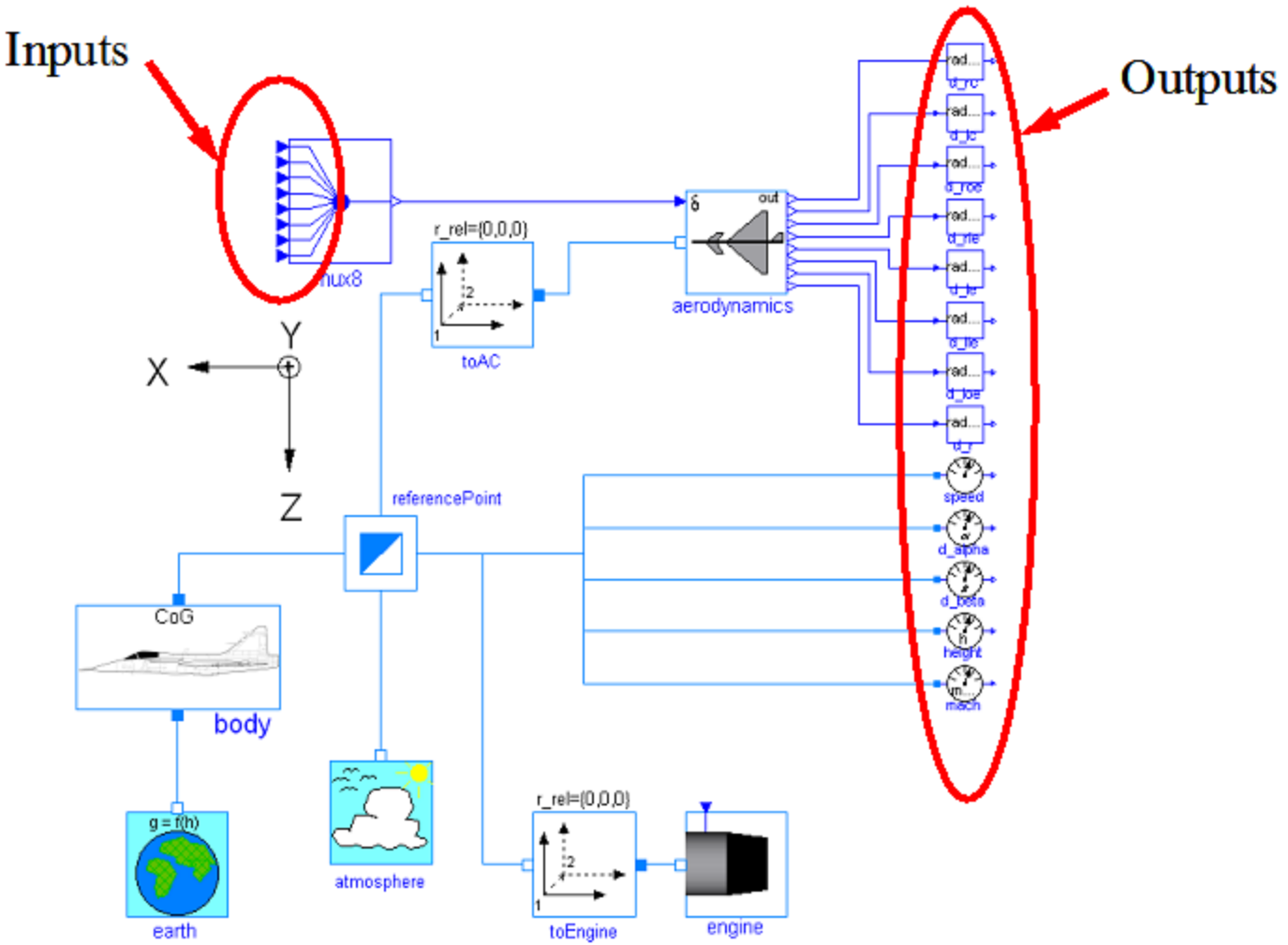
MathCore

MODELICA

Modelica in Automotive Industry



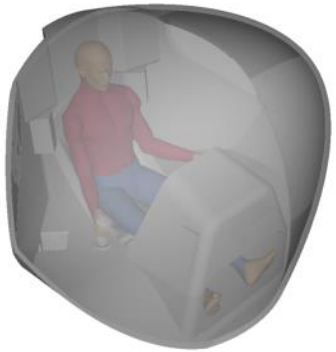
Modelica in Avionics



MODELICA

Application of Modelica in Robotics Models Real-time Training Simulator for Flight, Driving

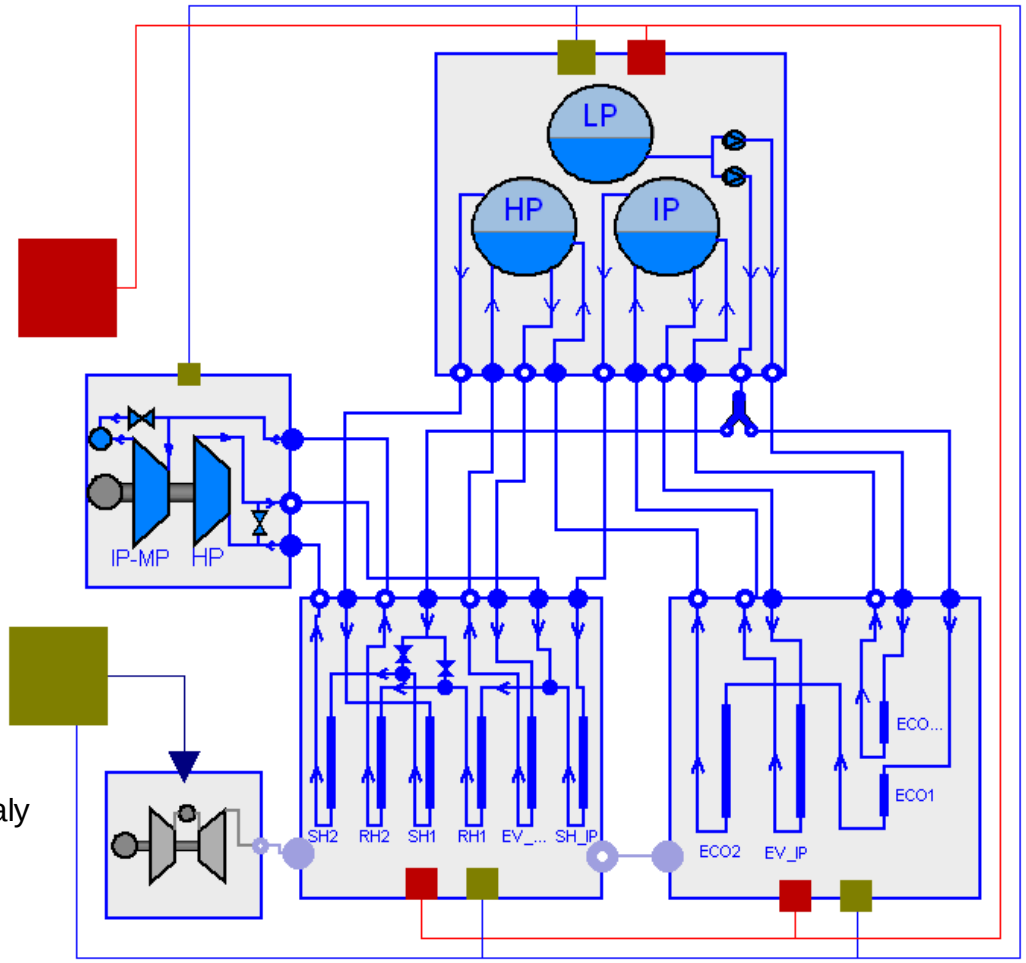
- Using Modelica models generating real-time code
- Different simulation environments (e.g. Flight, Car Driving, Helicopter)
- Developed at DLR Munich, Germany
- Dymola Modelica tool



Courtesy of Tobias Bellmann, DLR, Oberphaffenhofen, Germany

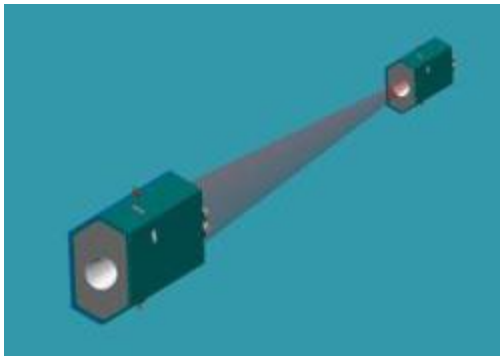
Combined-Cycle Power Plant
Plant model – system level

- GT unit, ST unit, Drum boilers unit and HRSG units, connected by thermo-fluid ports and by signal buses
- Low-temperature parts (condenser, feedwater system, LP circuits) are represented by trivial boundary conditions.
- GT model: simple law relating the electrical load request with the exhaust gas temperature and flow rate.



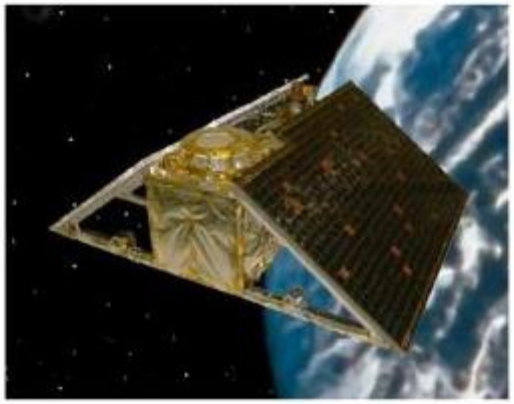
Courtesy Francesco Casella, Politecnico di Milano – Italy
and Francesco Pretolani, CESI SpA - Italy

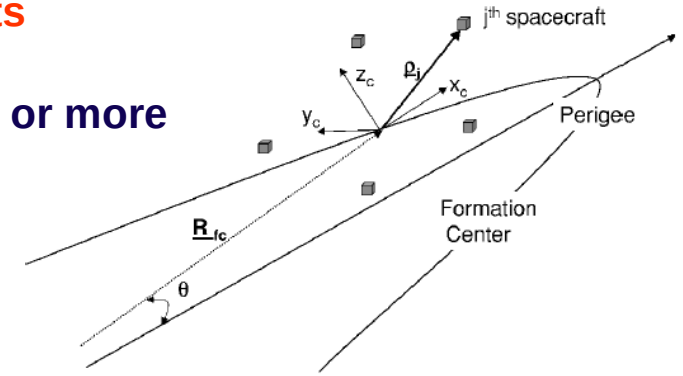
Modelica Spacecraft Dynamics Library



Formation flying on elliptical orbits

Control the relative motion of two or more spacecraft





Attitude control for satellites using magnetic coils as actuators

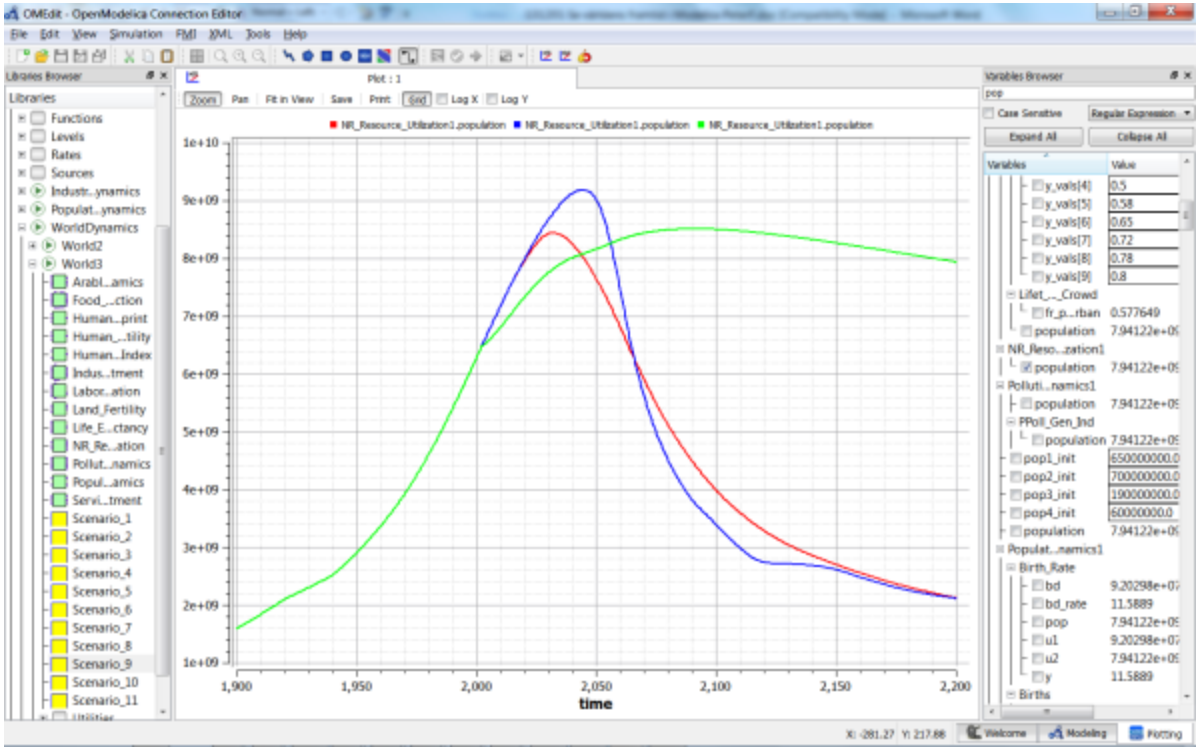
Torque generation mechanism: interaction between coils and geomagnetic field

Courtesy of Francesco Casella, Politecnico di Milano, Italy

MODELICA

System Dynamics – World Society Simulation

Limits to Material Growth; Population, Energy and Material flows



Left. World3 simulation with OpenModelica

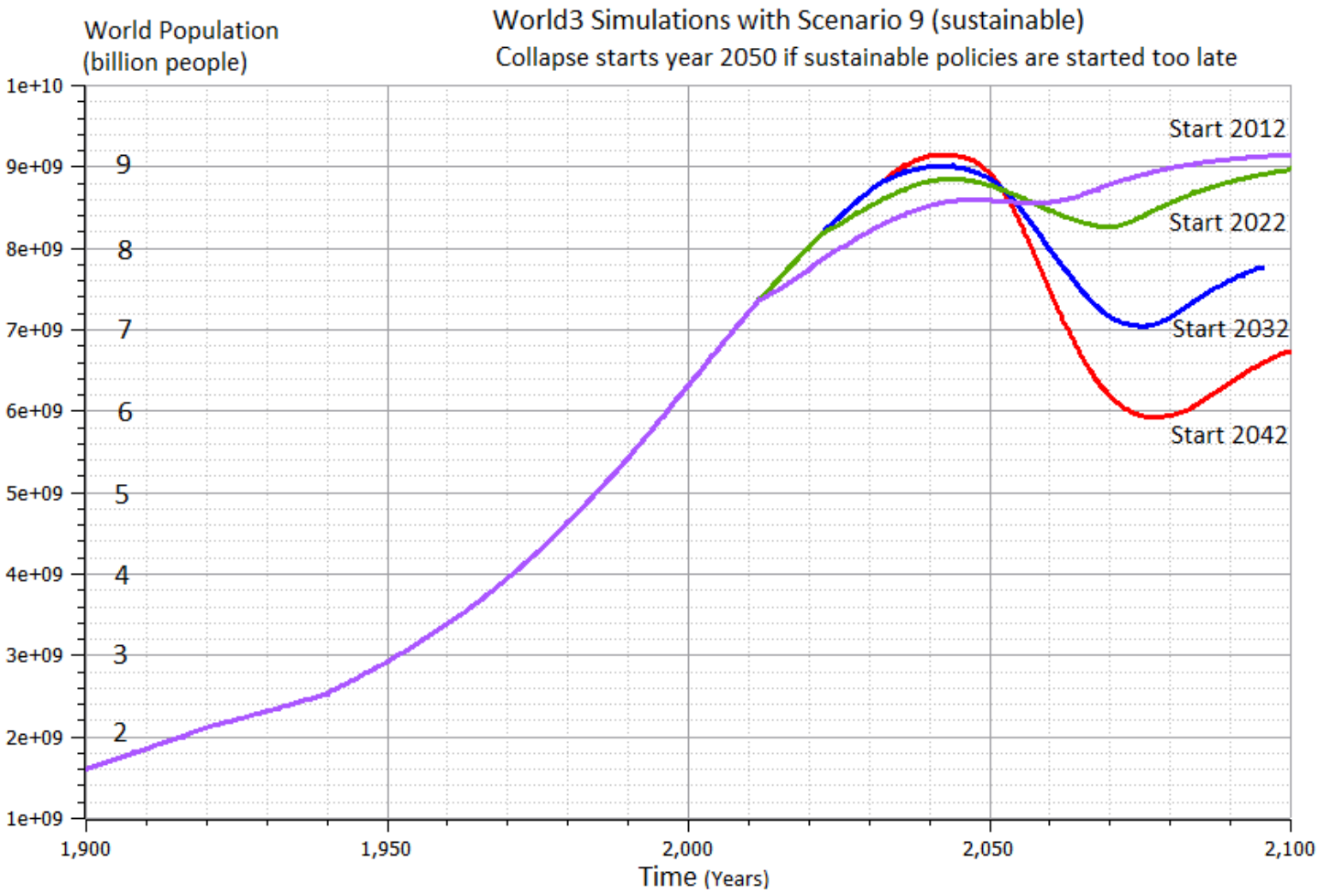
- 2 collapse scenarios (close to current developments)
- 1 sustainable scenario (green).

CO_2 Emissions per person:

- USA 17 ton/yr
- Sweden 7 ton/yr
- India 1.4 ton/yr
- Bangladesh 0.3 ton/yr

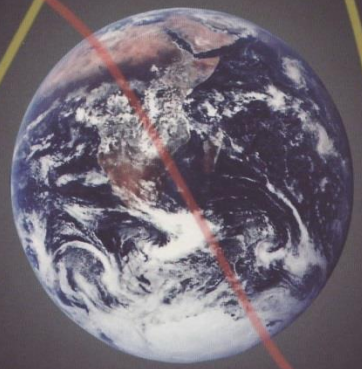
- System Dynamics Modelica library by Francois Cellier (ETH), et al in OM distribution.
- Warming converts many agriculture areas to deserts (USA, Europe, India, Amazonas)
- Ecological breakdown around 2080-2100, drastic reduction of world population
- To **avoid** this: Need for massive investments in sustainable technology and renewable energy sources

World3 Simulations with Different Start Years for Sustainable Policies – Collapse if starting too late



MODELICA

LIMITS TO GROWTH



The 30-Year Update

DONELLA MEADOWS | JORGEN RANDERS | DENNIS MEADOWS

THE *NEW YORK TIMES* BESTSELLER

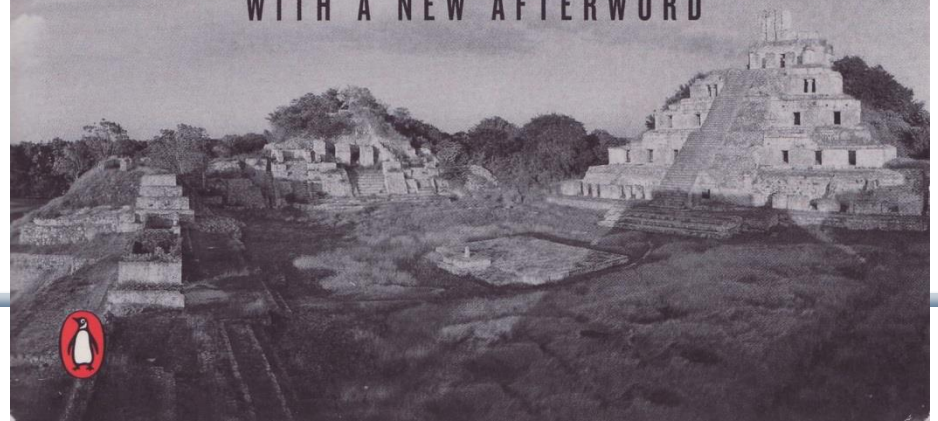
COLLAPSE

HOW SOCIETIES CHOOSE TO FAIL OR SUCCEED

JARED DIAMOND

author of the Pulitzer Prize–winning *GUNS, GERMS, and STEEL*

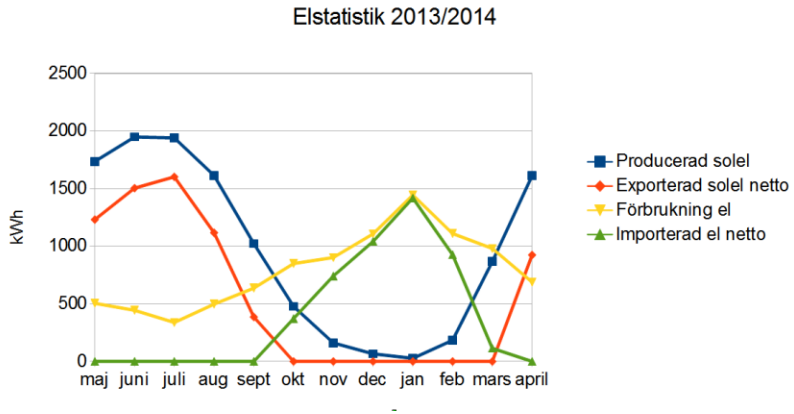
WITH A NEW AFTERWORD



What Can You Do?
Need Global Sustainability Mass Movement

- Develop smart Cyber-Physical systems for reduced energy and material footprint
- Model-based circular economy for re-use of products and materials
- Promote sustainable lifestyle and technology
- Install electric solar PV panels
- Buy shares in cooperative wind power

20 sqm solar panels on garage roof, Nov 2012
Generated 2700 W at noon March 10, 2013



Expanded to 93 sqm, 12 kW, March 2013
House produced 11600 kwh, used 9500 kwh
Avoids 10 ton CO_2 emission per year

MODELICA

Example Electric Cars

Can be charged by electricity from own solar panels

Renault ZOE; 5 seat; Range:

- **EU-drive cycle 210 km**
- **Realistic Swedish drive cycle:**
- **Summer: 165 km**
- **Winter: 100 – 110 km**

Cheap fast supercharger

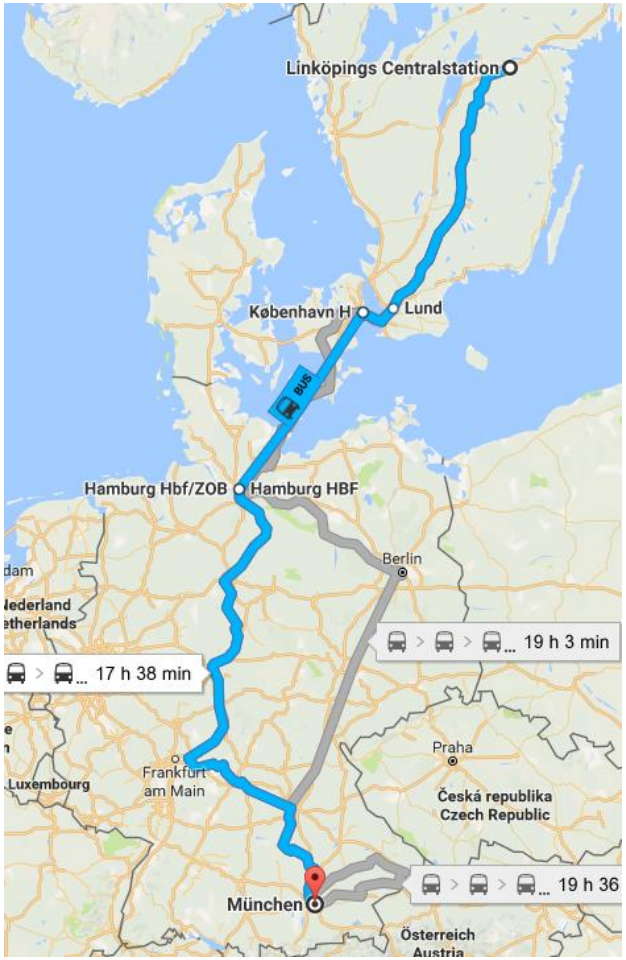
Tesla model S range 480 km

DLR ROboMObil

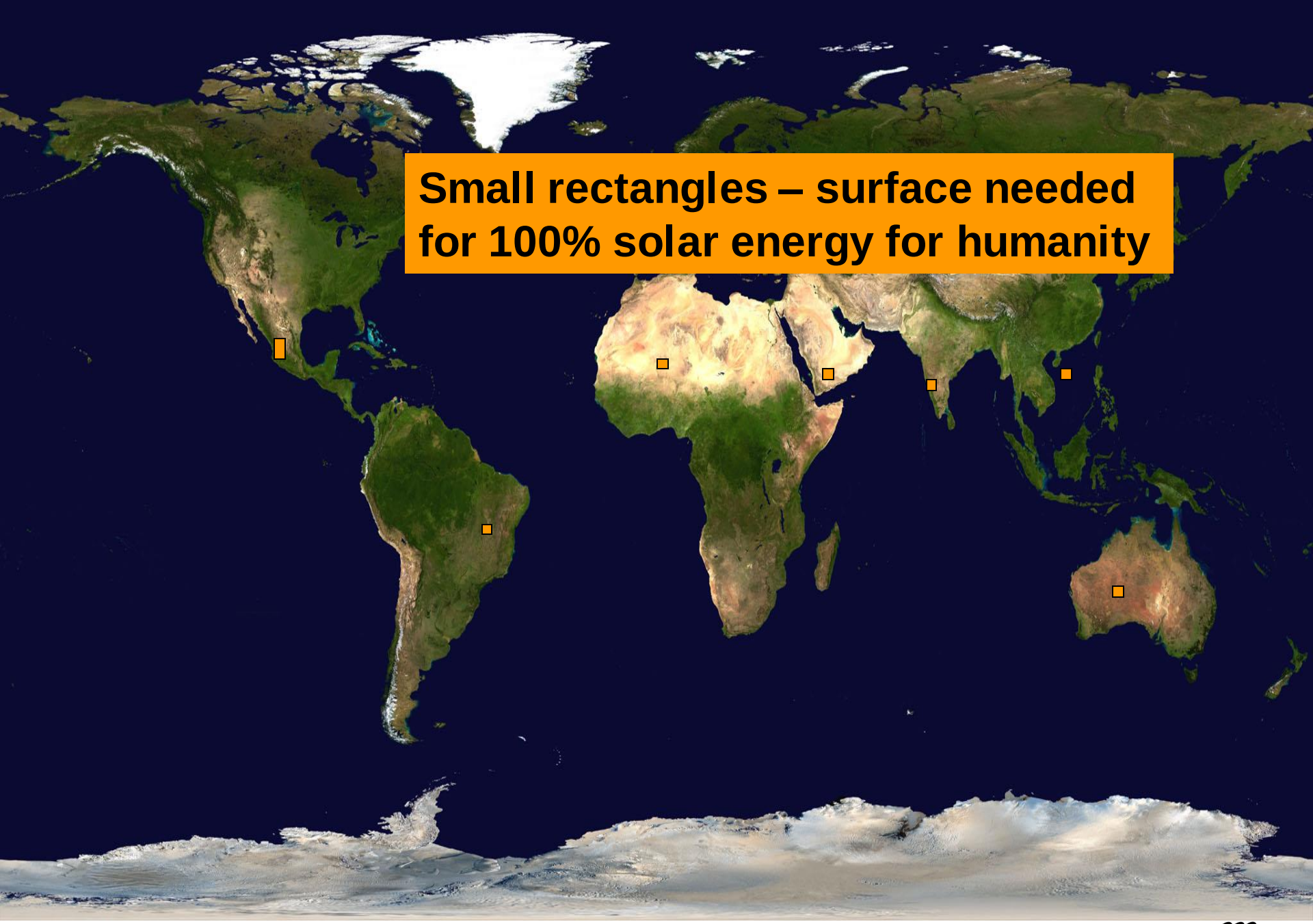
- **experimental electric car**
- **Modelica models**

What Can You Do?
More Train Travel – Less Air Travel

- Air travel by Swedish Citizens – about the same emissions as all personal car traffic in Sweden!
- By train from Linköping to Munich and back – saves almost 1 ton of CO2e emissions compared to flight
- Leave Linköping 07.00 in Munich 23.14



MODELICA


Small rectangles – surface needed
for 100% solar energy for humanity

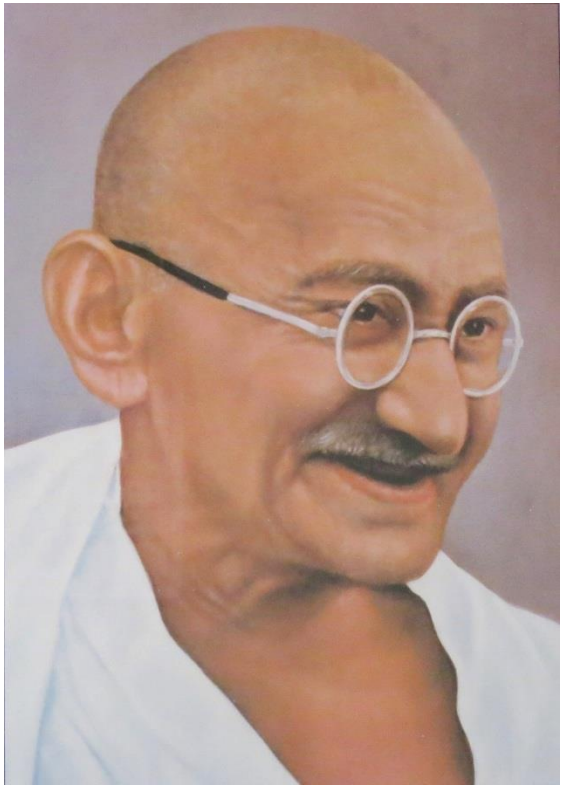
Sustainable Society Necessary for Human Survival

Almost Sustainable

- India, 1.4 ton C02/person/year
- Healthy vegetarian food
- Small-scale agriculture
- Small-scale shops
- Simpler life-style (Mahatma Gandhi)

Non-sustainable

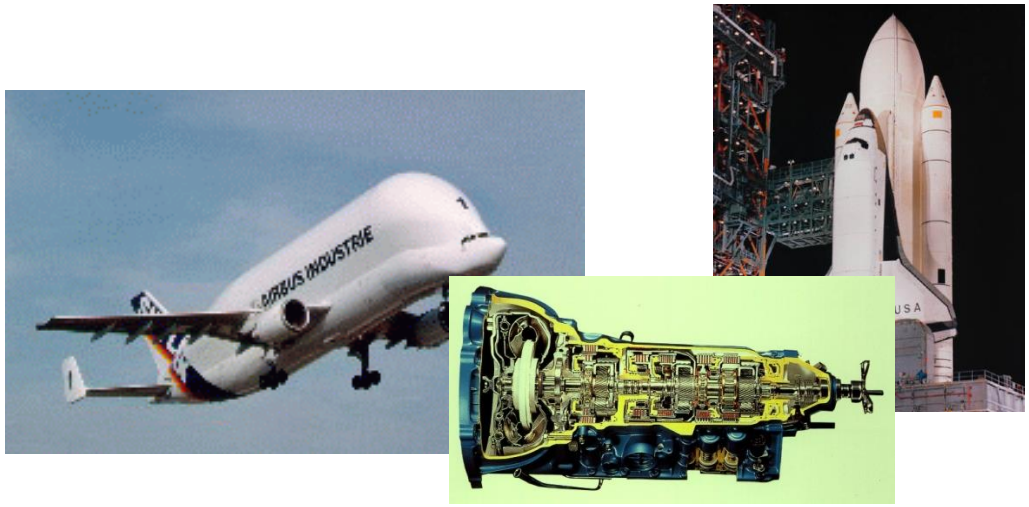
- USA 17 ton CO2, Sweden 7 ton CO2/yr
- High meat consumption (1 kg beef uses ca 4000 L water for production)
- Hamburgers, unhealthy , includes beef
- Energy-consuming mechanized agriculture
- Transport dependent shopping centres
- Stressful materialistic lifestyle



Gandhi – role model for future less materialistic life style

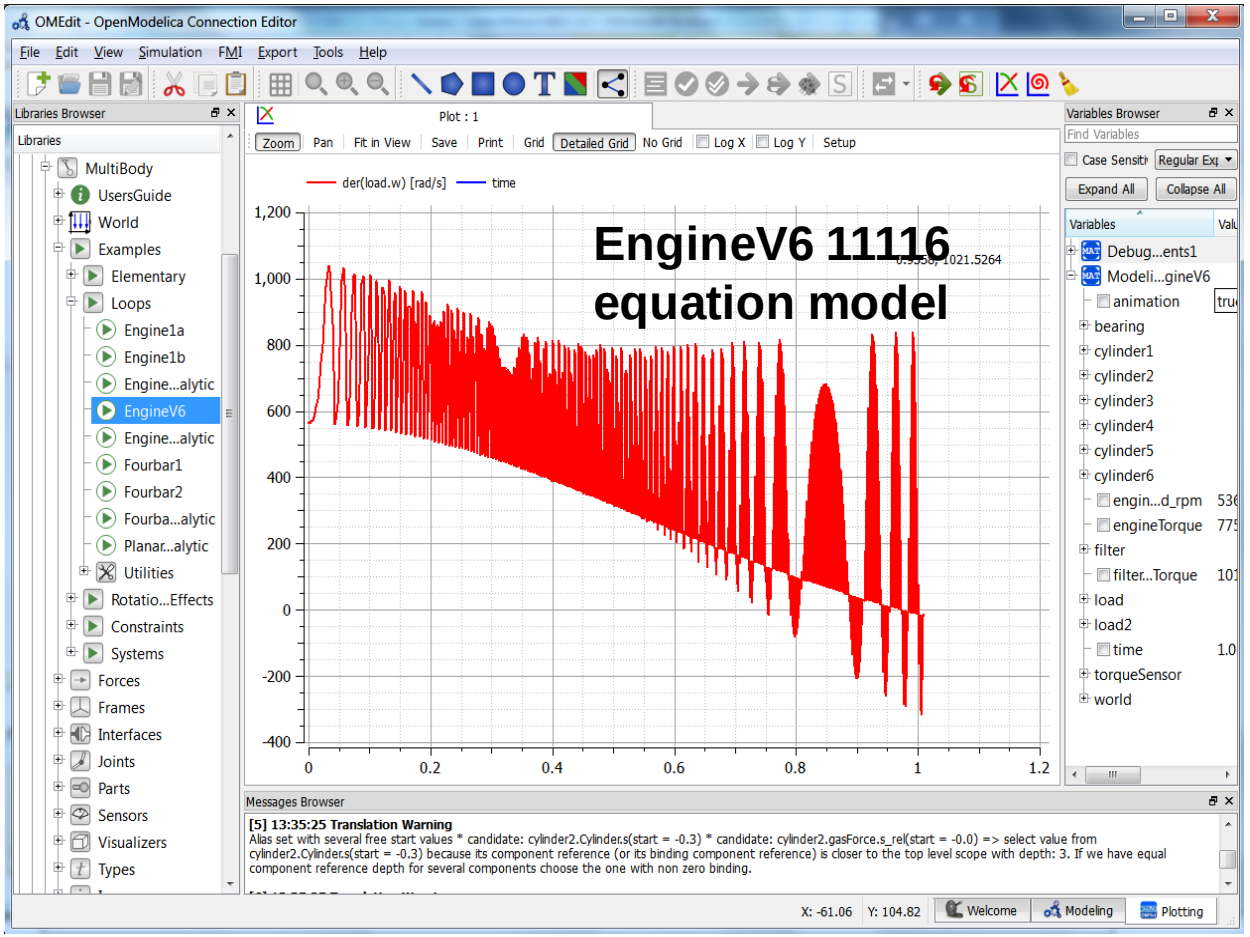
Part II

Introduction to the OpenModelica Environment



OpenModelica – Free Open Source Tool
developed by the Open Source Modelica Consortium (OSMC)

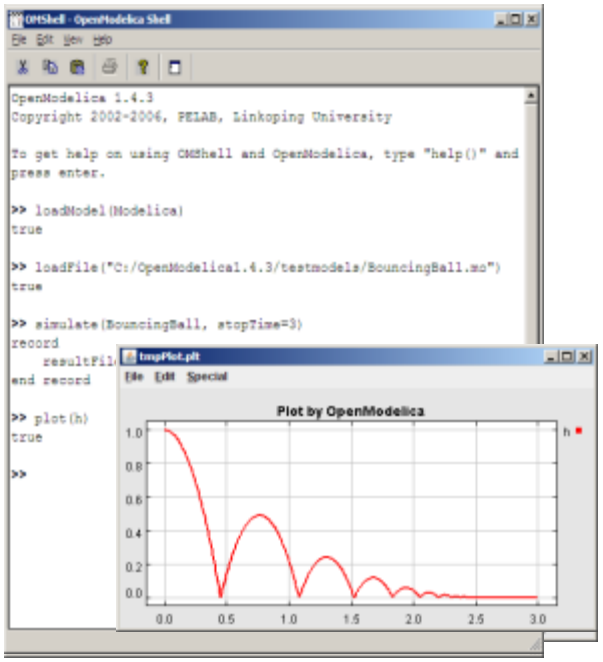
- Graphical editor
- Model compiler and simulator
- Debugger
- Performance analyzer
- Dynamic optimizer
- Symbolic modeling
- Parallelization
- Electronic Notebook for teaching

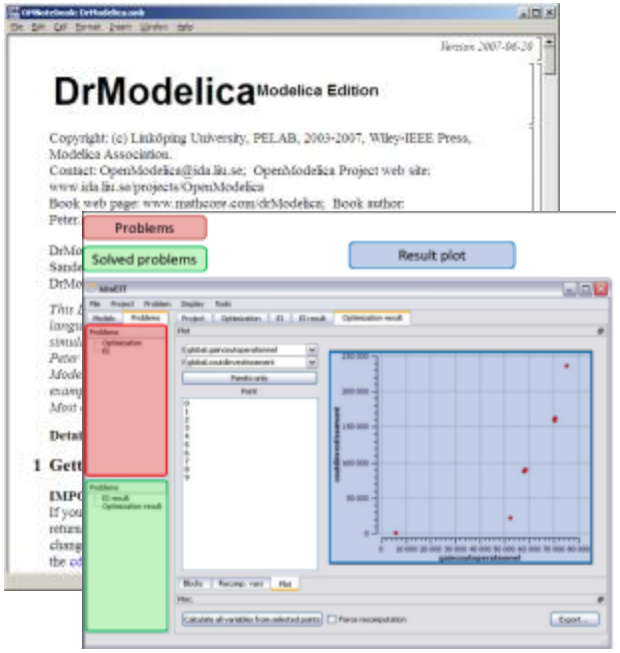


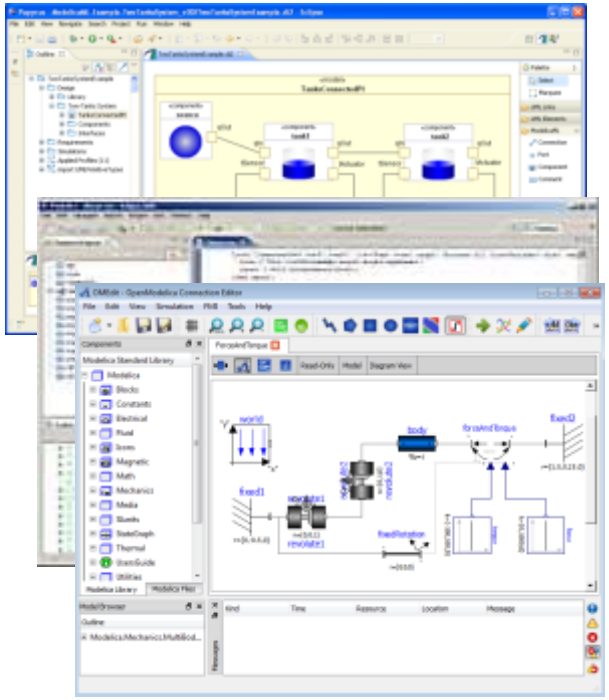
The OpenModelica Open Source Environment
www.openmodelica.org

- Advanced Interactive Modelica compiler (OMC)
  - Supports most of the Modelica Language
  - **Modelica** and **Python scripting**
- Basic environment for creating models
  - **OMShell** – an interactive command handler
  - **OMNotebook** – a literate programming notebook
  - **MDT** – an advanced textual environment in Eclipse

- **OMEdit** graphic Editor
- **OMDebugger** for equations
- **OMOptim** optimization tool
- **OM Dynamic optimizer** collocation
- **ModelicaML** UML Profile
- **MetaModelica** extension
- **ParModelica** extension







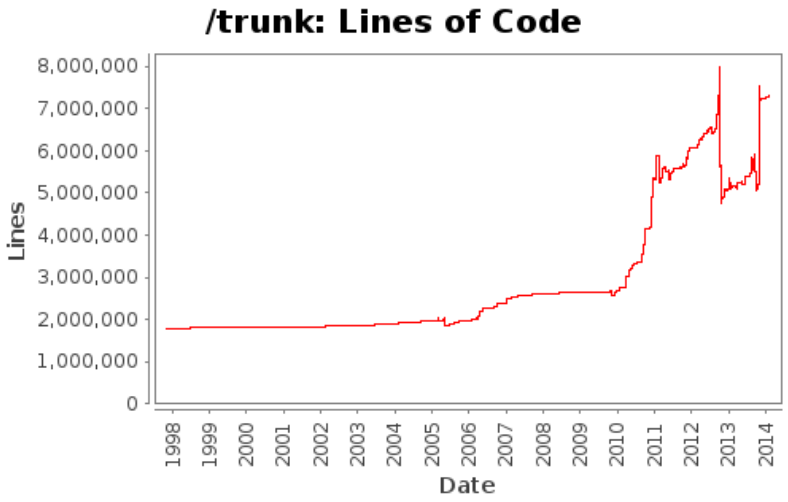
OSMC – International Consortium for Open Source Model-based Development Tools, 48 members Jan 2016

Founded Dec 4, 2007

Open-source community services

- **Website and Support Forum**
- **Version-controlled source base**
- **Bug database**
- **Development courses**
- **www.openmodelica.org**

Code Statistics



Industrial members

- ABB AB, Sweden
- Bosch Rexroth AG, Germany
- Siemens Turbo, Sweden
- CDAC Centre, Kerala, India
- Creative Connections, Prague
- DHI, Aarhus, Denmark
- Dynamica s.r.l., Cremona, Italy
- EDF, Paris, France
- Equa Simulation AB, Sweden
- Fraunhofer IWES, Bremerhaven
- IFPEN, Paris, France
- ISID Dentsu, Tokyo, Japan
- Maplesoft, Canada
- Ricardo Inc., USA
- RTE France, Paris, France
- Saab AB, Linköping, Sweden
- Scilab Enterprises, France
- SKF, Göteborg, Sweden
- TLK Thermo, Germany
- Sozhou Tongyuan, China
- VTI, Linköping, Sweden
- VTT, Finland
- Wolfram MathCore, Sweden

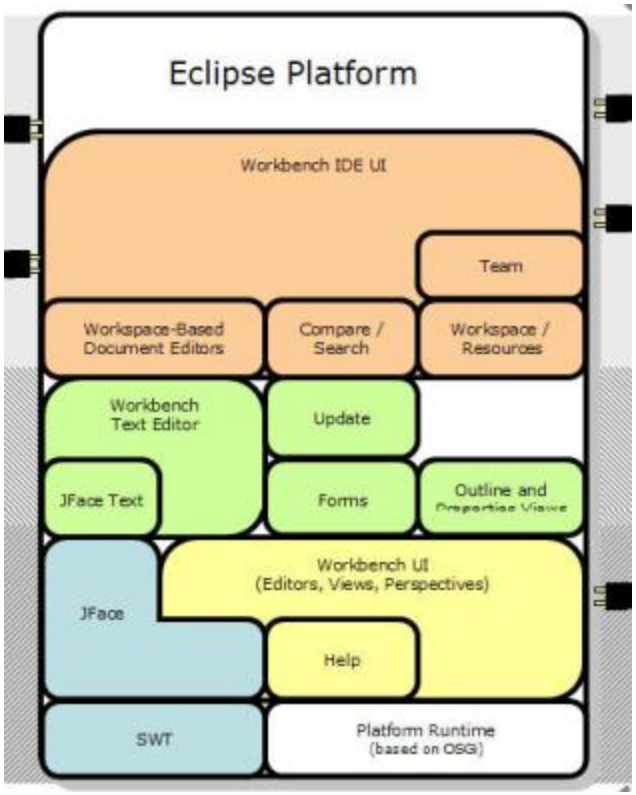
University members

- Austrian Inst. of Tech, Austria
- TU Berlin, Inst. UEBB, Germany
- FH Bielefeld, Bielefeld, Germany
- TU Braunschweig, Germany
- University of Calabria, Italy
- Univ California, Berkeley, USA
- Chalmers Univ Techn, Sweden
- TU Dortmund, Germany
- TU Dresden, Germany
- Université Laval, Canada
- Ghent University, Belgium
- Halmstad University, Sweden
- Heidelberg University, Germany
- Linköping University, Sweden
- TU Hamburg/Harburg Germany
- IIT Bombay, Mumbai, India
- KTH, Stockholm, Sweden
- Univ of Maryland, Syst Eng USA
- Univ of Maryland, CEEE, USA
- Politecnico di Milano, Italy
- Ecoles des Mines, CEP, France
- Mälardalen University, Sweden
- Univ Pisa, Italy
- StellenBosch Univ, South Africa
- Telemark Univ College, Norway

MODELICA

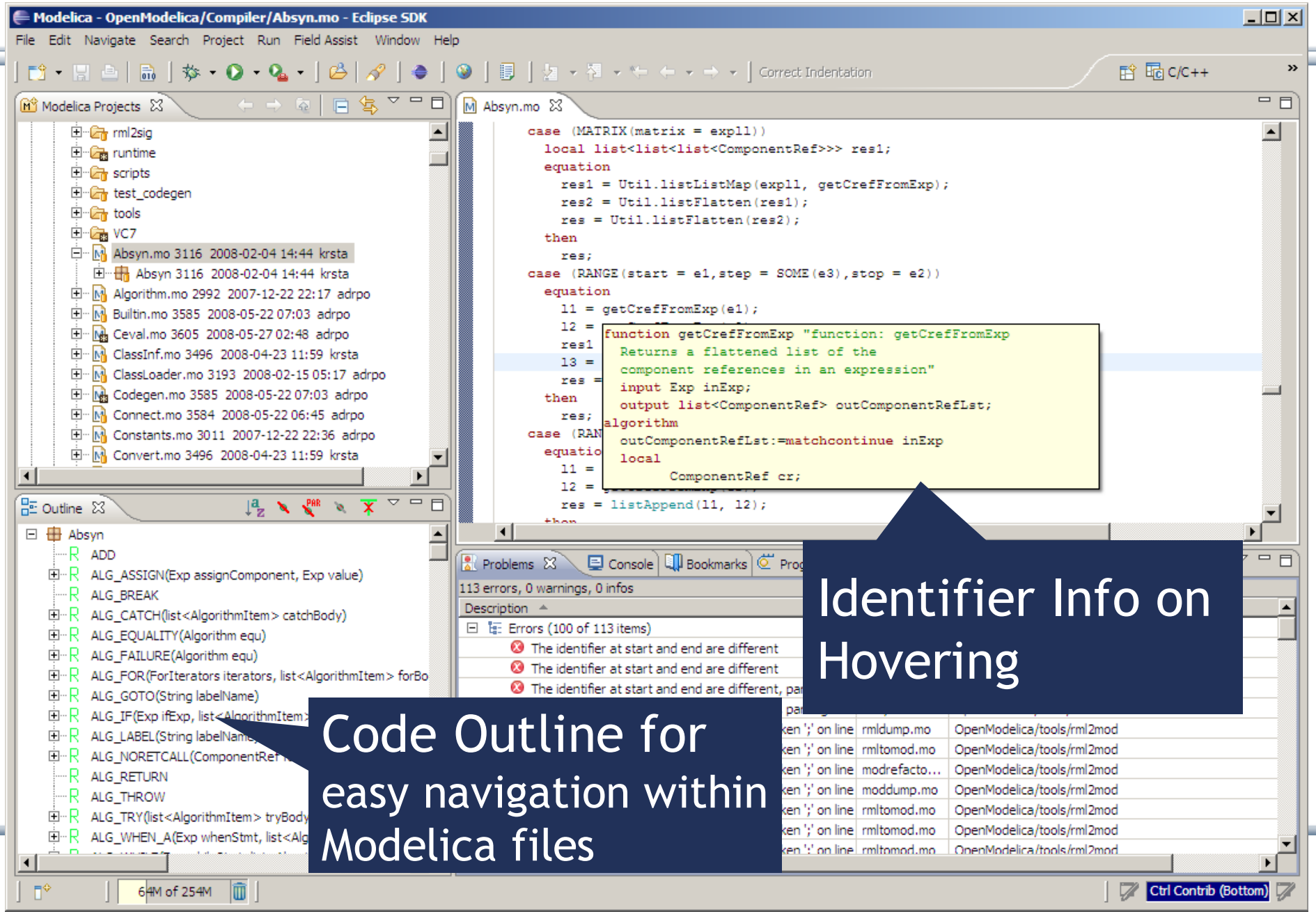
OpenModelica MDT – Eclipse Plugin

- Browsing of packages, classes, functions
- Automatic building of executables; separate compilation
- Syntax highlighting
- Code completion,
  Code query support for developers
- Automatic Indentation
- Debugger



MODELICA

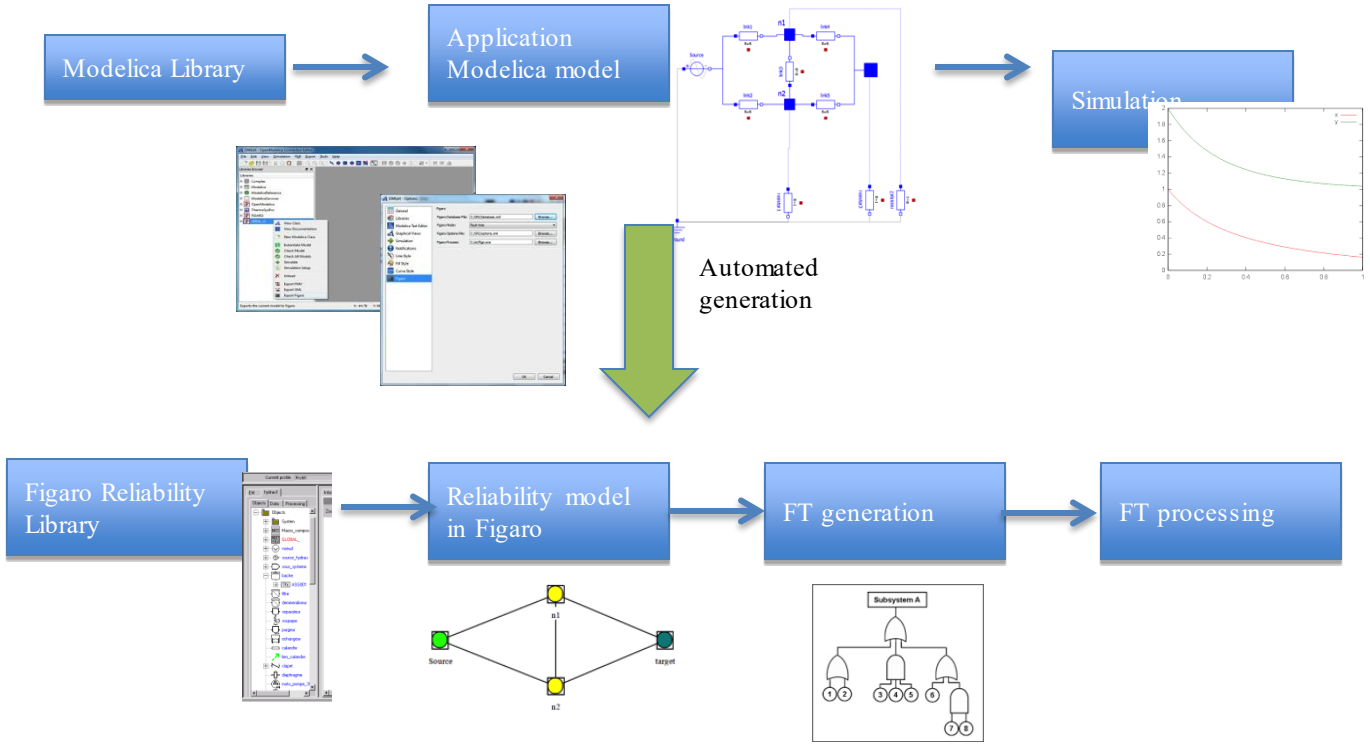
OpenModelica Eclipse MDT: Code Outline and Hovering Info



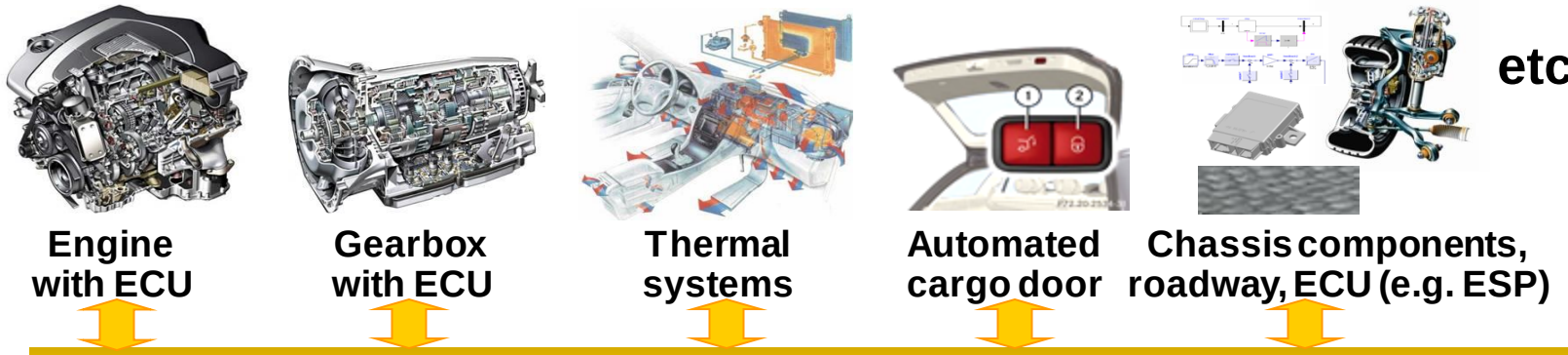
Model-based Failure Mode and Effects Analysis

(Marc Bouissou and Lena Buffoni)

- Modelica models augmented with reliability properties can be used to generate reliability models in Figaro, which in turn can be used for static reliability analysis
- Prototype in OpenModelica integrated with Figaro tool (which is becoming open-source)



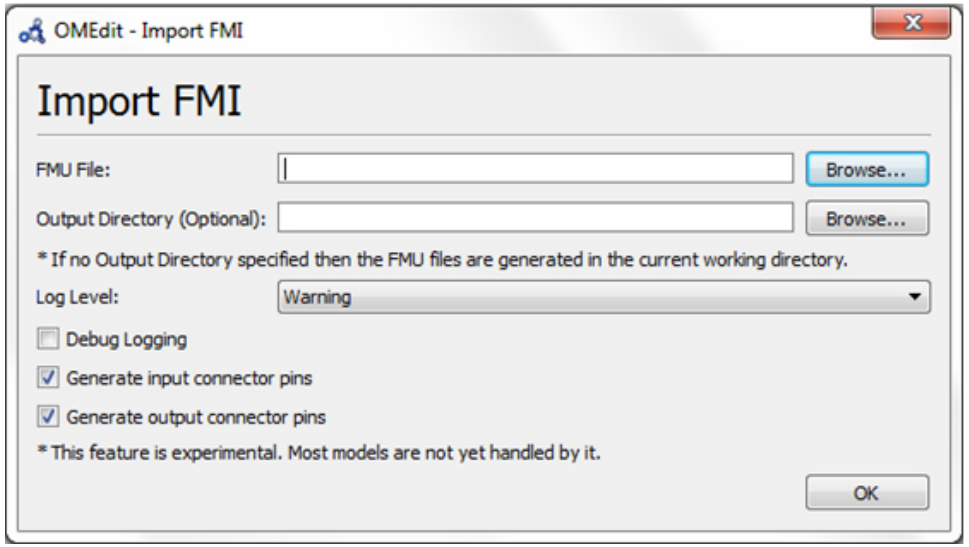
General Tool Interoperability & Model Exchange Functional Mock-up Interface (FMI)



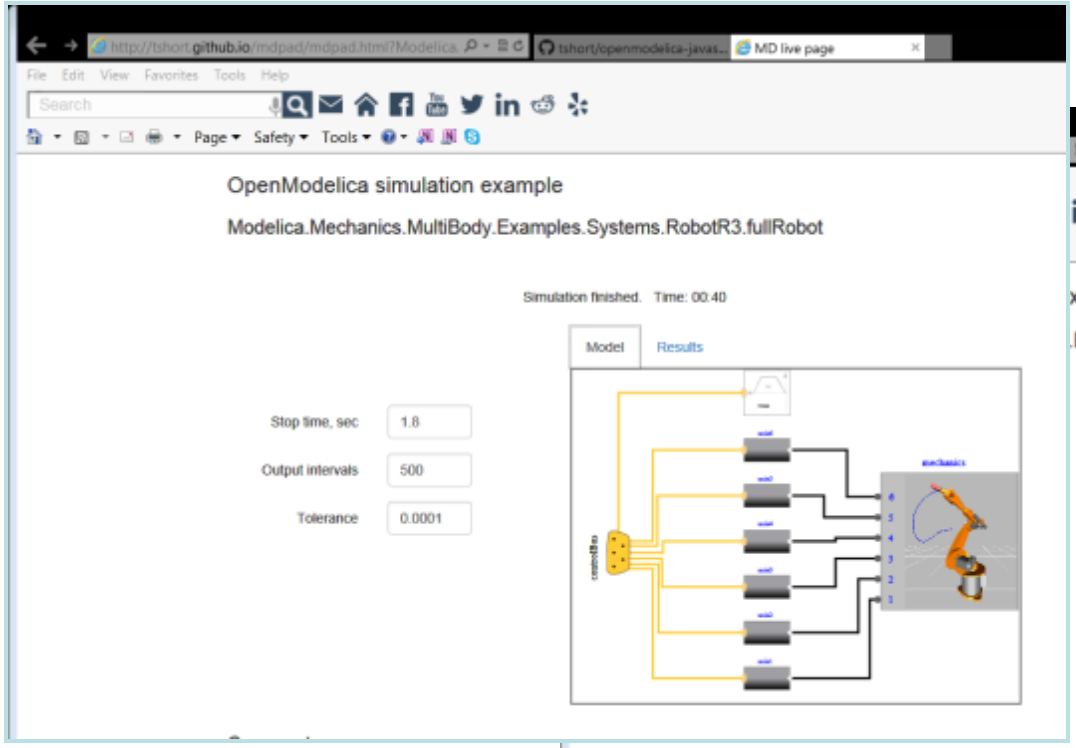
- FMI development was started by ITEA2 MODELISAR project. FMI is a Modelica Association Project now
- **Version 1.0**
- FMI for Model Exchange (released Jan 26,2010)
- FMI for Co-Simulation (released Oct 12,2010)
- **Version 2.0**
- FMI for Model Exchange and Co-Simulation (released July 25,2014)
- **> 60 tools** supporting it (https://www.fmi-standard.org/tools)

FMI in OpenModelica

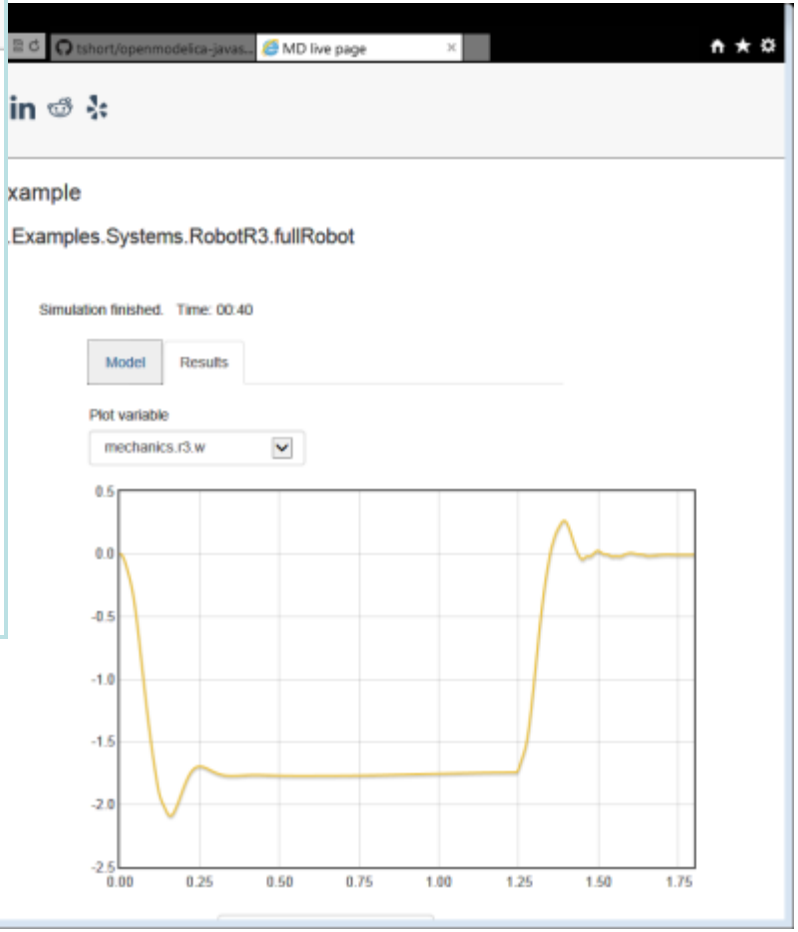
- FMI Model Exchange implemented (FMI 1.0 and FMI 2.0)
- A prototype of FMI 2.0 co-simulation is available
- Ongoing work to support full FMI 2.0 co-simulation
- The FMI interface is accessible via the **OpenModelica scripting environment** and the **OpenModelica connection editor**



OpenModelica Simulation in Web Browser Client



MultiBody RobotR3.FullRobot

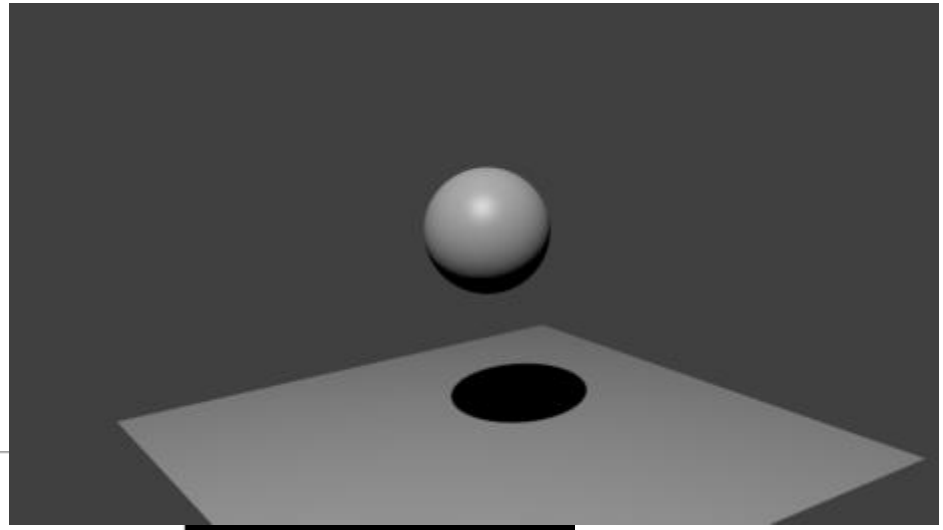


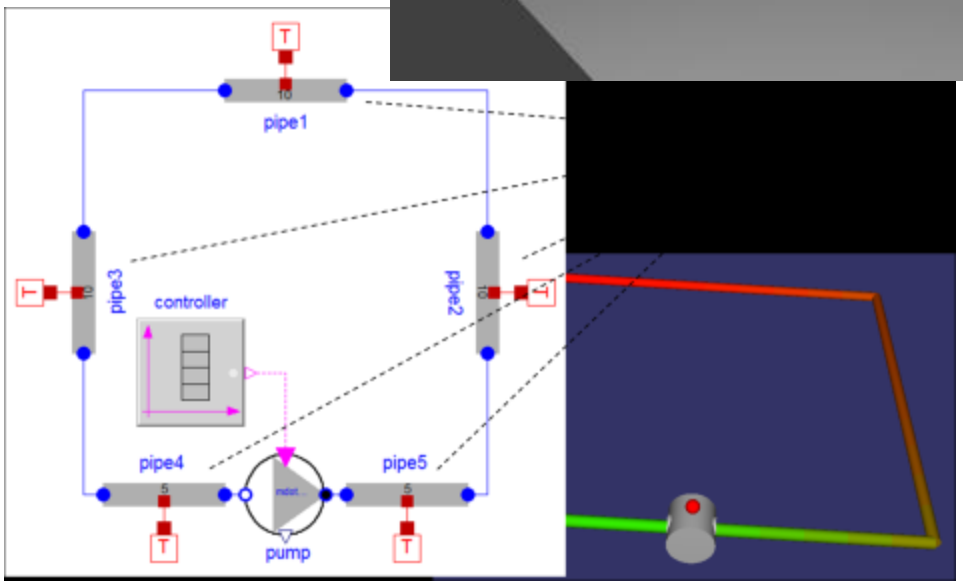
OpenModelica compiles to efficient Java Script code which is executed in web browser

MODELICA

Modelica3D Library with OpenModelica

- Modelica 3D Graphics Library by Fraunhofer FIRST, Berlin
- Part of OpenModelica distribution
- Can be used for 3D graphics in OpenModelica





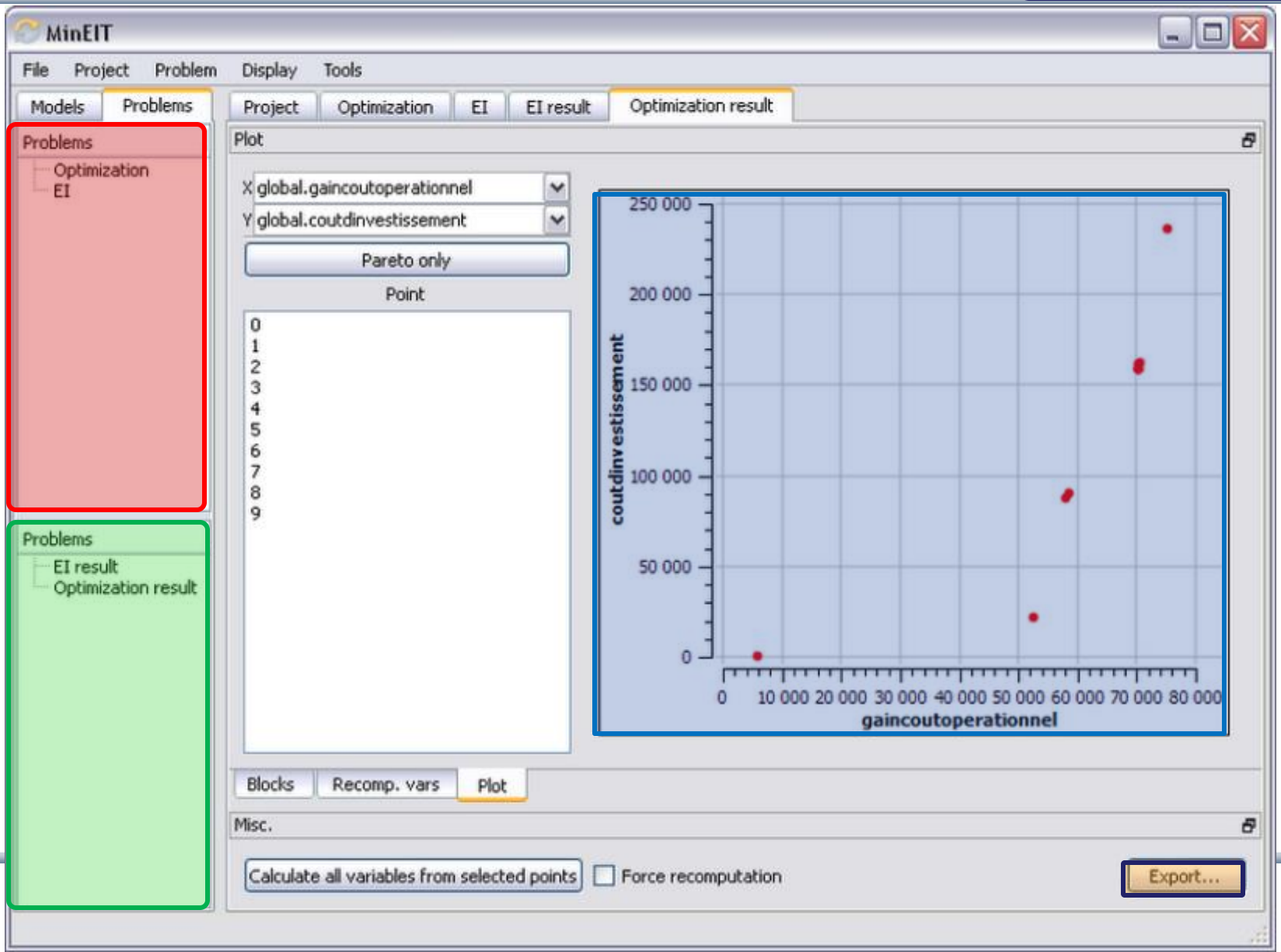
OMOptim – Parameter Sweep Design Optimization

Problems

Solved problems

Result plot

Export result data .csv



Here Pareto front optimiza-tion

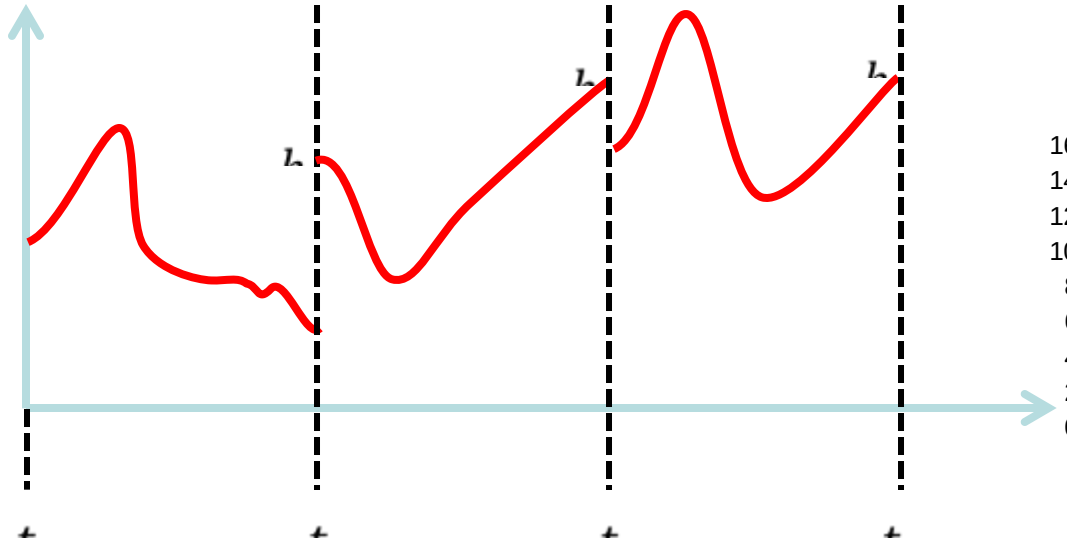
MODELICA

Optimization of Dynamic Trajectories Using Multiple-Shooting and Collocation

- Minimize a goal function subject to model equation constraints, useful e.g. for NMPC
- Multiple Shooting/Collocation
  - Solve sub-problem in each sub-interval

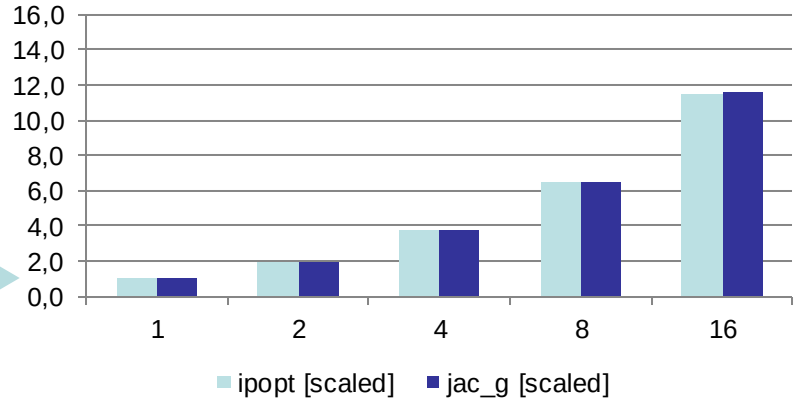
In OpenModelica 1.9.1 beta release Jan 2014.

$$x_i(t_{i+1}) = h_i + \int_{t_i}^{t_{i+1}} f(x_i(t), u(t), t)\, dt \approx F(t_i, t_{i+1}, h_i, u_i), \qquad x_i(t_i) = h_i$$



Example speedup, 16 cores:

MULTIPLE_COLLOCATION



MODELICA

OMnotebook Interactive Electronic Notebook Here Used for Teaching Control Theory

OMNotebook: Kalman.onb

File Edit Cell Format Insert Window Help

1 Kalman Filter

Often we don't have access to the internal states of
have to reconstruct the state of the system based o
The idea with an observer is that we feedback the
the estiamtion is correct then the difference should

Another difficulty is that the measured quantities oft

$$\begin{cases} \dot{\hat{x}} = \\ \end{cases}$$

Here are e denoting a disturbance in the input signa
be evaluated by the difference

$$K(y(t)$$

By using this quantity as feedback we obtain the ob

$$\dot{\hat{x}} = A\hat{x}(t) + Bu(t$$

Now form the error as

The differential error is

Ready

OMNotebook: Kalman.onb*

File Edit Cell Format Insert Window Help

```
model KalmanFeedback
  parameter Real A[:,size(A, 1)] = {{0,1},{1,0}} ;
  parameter Real B[size(A, 1),:] = {{0},{1}};
  parameter Real C[:,size(A, 1)] = {{1,0}};
  parameter Real[2,1] K = [2.4;3.4];
  parameter Real[1,2] L = [2.4,3.4];
  parameter Real[:,:] ABL = A-B*L;
  parameter Real[:,:] BL = B*L;
  parameter Real[:,:] Z = zeros(size(ABL,2),size(AKC,1));
  parameter Real[:,:] AKC = A-K*C;
  parameter Real[:,:] Anew = [0,1,0,0 ; -1.4, -3.4, 2.4,3.4; 0,0,-2.4,1;0,0,-2.4,0];
  parameter Real[:,:] Bnew = [0;1;0;0];
  parameter Real[:,:] Fnew = [1;0;0;0];
  stateSpaceNoise Kalman(stateSpace.A=Anew,stateSpace.B=Bnew,  stateSpace.C=[1,0,0,0],
stateSpace.F = Fnew);
  stateSpaceNoise noKalman;
end KalmanFeedback;
```

```
simulate(KalmanFeedback,stopTime=3)
plot({Kalman.stateSpace.y[1],noKalman.stateSpace.y[1]})
```

true

Ready Ln 12, Col 39

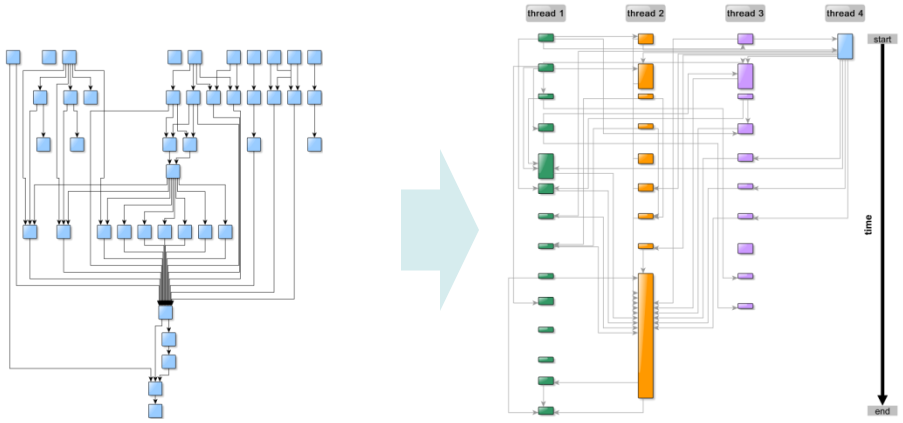
MODELICA

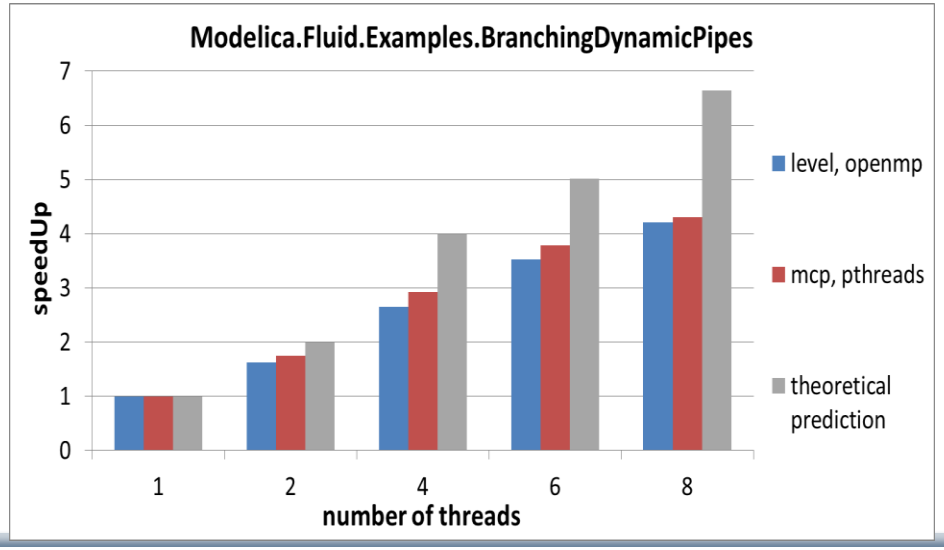
MetaModelica Language Extension for Model Transformations and Advanced Applications

- **Large-scale** existing application – OpenModelica compiler written in MetaModelica, compiling itself
- MetaModelica language extension
  - single assignment equations (with opt. patterns)
  - **tree** data structures, garbage collection
  - **pattern** equations
  - **matching**, backtracking
  - Very **efficient portable** implementation (compiles to C)
- Now ongoing **standardization** in Modelica Association

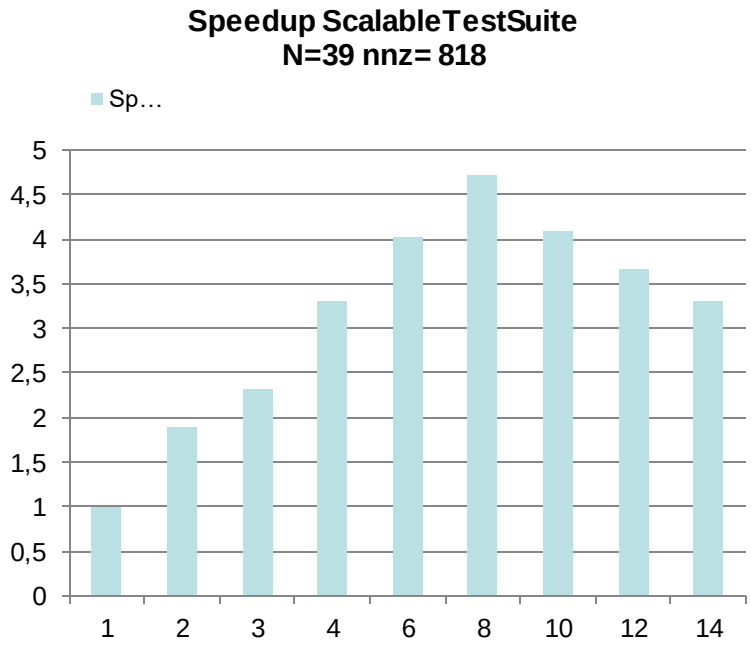
OpenModelica Model Parallelization
Faster Simulation on Multi-Core

Automated parallelization of models





Parallelizing numeric Jacobian computations in simulation

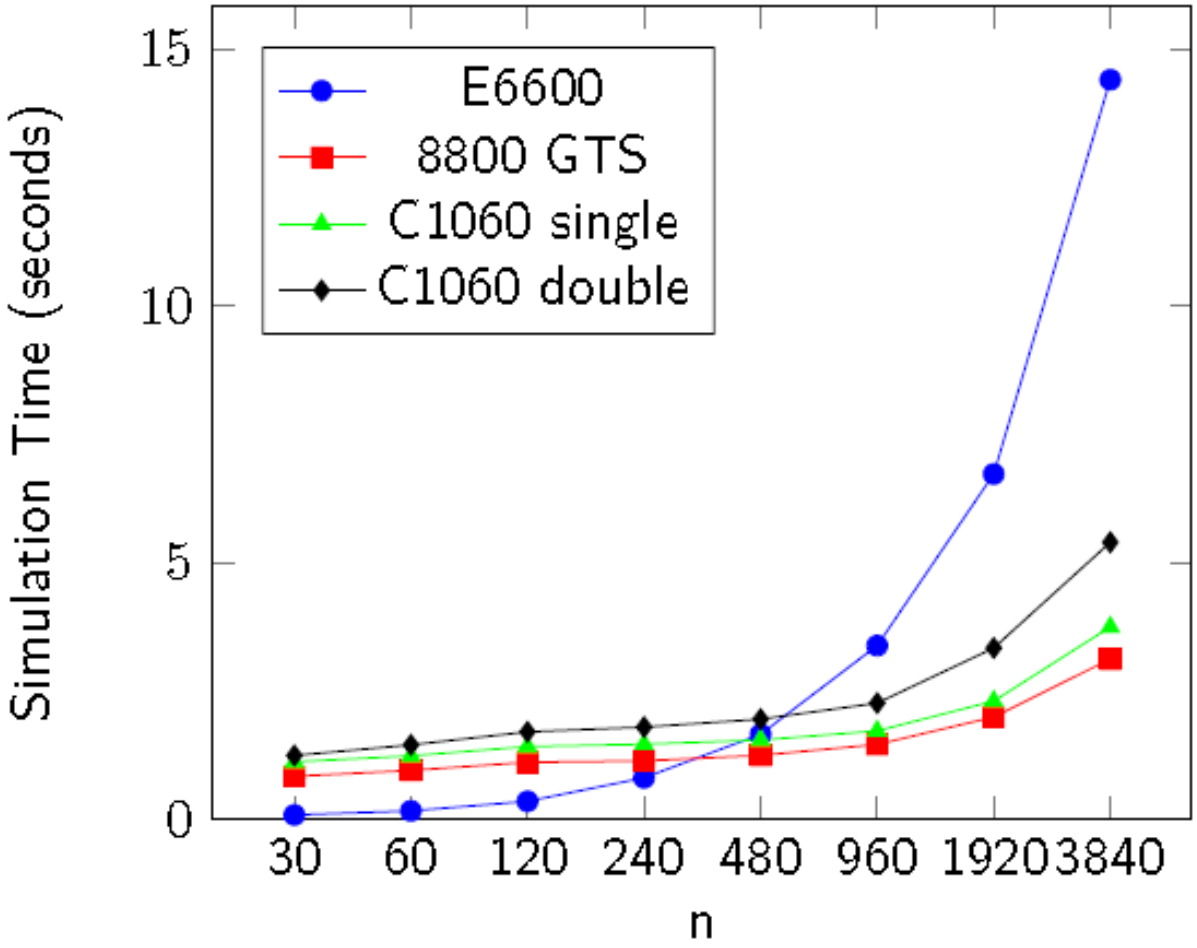


Speedup about 4 using 8 threads

MODELICA

Faster Simulation – Compiling Modelica to Multi-Core

Speedup on NVIDIA, Modelica Model, Generated Code, n Problem Size



Recent Large-scale ABB OpenModelica Application
Generate code for controlling 7.5 to 10% of German Power Production

ABB OPTIMAX PowerFit

- Real-time optimizing control of large-scale virtual power plant for system integration
- **Software including OpenModelica** now used in managing more than 2500 renewable plants, total up to 1.5 GW

High scalability supporting growth

- 2012: initial delivery (for 50 plants)
- 2013: SW extension (500 plants)
- 2014: HW+SW extension (> 2000)
- 2015: HW+SW extension, incl. OpenModelica generating optimizing controller code in FMI 2.0 form

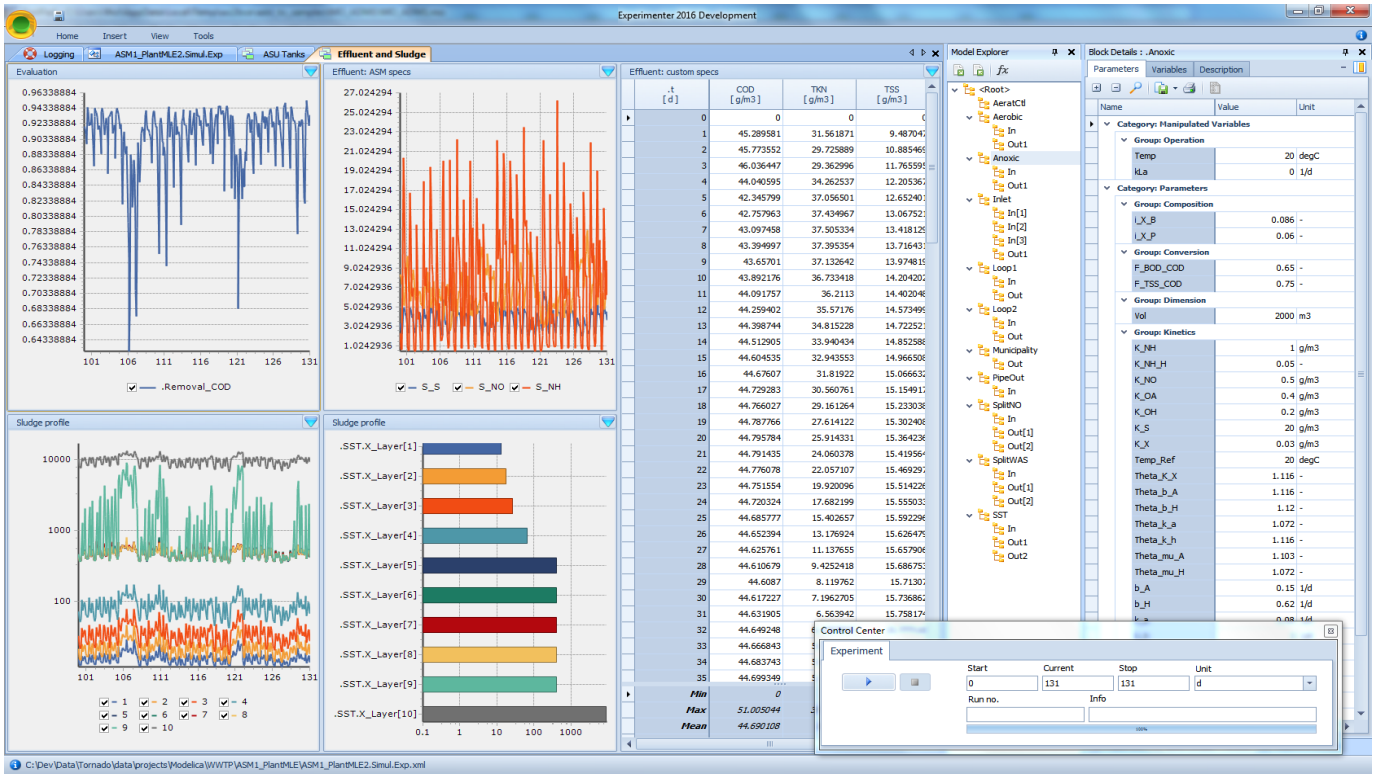
Manage 7.5% - 10% of German Power

- 2015, Aug: OpenModelica Exports FMUs for real-time optimizing control (seconds) of about **5.000 MW (7.5%) of power in Germany**

MODELICA

Industrial Product with OEM Usage of OpenModelica – MIKE by DHI, WEST Water Quality

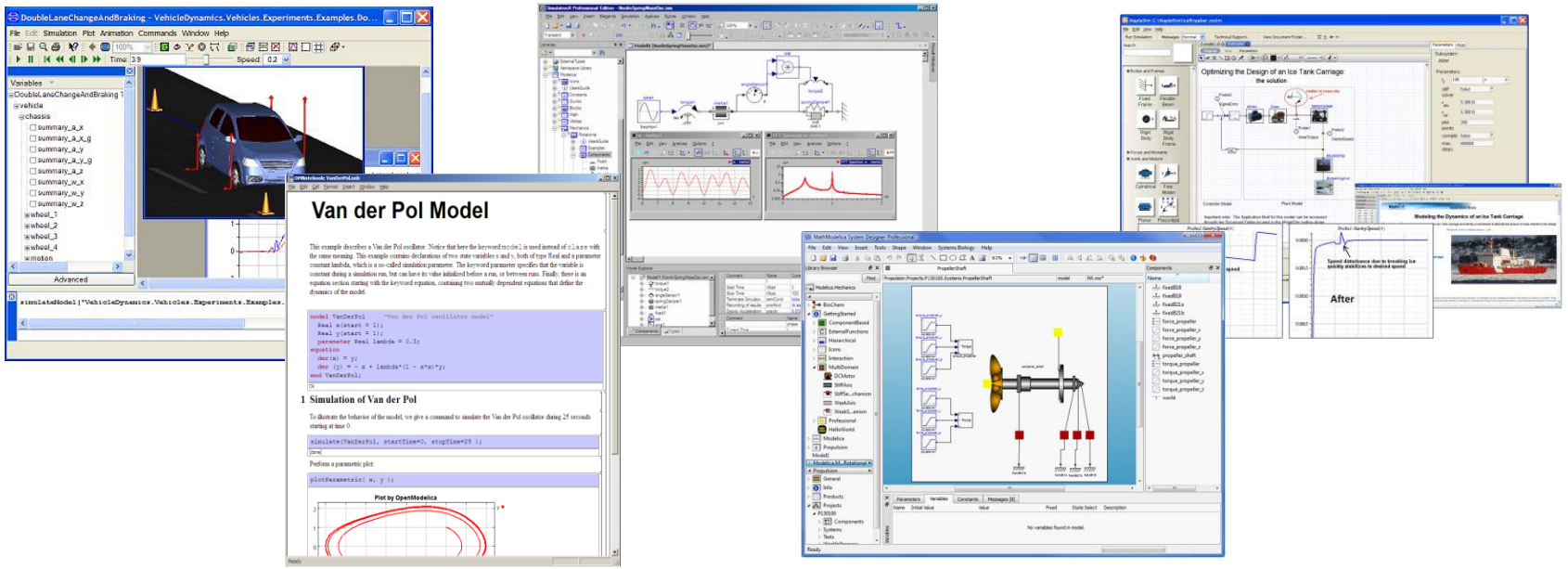
- **MIKE by DHI**, www.mikebydhi.com, **WEST Water Quality** modeling and simulation environment
- Includes a large part of the OpenModelica compiler using the OEM license.
- Here a water treatment effluent and sludge simulation.



Part III

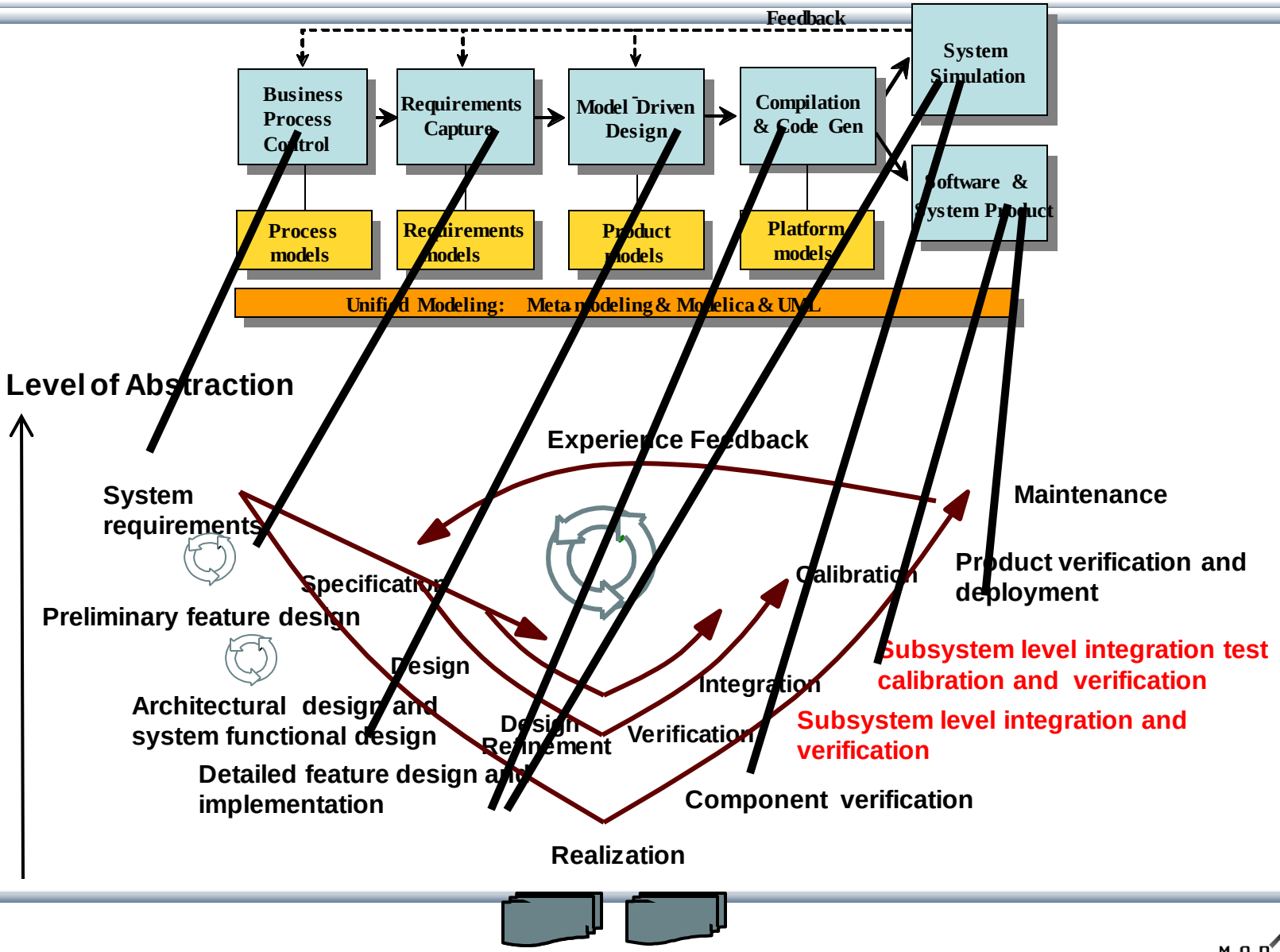
Dynamic Verification/Testing of Requirements vs Usage Scenario Models

Wladimir Schamai, Lena Buffoni, Peter Fritzson and contributions from MODRIO partners



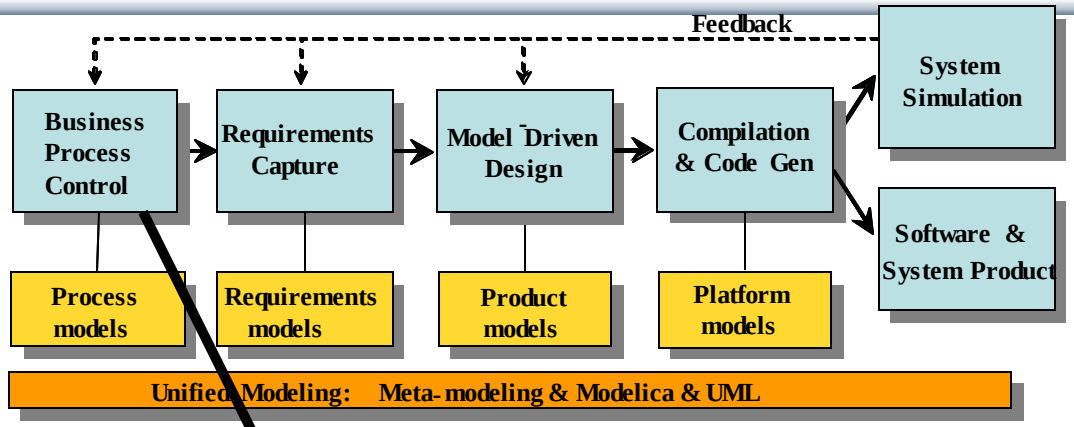
MODELICA

OpenModelica and Papyrus Based Model-Based Development Environment to Cover Product-Design V



MODELICA

Business Process Control and Modeling



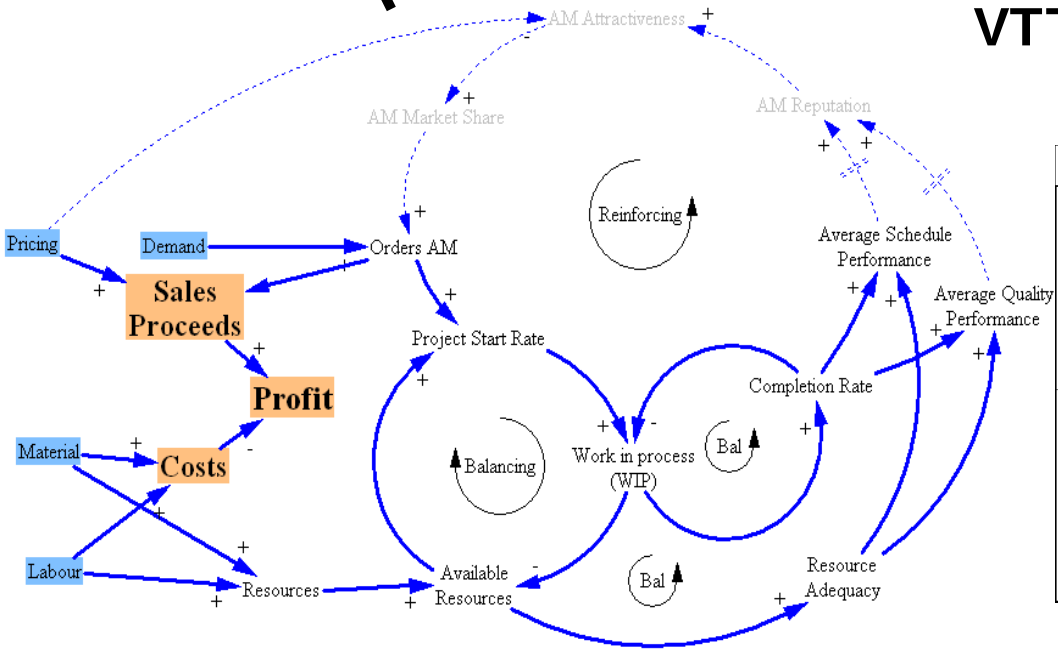
VTT Simantics
Business process modeler

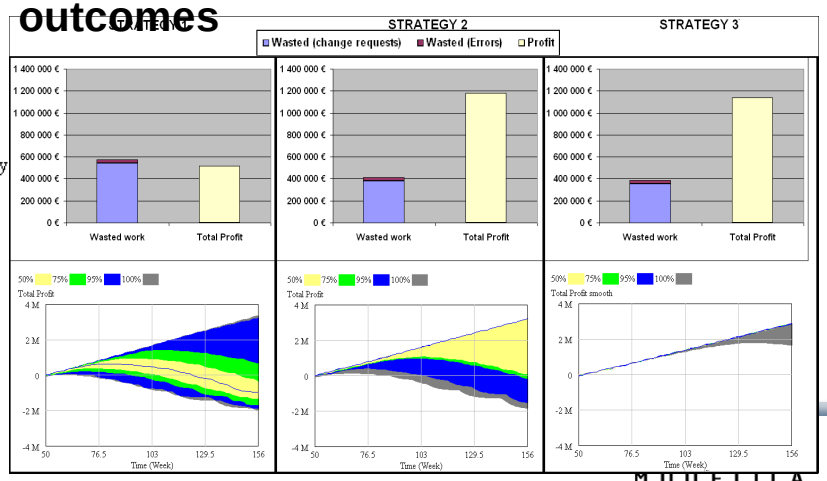
OpenModelica
compiler & simulator

OpenModelica based simulation

Metso Business model & simulation
VTT Simantics Graphic Modeling Too

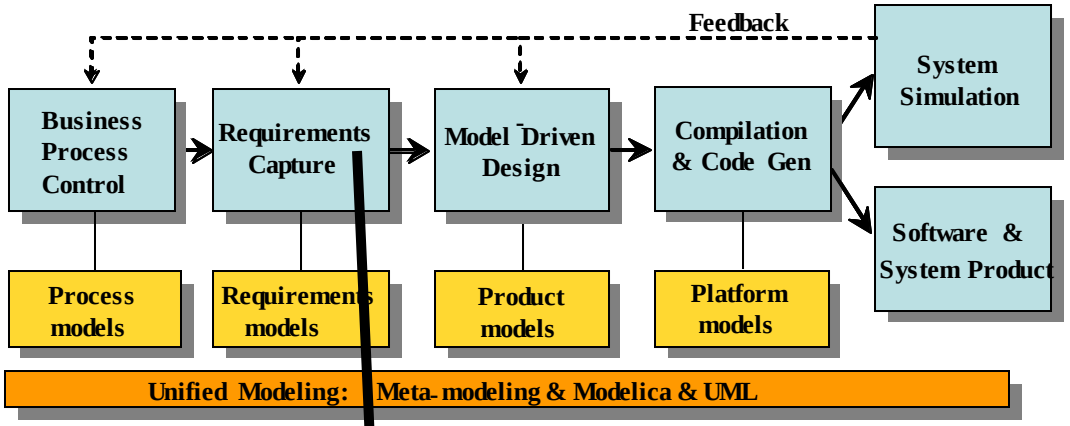
Simulation of 3 strategies with outcomes





MODELICA

Requirement Capture



vVDR (virtual Verification of Designs against Requirements)

in ModelicaML UML/Modelica Profile, part of OpenModelica

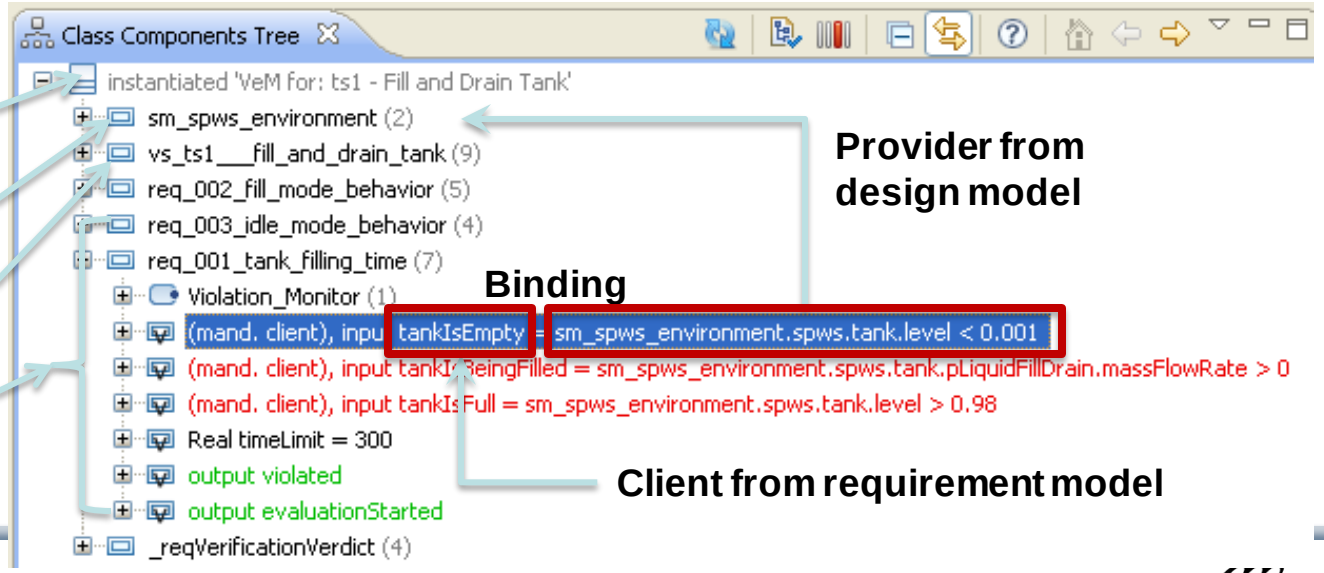
OpenModelica based simulation

Verification Model

Design Model

Scenario Model

Requirement Models



OpenModelica – ModelicaML UML Profile
Based on Open-Source Papyrus UML and OpenModelica

- ModelicaML is a UML Profile for SW/HW modeling
  - Applicable to "pure" UML or to other UML profiles, e.g. SysML
- Standardized Mapping UML/SysML to Modelica
  - Defines transformation/mapping for **executable** models
  - Being **standardized** by OMG
- ModelicaML
  - Defines graphical concrete syntax (graphical notation for diagram) for representing Modelica constructs integrated with UML
  - Includes graphical formalisms (e.g. State Machines, Activities, Requirements)
    - Which do not yet exist in Modelica language (extension work ongoing)
    - Which are translated into executable Modelica code
  - Is defined towards generation of executable Modelica code
  - Current implementation based on the Papyrus UML tool + OpenModelica

Example: Simulation and Requirements Evaluation

«model»
(TwoTanksSystemExample::SystemSimulations)
TankSystemSimulation

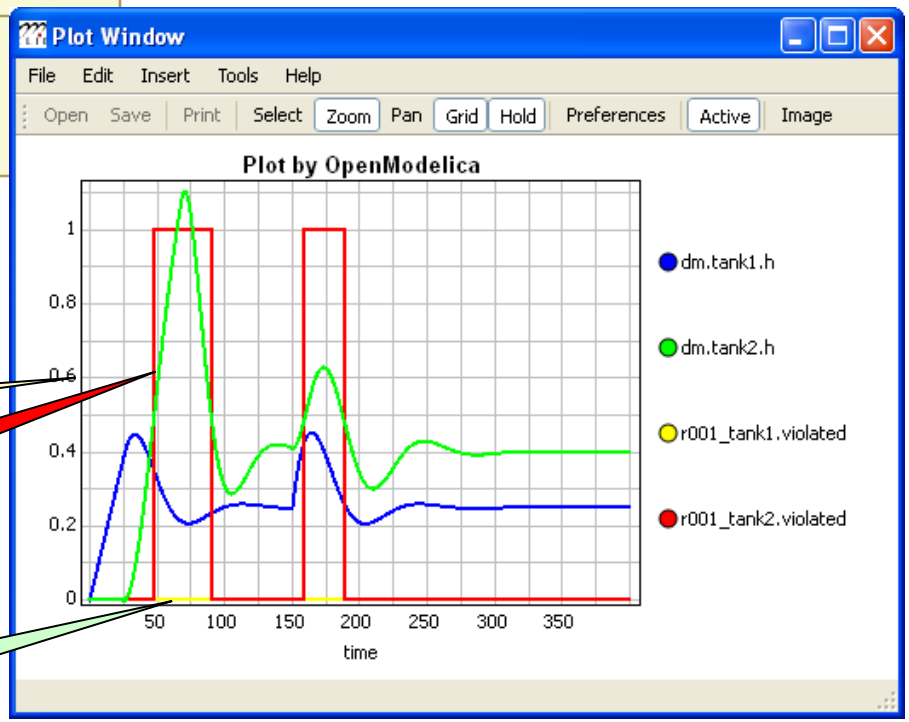
- «component» dm: TanksConnectedPI
- «requirementInstance» r001_tank1: Max level of liquid in a tank
- «requirementInstance» r001_tank2: Max level of liquid in a tank
- «requirementInstance» r002_tank1: Volume of the tank1

Req. 001 is instantiated 2 times (there are 2 tanks in the system)

tank-height is 0.6m

Req. 001 for the tank2 is violated

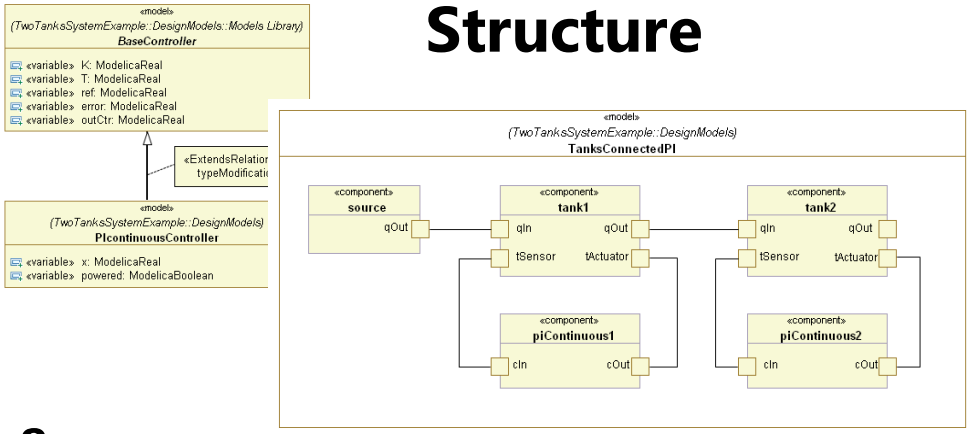
Req. 001 for the tank1 is not violated



MODELICA

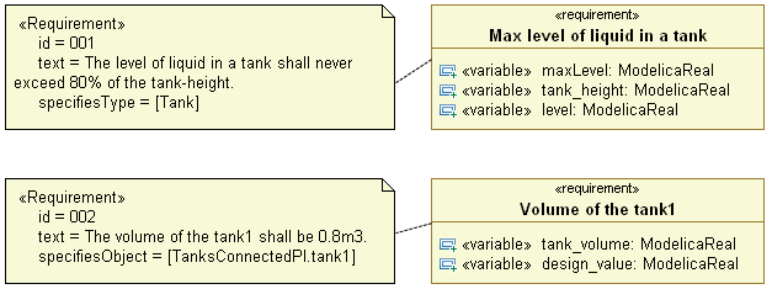
ModelicaML: Graphical Notation

Structure

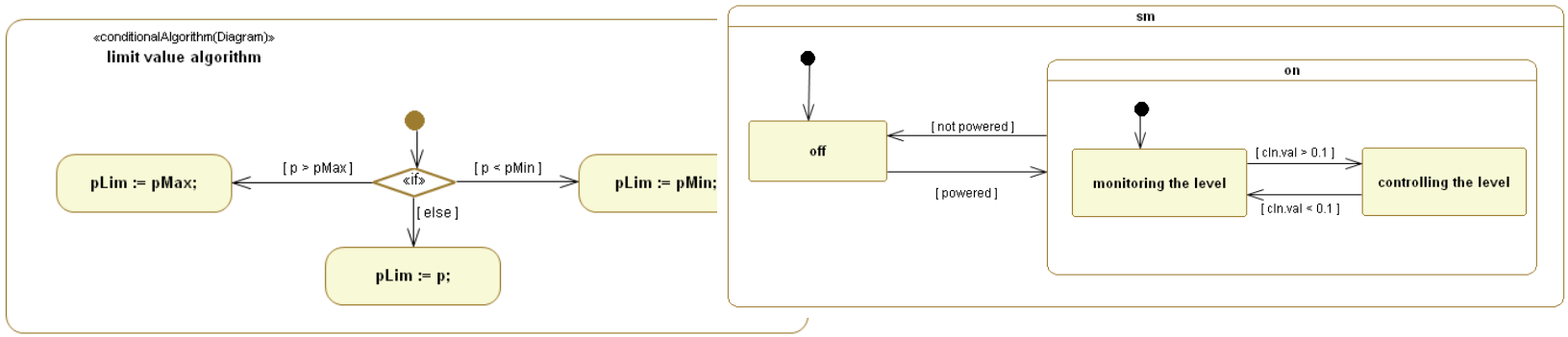


a

Requirements

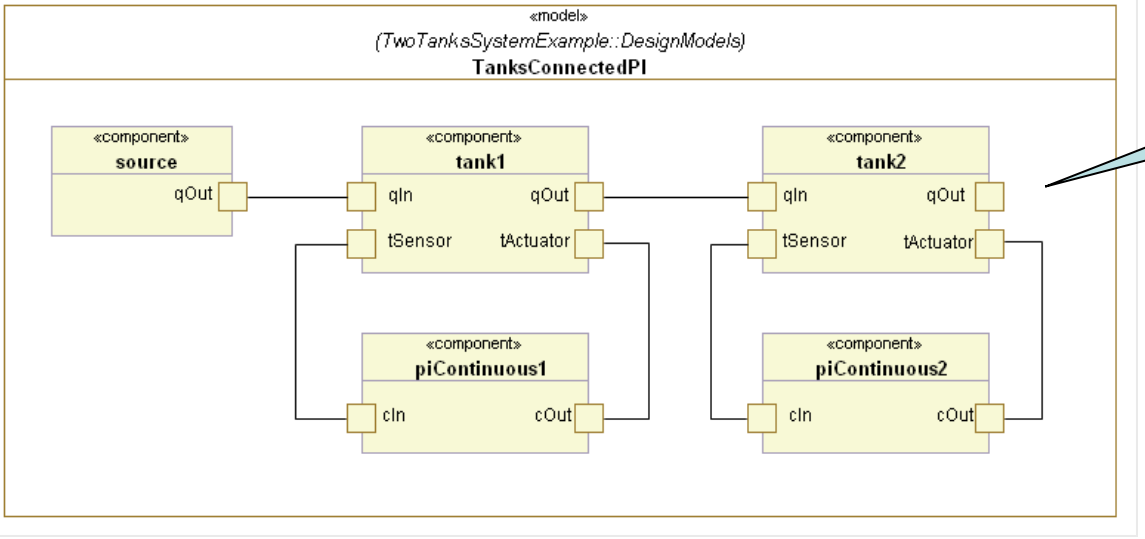


Behavior



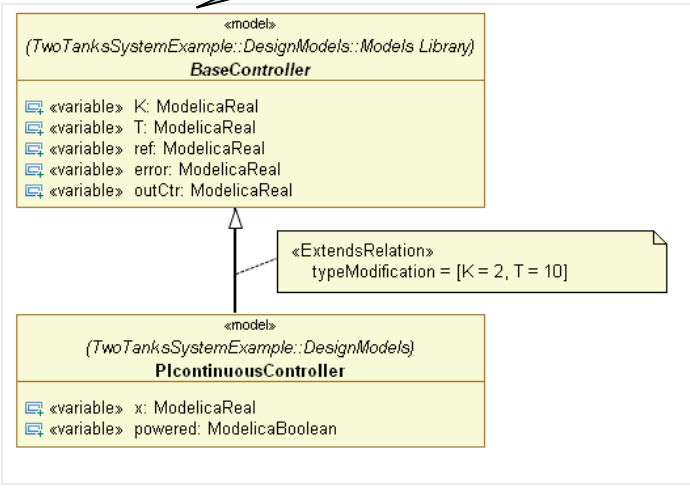
MODELICA

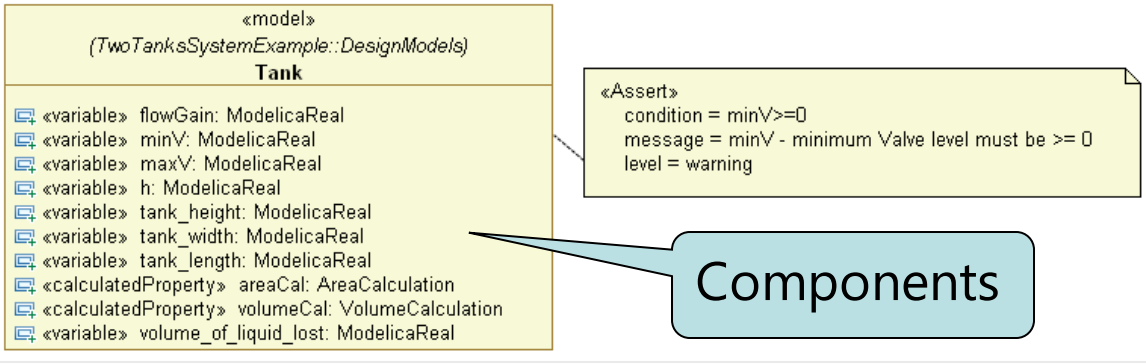
Example: Representation of System Structure



Interconnections

Inheritance

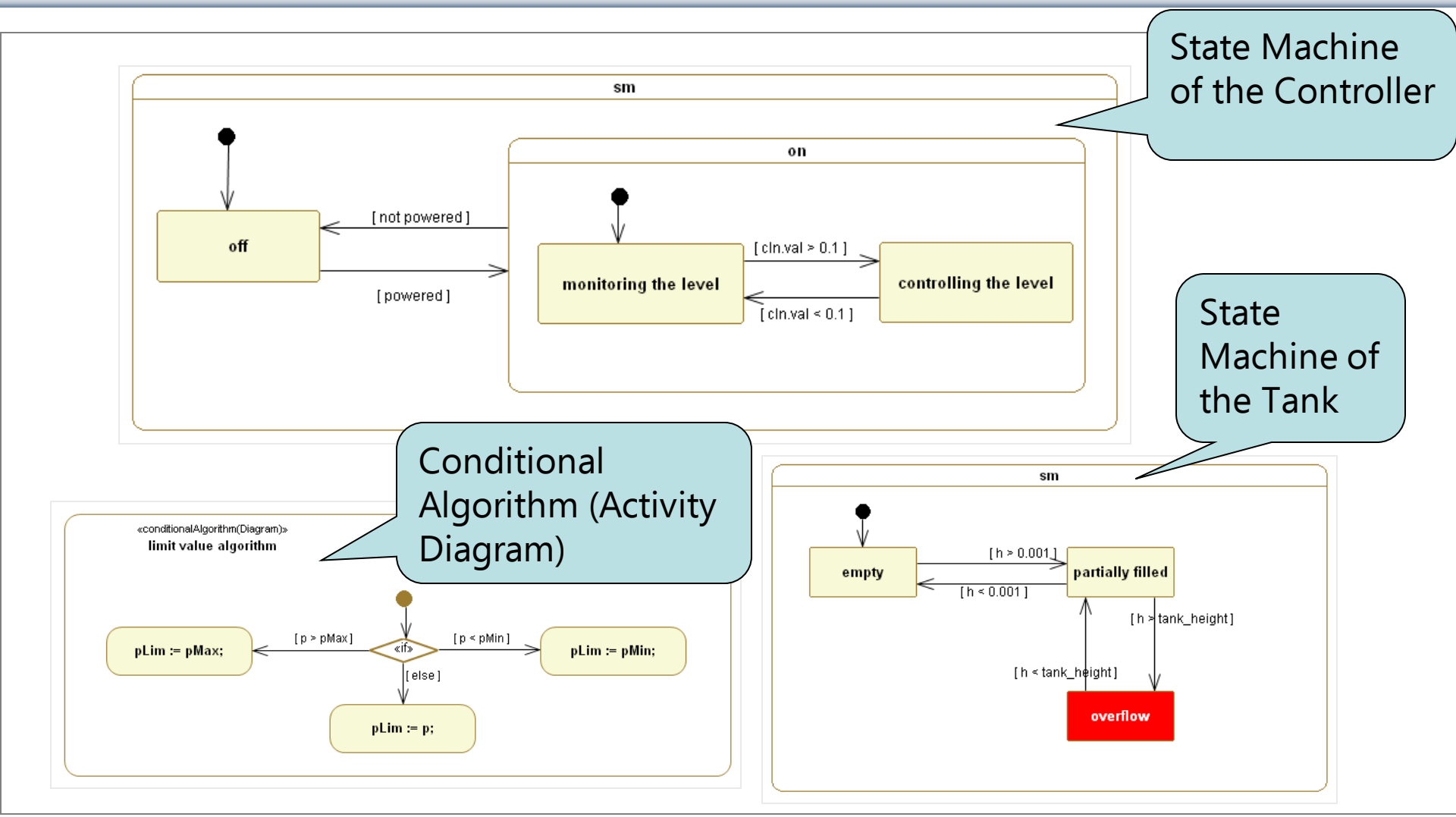




Components

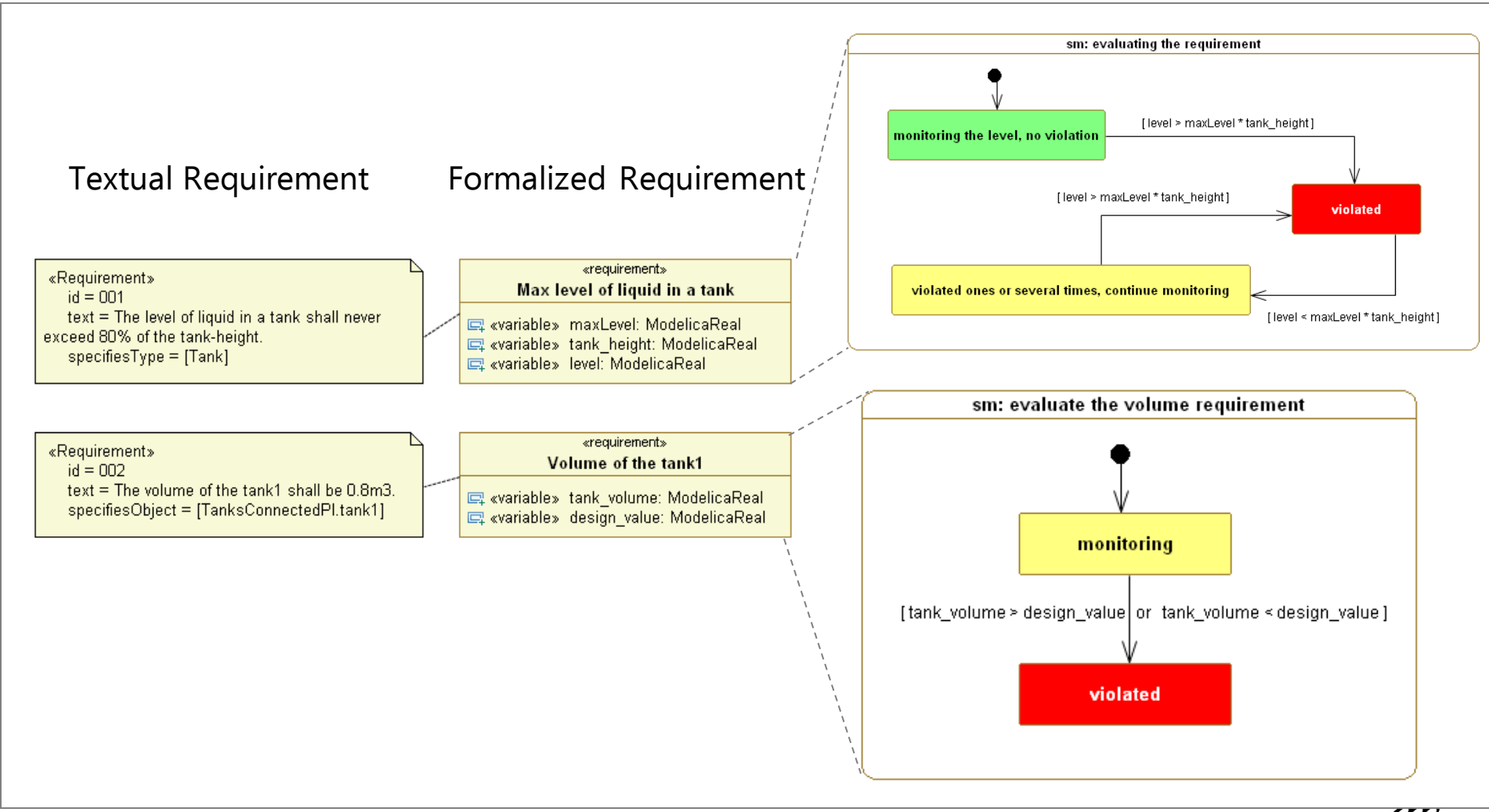
MODELICA

Example: Representation of System Behavior



MODELICA

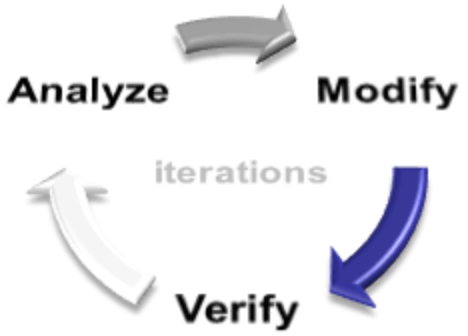
Example: Representation of System Requirements



MODELICA

vVDR Method – virtual Verification of Designs vs Requirements

Actor	Task	Created Artifact
	Formalize Requirements	RMM — Requirement Monitor Models
	Formalize Designs	DAM — Designs Alternative Models
	Formalize Scenarios	SM — Scenario Models
AUTOMATED	Create Verification Models *	VM — Verification Models
AUTOMATED	Execute and Create Report	Reports
	Analyze Results	



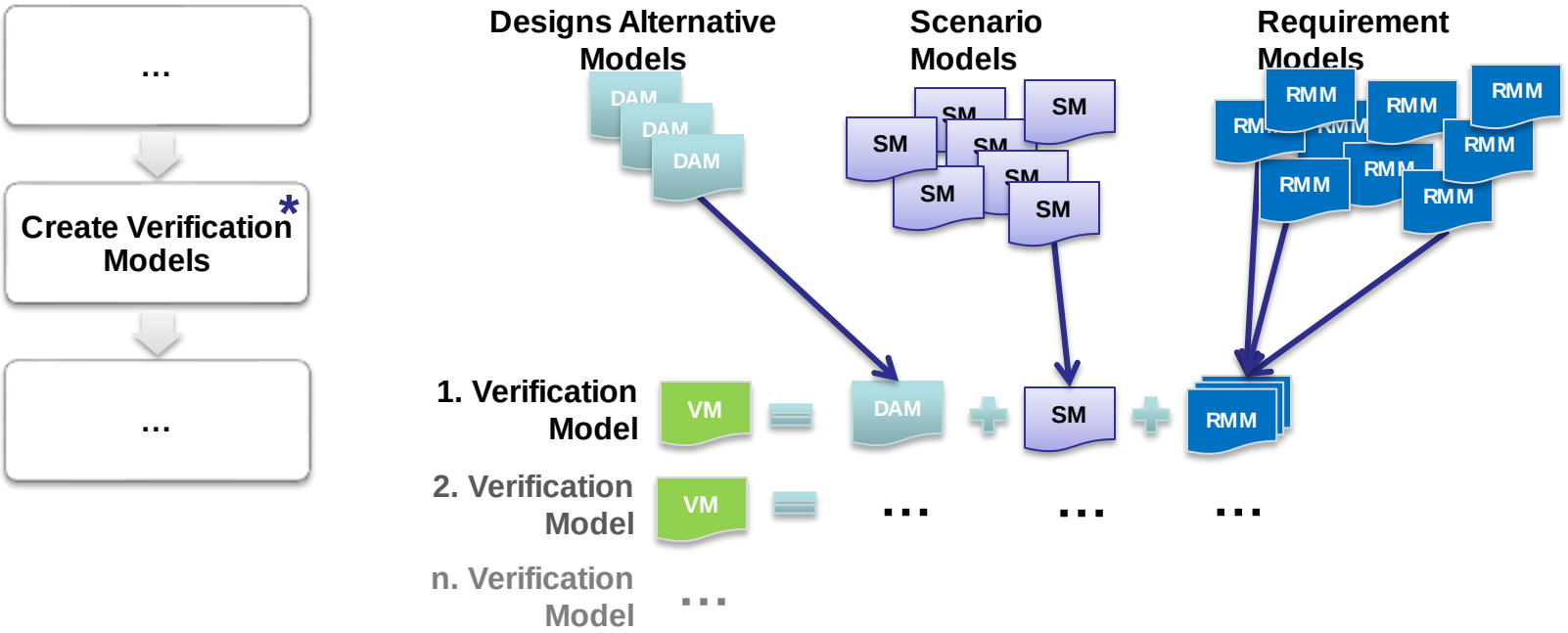
Goal: Enable on-demand verification of designs against requirements using automated model composition at any time during development.

MODELICA

Challenge

We want to verify **different design alternatives** against **sets of requirements** using **different scenarios**. Questions:

1) How to **find valid combinations** of **design alternatives**, **scenarios** and **requirements** in order to enable an automated composition of verification models?
2) Having found a valid combination: How to **bind all components correctly**?



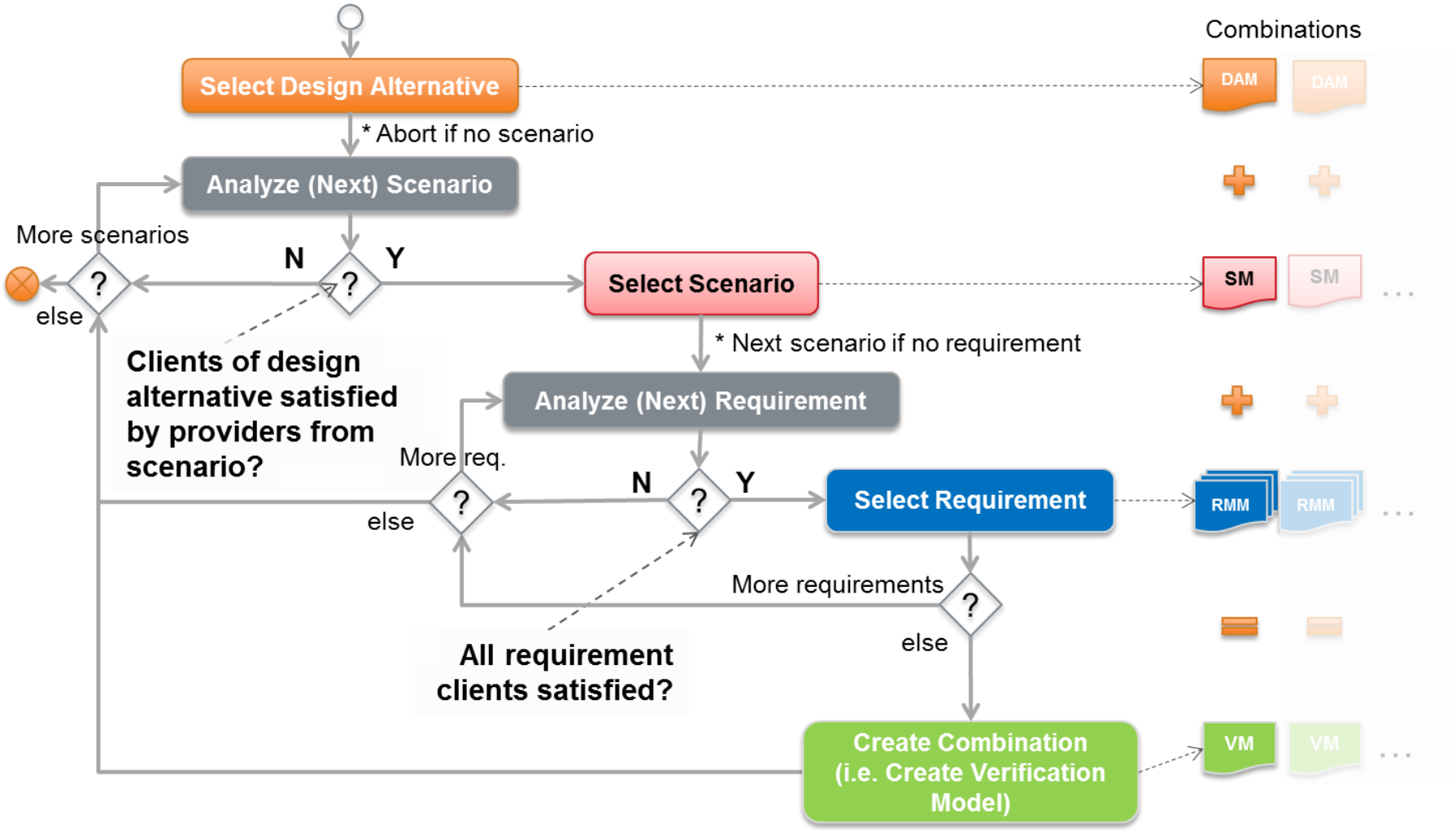
Composing Verification Models
main idea

- Collect all **scenarios**, **requirements**, import **mediators**
- Generate/compose *verification models* automatically:
  - Select the **system model** to be verified
  - Find all **scenarios** that can stimulate the selected system model (i.e., for each mandatory client check whether the binding expression can be inferred)
  - Find **requirements** that are implemented in the selected system model (i.e., **check** whether for **each requirement** for all mandatory clients binding expressions can be inferred)

- Present the list of scenarios and requirements to the user
  - The user can select only a subset or scenarios or requirements he/she wishes to consider

Generating/Composing Verification Models

algorithm



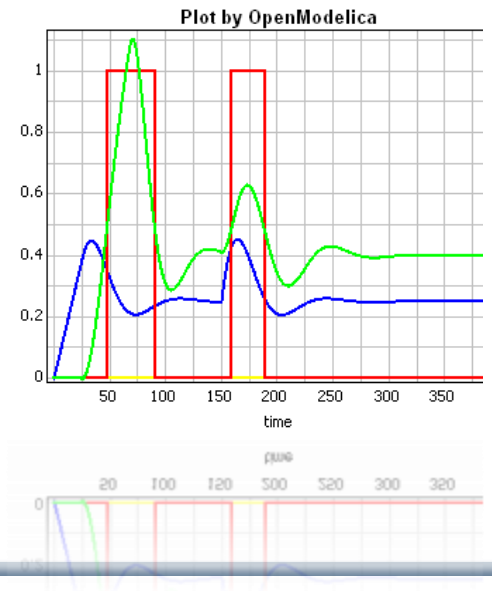
MODELICA

Simulation and Report Generation in ModelicaML

Verification models are simulated.

The generated **Verification Report** is a prepared summary of:

- Configuration, bindings
- Violations of requirements
- etc.



Verification models number (**3**), executed (**3**), passed (**0**), failed (**3**)

Failed VeM for: s1-Fill and Drain Tank (Plot)
Failed VeM for: s2-Fill tank (Plot)
Failed VeM for: s3-Drain tank (Plot)

Failed VeM for: s1-Fill and Drain Tank (Plot)
(ModelicaMLModel::GenVeMs for: SPWS Environment_1::VeM for: s1-Fill and Drain Tank)

Settings: startTime = 0, **stopTime = 1500**, tolerance = default, intervals = 0, outputFormat = plt

verdict allRequirementsEvaluated : **yes**
verdict someRequirementsViolated : **yes**

Model to be verified: SPWS Environment
(ModelicaMLModel::Design::SPWS Environment)

Verification Scenario: s1-Fill and Drain Tank
(ModelicaMLModel::Verification Scenarios::s1-Fill and Drain Tank)

madantory client: vs_s1_fill_and_drain_tank.tankHeight (**changed its value**)
Type : = ModelicaReal
Variability : = continuous
Binding code : = sm_spws_environment.spws.tank.height

Violated Requirement: Drain mode behavior (ID 004)
(ModelicaMLModel::Requirements::Drain mode behavior)
Text: When the system is drained only the fill/drain valve should be open, all other valves should be closed.

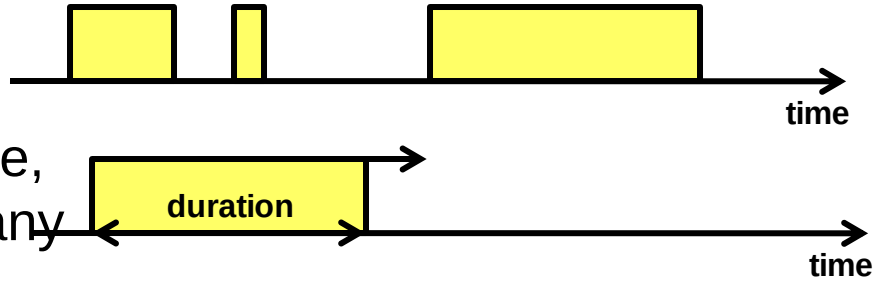
verdict evaluated : **yes**
verdict violated : **yes**

madantory client: req_004_drain_mode_behavior.fillDrainValveIsOpen (**changed its value**)
Type : = ModelicaBoolean
Variability : = continuous
Binding code : = sm_spws_environment.spws.fillDrainValve.isFullyOpen

madantory client: req_004_drain_mode_behavior.otherValvesAreClosed (**changed its value**)
Type : = ModelicaBoolean
Variability : = continuous
Binding code : = if sm_spws_environment.spws.overFlowValve.isFullyClosed and sm_spws_environment.spws.supplyVavle.isFullyClosed then true else false

Continuous and Discrete Time Locators for Time-related Requirements

- A Continuous Time Locator(CTL) specifies one or more time intervals
  - Time intervals have a duration
  - They usually have a position in time, but a sliding time window defines any time period of a given duration



- A Discrete Time Locator (DTL) defines one or more positions in time and has no duration
  - An event is associated with a DTL that specifies when the event occurred
  - The difference between events and DTLs is that a DTL is not an object
  - That position may be relative to the initialisation of the system or to another DTL

Time Locators Expressed in Modelica

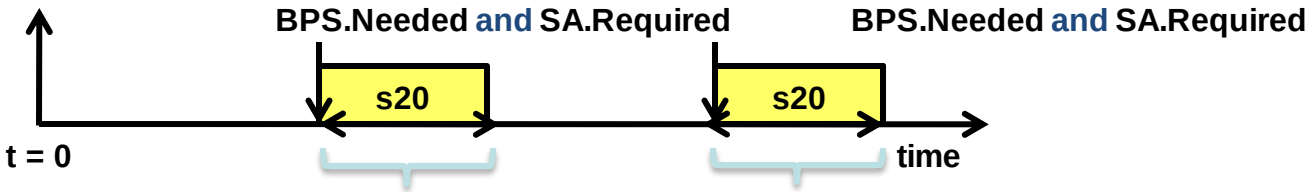
Special FORML-L syntax	Standard Modelica syntax
duringAny duration	**duringAny**(duration)
after event	**after**(event)
after event1 ***untilNext*** event2	**afterUntil**(event1, event2)
after event ***for*** duration	**afterFor**(event, duration)
after event ***within*** duration	**afterWithin**(event, duration)
until event	**until**(event)
every duration1 ***for*** duration2	**everyFor**(duration1, duration2)
when condition ***changes***	Maps to Modelica **if**

From Text to Simulated Requirement – Modelica Extended with new Operators

From a text requirement expressing a condition:

A - In the absence of any Backup Power Supply (BPS) component failure or in the presence of a single sensor failure, when the BPS is not under maintenance, in case of loss of MPS, and if safety injection is required, Set1 must be powered within 20 s

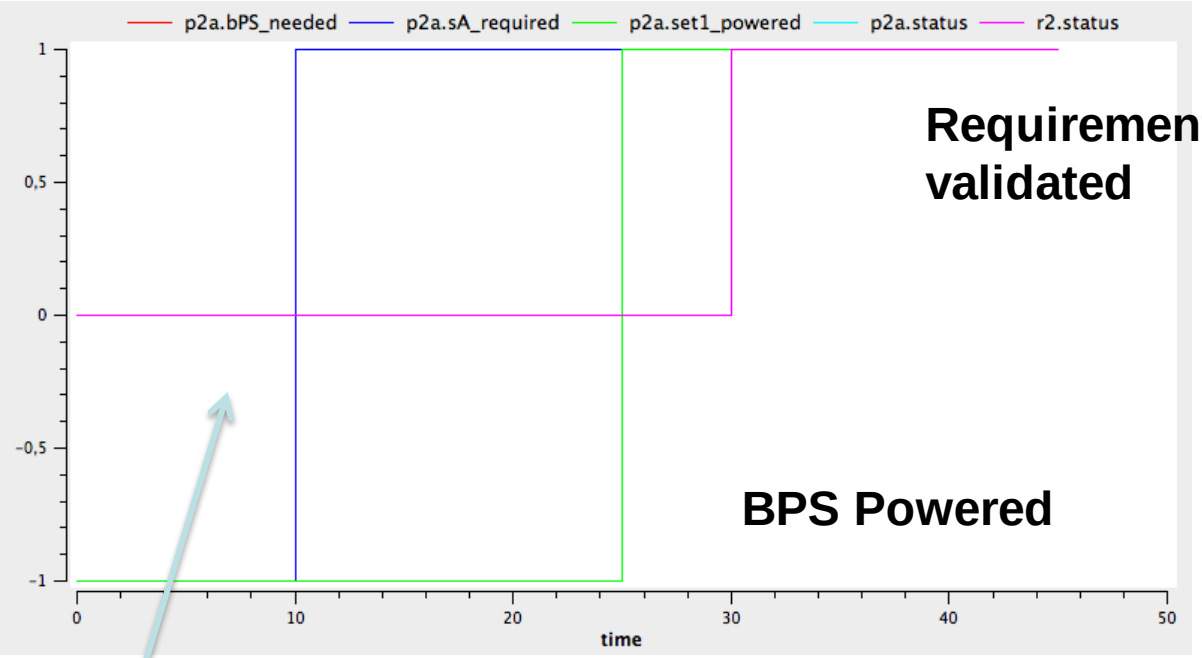
```
model P2a   extends Condition;
   input ConditionStatus bPSNeeded, sARequired, set1Powered;
equation
   status = if afterWithin (bPSNeeded == notViolated and
                                    sARequired == notViolated, 20) then
            if set1Powered == notViolated then
                  notViolated else violated else undefined;
end P2a;
```



Set1.Powered must become true within the timeframe s20 and remain true afterwards

MODELICA

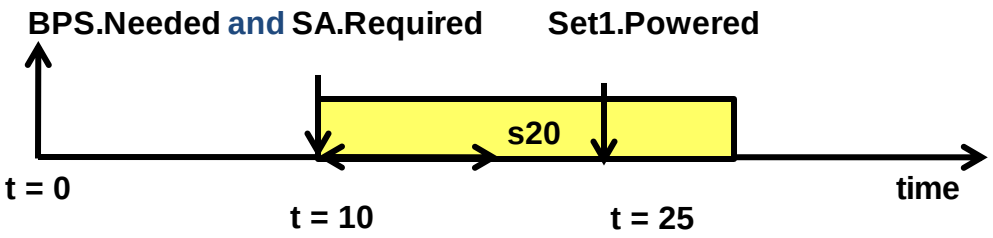
From Text to Simulated Requirement – Requirement not Violated – OpenModelica Simulation



3-valued logic prototype:
1 – true
0 – false
-1 – undefined

Requirement undefined outside the specified time window

Within 20s



MODELICA

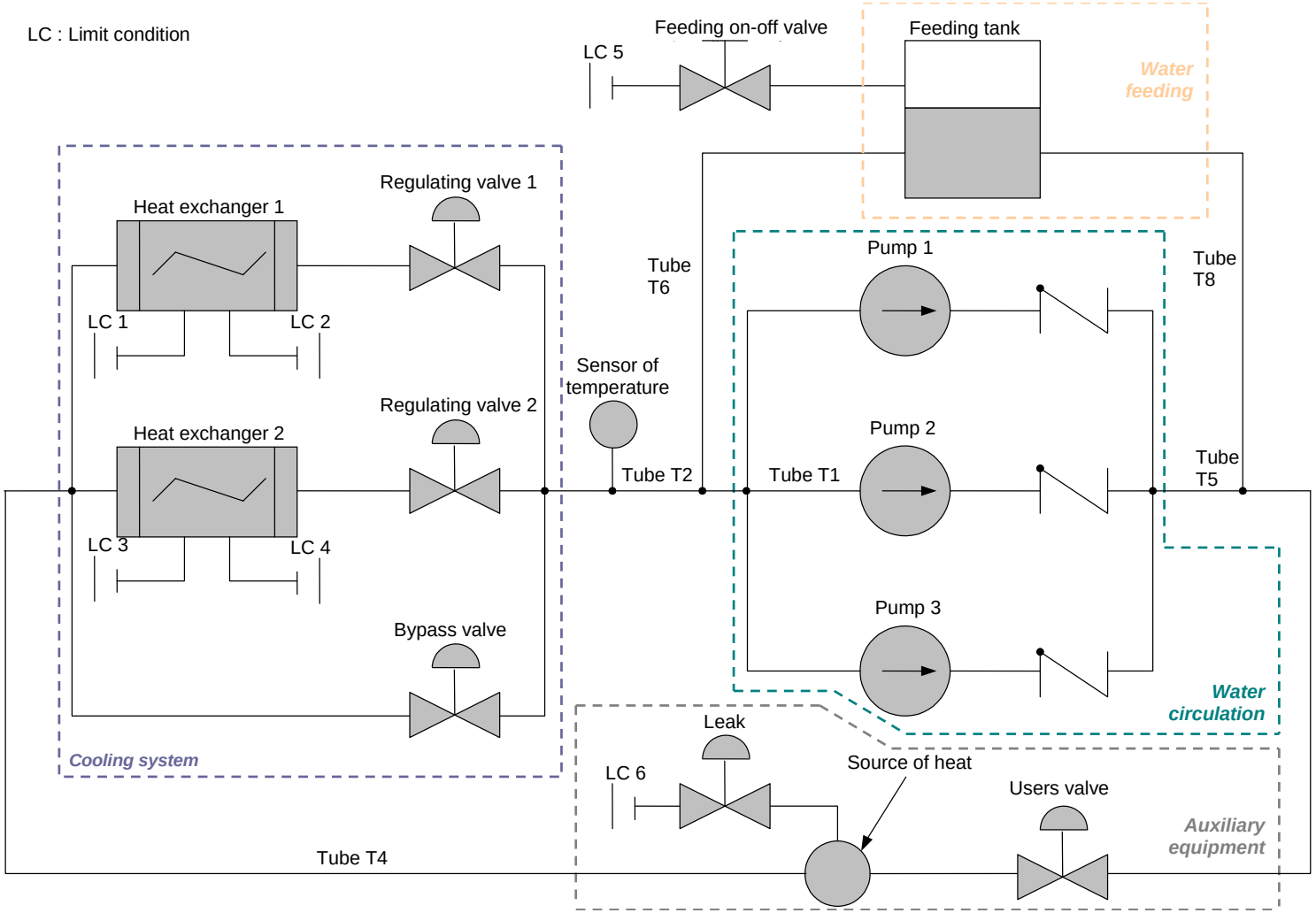
Industrial Use Case for Requirements Verification and Model Composition in ModelicaML

OPENPROD-Project Case Study

•Wladimir SCHAMAI (EADS Innovation Works, Germany)
•Peter Fritzson (Linköping University)

•Audrey JARDIN (EDF - R&D, France)
•Daniel BOUSKELA (EDF - R&D, France)

EDF Use Case – System Description of SRI system
(Intermediate Cooling System) in turbine hall of a nuclear power plant



MODELICA

System Requirements

- *#002:* The set point of the SRI water temperature must be held at a minimum value of 17°C.

- *#003:* In a normal operating mode, the water temperature of the SRI circuit should be between Ts - e and Ts + e (Ts : set point temperature).

- *#0083:* A pump must not start more than 3 times per hour.

- *#013:* In a normal operating mode, there must not be less than 2 operating pumps during more than 2s.

- *#007:* The water temperature must not vary more than 10°C/hour.

SRI Case Study Conclusion and Lessons Learnt

- Showed **applicability of vVDR method** to **realistic industrial** applications
- ModelicaML is a **promising prototype** implementation of the vVDR method, needs improved usability and stability
- Lessons learnt:
  - **Formalized requirements** should be **tested separately** in order to ensure correctness
  - **Model validity asserts** must be included
  - Parameterized **requirement monitors** can be re-used as **library** components (later realized in MODRIO project)
- Later work, now ongoing
  - Stochastic aspects (model uncertainties, tolerances in requirements, ...) should be taken into account

Part IV

Equation-Based Model Dynamic Debugging

Need for Debugging Tools
Map Low vs High Abstraction Level

- A **major part** of the total **cost** of software projects is due to testing and debugging
- US-Study 2002:
  Software errors cost the US economy **annually~ 60 Billion $**

- **Problem: Large Gap in Abstraction Level**
  from **Equations** to **Executable Code**
- Example error message (hard to understand)

  Error solving nonlinear system 132

  time = 0.002

  residual[0] = 0.288956

  x[0] = 1.105149

  residual[1] = 17.000400

  x[1] = 1.248448

  ...

Example Symbolic Transformations with Compiler Debug Trace

- **Complicated to understand source of some errors**
- **Efficient trace of transformations < 1 % overhead**

Example: 0 = y + der(x * time * z); z = 1.0;

(1) substitution:

y + der(x * (time * z))

=>

y + der(x * (time * 1.0))

(2) simplify:

y + der(x * (time * 1.0))

=>

y + der(x * time)

(3) expand derivative (symbolic diff):

y + der(x * time)

=>

y + (x + der(x) * time)

(4) solve:

0.0 = y + (x + der(x) * time)

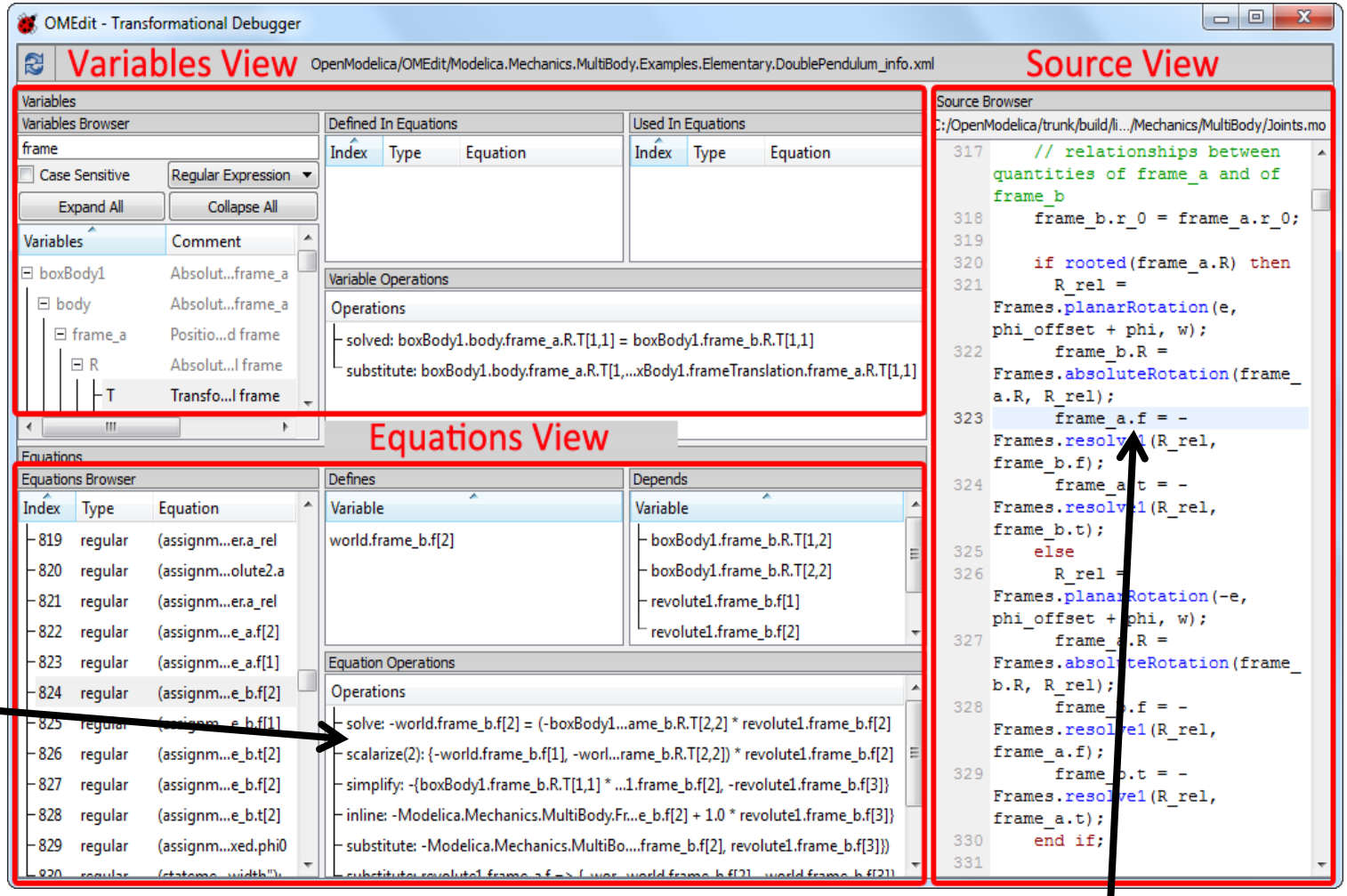
=>

der(x) = ((-y) - x) / time

Integrated Static-Dynamic OpenModelica Equation Model Debugger

Efficient handling of **Large** Equation Systems

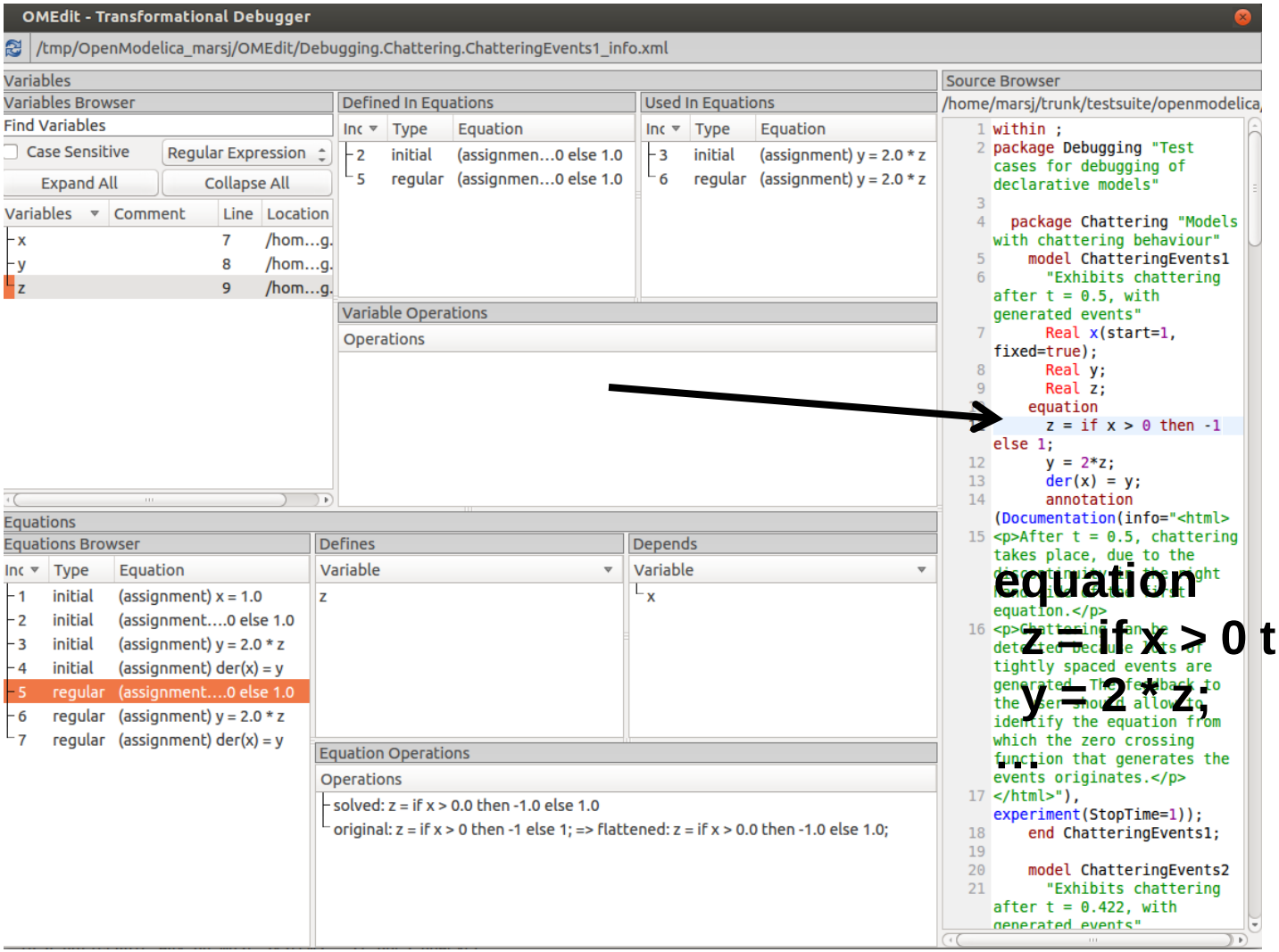
Showing equation transfor mations of a model:



Mapping dynamic run-time error to source model position

MODELICA

Example – Detecting Source of Chattering (excessive event switching) causing bad performance



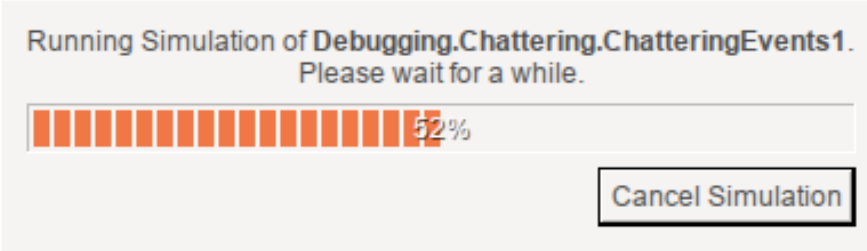
equation

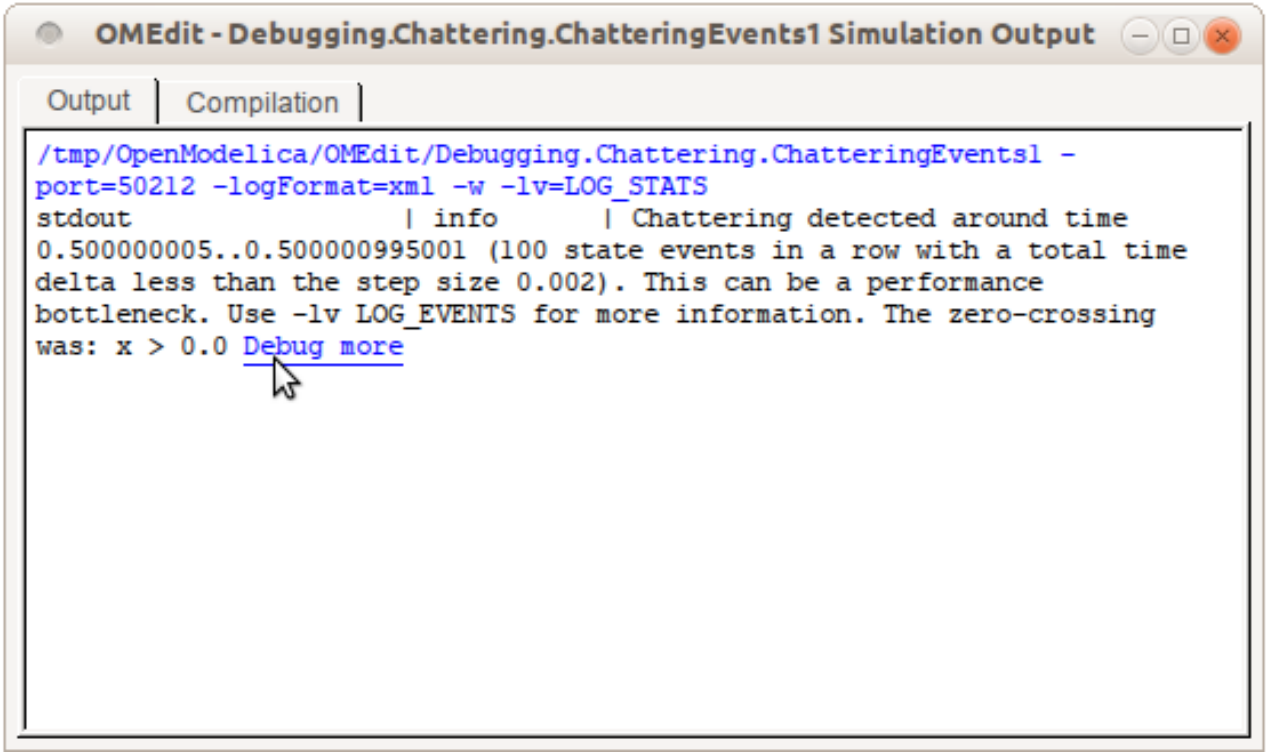
z = if x > 0 then -1 else 1;

y = 2*z;

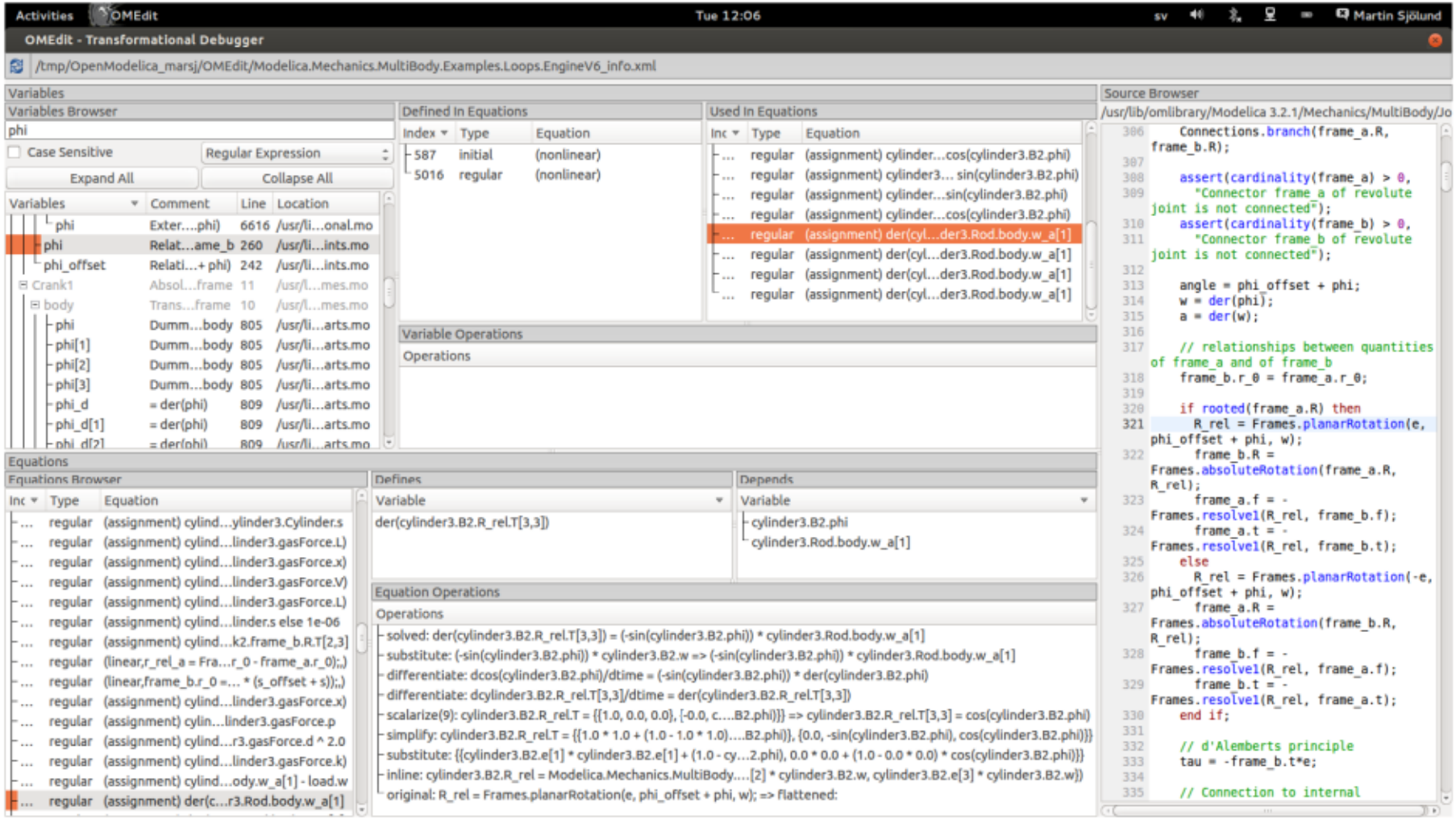
...

Error Indication – Simulation Slows Down





Transformations Browser – EngineV6 Overview (11 116 equations in model)



MODELICA

Performance Profiling
(Here: Profiling all equations in MSL 3.2.1 DoublePendulum)

- Measuring performance of equation blocks to find bottlenecks
  - Useful as input before model simplification for real-time platforms
- Integrated with the debugger so it is possible to show what the slow equations compute
- Suitable for real-time profiling (less information), or a complete view of all equation blocks and function calls

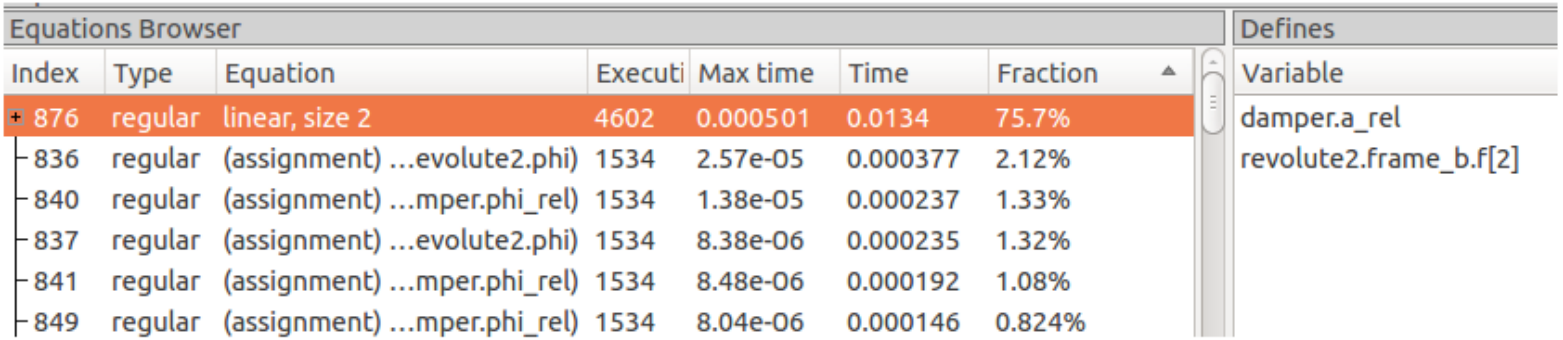
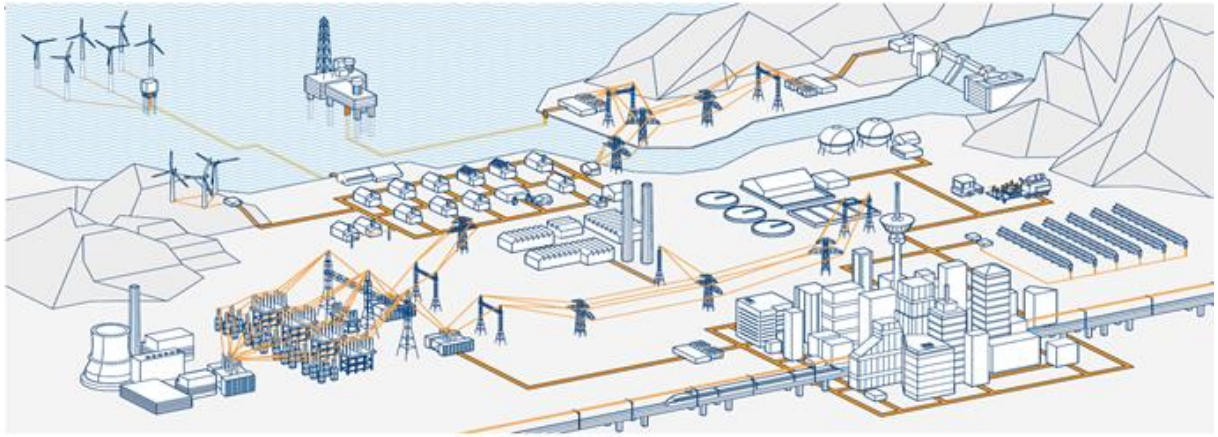


ABB Commercial Application Use of Debugger

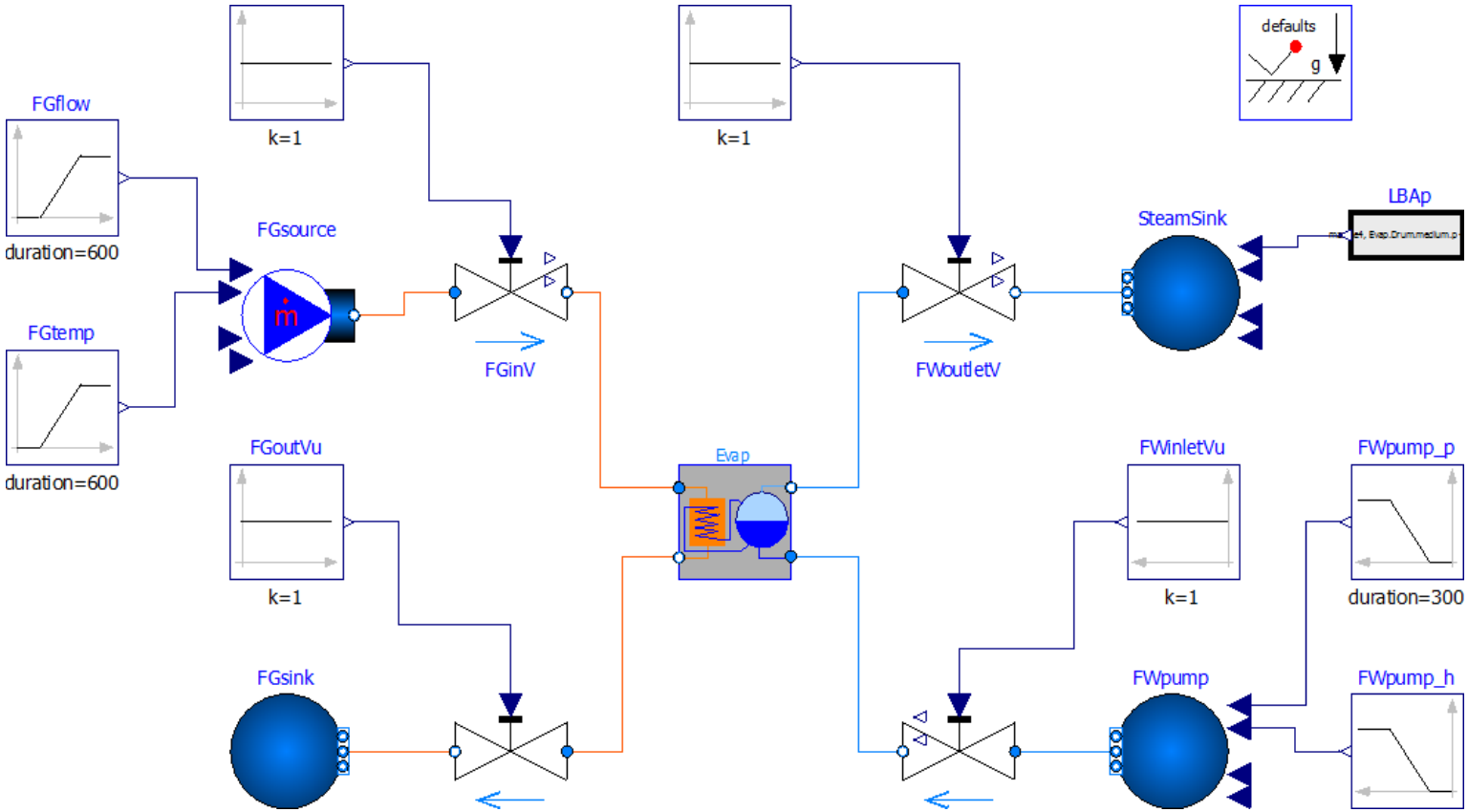
- ABB OPTIMAX® provides advanced model based control products for power generation and water utilities.



- ABB: “OpenModelica provides outstanding debugging features that help to save a lot of time during model development.”

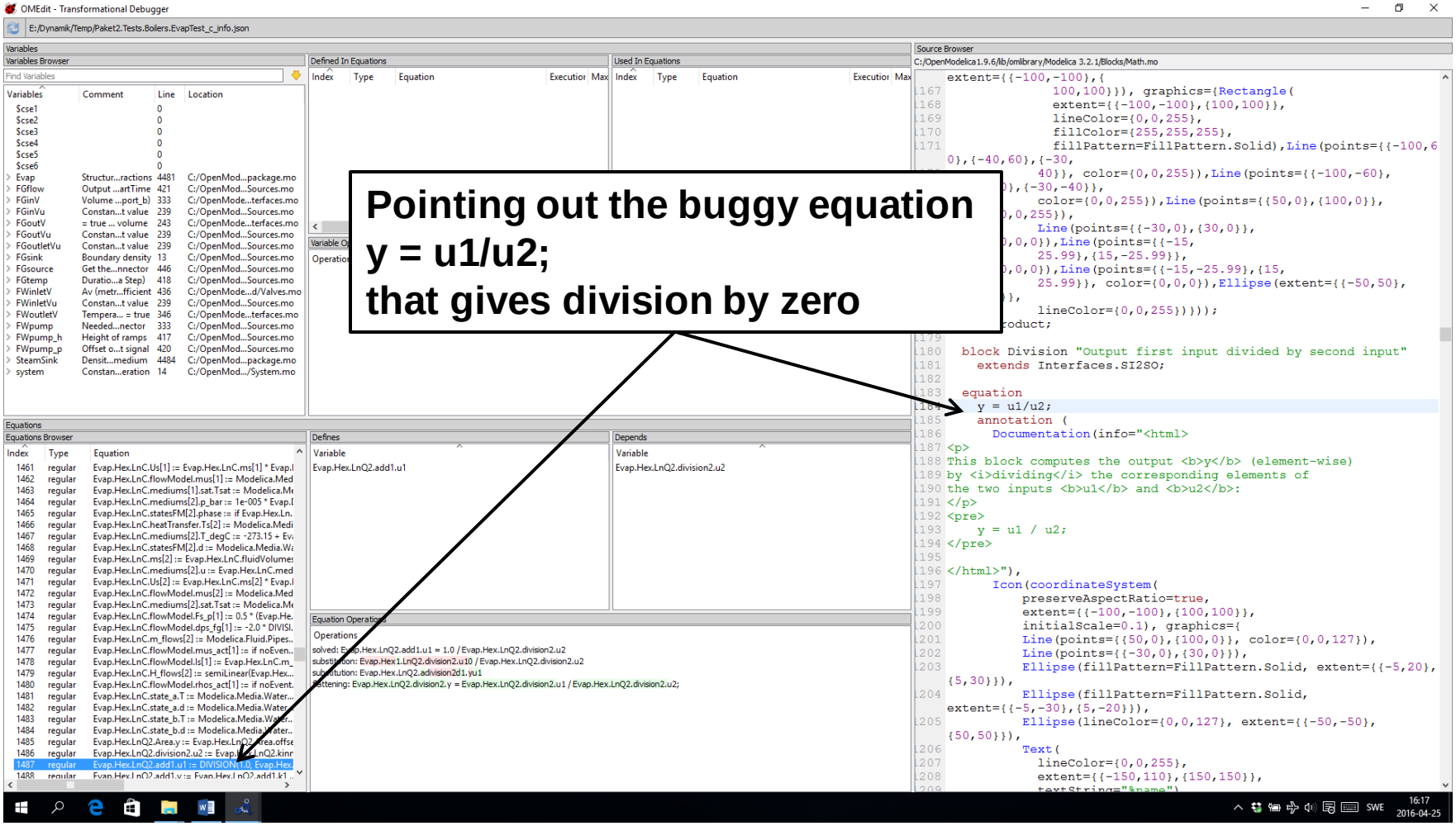
Equation Model Debugging on Siemens Model

(used on Siemens Evaporator test model, 1100 equations)



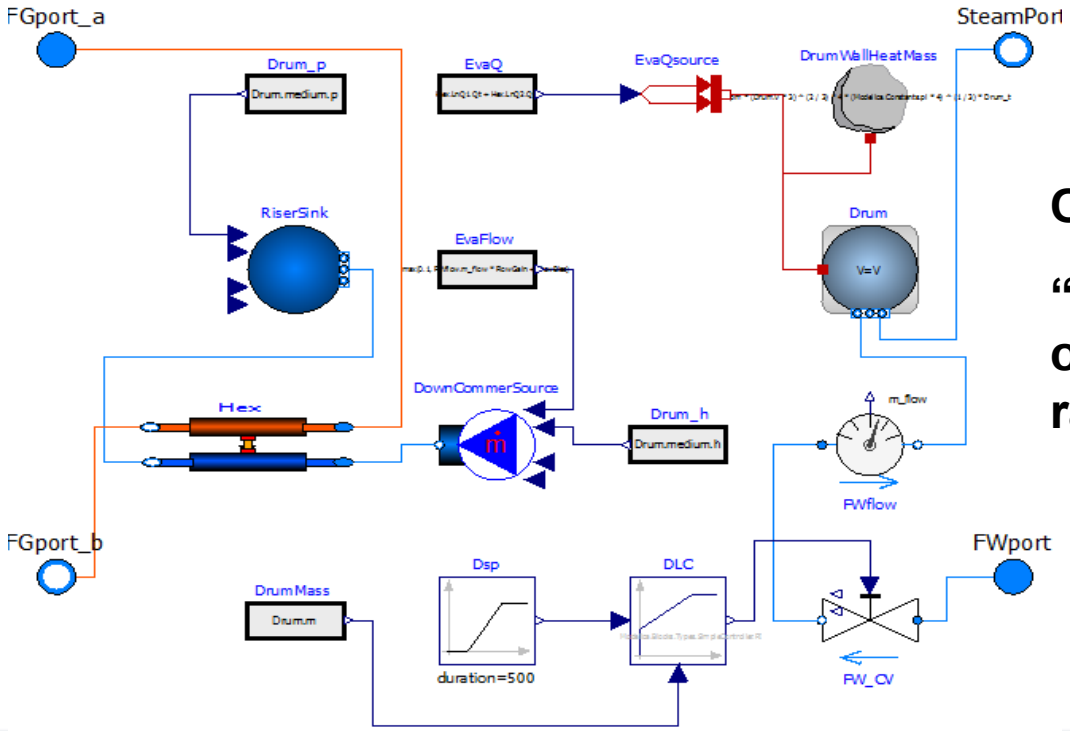
Equation Model Debugger on Siemens Model

(Siemens Evaporator test model, 1100 equations)



Performance Profiling for faster Simulation
(Here: Profiling equations of Siemens Drum boiler model with evaporator

- Measuring **performance** of equation blocks to find bottlenecks
  - Useful as input before model simplification for real-time applications
- Integrated with the debugger to **point out the slow equations**
- Suitable **for real-time profiling** (collect less information), or a complete view of all equation blocks and function calls



Conclusion from the evaluation:

"…the profiler makes the process of performance optimization radically shorter."

MODELICA

Part V

New Project OPENCPS
Open Cyber-Physical Development

Integrating

OpenModelica (Modelica) – Papyrus (UML)

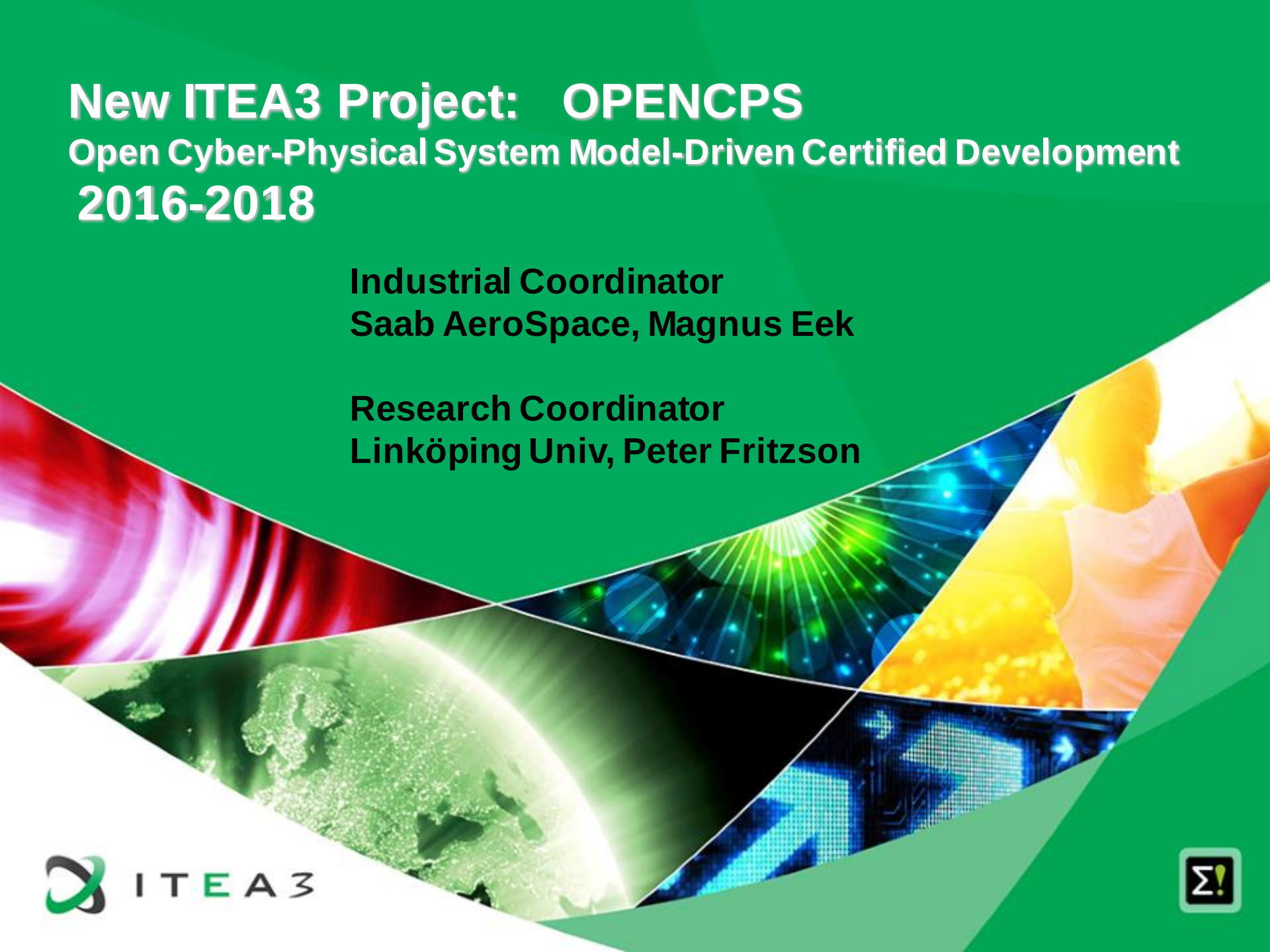
New ITEA3 Project: OPENCPS

Open Cyber-Physical System Model-Driven Certified Development

2016-2018

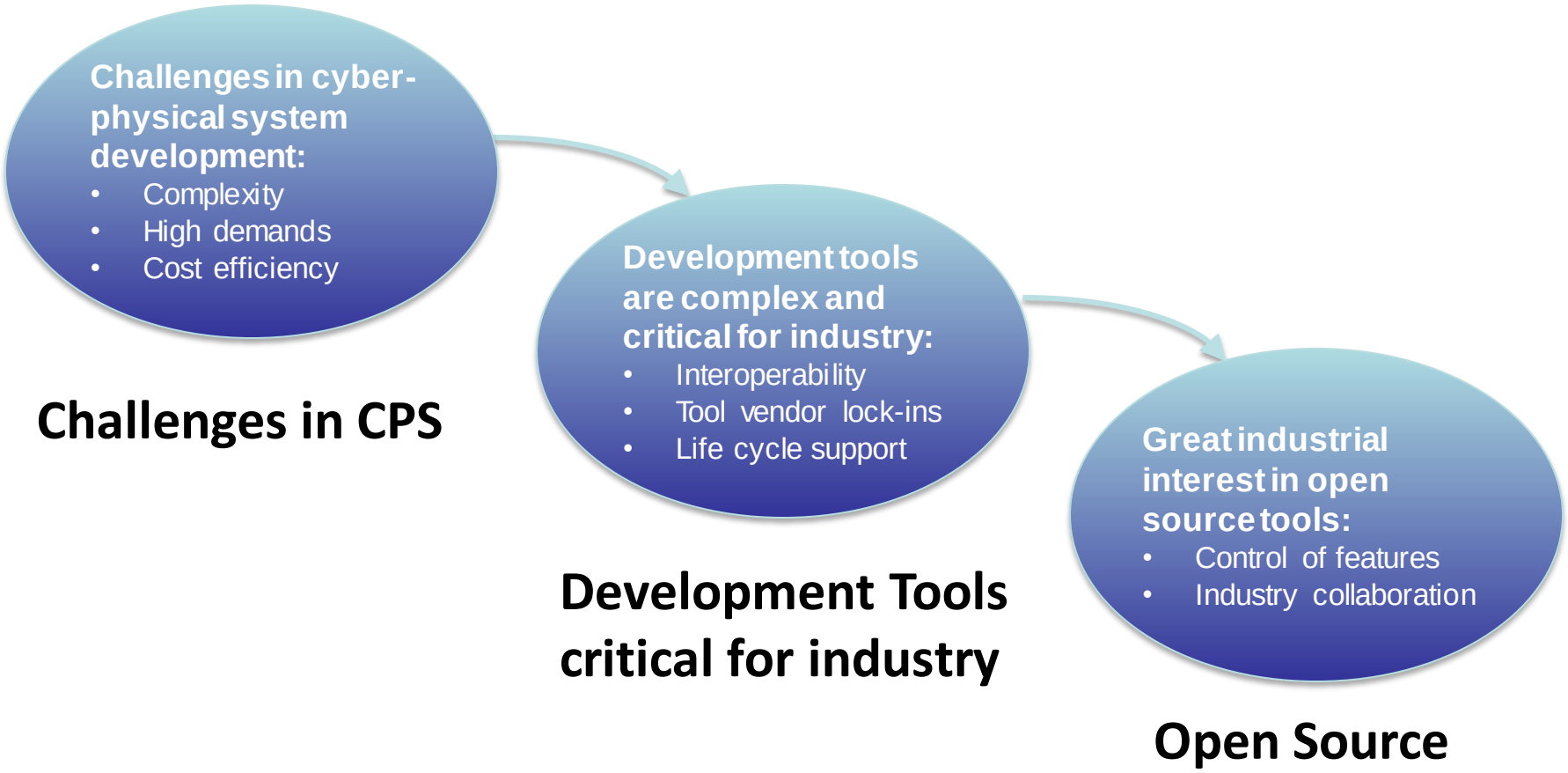
Industrial Coordinator
Saab AeroSpace, Magnus Eek

Research Coordinator
Linköping Univ, Peter Fritzson



ITEA3

OPENCPS – Integrating OpenModelica and Papyrus for Cyber-Physical System Development

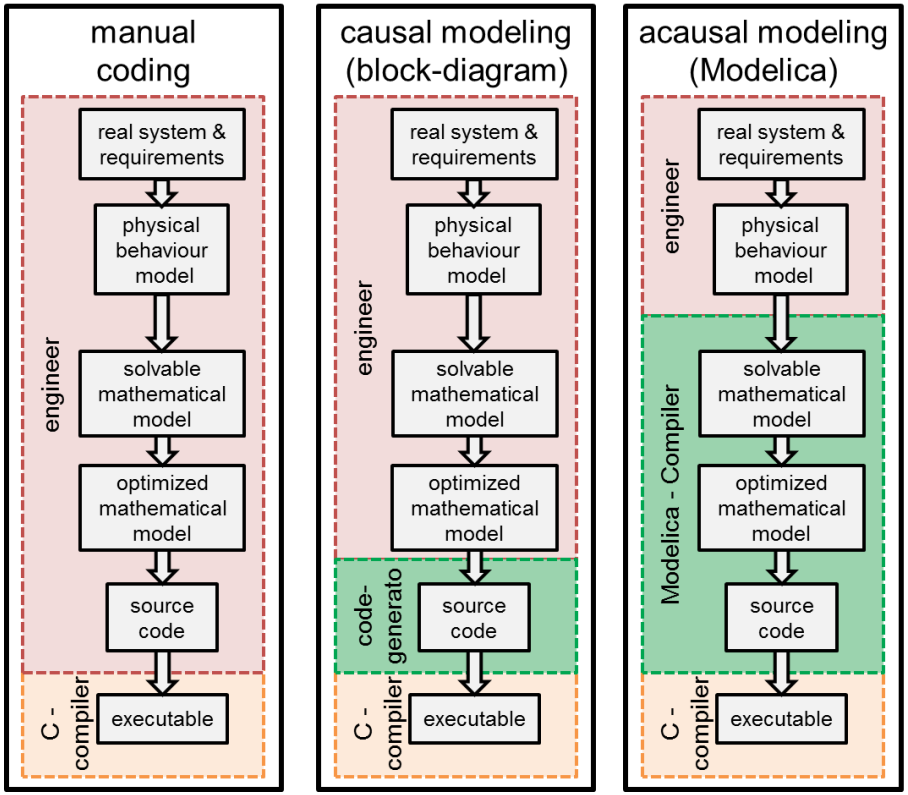


The two leading open source tools OpenModelica and Papyrus will be significantly developed and integrated for cyber-physical development

MODELICA

OPENCPS WP3: Translation and Validation of Code for Real-Time Embedded Applications

Goal: OpenModelica production code generation for advanced control applications supporting the translation validation of discrete and continuous-time models based on (acausal) equations.



Summary and Questions

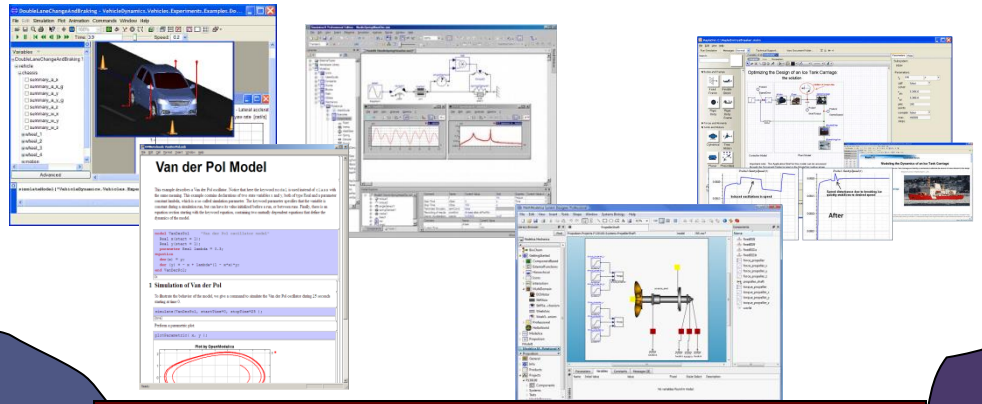
Multi-Domain Modeling

Visual Acausal Component Modeling

MODELICA

www.modelica.org – Language, Standard Library

www.openmodelica.org – Open Source Tool



Typed Declarative Textual Language

Thanks for listening!

Hybrid Modeling