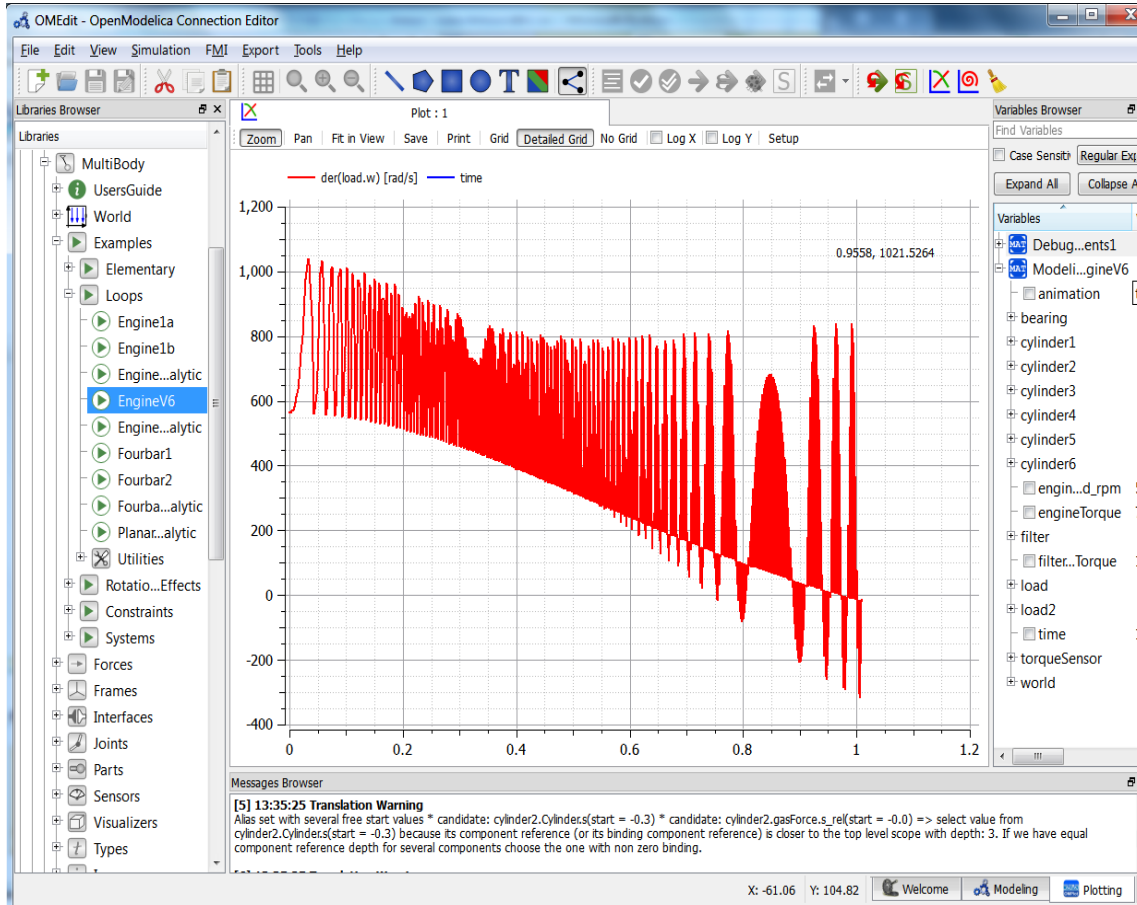


Modeling, Simulation, and Development of Cyber-Physical Systems with OpenModelica and FMI



Presentation at
RISE SICS East, Sweden

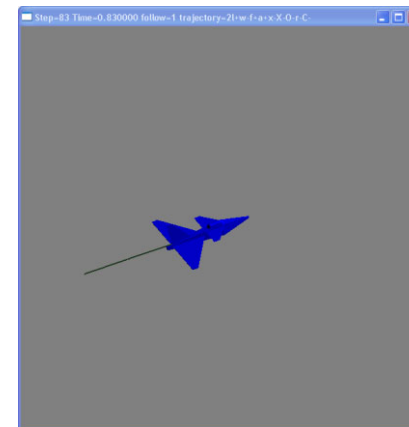
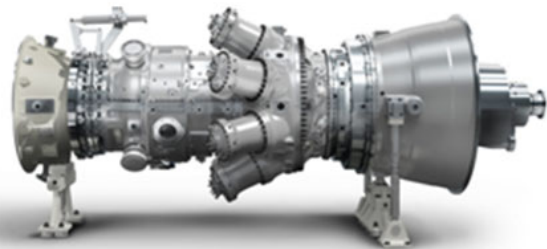
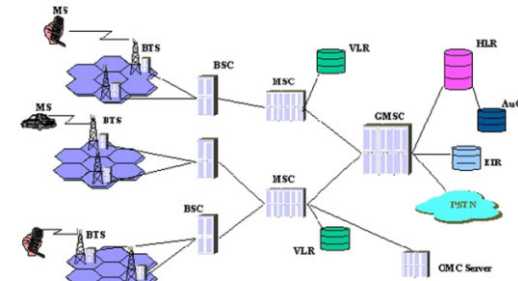
October 6, 2018

Peter Fritzson
peter.fritzson@liu.se

Full Professor at Linköping University
Director Open Source Modelica Consortium
Vice Chairman of Modelica Association
Director of the MODPROD Center for
model-based development

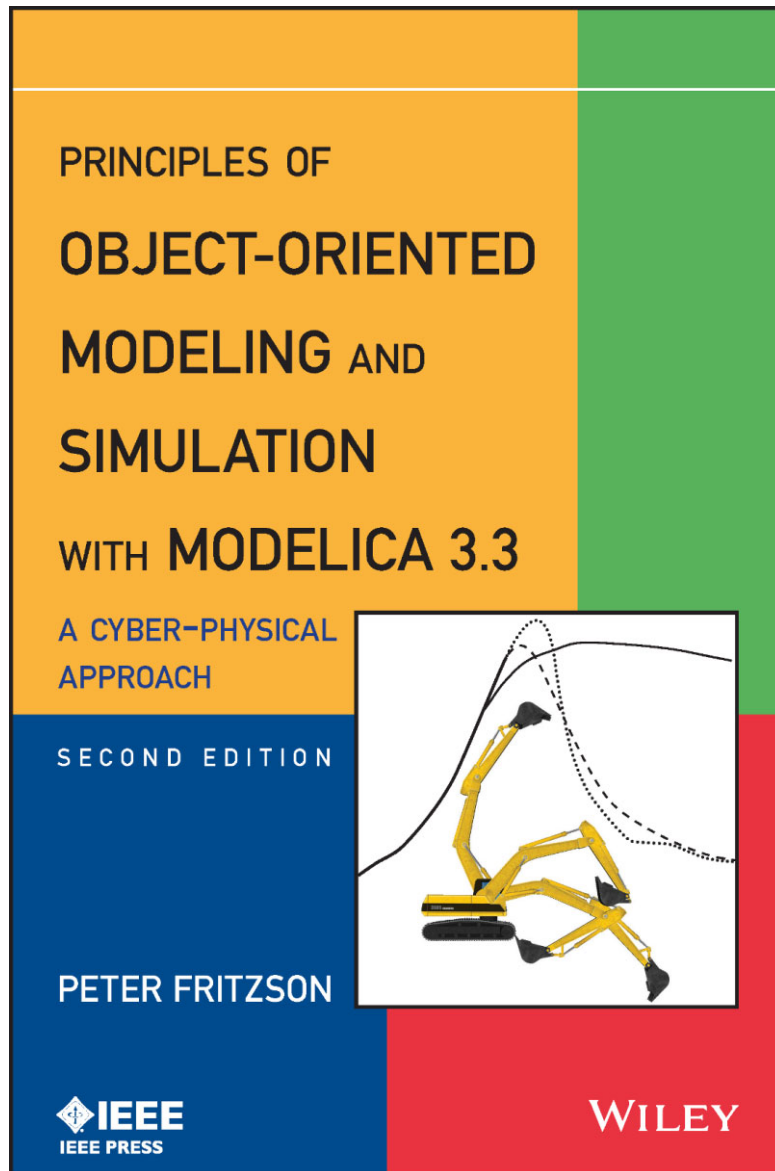
Industrial Challenges for Complex Cyber-Physical System Products of both Software and Hardware

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



Big Book on Modelica and Technology, Dec 2014

Download Free OpenModelica Software



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

Introductory Modelica Book

September 2011
232 pages

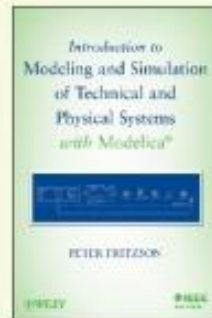
Translations
available in
Chinese,
Japanese,
Spanish

Wiley
IEEE Press

**For Introductory
Short Courses on
Object Oriented
Mathematical Modeling**

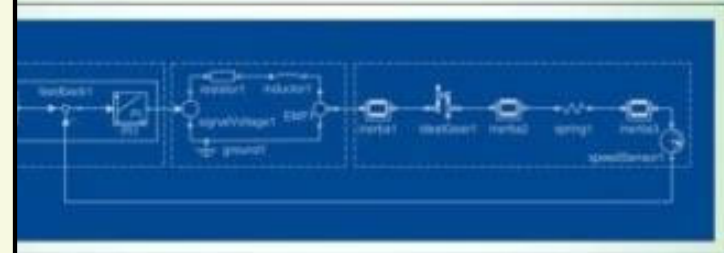
Modelica语言导论
——技术物理系统建模与仿真
(中文版)

Peter Fritzon 著
陈立平 译



科学出版社

Introduction to
Modeling and Simulation
of Technical and
Physical Systems
with Modelica



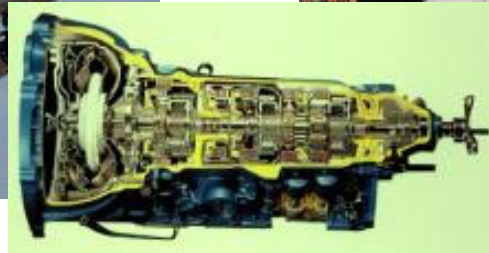
PETER FRITZSON

 **WILEY**

 **IEEE**
IEEE PRESS

Part I

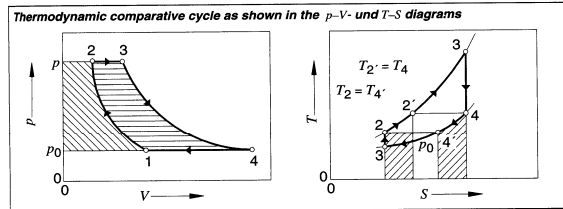
Introduction to Modelica



Modelica Background: Stored Knowledge

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

Internal-combustion engines 417



from T_2 to T_2' , supplied by the heat exchanger is coupled with a thermal discharge ($4 \rightarrow 4'$). 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 $p_2/p_1 = (T_2/T_1)^{\frac{\gamma}{\gamma-1}} = (T_3/T_4)^{\frac{\gamma}{\gamma-1}}$ and $T_4 = T_3 \cdot (T_1/T_2)$ thus

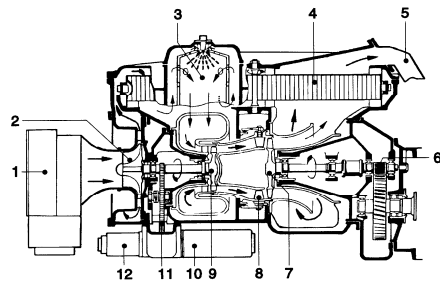
$$\eta_{th} = 1 - (T_2/T_3)$$

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 impressae, & fieri secundum lineam rectam qua vis illa imprimitur.

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

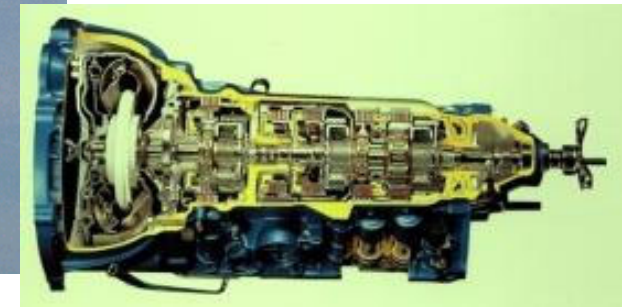
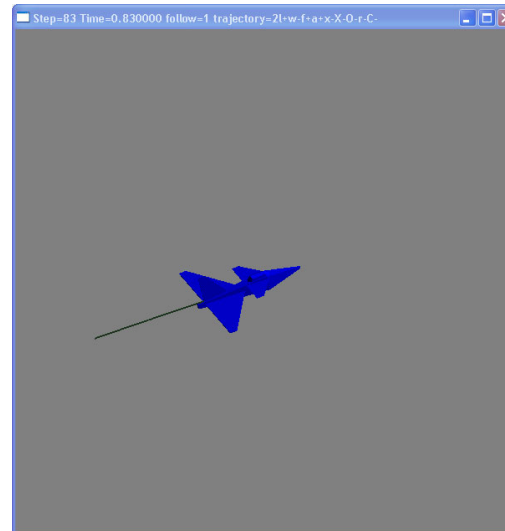
$v = \text{INTEG}(F) / m$

Programming languages usually do not allow equations!

What is Modelica?

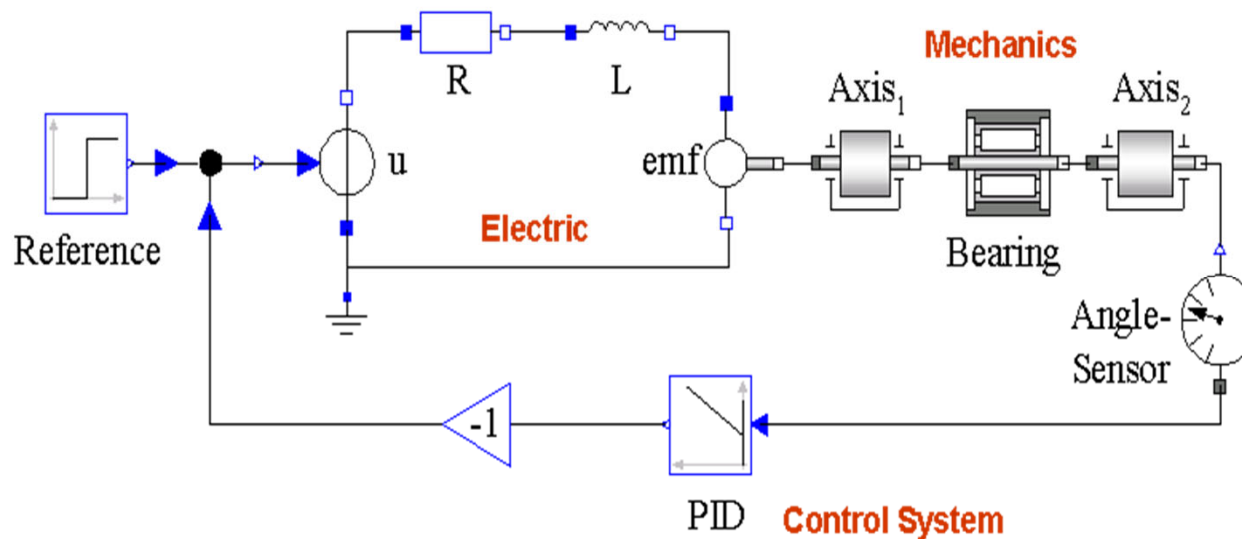
A language for modeling of **complex cyber-physical systems**

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



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 – ESI Group
- MapleSim from MapleSoft
- AMESIM from LMS
- JModelica.org from Modelon
- MWORKS from Tongyang Sw & Control
- IDA Simulation Env, from Equa

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*:

$$\mathbf{R * i = v;}$$

can be used in three ways:

$$\mathbf{i := v/R;}$$

$$\mathbf{v := R*i;}$$

$$\mathbf{R := v/i;}$$

What is Special about Modelica?

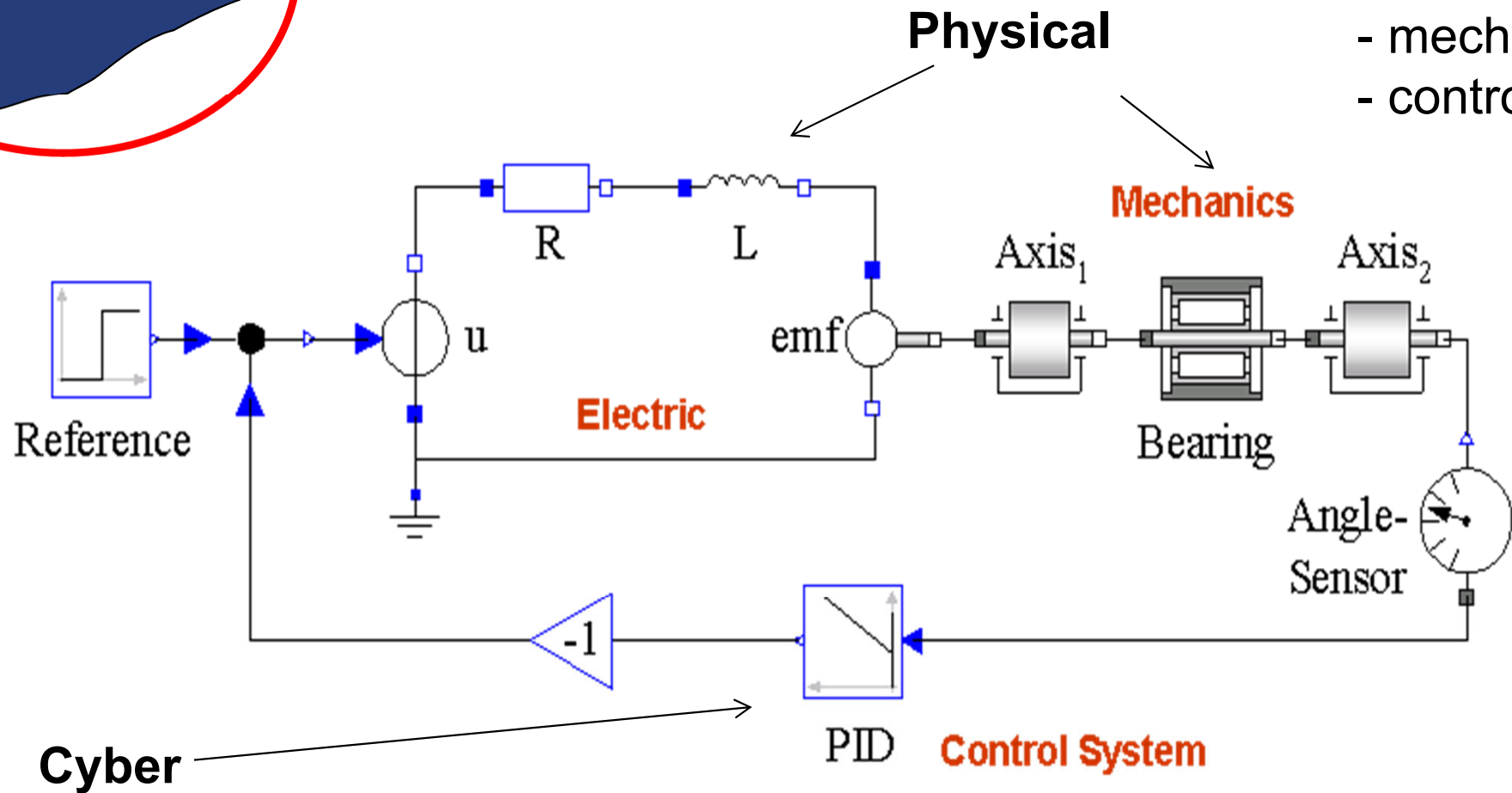
- Multi-Domain Modeling
- Visual acausal hierarchical component modeling
- Typed declarative equation-based textual language
- Hybrid modeling and simulation

What is Special about Modelica?

Multi-Domain Modeling

Cyber-Physical Modeling

- 3 domains
- electric
- mechanics
- control



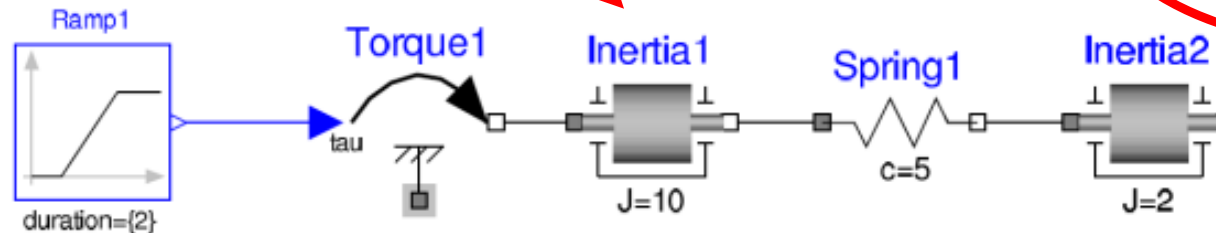
What is Special about Modelica?

Multi-Domain
Modeling

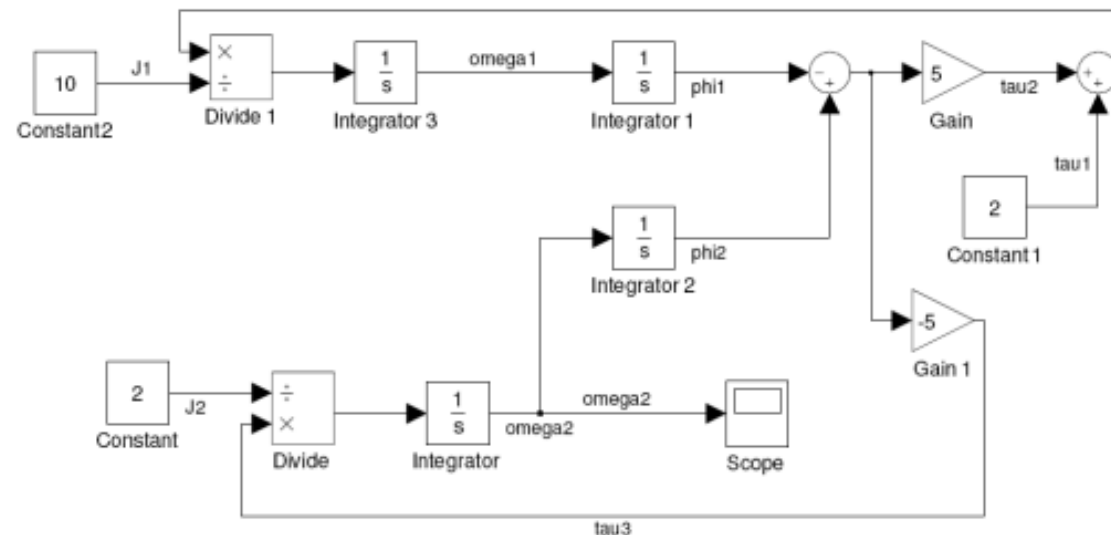
Visual Acausal
Hierarchical
Component
Modeling

Keeps the physical
structure

Acausal model
(Modelica)



Causal
block-based
model
(Simulink)

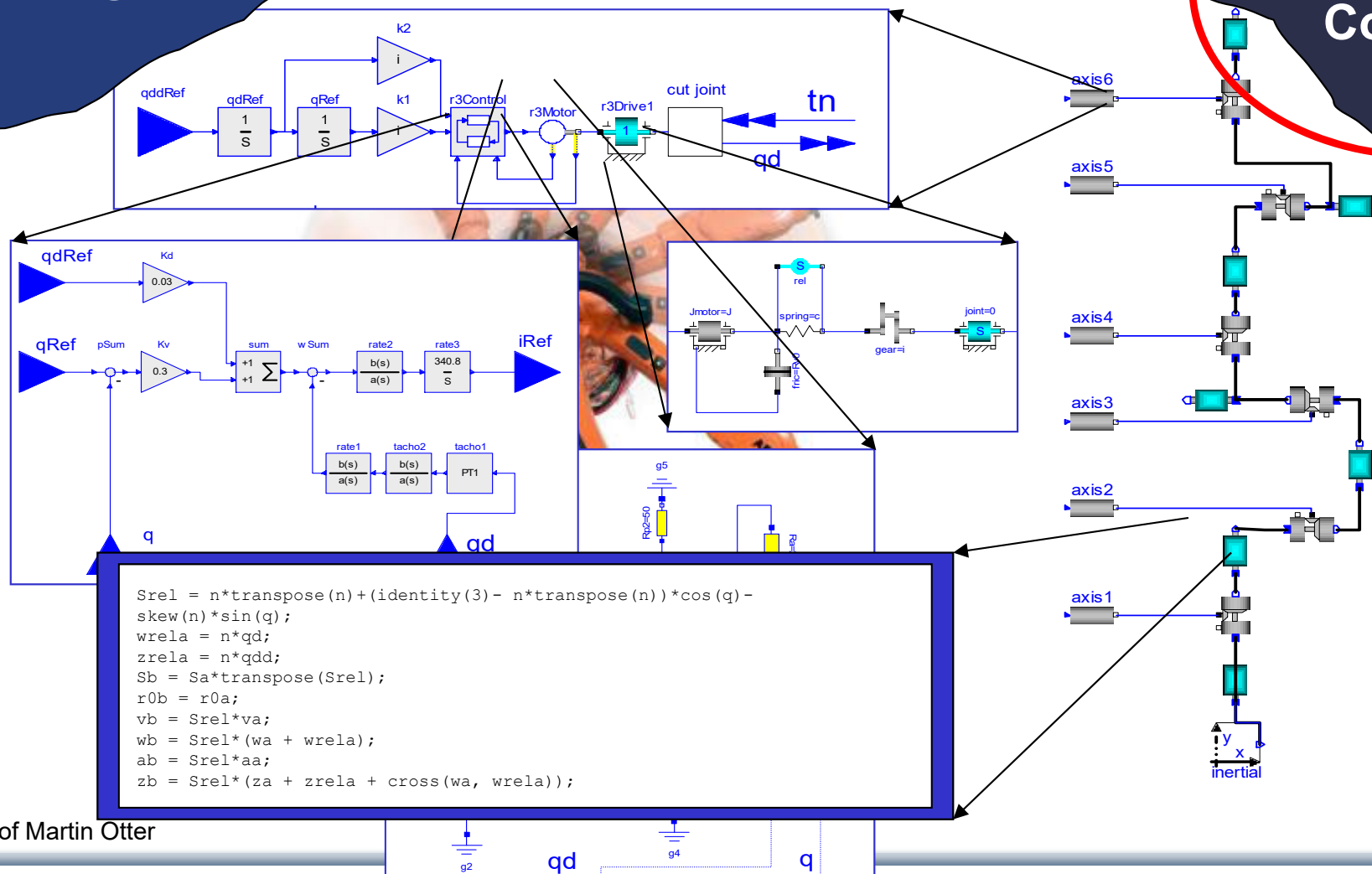


What is Special about Modelica?

Multi-Domain
Modeling

Hierarchical system
modeling

Visual Acausal
Hierarchical
Component
Modeling



Courtesy of Martin Otter

What is Special about Modelica?

Multi-Domain
Modeling

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

Visual Acausal
Hierarchical
Component
Modeling

Behaviour described declaratively using

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

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;
```

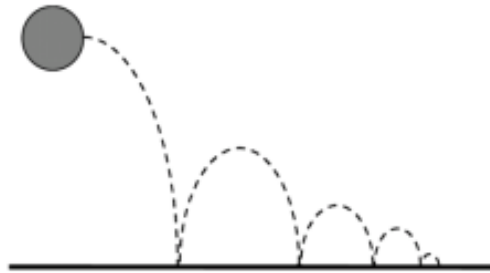
Differential equations

Typed
Declarative
Equation-based
Textual Language

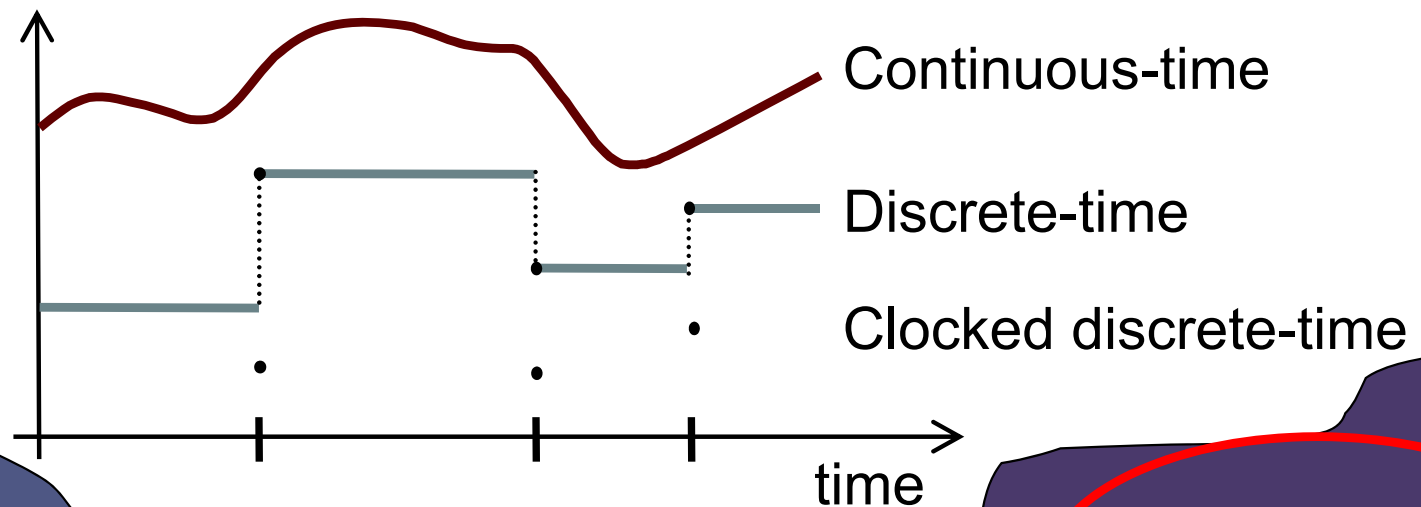
What is Special about Modelica?

Multi-Domain
Modeling

Visual Acausal
Component
Modeling



Hybrid modeling =
continuous-time + discrete-time modeling



Typed
Declarative
Equation-based
Textual Language

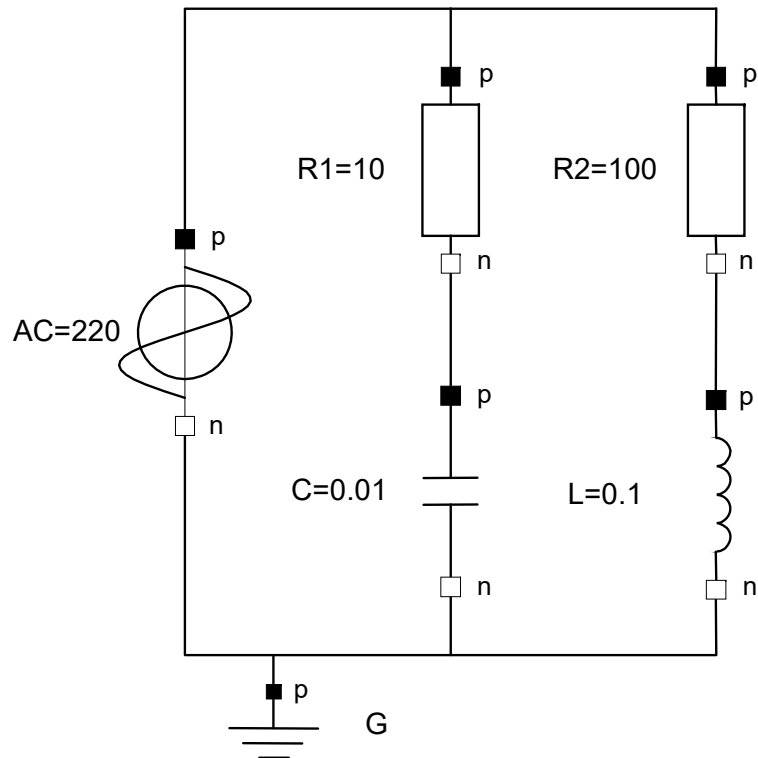
Hybrid
Modeling

Modelica vs Simulink Block Oriented Modeling

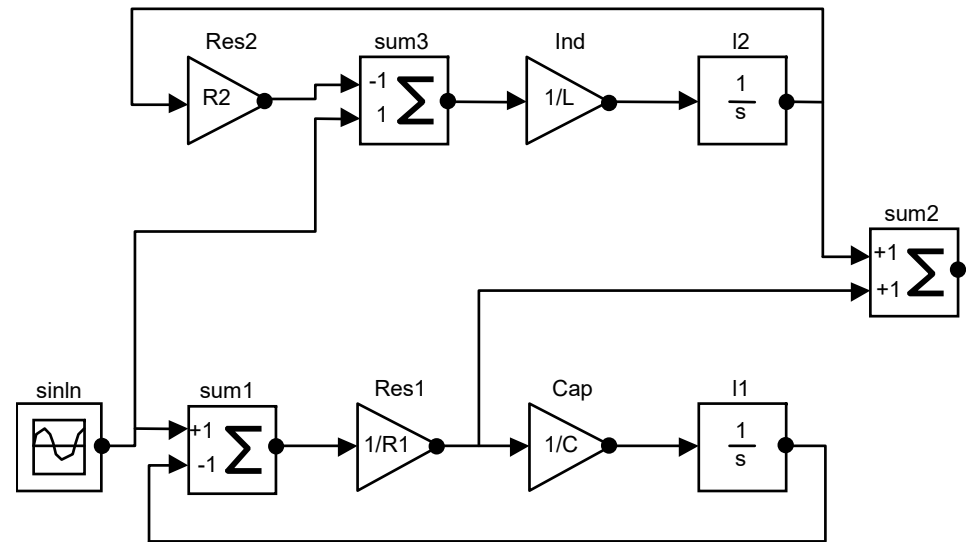
Simple Electrical Model

Modelica:
Physical model –
easy to understand

Keeps the
physical
structure

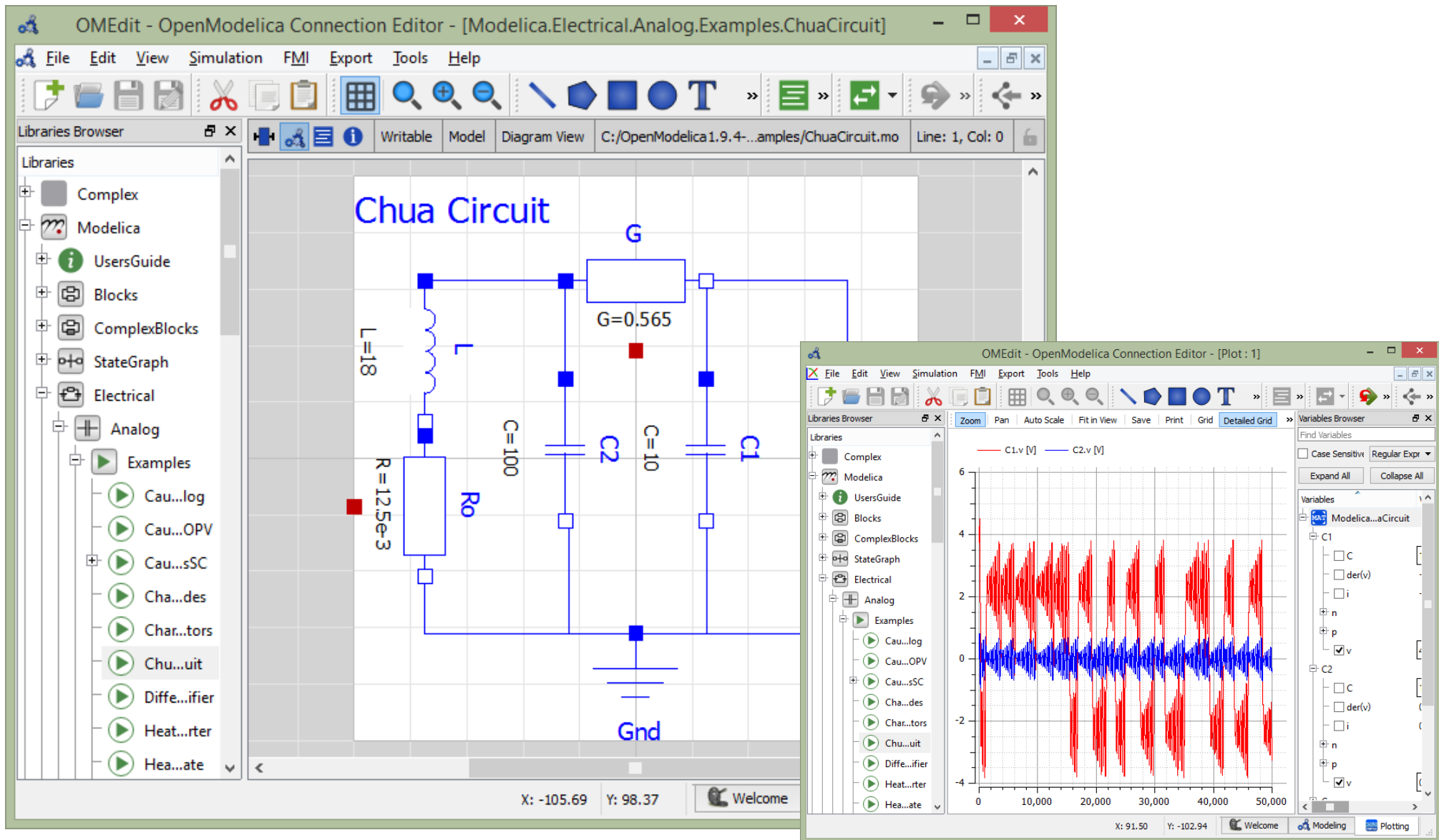


Simulink:
Signal-flow model – hard to
understand

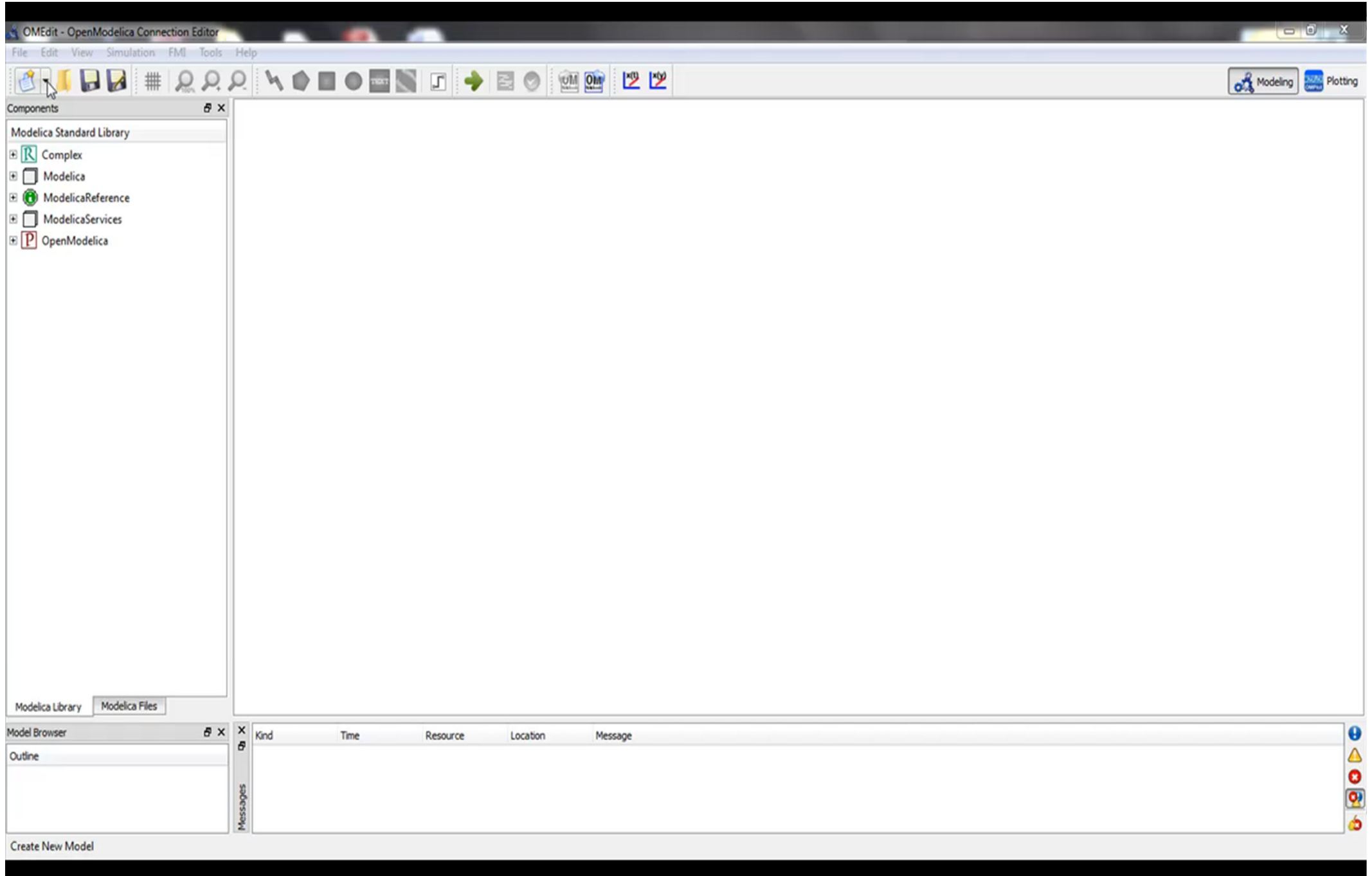


OpenModelica Tool Graphical Editor and Plotting

Graphical Modeling Using Drag and Drop



Graphical Modeling with OpenModelica Environment



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);
```

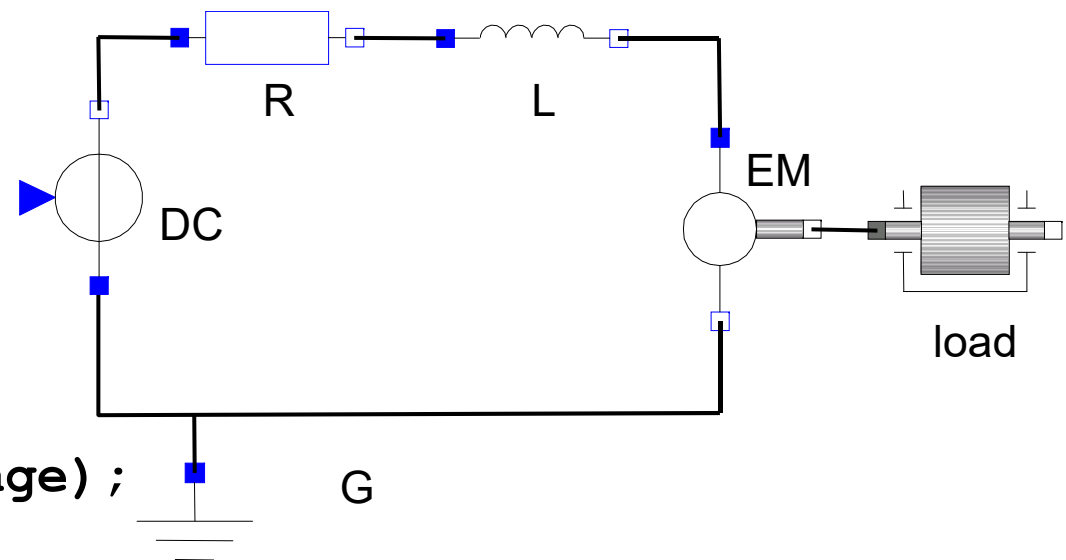
```
connect (L.p, EM.n);
```

```
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:

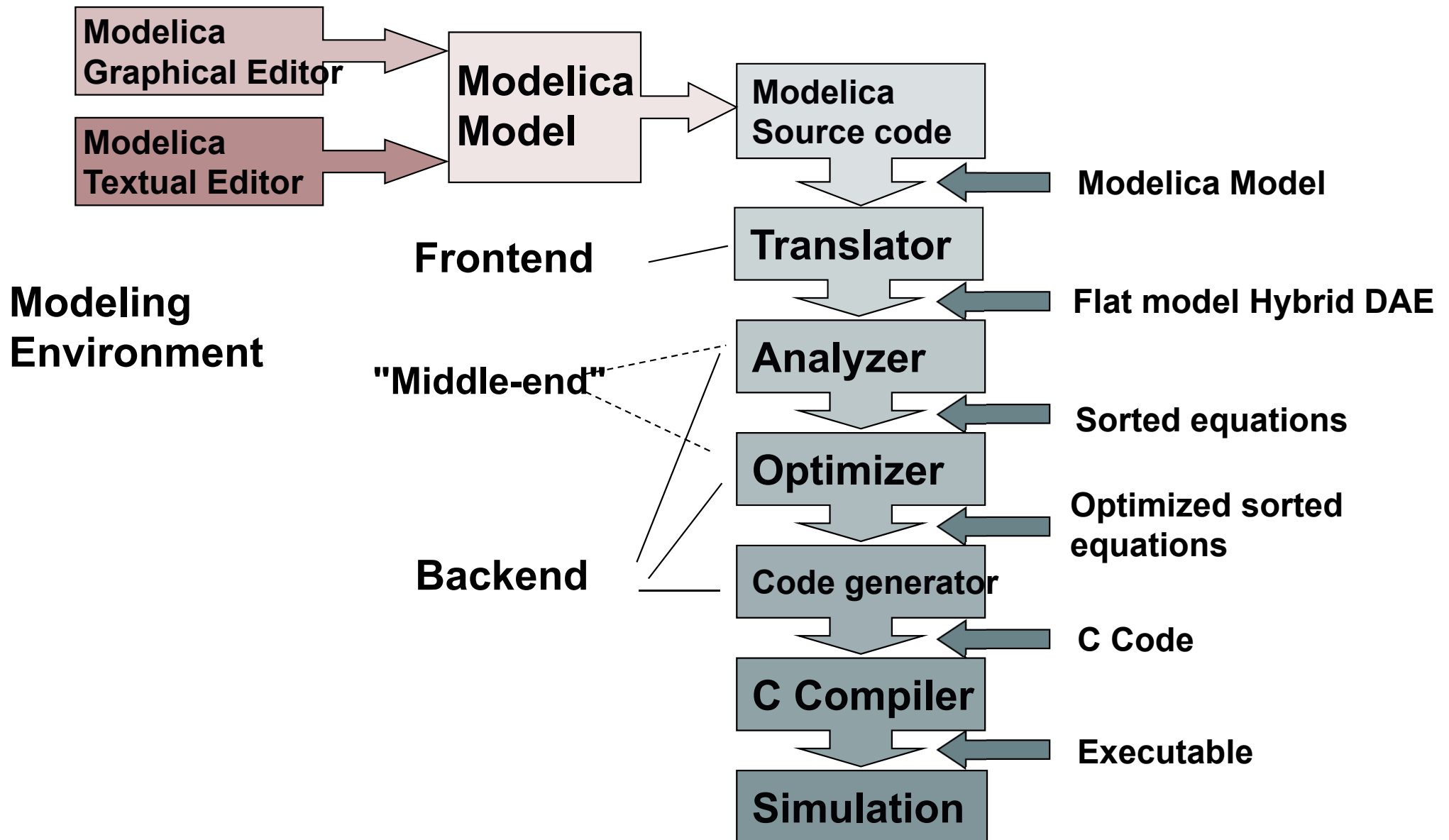
0 == DC.p.i + R.n.i	EM.u == EM.p.v - EM.n.v	R.u == R.p.v - R.n.v
DC.p.v == R.n.v	0 == EM.p.i + EM.n.i	0 == R.p.i + R.n.i
	EM.i == EM.p.i	R.i == R.p.i
0 == R.p.i + L.n.i	EM.u == EM.k * EM.omega	R.u == R.R * R.i
R.p.v == L.n.v	EM.i == EM.M / EM.k	
	EM.J * EM.omega == EM.M - EM.b * EM.omega	L.u == L.p.v - L.n.v
0 == L.p.i + EM.n.i		0 == L.p.i + L.n.i
L.p.v == EM.n.v	DC.u == DC.p.v - DC.n.v	L.i == L.p.i
	0 == DC.p.i + DC.n.i	L.u == L.L * L.i'
0 == EM.p.i + DC.n.i	DC.i == DC.p.i	
EM.p.v == DC.n.v	DC.u == DC.Amp * Sin[2 * pi * DC.f * t]	
0 == DC.n.i + G.p.i		
DC.n.v == G.p.v		

(load component not included)

Automatic transformation to ODE or DAE for simulation:

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

Model Translation Process to Hybrid DAE to Code

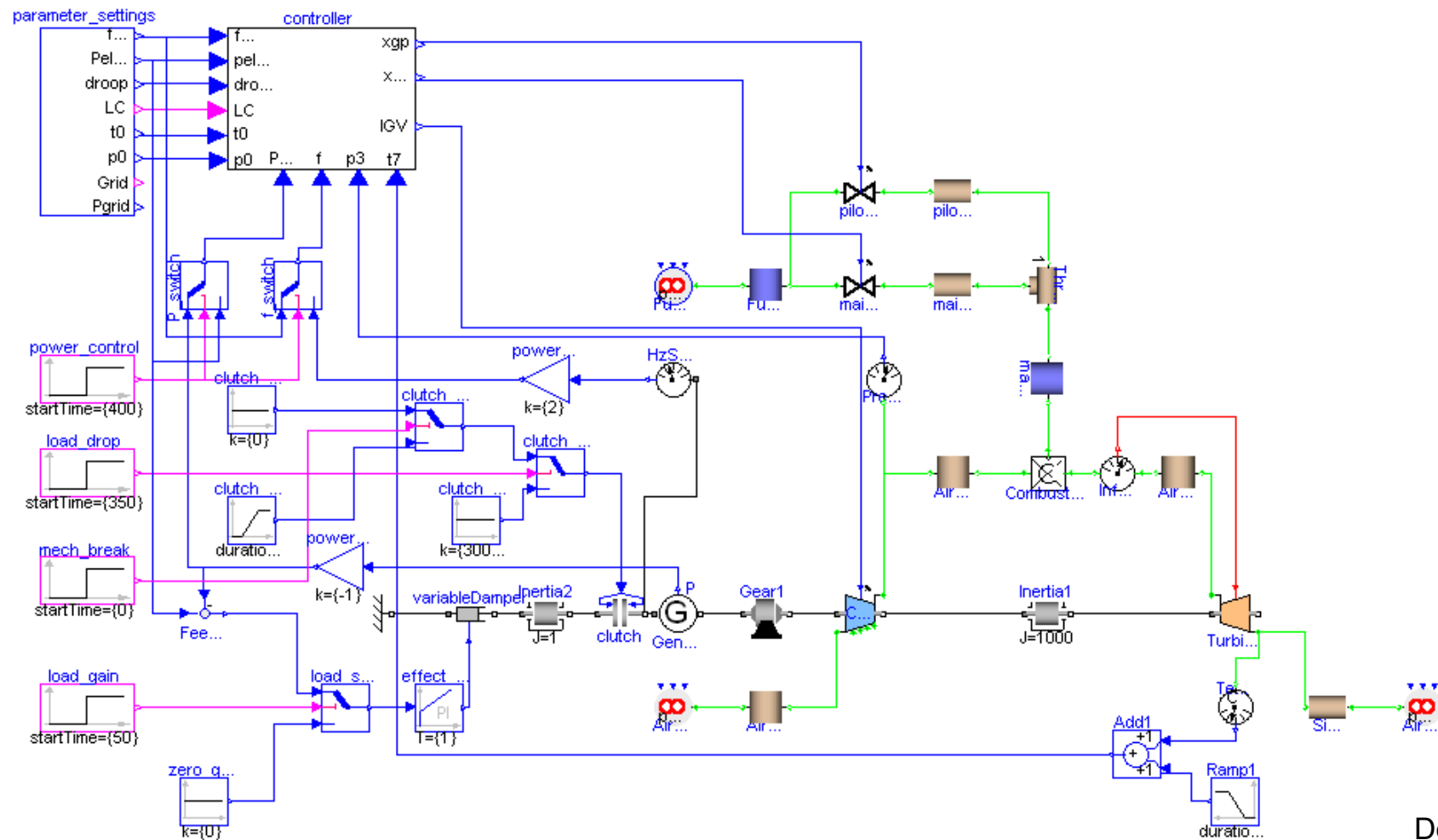


Brief Modelica History

- 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
 - 3.4 released May 2017
- Modelica Association established 2000 in Linköping
 - Open, non-profit organization

Modelica in Power Generation

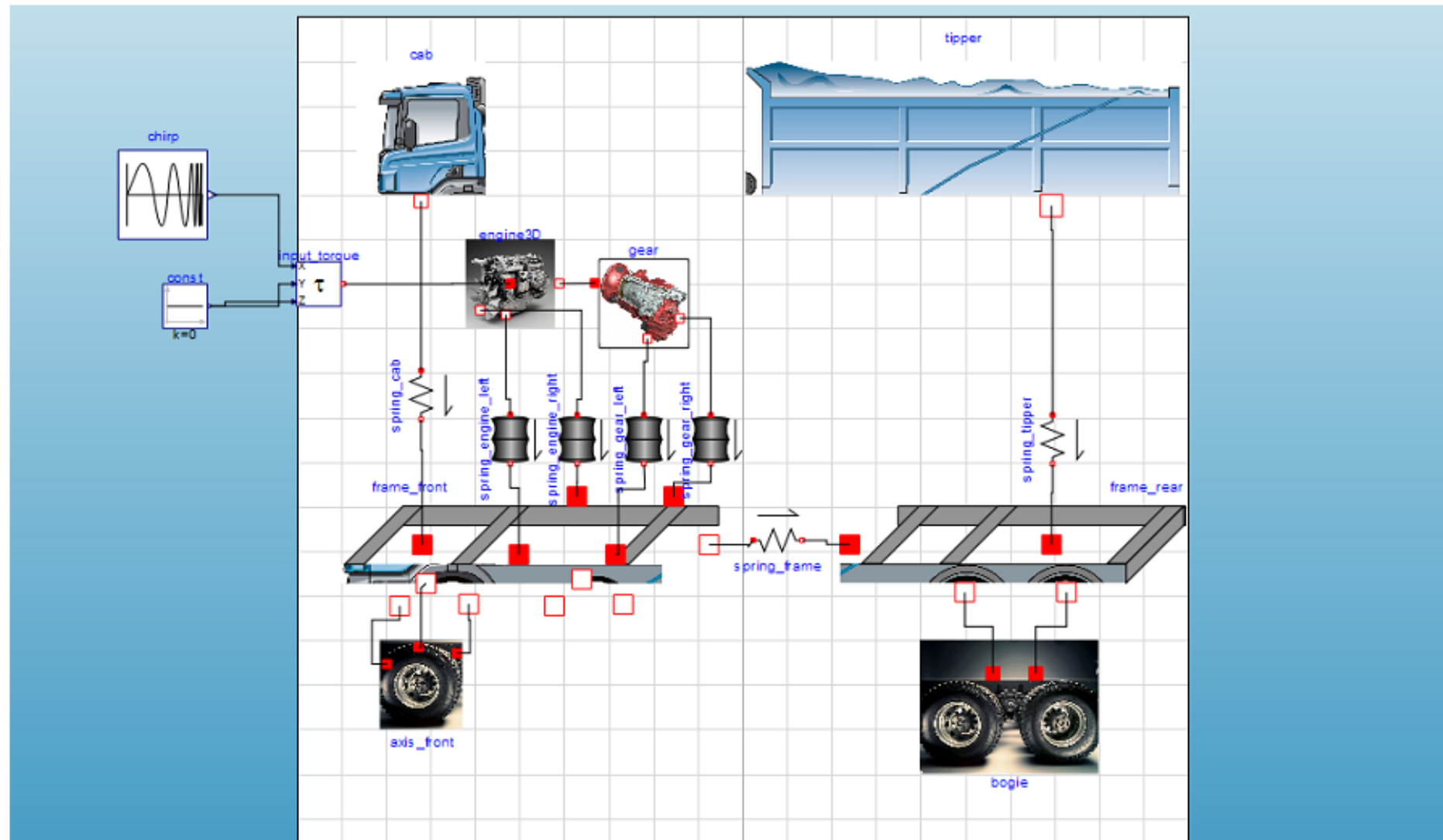
GTX Gas Turbine Power Cutoff Mechanism



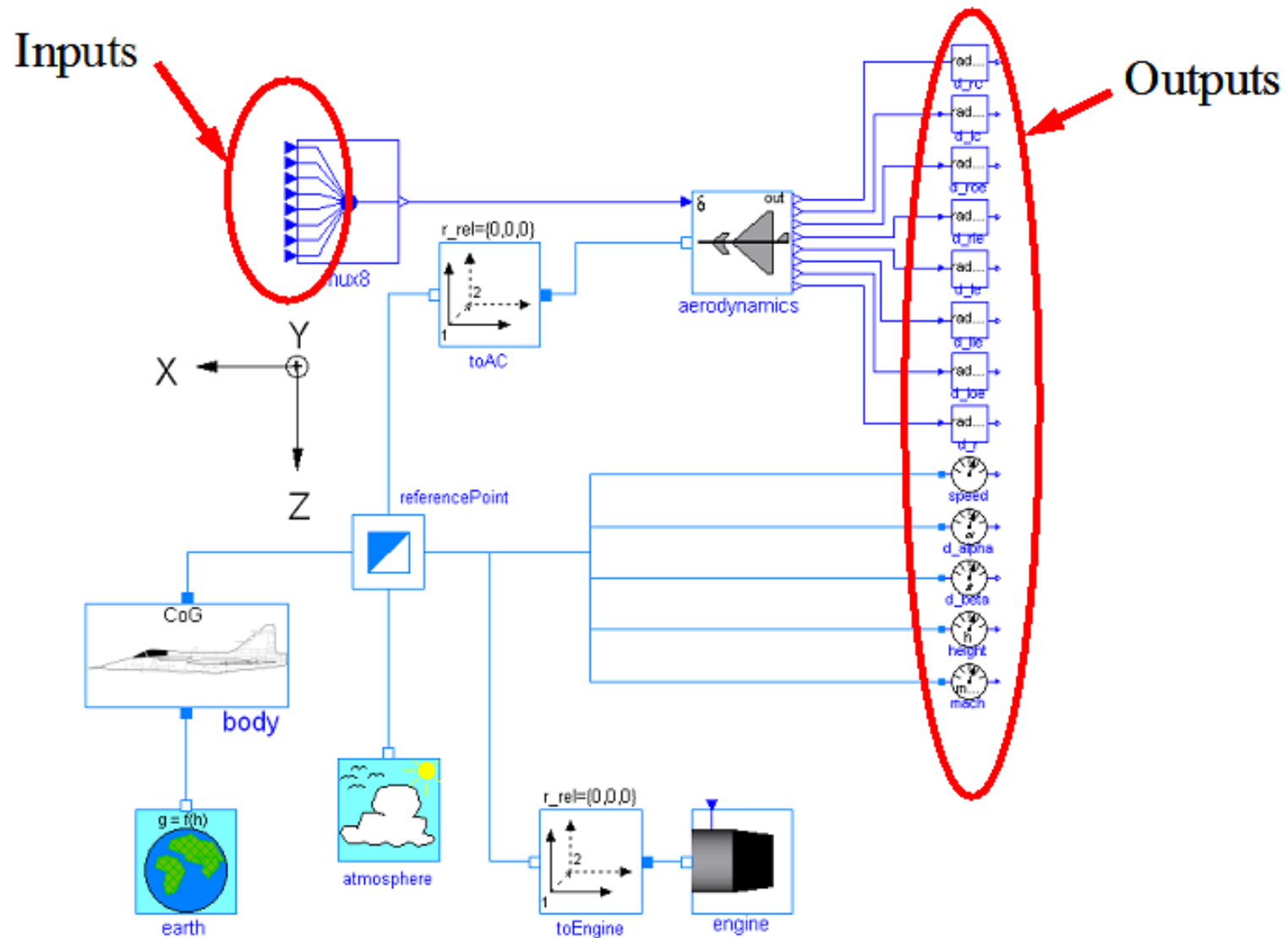
Developed
by MathCore
for Siemens

Courtesy of Siemens Industrial Turbomachinery AB

Modelica in Automotive Industry



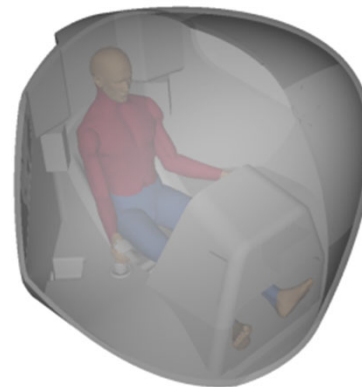
Modelica in Avionics



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,
Oberpfaffenhofen, Germany

Large Robotic Flight Simulator (Demo)

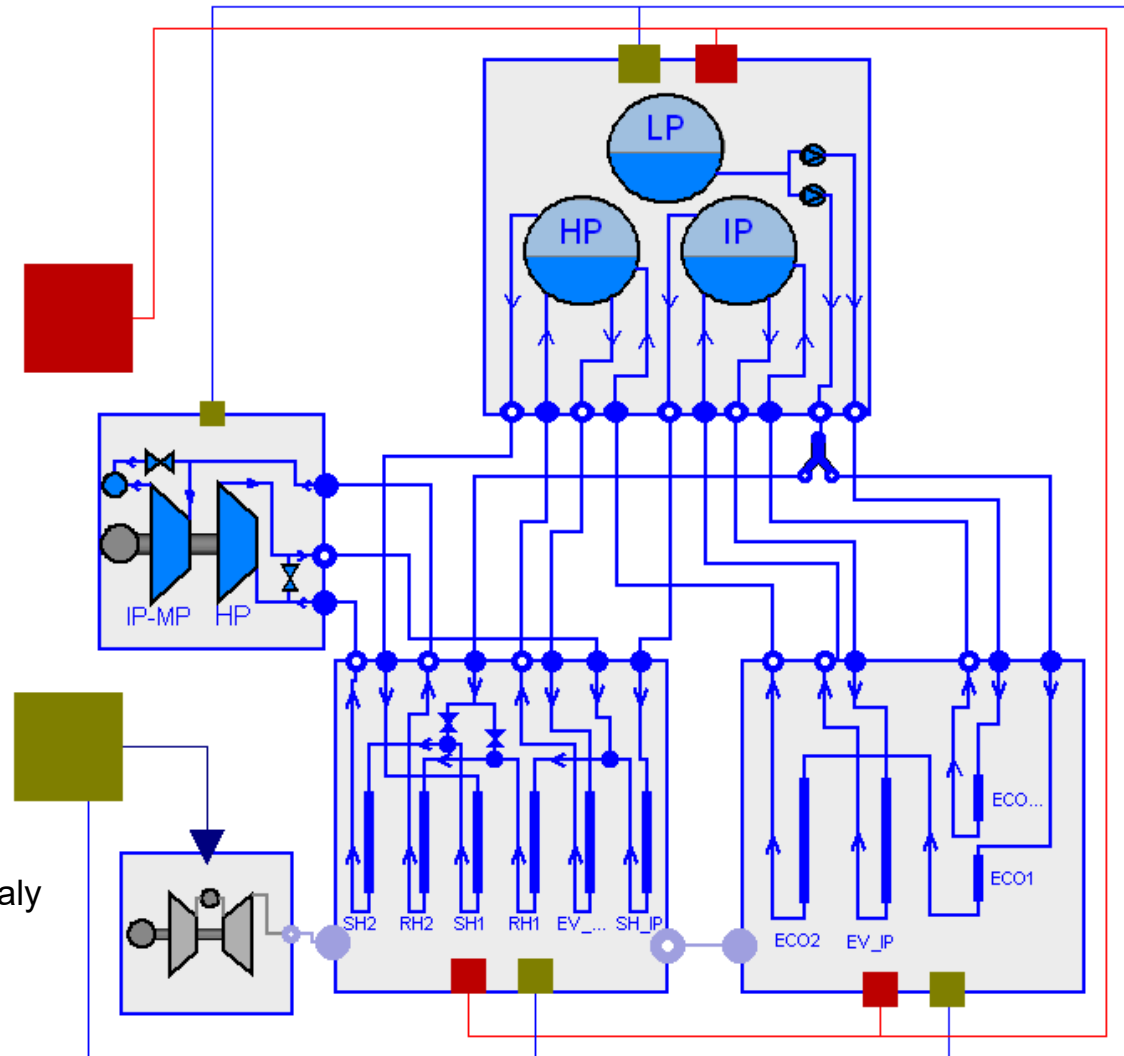


Combined-Cycle Power Plant

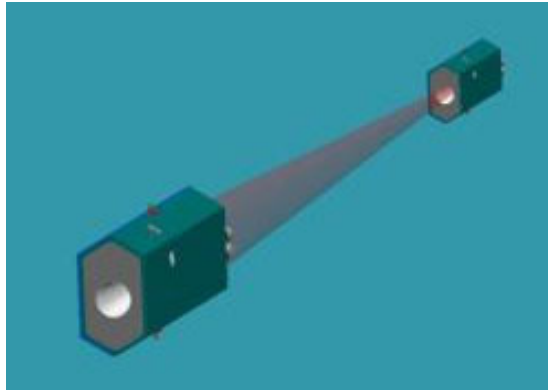
Plant model – system level

- 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

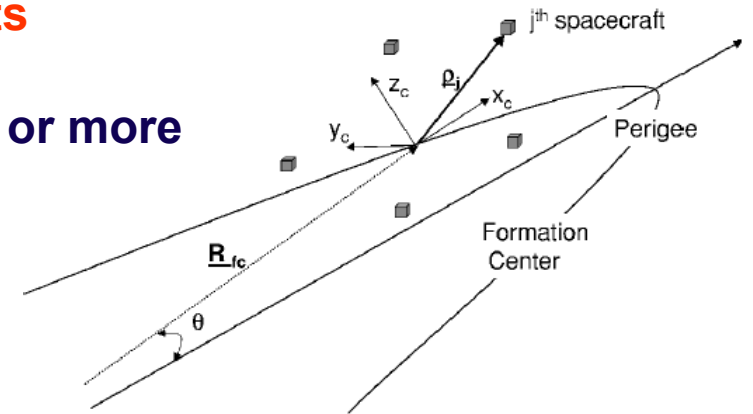
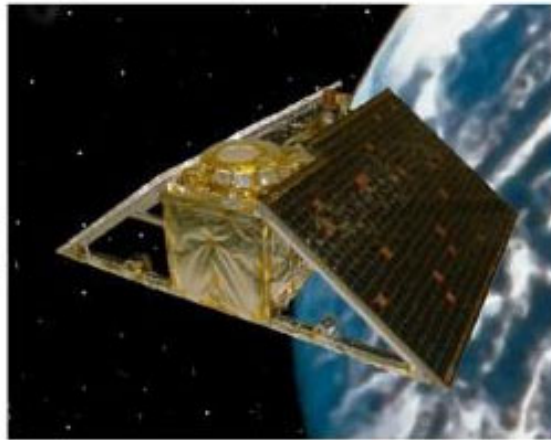


Modelica Spacecraft Dynamics Library



Formation flying on elliptical orbits

Control the relative motion of two or more spacecraft



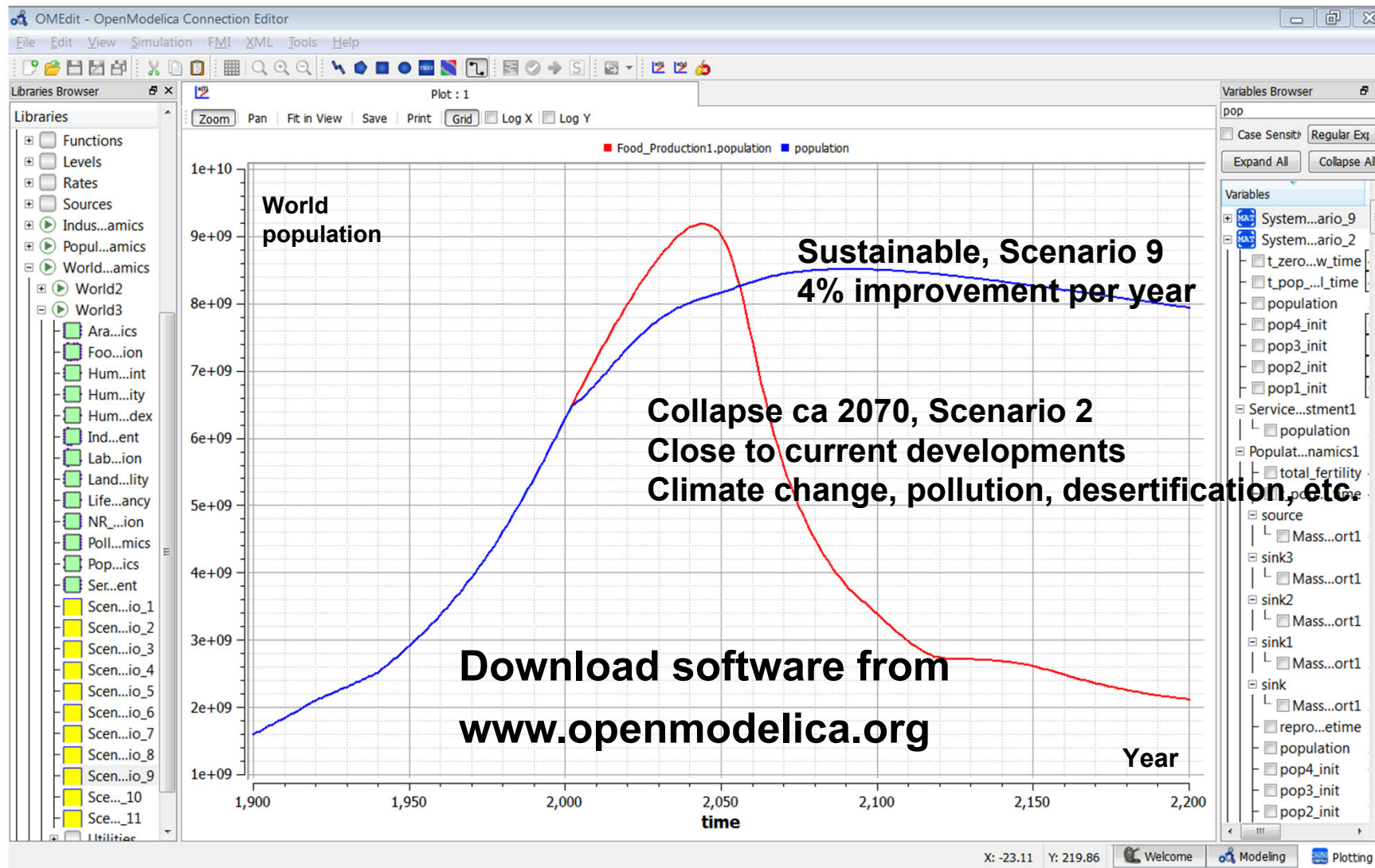
Attitude control for satellites using magnetic coils as actuators

Torque generation mechanism: interaction between coils and geomagnetic field

Courtesy of Francesco Casella, Politecnico di Milano, Italy



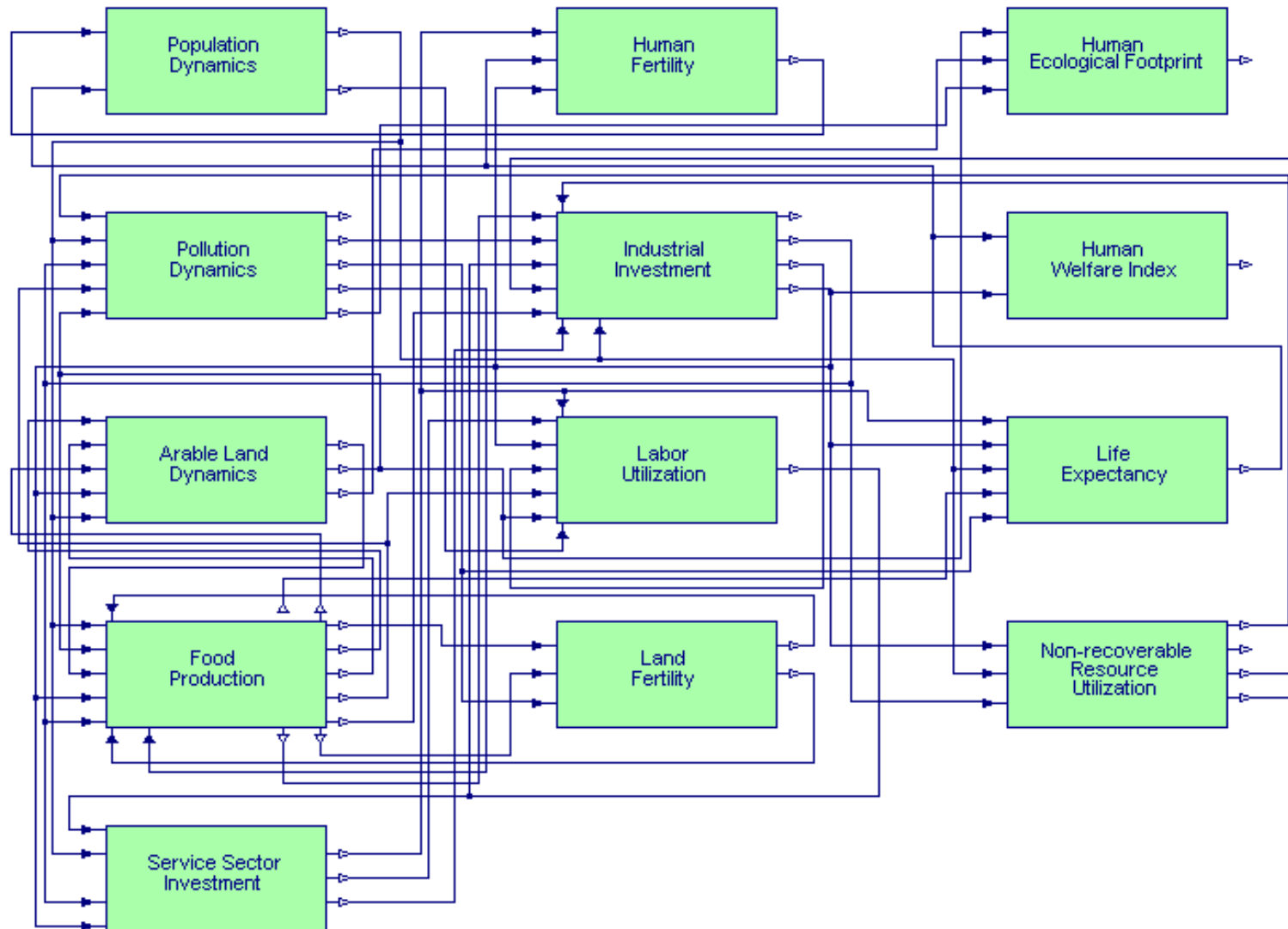
Biggest Immediate Challenge for Humanity – Create a Sustainable Society – Avoid Global Collapse in 50 Years



World System Dynamics Simulation with OpenModelica – World3 model, Meadows et al

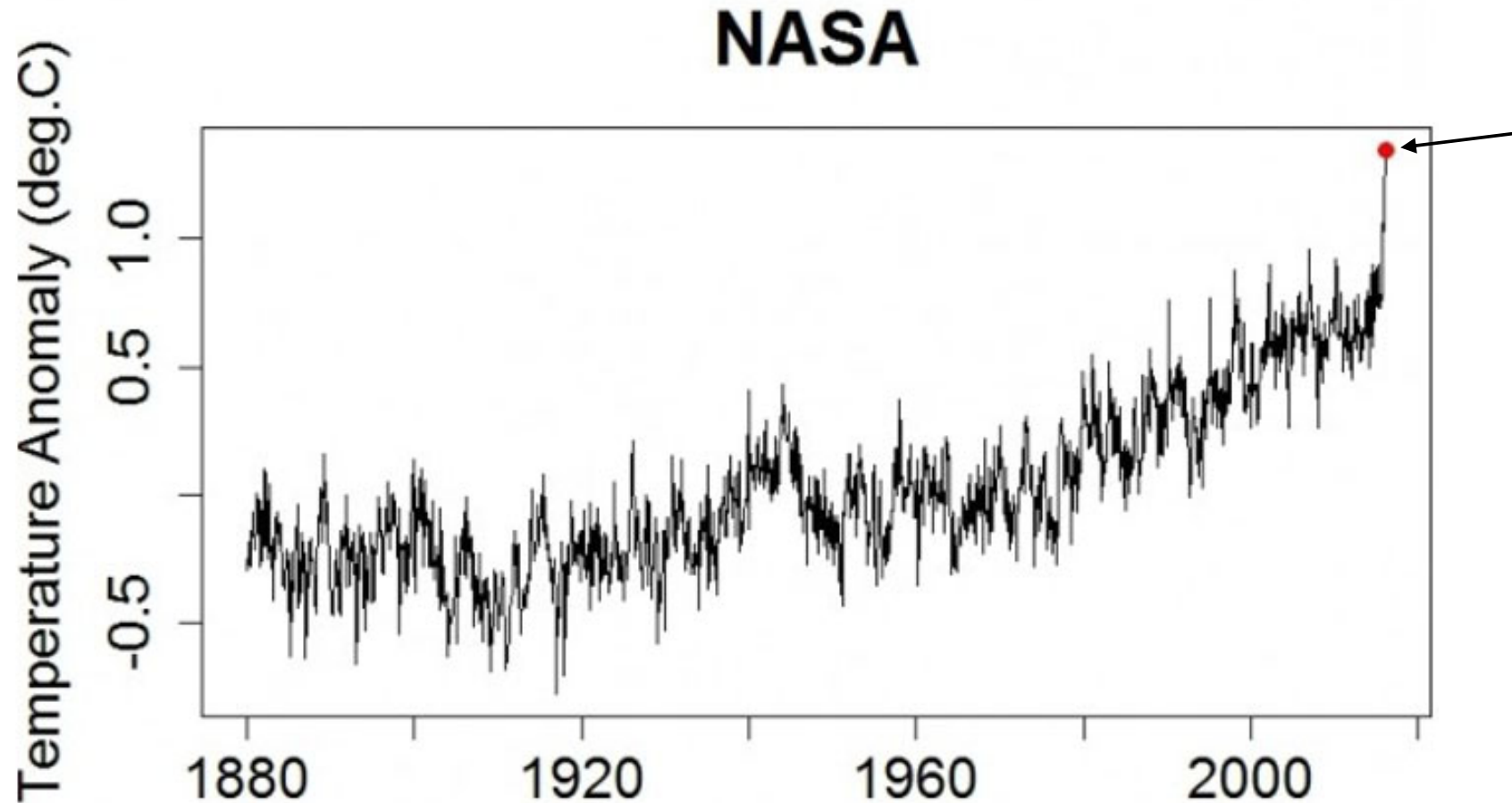
World3 Model in Modelica, Meadows et al, Cellier

Comprehensive model – 13 areas

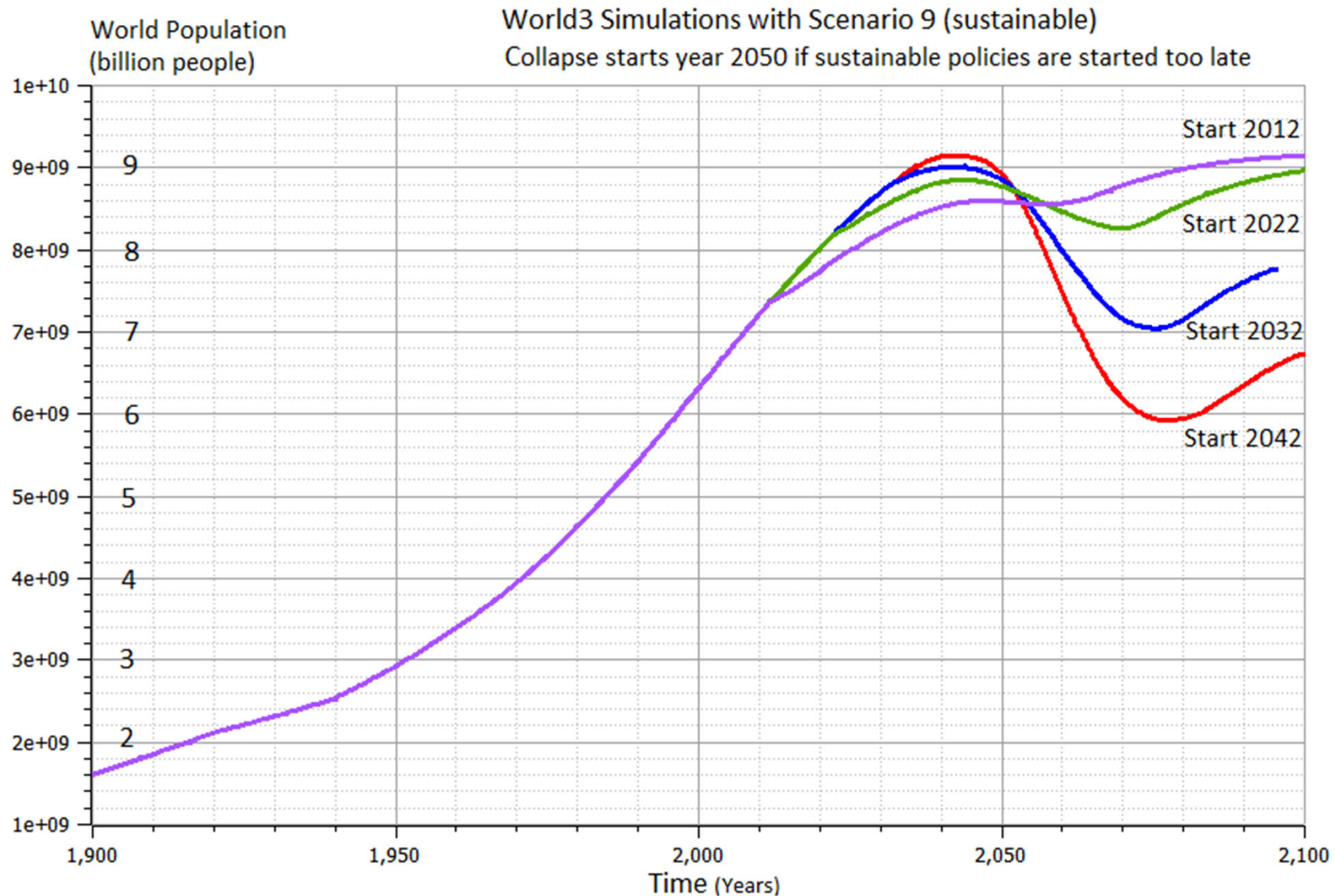


- Population dynamics
- Human fertility
- Human ecological footprint
- Pollution dynamics
- Industrial investment
- Human welfare
- Arable land
- Labor utilization
- Life expectancy
- Food production
- Land fertility
- Non-recoverable resource
- Service sector

Each Year New Record for Global Mean Temperature This is February 2016



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



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

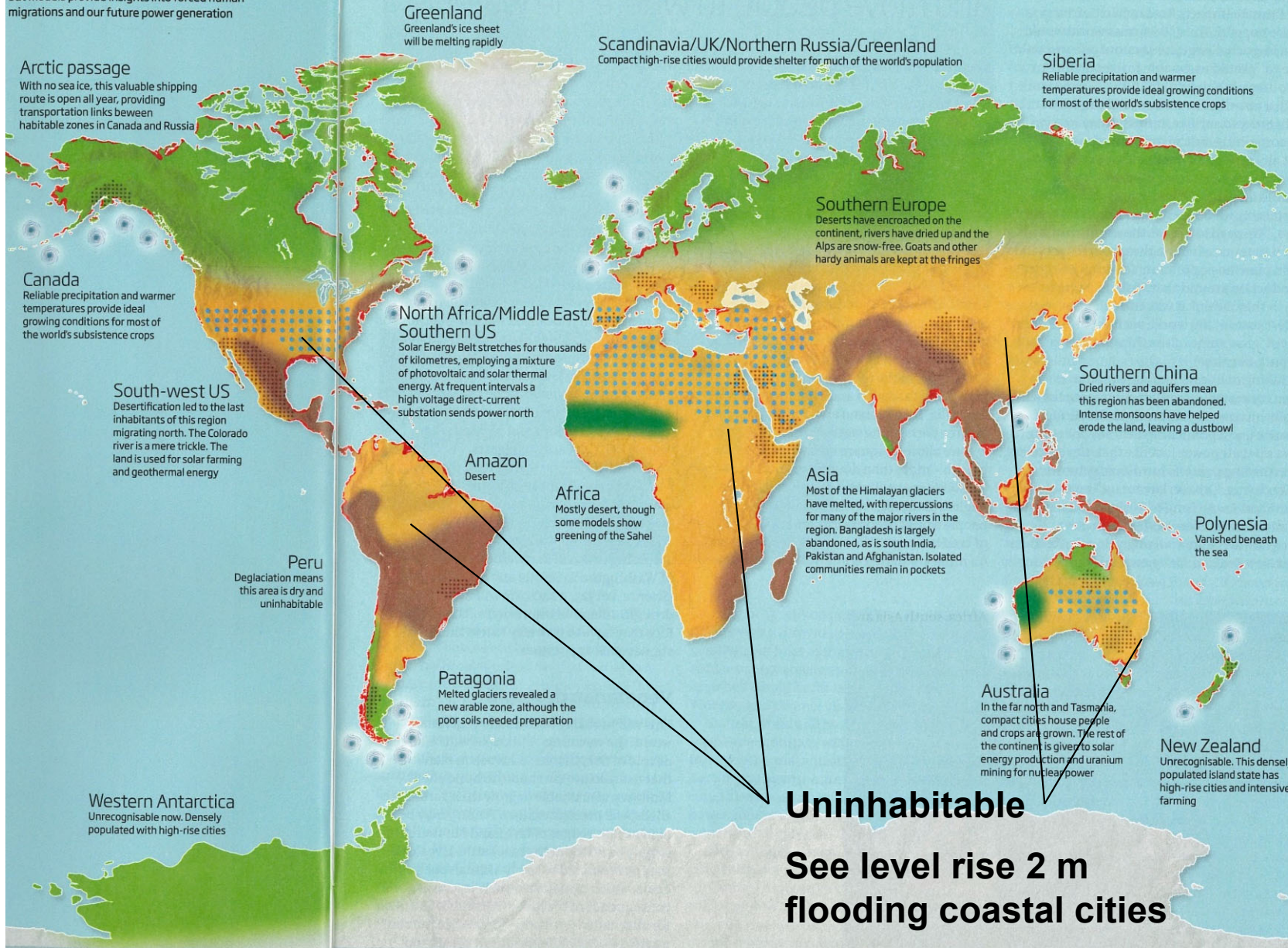


How the world could be in 80-100 years at a global warming of 4 degrees

Business-as-usual scenario, IPCC

The world: 4°C warmer

No one knows exactly what this world will look like, but models provide insights into forced human migrations and our future power generation



- Cities, agriculture
- Uninhabitable desert
- Uninhabitable due to extreme weather
- Flooded

Massive migration to northern Europe, Russia, and Canada

- Example Emissions
CO₂e / person
- Earth can handle 2 ton/yr
 - Flight Spain – 1 ton
 - Flight Canary Isl – 2 ton
 - Flight Thailand – 4 ton

Uninhabitable
See level rise 2 m
flooding coastal cities

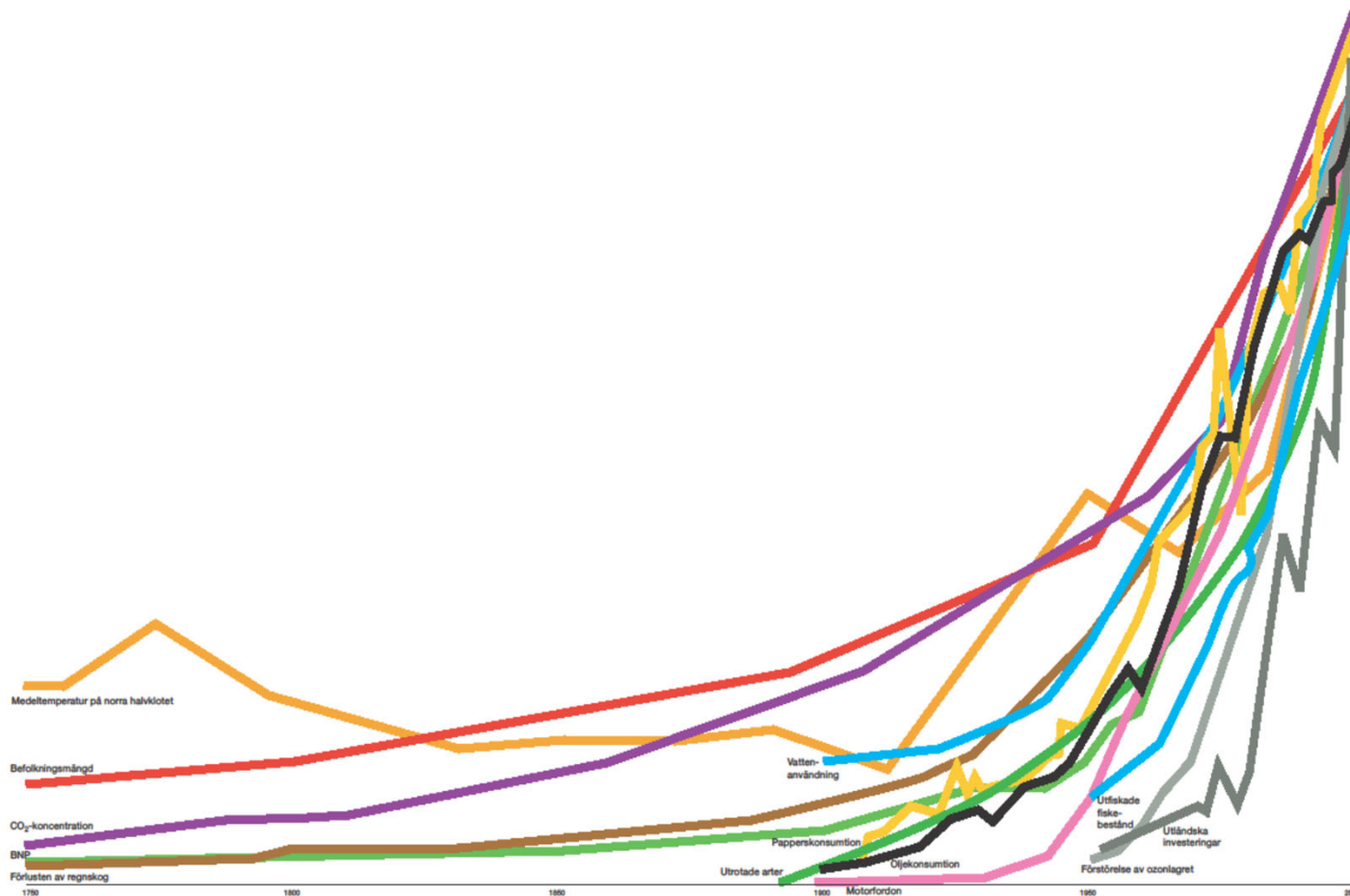
 Food-growing zones / Compact high-rise cities	 Uninhabitable desert	 Uninhabitable due to floods, drought or extreme weather	 Potential for reforestation	 Land lost due to rising sea levels, assuming a 2-metre rise	 Solar energy	 Geothermal energy	 Wind energy
---	--	---	---	---	---	--	---

References

New Scientist, 28 february 2009
 IPCC, business as usual scenario
www.climate-lab-book.ac.uk
www.atmosfair.de

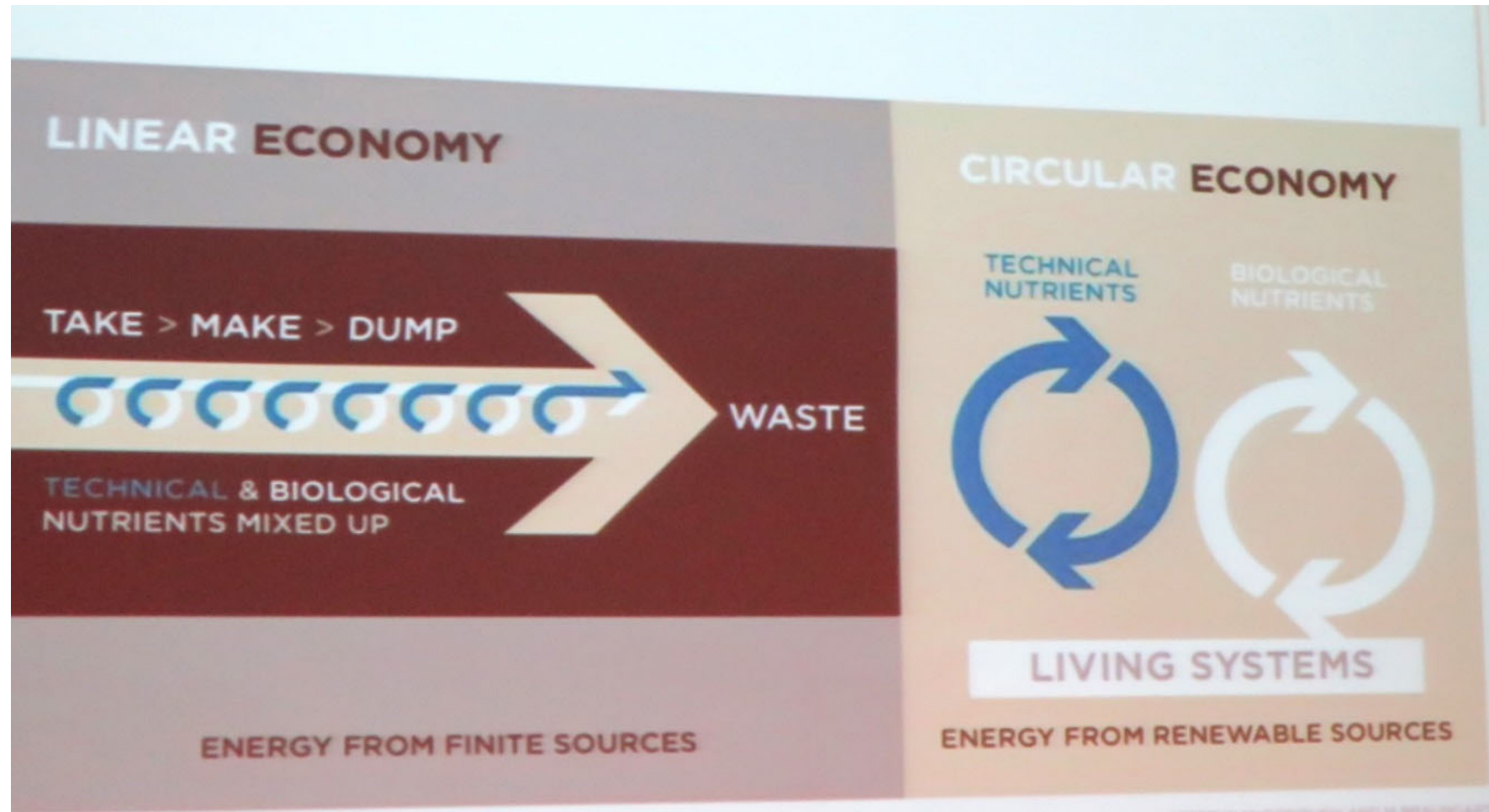
A Unique Point in History – Exponential Trends Approaches Planet Earth Boundaries

Year 1750-2000:



- Mean temperature north hemisphere,
- Population,
- CO₂-concentration,
- BNP,
- Loss av rain forest,
- Water usage
- Paper consumption,
- Exterminated species
- Oil consumption,
- Motor vehicles
- Destroyed fish populations
- Destruction of ozone layer
- Foreign investments

Need Smart Systems to Support a Circular Economy for a Sustainable Society



- **Circular** management of products, material, throughout the life-cycle
- Optimize manufacturing and usage over the **entire life cycle**

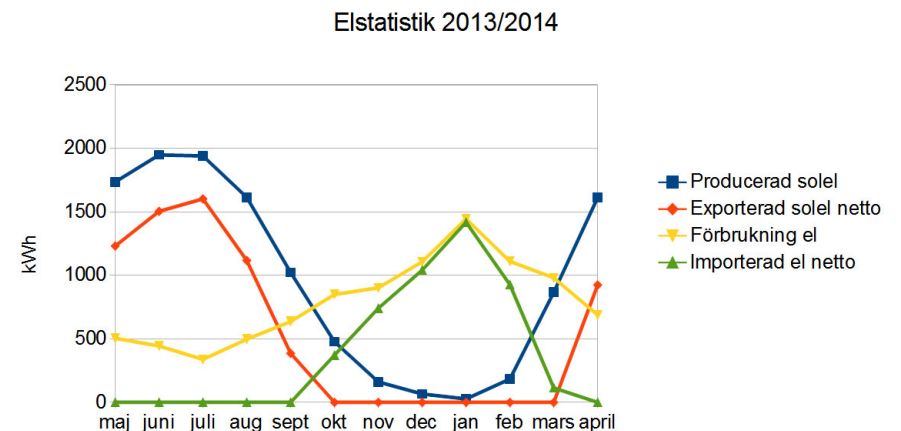
What Can You Do?

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 CO2 emission per year

Example Electric Cars

Can be charged by electricity from own solar panels



- Renault ZOE; 5 seat; Range:**
22kw (2014) vs 41 kw battery (2017)
- **Realistic Swedish drive cycle:**
 - **Summer: 165 km, now 300 km**
 - **Winter: 110 km, now 200 km**
- Cheap fast AC chargers (22kw, 43kw)**



Tesla model S
range 480 km



DLR ROboMObil

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

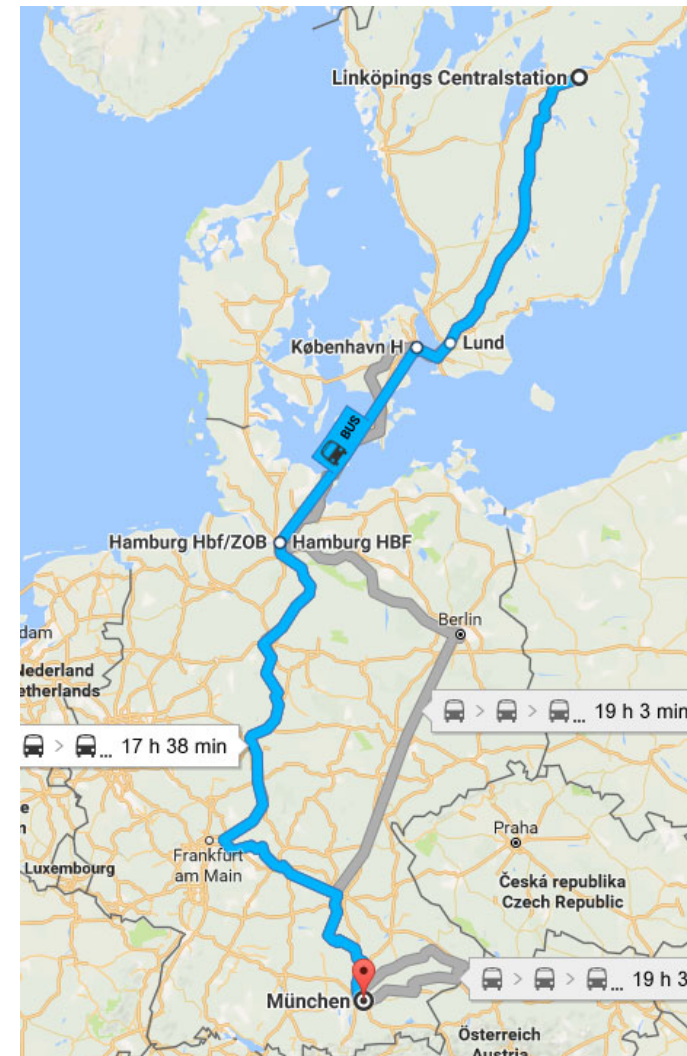
What Can You Do?

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

More Examples, PF travel 2016:

- Train Linköping-Paris, Dec 3-6, EU project meeting
- Train Linköping-Dresden, Dec 10-16, 1 week workshop



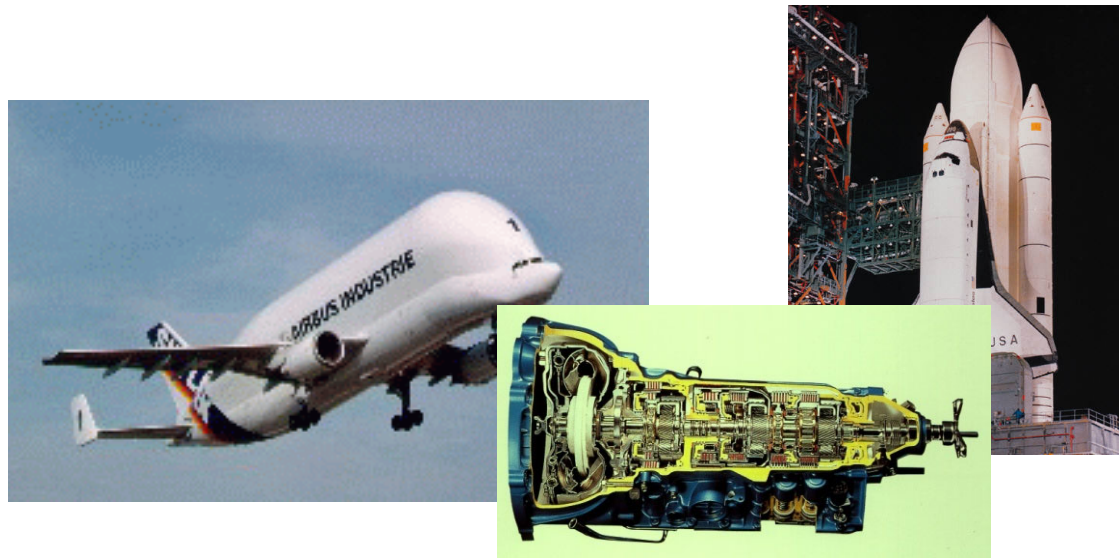
Train
travel
Linköping
- Munich

A satellite-style world map showing the continents. Overlaid on the map are several small, semi-transparent orange rectangles. One rectangle is located in Mexico, and others are scattered across Africa, the Middle East, India, Southeast Asia, and Australia. A large orange text box is centered over the map.

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

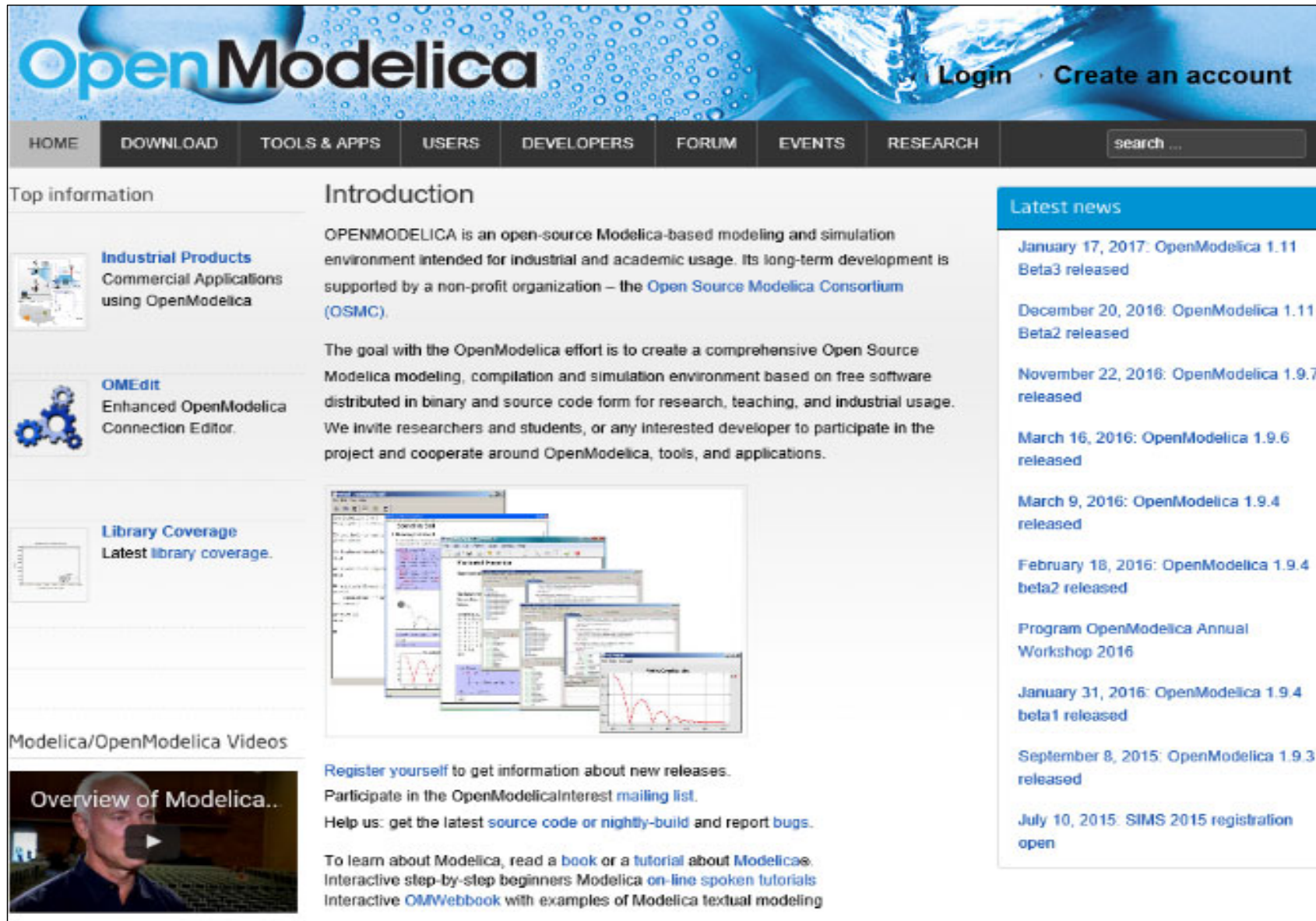
Part II

Introduction to the OpenModelica Environment



The OpenModelica Environment

www.openmodelica.org



The screenshot shows the OpenModelica website homepage. At the top, there is a navigation bar with the following links: HOME, DOWNLOAD, TOOLS & APPS, USERS, DEVELOPERS, FORUM, EVENTS, RESEARCH, and a search box. The main content area is divided into several sections:

- Top information:** A sidebar on the left containing links to "Industrial Products" (Commercial Applications using OpenModelica), "OMEdit" (Enhanced OpenModelica Connection Editor), and "Library Coverage" (Latest library coverage).
- Introduction:** A central section with the heading "Introduction". It describes OPENMODELICA as an open-source Modelica-based modeling and simulation environment intended for industrial and academic usage. It mentions that its long-term development is supported by the Open Source Modelica Consortium (OSMC). The text states the goal is to create a comprehensive Open Source Modelica modeling, compilation and simulation environment based on free software distributed in binary and source code form for research, teaching, and industrial usage. It invites researchers and students, or any interested developer to participate in the project and cooperate around OpenModelica, tools, and applications. Below the text is a screenshot of the OpenModelica software interface, showing various windows and plots.
- Latest news:** A sidebar on the right with a blue header, listing recent releases and events:
 - January 17, 2017: OpenModelica 1.11 Beta3 released
 - December 20, 2016: OpenModelica 1.11 Beta2 released
 - November 22, 2016: OpenModelica 1.9.7 released
 - March 16, 2016: OpenModelica 1.9.6 released
 - March 9, 2016: OpenModelica 1.9.4 released
 - February 18, 2016: OpenModelica 1.9.4 beta2 released
 - Program OpenModelica Annual Workshop 2016
 - January 31, 2016: OpenModelica 1.9.4 beta1 released
 - September 8, 2015: OpenModelica 1.9.3 released
 - July 10, 2015: SIMS 2015 registration open
- Modelica/OpenModelica Videos:** A section at the bottom left featuring a video thumbnail titled "Overview of Modelica..".

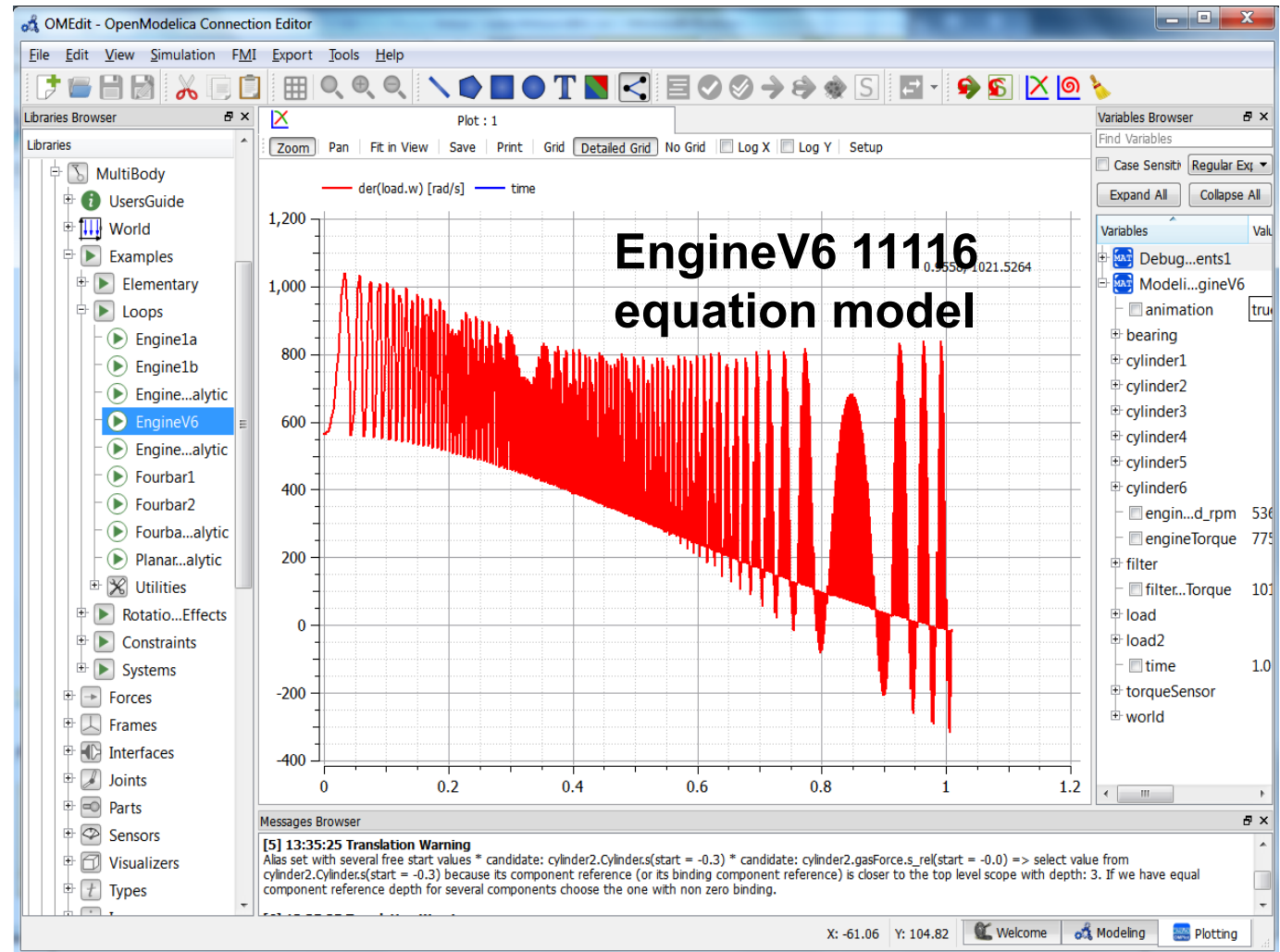
Below the introduction text, there are several links and instructions:

- Register yourself to get information about new releases.
- Participate in the OpenModelicaInterest mailing list.
- Help us: get the latest source code or nightly-build and report bugs.
- To learn about Modelica, read a book or a tutorial about Modelica.
- Interactive step-by-step beginners Modelica on-line spoken tutorials
- Interactive OMWebbook with examples of Modelica textual modeling

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 and OMWebbook for teaching
- Spokentutorial for teaching

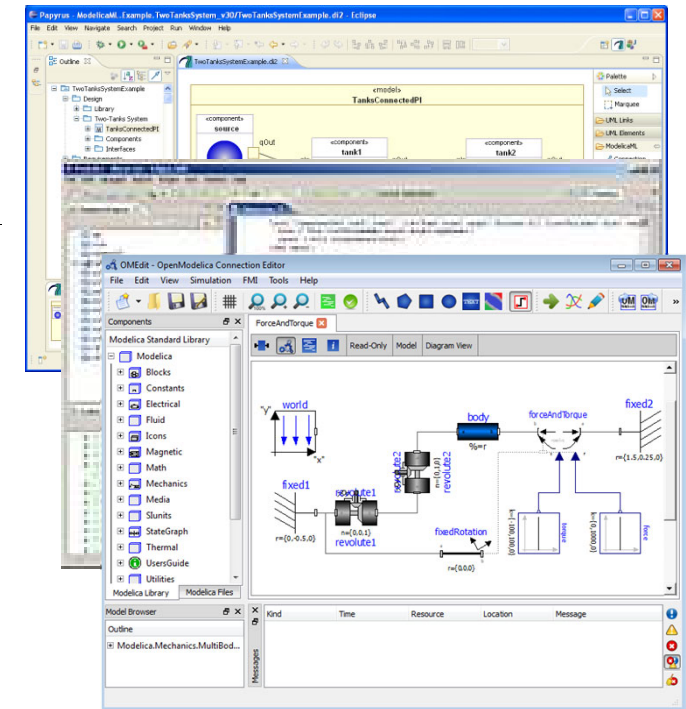
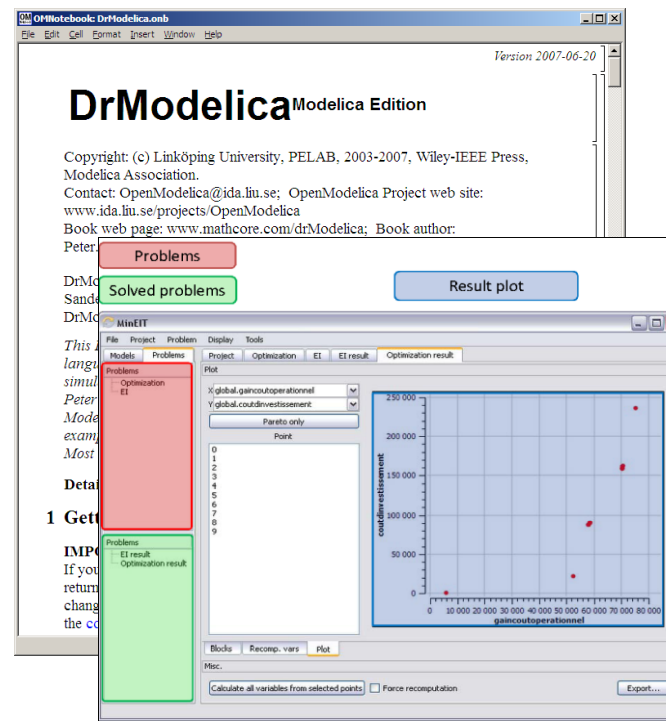
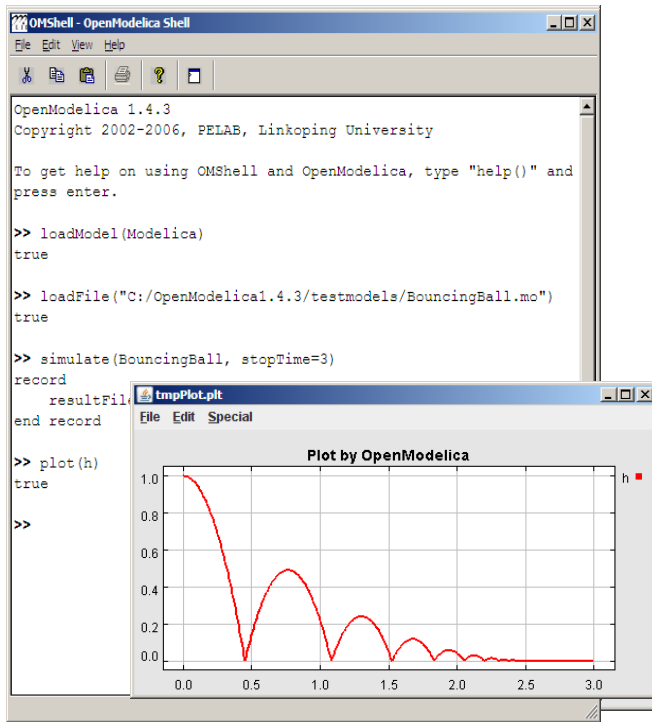


The OpenModelica Open Source Environment

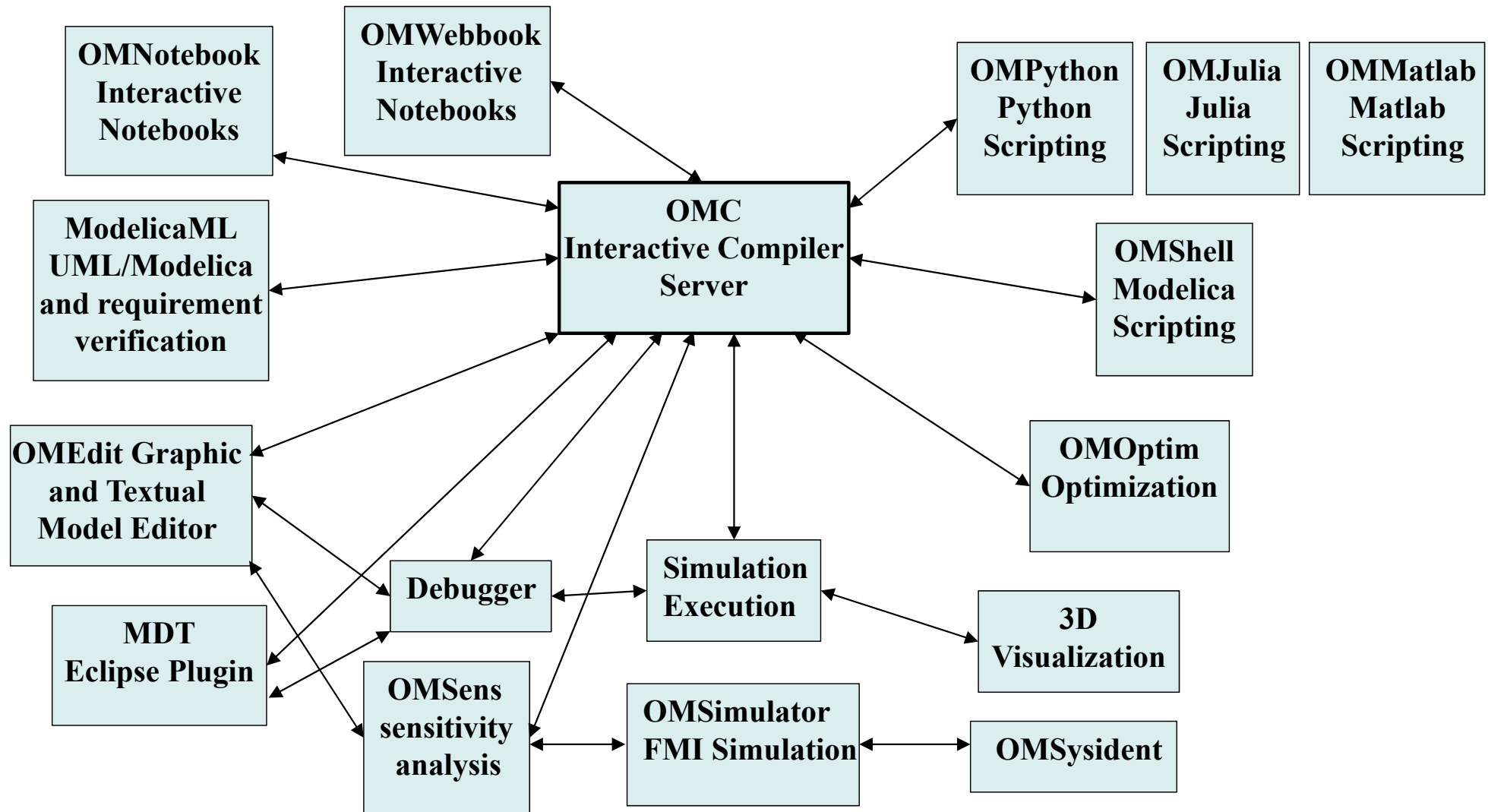
www.openmodelica.org

- **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
- **OMSimulator** – FMI/TLM simulator

new →



The OpenModelica Tool Architecture



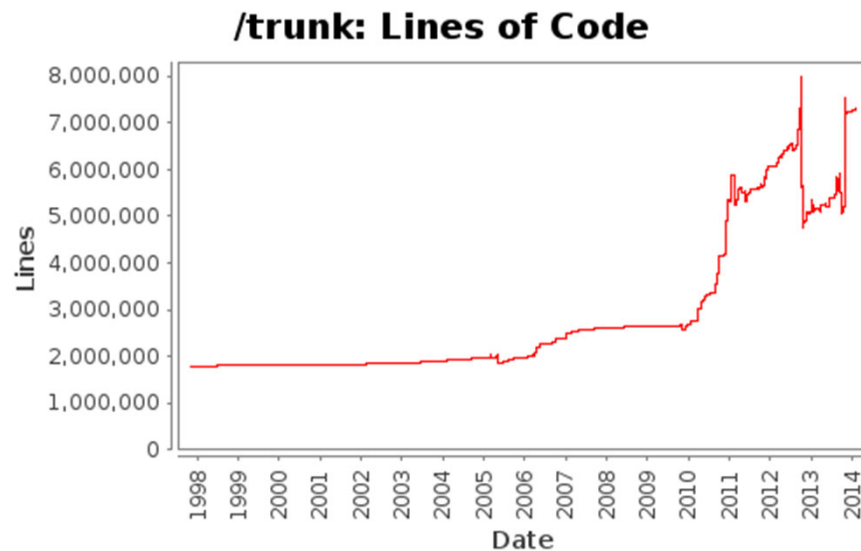
OSMC – International Consortium for Open Source Model-based Development Tools, 53 members Febr 2018

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



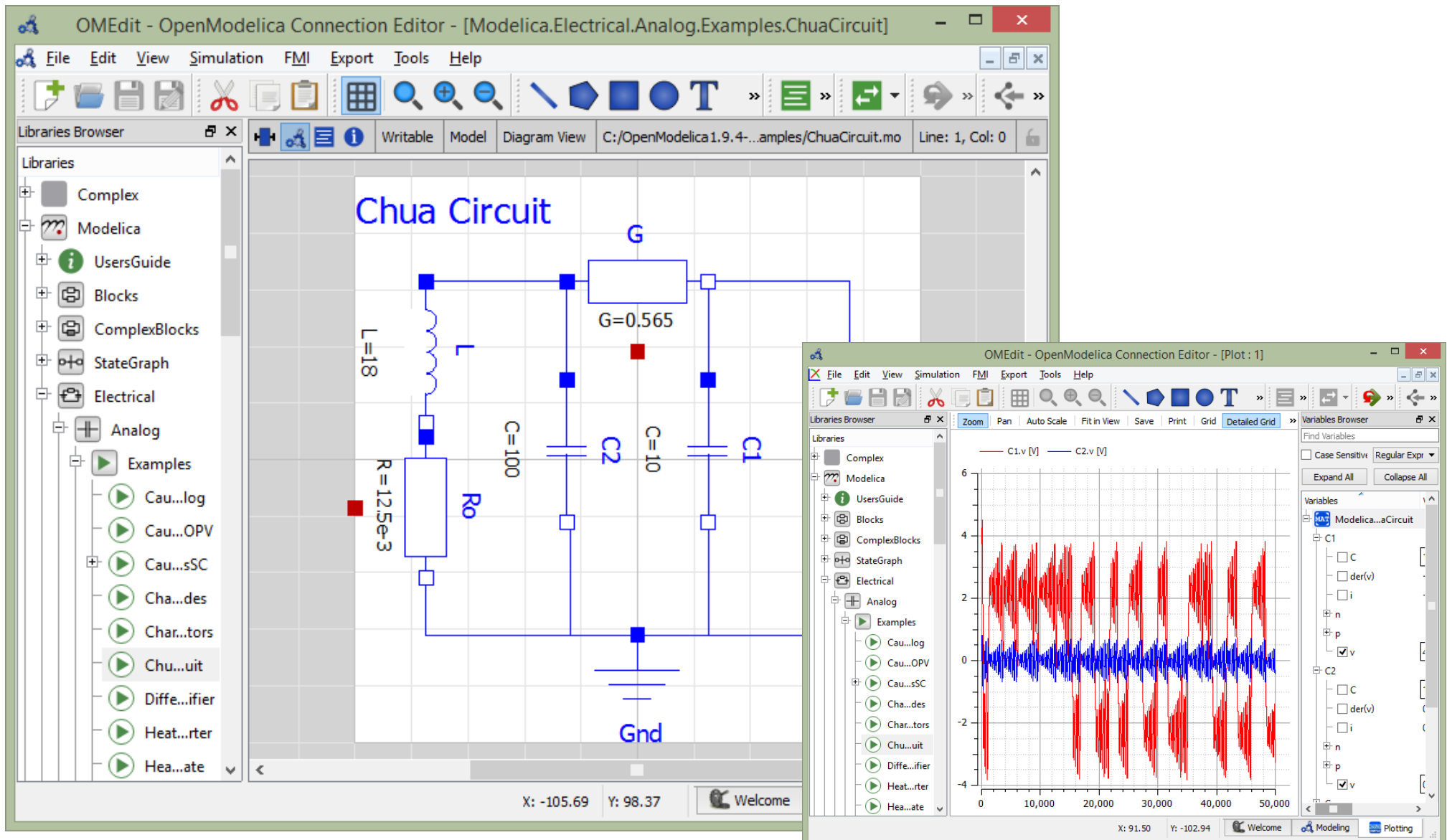
Industrial members

- ABB AB, Sweden
- Berger IT-Cosmos, Germany
- Bosch Rexroth AG, Germany
- Brainheart Energy AB, 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
- INRIA, Rennes, France
- ISID Dentsu, Tokyo, Japan
- Maplesoft, Canada
- RTE France, Paris, France
- Saab AB, Linköping, Sweden
- Scilab Enterprises, France
- SKF, Göteborg, Sweden
- TLK Thermo, Germany
- Siemens Turbo, Sweden
- Sozhou Tongyuan, China
- Talent Swarm, Spain
- VTI, Linköping, Sweden
- VTT, Finland
- Wolfram MathCore, Sweden

University members

- FH Bielefeld, Bielefeld, Germany
- University of Bolivar, Colombia
- TU Braunschweig, Germany
- University of Calabria, Italy
- Univ California, Berkeley, USA
- Chalmers Univ, Control, Sweden
- Chalmers Univ, Machine, Sweden
- TU Darmstadt, Germany
- TU Delft, The Netherlands
- TU Dresden, Germany
- Université Laval, Canada
- Georgia Inst of Technology, USA
- Ghent University, Belgium
- Halmstad University, Sweden
- Heidelberg University, Germany
- TU Hamburg/Harburg Germany
- IIT Bombay, Mumbai, India
- KTH, Stockholm, Sweden
- Linköping University, 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
- Univ College SouthEast Norway
- Tsinghua Univ, Beijing, China
- Vanderbilt Univ, USA

OpenModelica Graphical Editor and Plotting



OpenModelica Simulation in Web Browser Client

OpenModelica simulation example
Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.fullRobot

Simulation finished. Time: 00:40

Stop time, sec: 1.8
Output intervals: 500
Tolerance: 0.0001

Model Results

MultiBody RobotR3.FullRobot

OpenModelica simulation example
Examples.Systems.RobotR3.fullRobot

Simulation finished. Time: 00:40

Model Results

Plot variable: mechanics.r3.w

0.5
0.0
-0.5
-1.0
-1.5
-2.0
-2.5

0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75

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

Spoken-Tutorial step-by-step OpenModelica and Modelica Tutorial Using OMEdit. Link from www.openmodelica.org

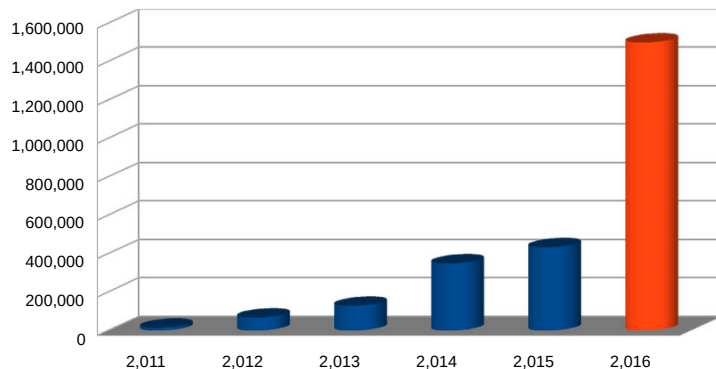


To learn about Modelica, read a [book](#) or a [tutorial](#) about [Modelica®](#).
 Interactive step-by-step beginners Modelica [on-line spoken tutorials](#)
 Interactive [OMWebbook](#) with examples of Modelica textual modeling

English (804) [Reset dropdowns](#)

OpenModelica is an open source modelling and simulation environment intended for industrial and academic usage. It is an object oriented declarative multi domain modelling language for complex systems. This environment can be used to work for both steady state as well as dynamic systems. Attractive strategy when dealing with design and optimization problems. As all the equations are solved simultaneously it doesn't matter whether the unknown variable in an input or output variable. [Read more](#)

Number of students/teachers trained in their colleges/schools



About 12 results found.

[Instruction Sheet](#)



1. Introduction to OMEdit

Foss : OpenModelica - English

Outline: Introduction to OpenModelica Introduction to OMEdit Perspectives in OMEdit Browsers in OMEdit View icons in OMEdit Open a Class from Libraries Browser Checking for correctnes..

Basic

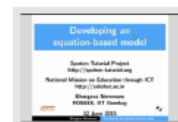


2. Examples through OMEdit

Foss : OpenModelica - English

Outline: Expand Modelica library Expand Electrical library Expand Analog library Open Rectifier Class Compare the values of IDC & Losses time vs Losses plot Expand Mechanics library ..

Basic



3. Developing an equation-based model

Foss : OpenModelica - English

Outline: Introduction to OMEdit Declaration of variables and equations Simulation of a model in

Basic

OMnotebook Interactive Electronic Notebook

Here Used for Teaching Control Theory

1 Kalman Filter

Often we don't have access to the internal states of a system and we have to reconstruct the state of the system based on measurements. The idea with an observer is that we feedback the error. If the estimation is correct then the difference should be zero.

Another difficulty is that the measured quantities of the system are noisy.

$$\begin{cases} \dot{\hat{x}} = A\hat{x} + Bu + e \\ y = C\hat{x} + v \end{cases}$$

Here e denoting a disturbance in the input signal and v can be evaluated by the difference between the measured and estimated values.

$$K(y(t) - C\hat{x}(t))$$

By using this quantity as feedback we obtain the observer dynamics:

$$\dot{\hat{x}} = A\hat{x}(t) + Bu(t) + K(y(t) - C\hat{x}(t))$$

Now form the error as $e = x - \hat{x}$.

The differential error is $\dot{e} = (A - KC)e + v$.

OMNotebook: Kalman.onb

File Edit Cell Format Insert Window Help

1 Kalman Filter

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(ARC,1));
parameter Real[:,:] ARC = 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
                    
```

Plot by OpenModelica

Legend: Kalman.stateSpace.y[1] (red), noKalman.stateSpace.y[1] (blue)

OM Web Notebook Generated from OMNotebook

Edit, Simulate, Plot Models on a Web Page

<http://omwebbook.openmodelica.org/>

OMNotebook

First Basic Class

1 HelloWorld

The program contains a declaration of a class called `HelloWorld` with two fields and one equation. The first field is the variable `x` which is initialized to a start value 2 at the time when the simulation starts. The second field is the variable `a`, which is a constant that is initialized to 2 at the beginning of the simulation. Such a constant is prefixed by the keyword parameter in order to indicate that it is constant during simulation but is a model parameter that can be changed between simulations.

The Modelica program solves a trivial differential equation: $x' = -a * x$. The variable `x` is a state variable that can change value over time. The `x'` is the time derivative of `x`.

```
class HelloWorld
  Real x(start = 1, fixed=true);
  parameter Real a = 1;
equation
  der(x) = - a * x;
end HelloWorld;
```

{HelloWorld}

2 Simulation of HelloWorld

```
simulate( HelloWorld, startTime=0, stopTime=3 )
```

record SimulationResult
 resultFile = "HelloWorld_res.mat",
 messages = ""
end SimulationResult;

Plot the results.

```
plot( x )
```

[done]

Zoom Pan Auto Scale Fit in View Save Print Grid Detailed Grid No Grid Log X Log Y Setup

Plot by OpenModelica

Ready

OMwebbook

First Basic Class

1 HelloWorld

The program contains a declaration of a class called `HelloWorld` with two fields and one equation. The first field is the variable `x` which is initialized to a start value 1 at the time when the simulation starts. The second field is the variable `a`, which is a constant that is initialized to 1 at the beginning of the simulation. Such a constant is prefixed by the keyword parameter in order to indicate that it is constant during simulation but is a model parameter that can be changed between simulations.

The Modelica program solves a trivial differential equation: $x' = -a * x$. The variable `x` is a state variable that can change value over time. The `x'` is the time derivative of `x`.

```
1 class HelloWorld
2   Real x(start = 1, fixed=true);
3   parameter Real a = 1;
4 equation
5   der(x) = - a * x;
6 end HelloWorld;
```

2 Simulation of HelloWorld

```
1 simulate( HelloWorld, startTime=0, stopTime=4 )
2
```

Plot the results.

```
1 plot( x )
2
```

OMPpython – Python Scripting with OpenModelica

- Interpretation of Modelica commands and expressions
- Interactive Session handling
- Library / Tool
- Optimized Parser results
- Helper functions
- Deployable, Extensible and Distributable

The screenshot shows a Python script named `test_execute_mode.py` in a text editor. The script uses the `OMPpython` library to load and simulate a Modelica model. The execution output in a terminal window shows the OMC server starting, the simulation parameters, and the resulting data. A separate window displays a plot of the simulation results, showing a damped oscillation.

```
import OMPpython

OMPpython.execute("loadFile('\\c:/OpenModelica1.8.1/testmodels/BouncingBall.mo")")
result=OMPpython.execute("simulate(BouncingBall, stopTime=2, method='Euler')")
print result
OMPpython.execute("plot(h)")

OMPpython.execute("quit()")
```

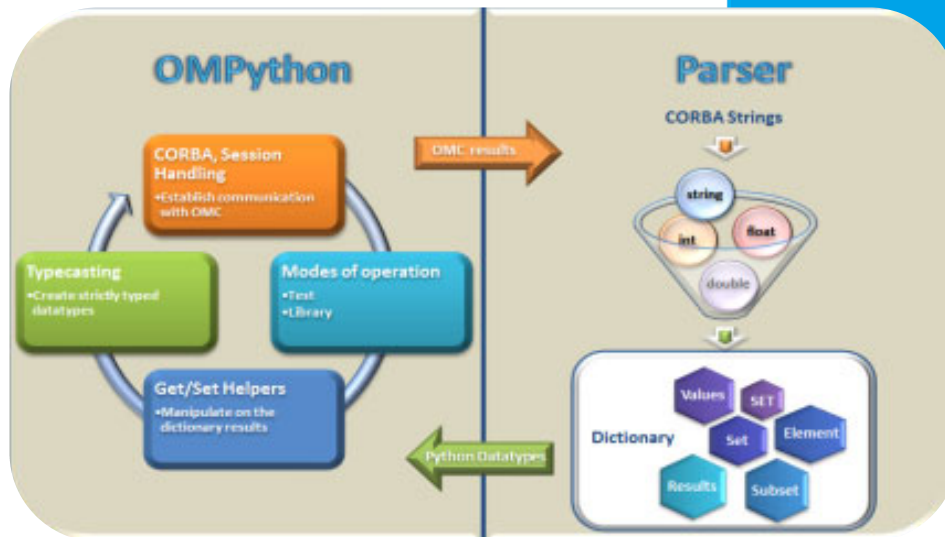
```
C:\Users\ganan642>python test_execute_mode.py
OMC Server is up and running at file:///c:/users/ganan642/appdata/local/temp/openmodelica.objid.20120825120756188000

{'simulationOptions': {'options': {'storeInTemp': False, 'cf_lags': {'simflags': {'variableFilter': 'noclean': False, 'outputFormat': 'mat', 'method': 'dassl', 'measureTime': False, 'stopTime': 2.0, 'startTime': 0.0, 'numberOfIntervals': 500, 'tolerance': 1e-06, 'fileNamePrefix': 'BouncingBall'}, 'simulationResults': {'timeCompile': 6.89815662792063, 'timeBackend': 0.0229111689831523, 'messages': {'timeFrontend': 0.0245992104508437, 'timeSimulation': 0.131418166559841, 'timeTemplate s': 0.0206379911344139, 'timeSimCode': 0.00999736303670383, 'time Total': 7.1078098383753, 'resultFile': 'C:/Users/ganan642/BouncingBall_res.mat'}}}}

OMC has been shutdown

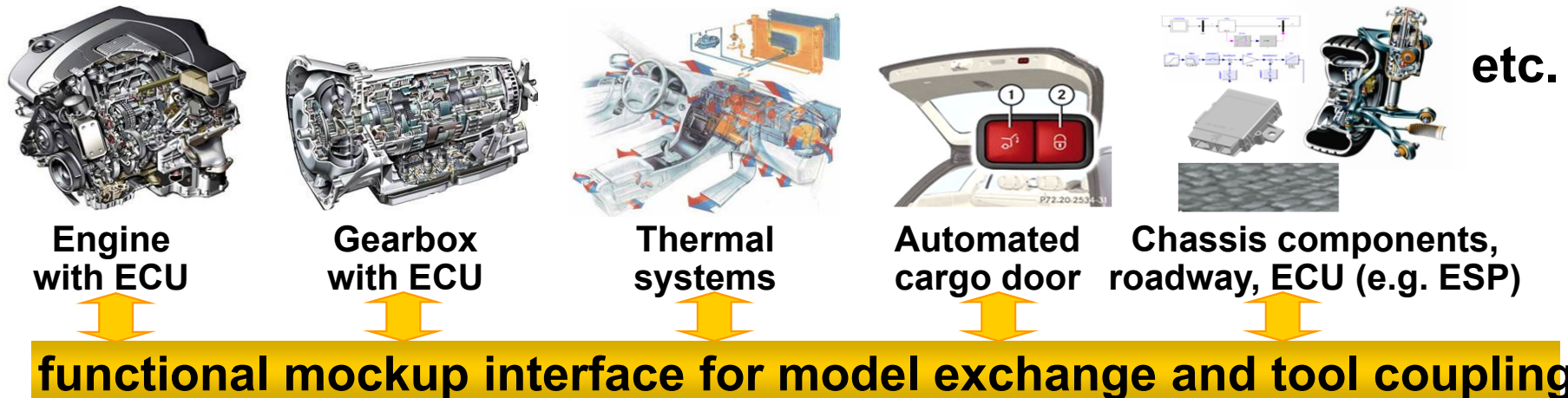
C:\Users\ganan642>
```

The plot shows a damped oscillation of a variable over time, with the amplitude decreasing as time progresses.



General Tool Interoperability & Model Exchange

Functional Mock-up Interface (FMI)

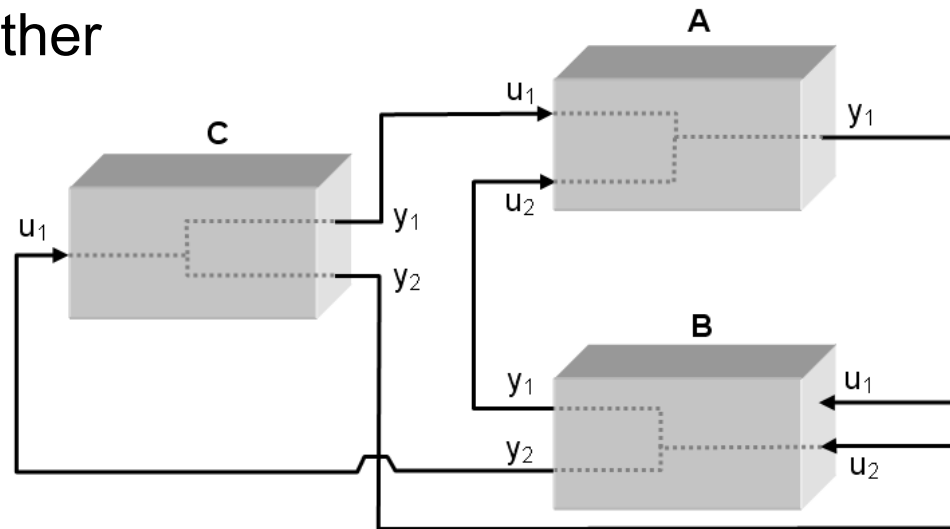


courtesy Daimler

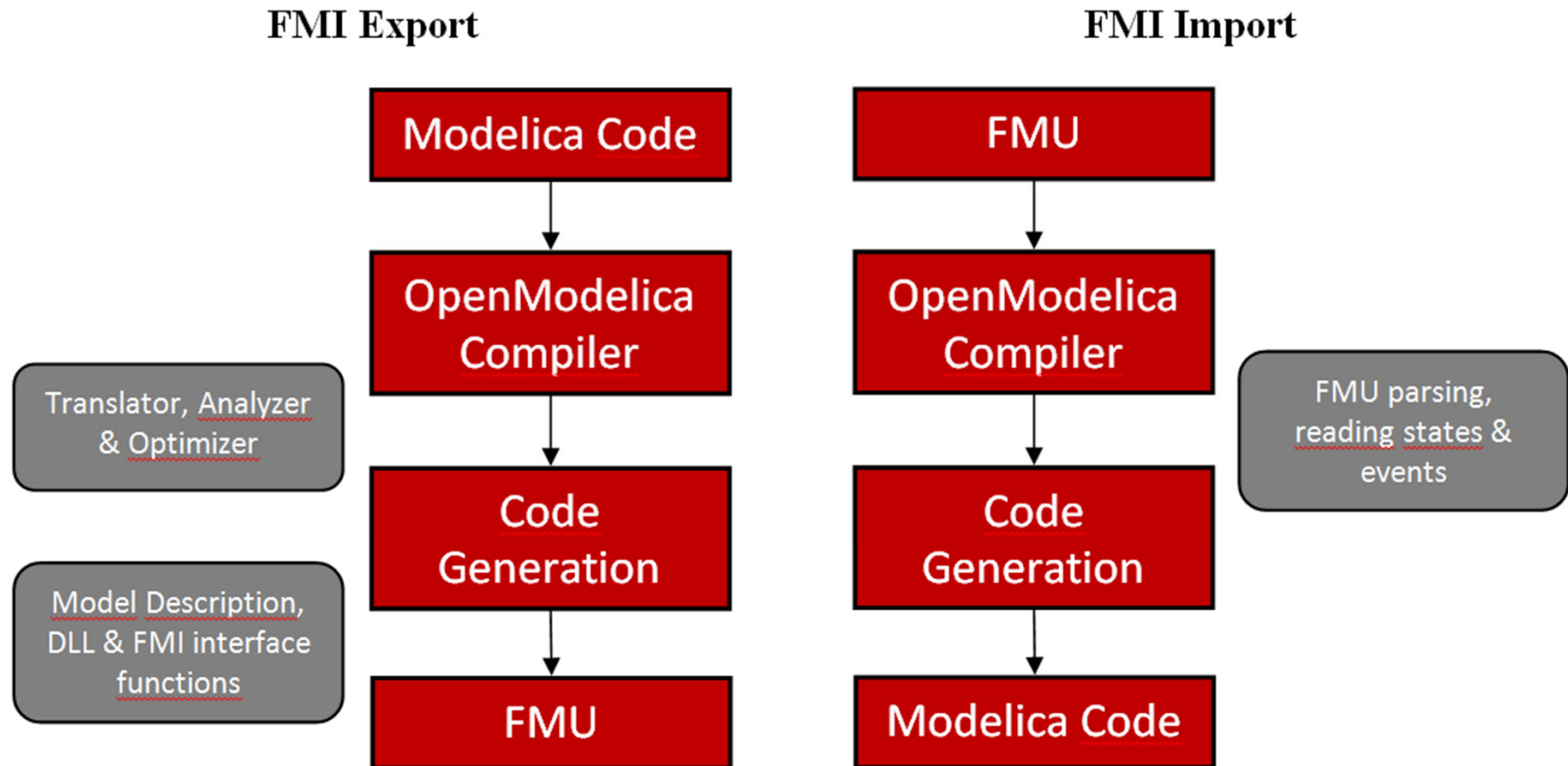
- FMI development was started by ITEA2 MODELISAR project. FMI is a Modelica Association Project now
- **Version 1.0**
- FMI for Model Exchange (released Jan 26,2010)
- FMI for Co-Simulation (released Oct 12,2010)
- **Version 2.0**
- FMI for Model Exchange and Co-Simulation (released July 25,2014)
- **> 80 tools** supporting it (<https://www.fmi-standard.org/tools>)

Functional Mockup Units

- Import and export of input/output blocks – **Functional Mock-Up Units – FMUs**, described by
 - differential-, algebraic-, discrete equations,
 - with time-, state, and step-events
- An **FMU** consists of **(compiled) C-code**, + interface description in **XML**
- An FMU can be large (e.g. 100 000 variables)
- An FMU can be used in an embedded system (small overhead)
- FMUs can be connected together

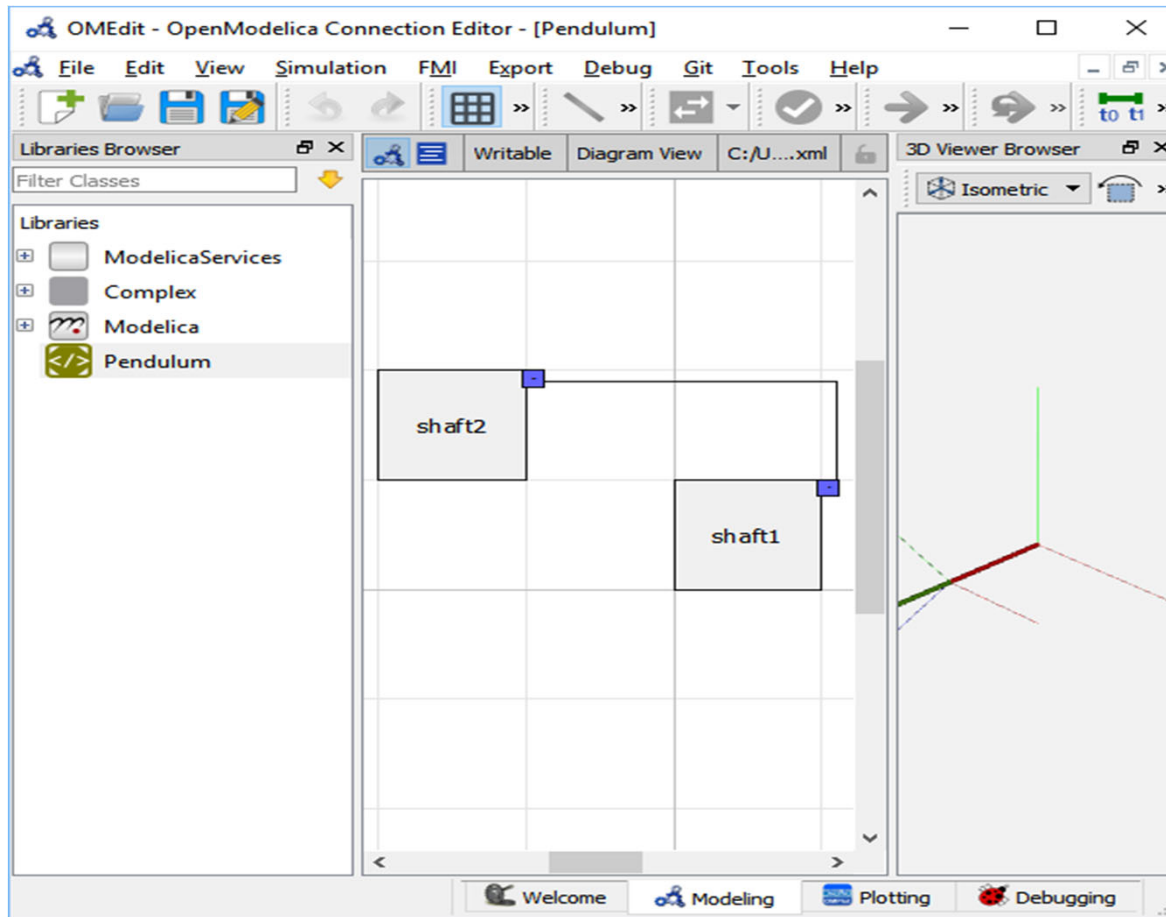


OpenModelica Functional Mockup Interface (FMI)



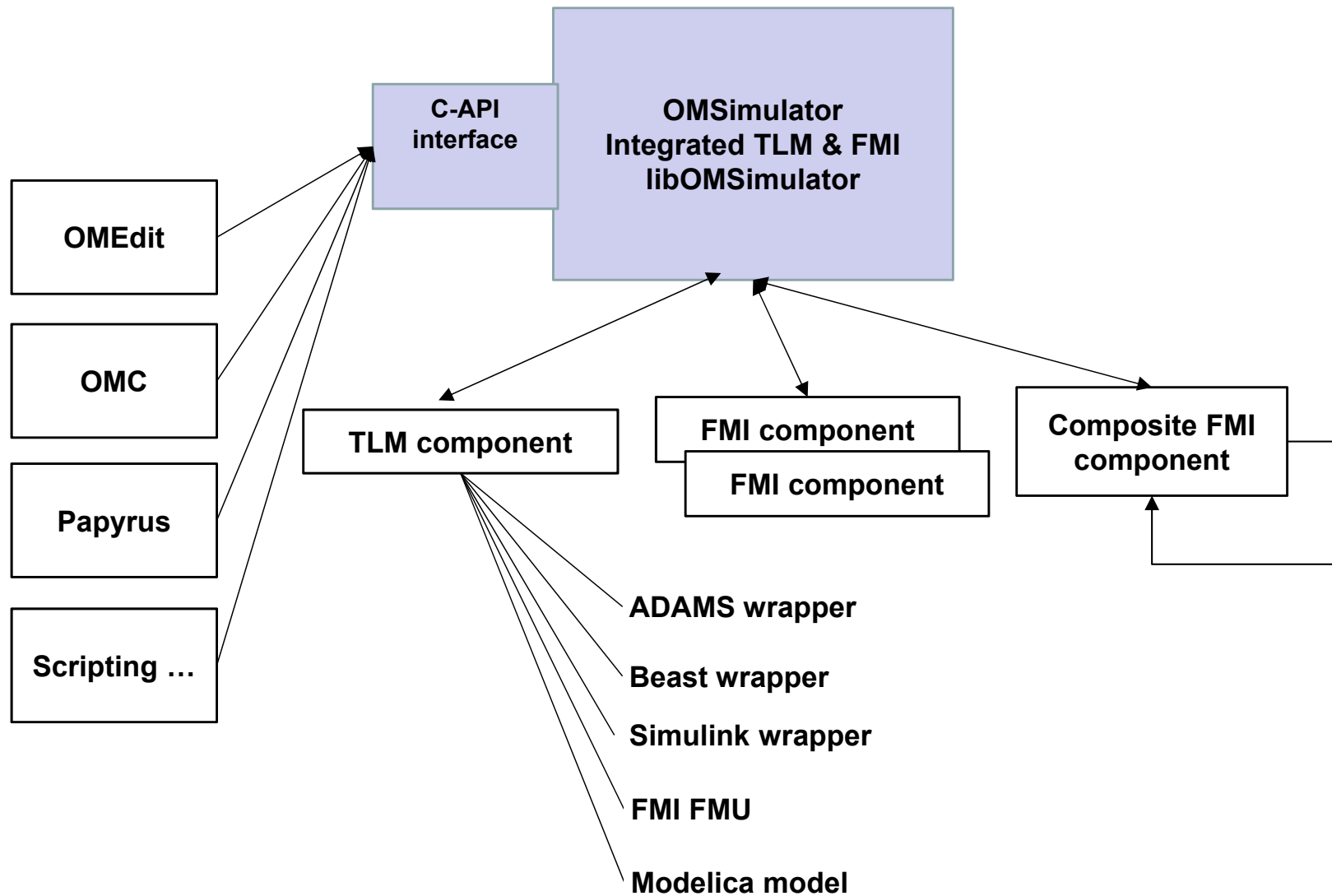
OMSimulator Composite Model Editor with 3D Viewer

Combine External (FMI) Models into New Models



- **Composite model editor** with 3D visualization of connected mechanical model components which can be FMUs, Modelica models, etc., or co-simulated components
- **3D animation** possible
- Composite model saved as **XML-file**

OMSimulator – Integrated FMI and TLM-based Cosimulator/Simulator in OpenModelica



Embedded System Support in OpenModelica

- Code generation of real-time Controllers from Modelica models for small foot-print platforms



Use Case: SBHS (Single Board Heating System)

Single board heating system (IIT Bombay)

- Use for teaching basic control theory
- Usually controlled by serial port (set fan value, read temperature, etc)
- OpenModelica can generate code targeting the ATmega16 on the board (AVR-ISP programmer in the lower left).
Program size is 4090 bytes including LCD driver and PID-controller (out of 16 kB flash memory available).



Movie Demo!

Example – Code Generation to SHBS



Code Generator Comparison, Full vs Simple

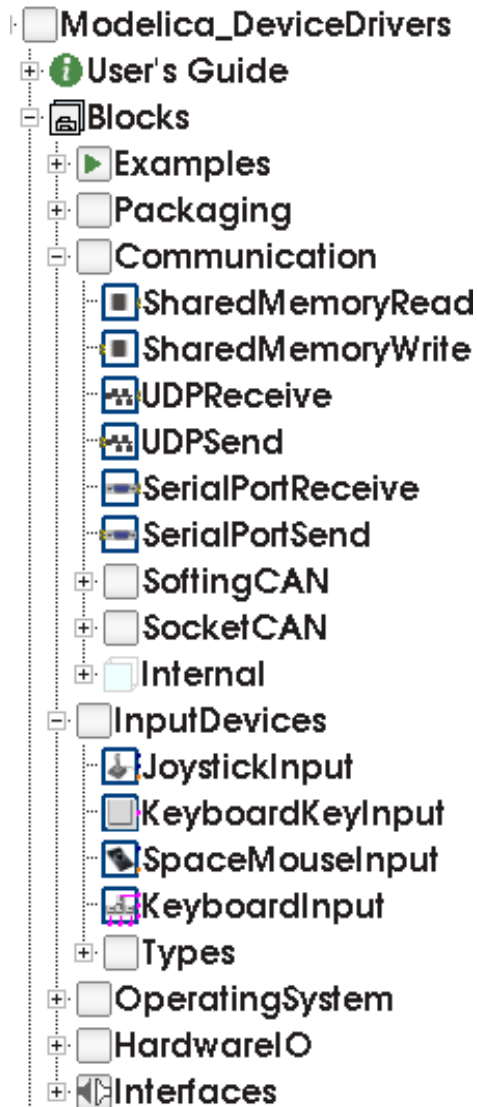
	Full Source-code FMU targeting 8-bit AVR proc	Simple code generator targeting 8-bit AVR proc
Hello World (0 equations)	43 kB flash memory 23 kB variables (RAM)	130 B flash memory 0 B variables (RAM)
SBHS Board (real-time PID controller, LCD, etc)	68 kB flash memory 25 kB variables (RAM)	4090 B flash memory 151 B variables (RAM)

The largest 8-bit AVR processor MCUs (Micro Controller Units) have 16 kB SRAM.

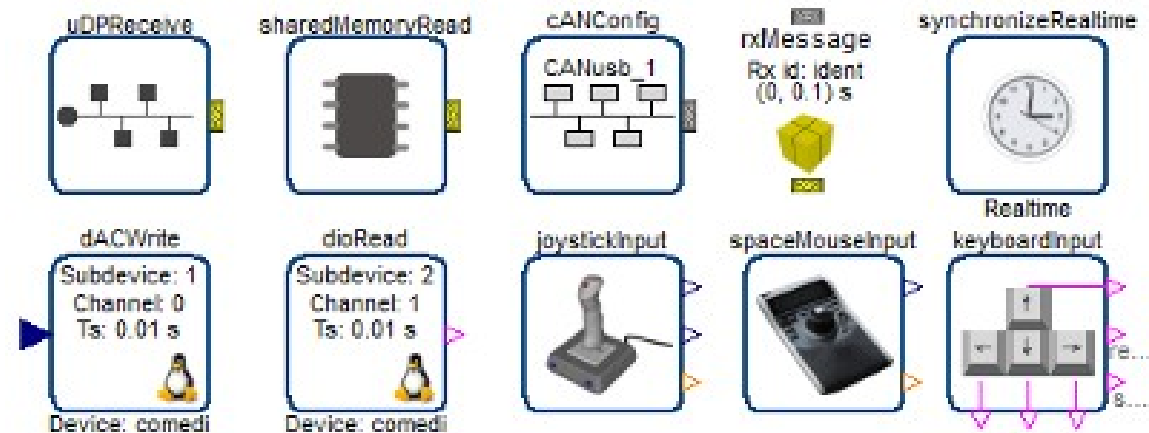
One of the more (ATmega328p; Arduino Uno) has 2 kB SRAM.

The ATmega16 we target has **1 kB SRAM available** (stack, heap, and global variables).

Communication & I/O Devices: MODELICA_DEVICEDRIVERS Library

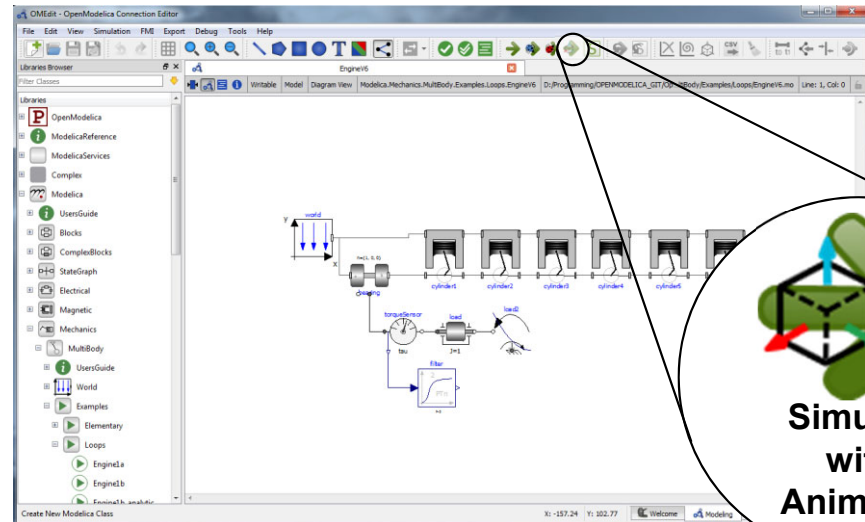


- **Free library** for interfacing hardware drivers
- **Cross-platform** (Windows and Linux)
- UDP, SharedMemory, CAN, Keyboard, Joystick/Gamepad
- DAQ cards for digital and analog IO (only Linux)
- Developed for **interactive real-time** simulations

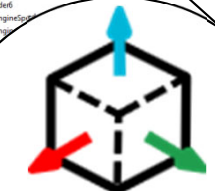
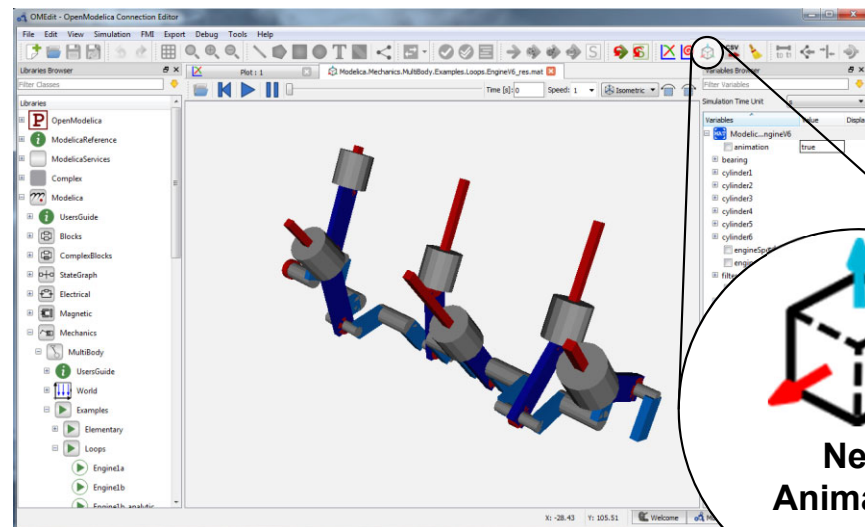
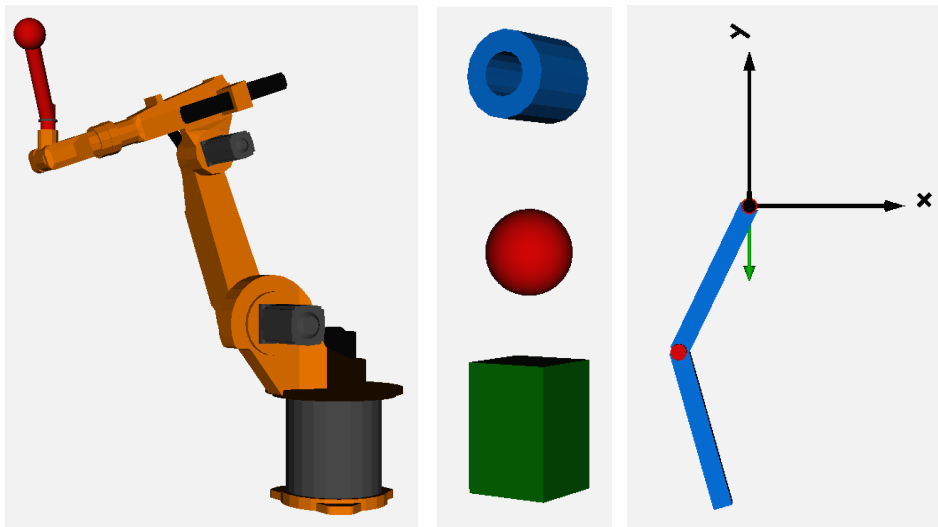


OMEdit 3D Visualization of Multi-Body Systems

- Built-in feature of OMEdit to animate MSL-Multi-Body shapes
- Visualization of simulation results
- Animation of geometric primitives and CAD-Files



**Simulate
with
Animation**

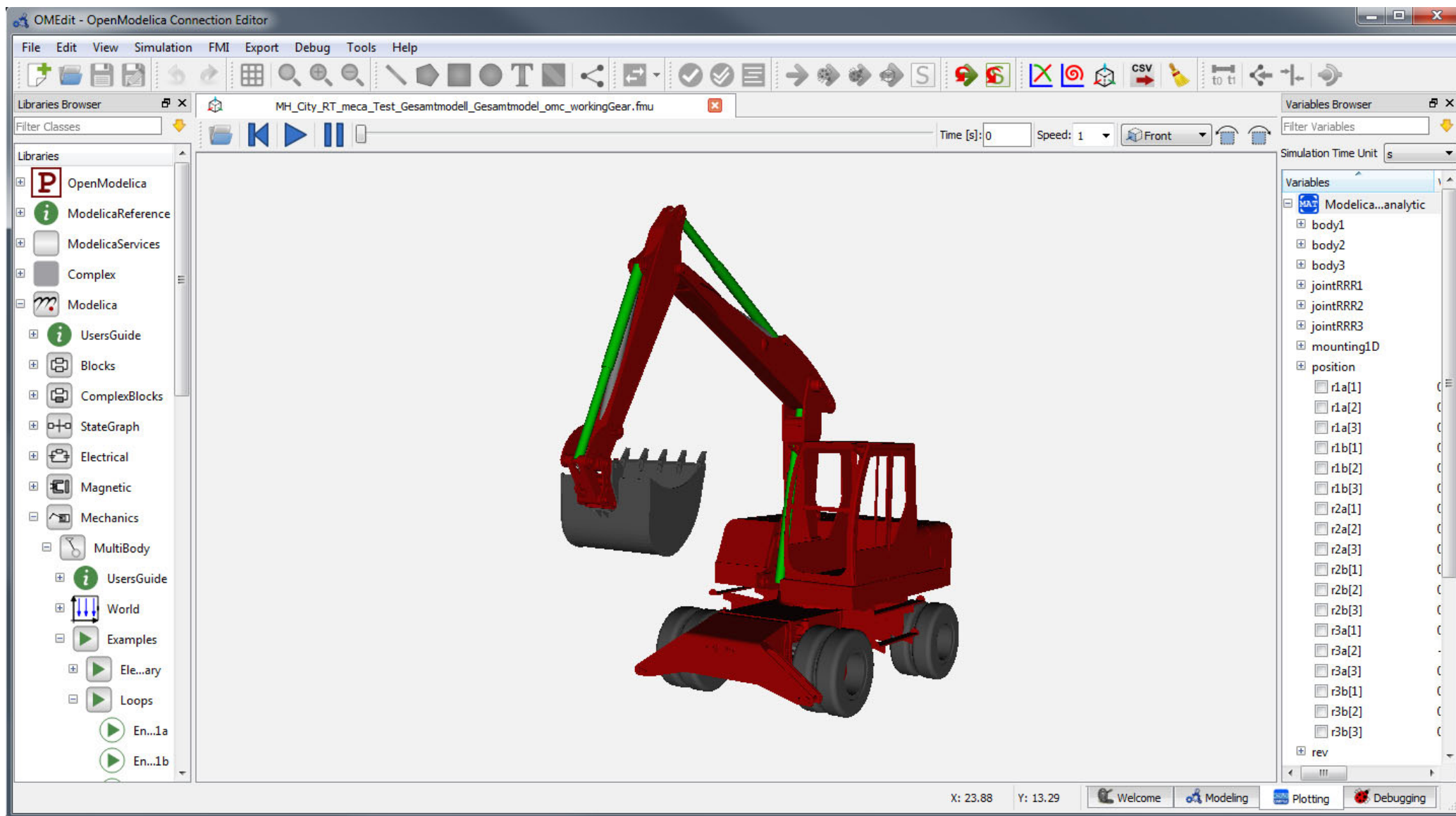


**New
Animation
Window**

OpenModelica 3D Animation Demo

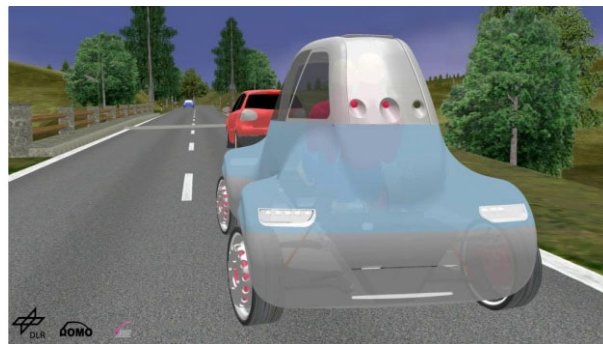
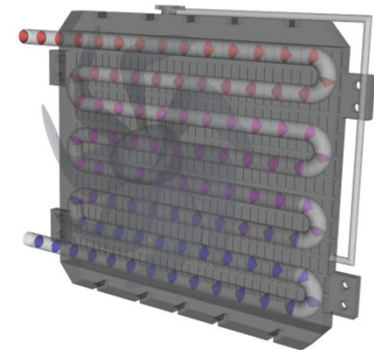
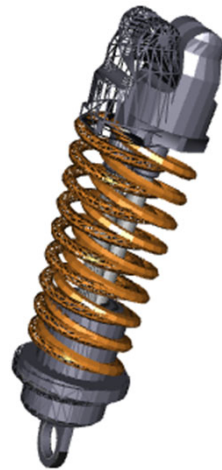
The screenshot shows the OpenModelica Connection Editor (OMEdit) interface. The main workspace displays a 3D model of a mechanical system. The system consists of a horizontal chain of six cylinders, labeled *cylinder1* through *cylinder6*, connected in series. A coordinate system *world* is shown at the top left, with axes *x* and *y*. A torque sensor *tau* is connected to the first cylinder, and a load *J=1* is connected to the last cylinder. A filter block is connected to the torque sensor, with parameters *f=5* and *PTn*. The Libraries Browser on the left lists various components, including *OpenModelica*, *ModelicaReference*, *ModelicaServices*, *Complex*, *Modelica*, *UsersGuide*, *Blocks*, *ComplexBlocks*, *StateGraph*, *Electrical*, *Magnetic*, *Mechanics*, *MultiBody*, *UsersGuide*, *World*, *Examples*, *Elementary*, *Loops*, *EngineLa*, *EngineLb*, *EngineLb_analytic*, *EngineV6*, *EngineV6_analytic*, *Fourbar1*, *Fourbar2*, *Fourbar_analytic*, *PlanarLoops_analytic*, *Utilities*, and *Rotational3DEffects*. The status bar at the bottom shows coordinates (X: 148.21, Y: -54.16) and active tools (Welcome, Modeling, Plotting, Debugging).

OpenModelica 3D Animation Demo – Excavator

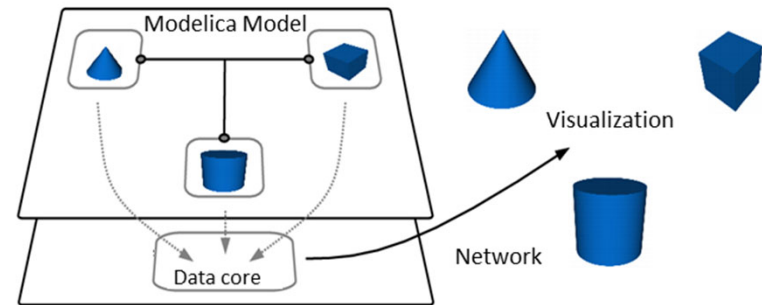


Visualization using Third-Party Libraries: DLR Visualization Library

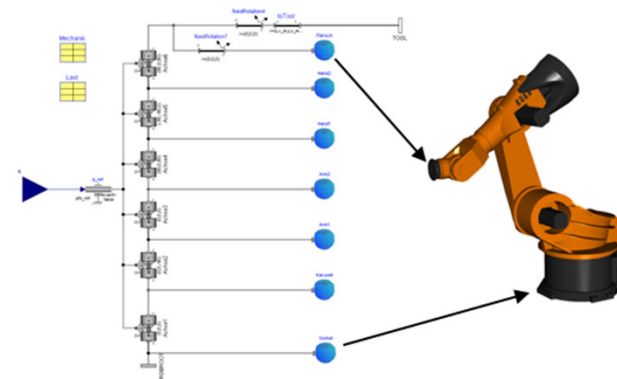
- Advanced, model-integrated and vendor-unspecific visualization tool for Modelica models
- Offline, online and real-time animation
- Video-export function
- Commercial library, feature reduced free Community Edition exists



Integration of visualizer blocks into the model and Communication to an external viewer (SimVis)



Additional C++ Code



Courtesy of Dr. Tobias Bellmann (DLR)

Problems

OMOptim – Parameter Sweep Design Optimization

Solved problems

Result plot

Export result data .csv

The screenshot shows the MinEIT software interface. The 'Problems' list on the left is highlighted in red, and the 'Solved problems' list is highlighted in green. The main window displays a 'Plot' of the Pareto front for the optimization problem. The X-axis is labeled 'gaincouteroperationnel' and the Y-axis is labeled 'coutinvestissement'. The plot shows five red data points representing the Pareto front. Below the plot, there are buttons for 'Blocks', 'Recomp. vars', and 'Plot'. At the bottom, there are checkboxes for 'Calculate all variables from selected points' and 'Force recomputation', and an 'Export...' button.

Point	gaincouteroperationnel	coutinvestissement
0	~5000	~0
1	~50000	~20000
2	~60000	~90000
3	~70000	~160000
4	~75000	~240000

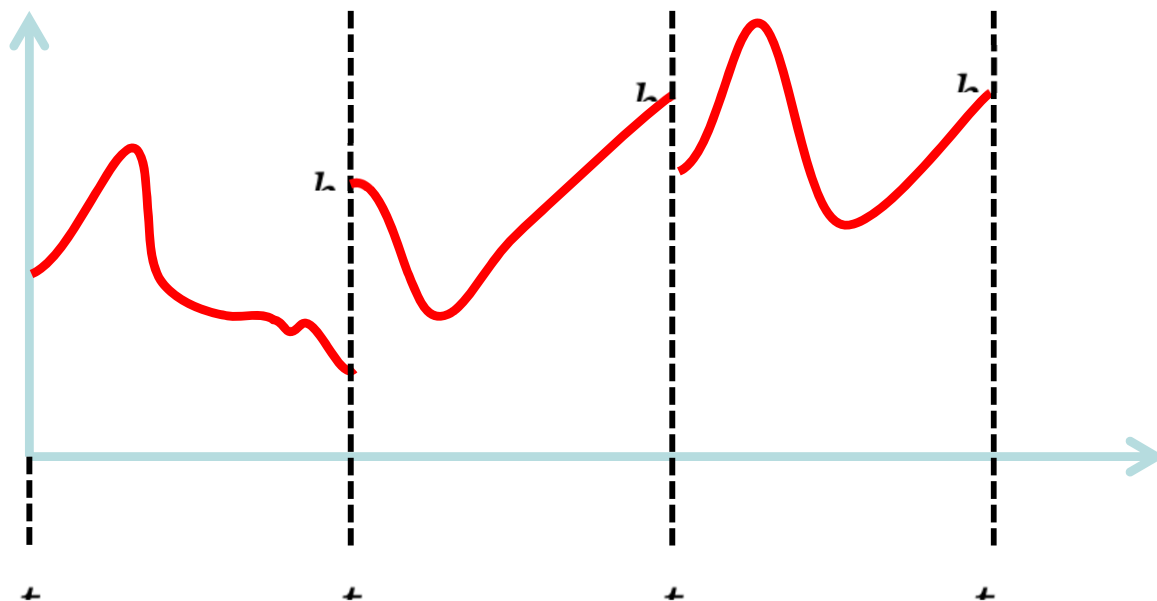
Here Pareto front optimization

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

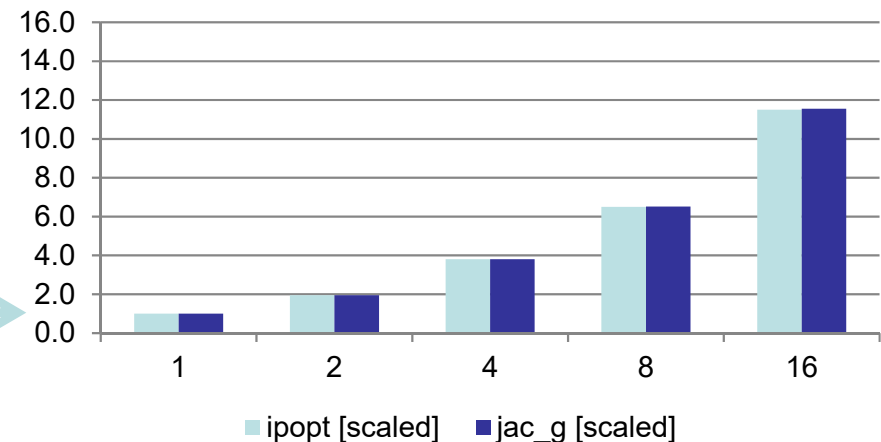
In OpenModelica 1.9.1
beta release Jan 2014.

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

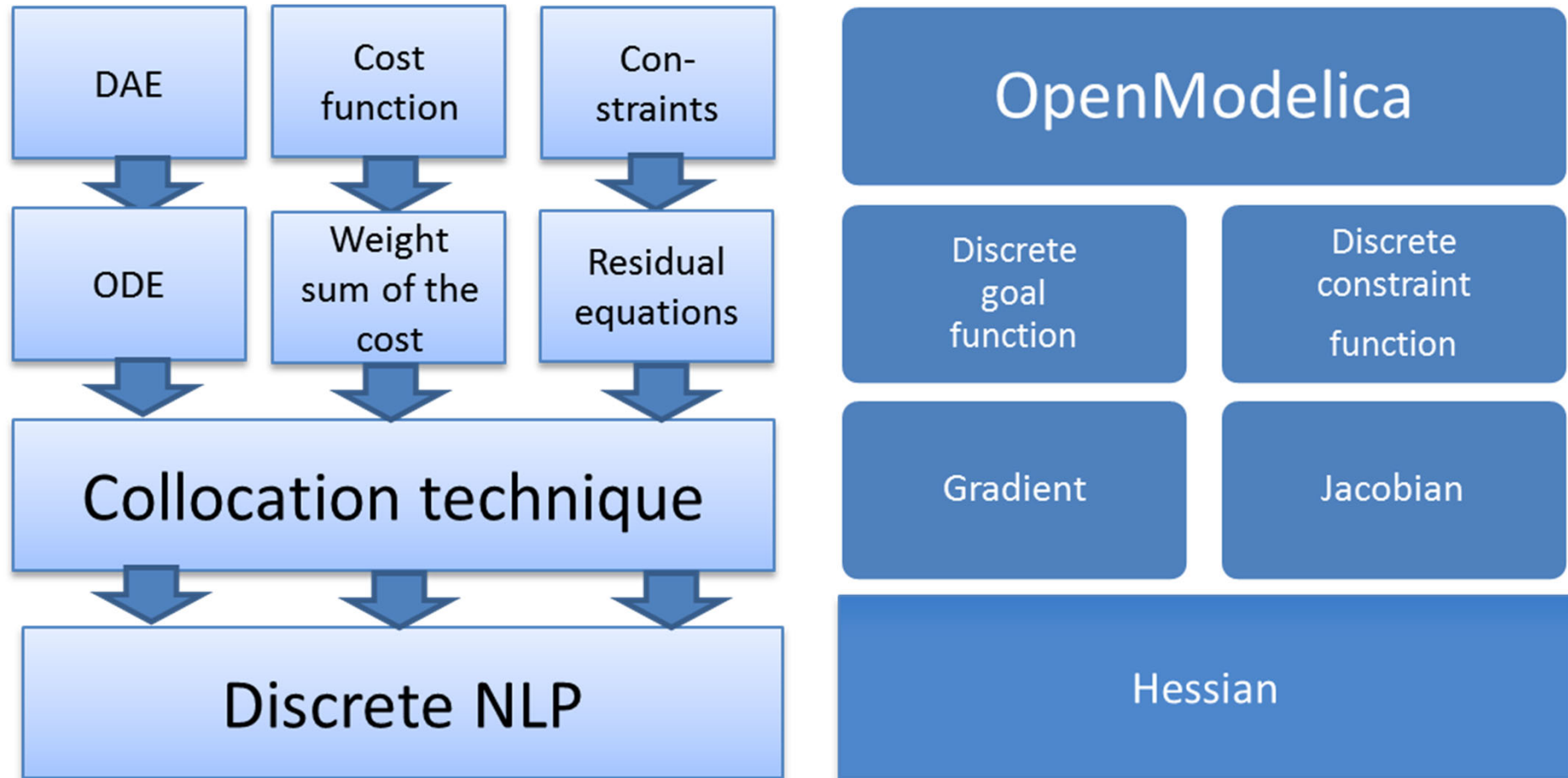


Example speedup, 16 cores:

MULTIPLE_COLLOCATION

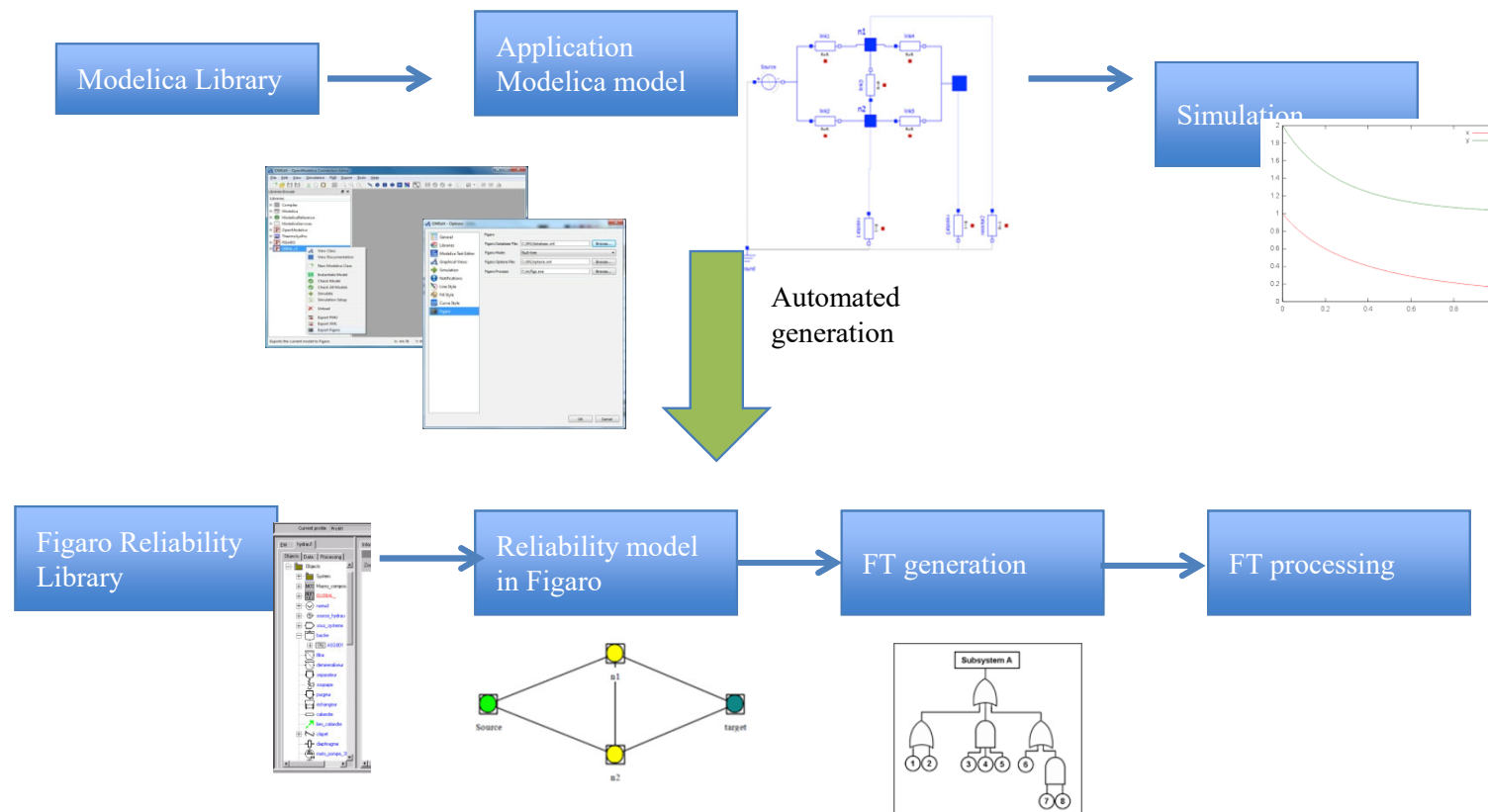


OpenModelica Dynamic Optimization Collocation



Failure Mode and Effects Analysis (FMEA) in OM

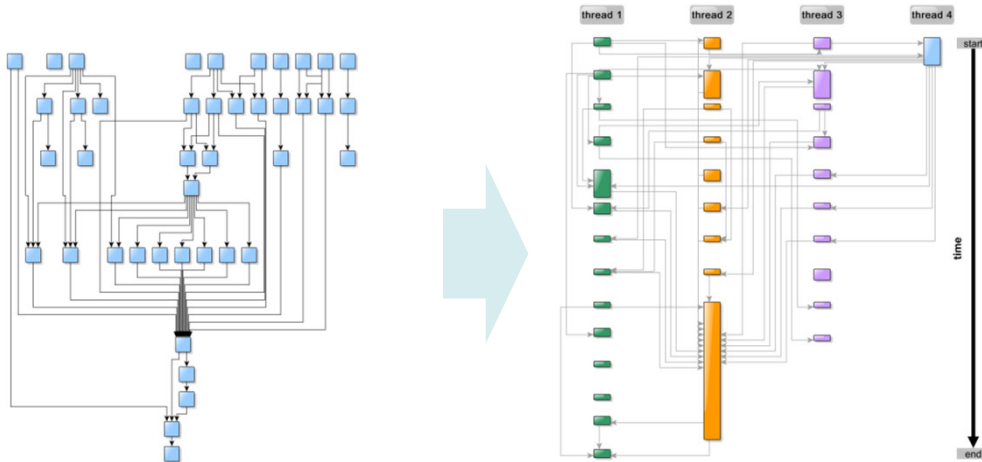
- Modelica models augmented with reliability properties can be used to generate reliability models in Figaro, which in turn can be used for static reliability analysis
- Prototype in OpenModelica integrated with Figaro tool



OpenModelica Model Parallelization

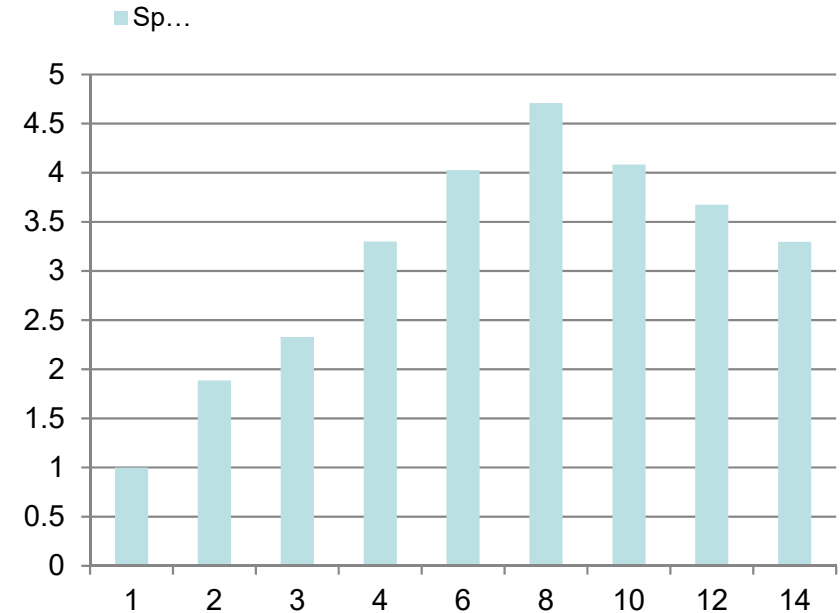
Faster Simulation on Multi-Core

Automated parallelization of models



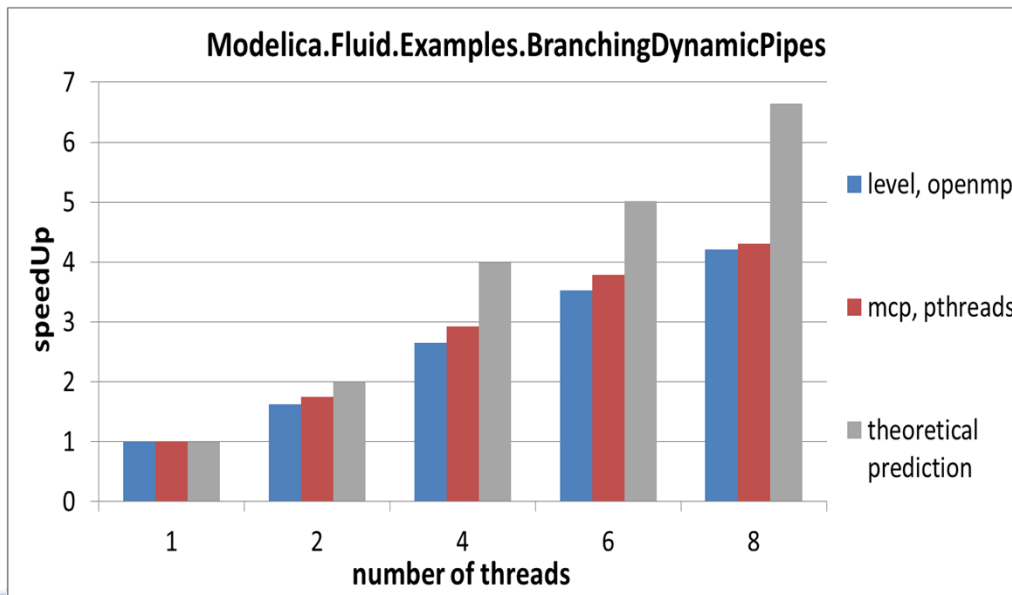
Parallelizing numeric Jacobian computations in simulation

Speedup ScalableTestSuite
N=39 nnz= 818



**Speedup about 4
using 8 threads**

number of
threads



Recent Large-scale ABB OpenModelica Application

Generate code for controlling 7.5 to 10% of German Power Production

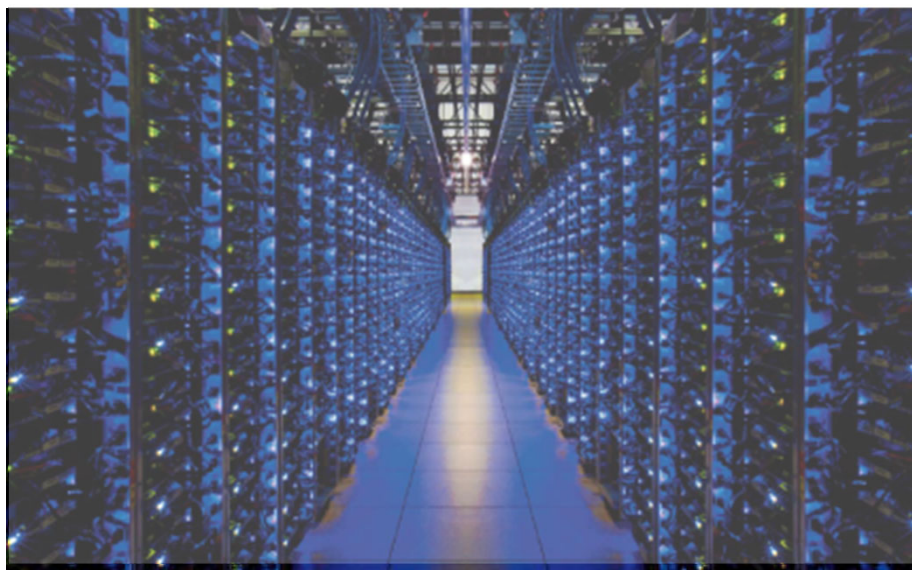


ABB OPTIMAX PowerFit

- 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**

Part III

Equation-Based Model Dynamic Debugging and Performance Analysis

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
...

OpenModelica Equation Model Debugger

The screenshot displays the OpenModelica Equation Model Debugger interface. It is divided into three main sections:

- Variables View:** Shows a tree structure of variables (frame, boxBody1, body, frame_a, R, T) and their comments. It also includes a 'Defined In Equations' and 'Used In Equations' table.
- Equations View:** Contains an 'Equations Browser' table listing equations by index, type, and equation text. Below it are 'Defines' and 'Depends' tables, and 'Equation Operations' showing the solver's steps.
- Source View:** Displays the source code of the model, with line numbers 317-331 visible.

Example of equation transformations on a model:

$$0 = y + \text{der}(x * \text{time} * z); z = 1.0;$$

(1) substitution:

$$y + \text{der}(x * (\text{time} * z)) \\ \Rightarrow \\ y + \text{der}(x * (\text{time} * 1.0))$$

(2) simplify:

$$y + \text{der}(x * (\text{time} * 1.0)) \\ \Rightarrow \\ y + \text{der}(x * \text{time})$$

(3) expand derivative (symbolic diff):

$$y + \text{der}(x * \text{time}) \\ \Rightarrow y + (x + \text{der}(x) * \text{time})$$

(4) solve:

$$0.0 = y + (x + \text{der}(x) * \text{time}) \\ \Rightarrow \\ \text{der}(x) = ((-y) - x) / \text{time} \\ \text{time} \ll 0$$

Integrated Static-Dynamic OpenModelica Equation Model Debugger

Efficient
handling
of
Large
Equation
Systems

Showing
equation
transformations
of a
model:

The screenshot displays the OMEdit - Transformational Debugger interface, which is divided into three main panels: Variables View, Equations View, and Source View.

- Variables View:** Located at the top left, it shows a tree structure of variables under 'boxBody1'. It includes a 'Variables Browser' with search options (Case Sensitive, Regular Expression, Expand All, Collapse All) and a table of variables with their comments. Below this is a 'Variable Operations' section showing operations like 'solved' and 'substitute'.
- Equations View:** Located at the bottom left, it shows a table of equations with their indices, types, and the equations themselves. Below the table is an 'Equation Operations' section showing operations like 'solve', 'scalarize', 'simplify', 'inline', and 'substitute'.
- Source View:** Located on the right, it shows the source code of the model. The code is color-coded and includes comments. A black arrow points from the 'Equations View' to the corresponding line in the 'Source View'.

The interface is titled 'OMEdit - Transformational Debugger' and shows the file path 'OpenModelica/OMEdit/Modelica.Mechanics.MultiBody.Examples.Elementary.DoublePendulum_info.xml'.

Mapping dynamic run-time error to source model position

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

The screenshot displays the OMEdit - Transformational Debugger interface. The main window is titled "OMEdit - Transformational Debugger" and shows the file path: `/tmp/OpenModelica_marsj/OMEdit/Modelica.Mechanics.MultiBody.Examples.Loops.EngineV6_info.xml`. The interface is divided into several panes:

- Variables Browser:** Shows a list of variables with columns for "Variables", "Comment", "Line", and "Location". The variable `phi` is highlighted.
- Defined In Equations:** A table showing where the variable `phi` is defined. It has two entries: index 587 (initial, nonlinear) and index 5016 (regular, nonlinear).
- Used In Equations:** A table showing where the variable `phi` is used. It lists several regular (assignment) equations involving `der(cylinder3.B2.phi)` and `der(cylinder3.Rod.body.w_a[1])`.
- Equations Browser:** Shows a list of equations with columns for "Inc", "Type", and "Equation". The equation `der(cylinder3.B2.R_rel.T[3,3])` is highlighted.
- Source Browser:** Shows the source code for the model, with lines 306-335 visible. The code includes assertions for joint connectivity, angle calculations, and frame transformations.

The "Equations Browser" pane shows the following equation operations:

- Variable: `der(cylinder3.B2.R_rel.T[3,3])`
- Equation Operations:
 - `- solved: der(cylinder3.B2.R_rel.T[3,3]) = (-sin(cylinder3.B2.phi)) * cylinder3.Rod.body.w_a[1]`
 - `- substitute: (-sin(cylinder3.B2.phi)) * cylinder3.B2.w => (-sin(cylinder3.B2.phi)) * cylinder3.Rod.body.w_a[1]`
 - `- differentiate: dcos(cylinder3.B2.phi)/dtime = (-sin(cylinder3.B2.phi)) * der(cylinder3.B2.phi)`
 - `- differentiate: dcylinder3.B2.R_rel.T[3,3]/dtime = der(cylinder3.B2.R_rel.T[3,3])`
 - `- scalarize(9): cylinder3.B2.R_rel.T = {{1.0, 0.0, 0.0}, [-0.0, c...B2.phi]] => cylinder3.B2.R_rel.T[3,3] = cos(cylinder3.B2.phi)`
 - `- simplify: cylinder3.B2.R_rel.T = {{1.0 * 1.0 + (1.0 - 1.0 * 1.0)...B2.phi}}, {0.0, -sin(cylinder3.B2.phi), cos(cylinder3.B2.phi)}`
 - `- substitute: {{cylinder3.B2.e[1] * cylinder3.B2.e[1] + (1.0 - cy...2.phi)}, 0.0 * 0.0 + (1.0 - 0.0 * 0.0) * cos(cylinder3.B2.phi)}`
 - `- inline: cylinder3.B2.R_rel = Modelica.Mechanics.MultiBody....[2] * cylinder3.B2.w, cylinder3.B2.e[3] * cylinder3.B2.w`
 - `- original: R_rel = Frames.planarRotation(e, phi_offset + phi, w); => flattened:`

Performance Profiling

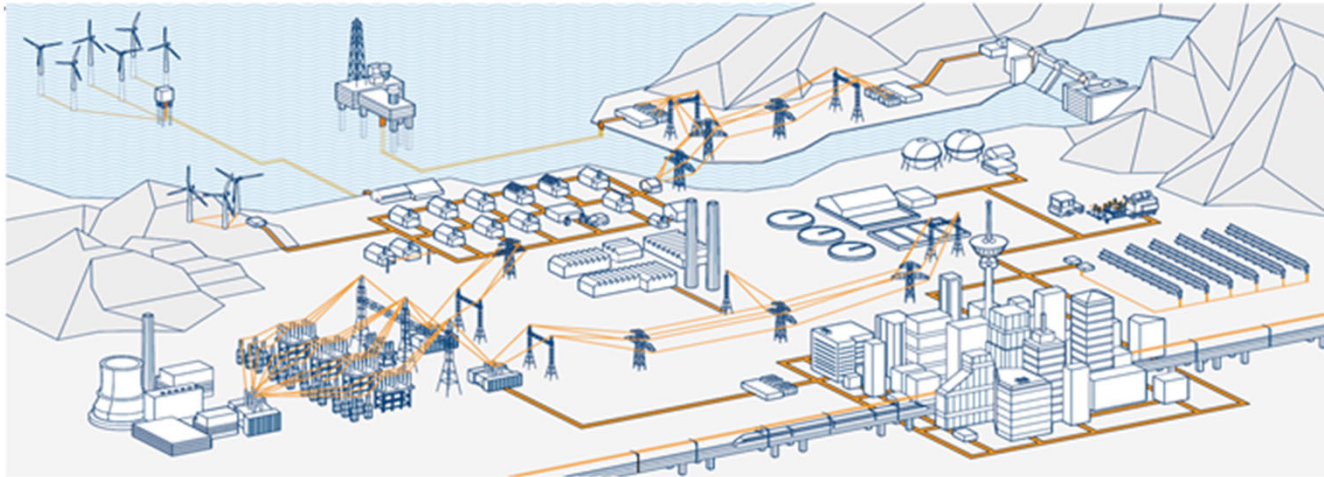
(Here: Profiling all equations in MSL 3.2.1 DoublePendulum)

- ▶ 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

Equations Browser							Defines
Index	Type	Equation	Executi	Max time	Time	Fraction	Variable
+ 876	regular	linear, size 2	4602	0.000501	0.0134	75.7%	damper.a_rel revolute2.frame_b.f[2]
- 836	regular	(assignment) ...evolute2.phi)	1534	2.57e-05	0.000377	2.12%	
- 840	regular	(assignment) ...mper.phi_rel)	1534	1.38e-05	0.000237	1.33%	
- 837	regular	(assignment) ...evolute2.phi)	1534	8.38e-06	0.000235	1.32%	
- 841	regular	(assignment) ...mper.phi_rel)	1534	8.48e-06	0.000192	1.08%	
- 849	regular	(assignment) ...mper.phi_rel)	1534	8.04e-06	0.000146	0.824%	

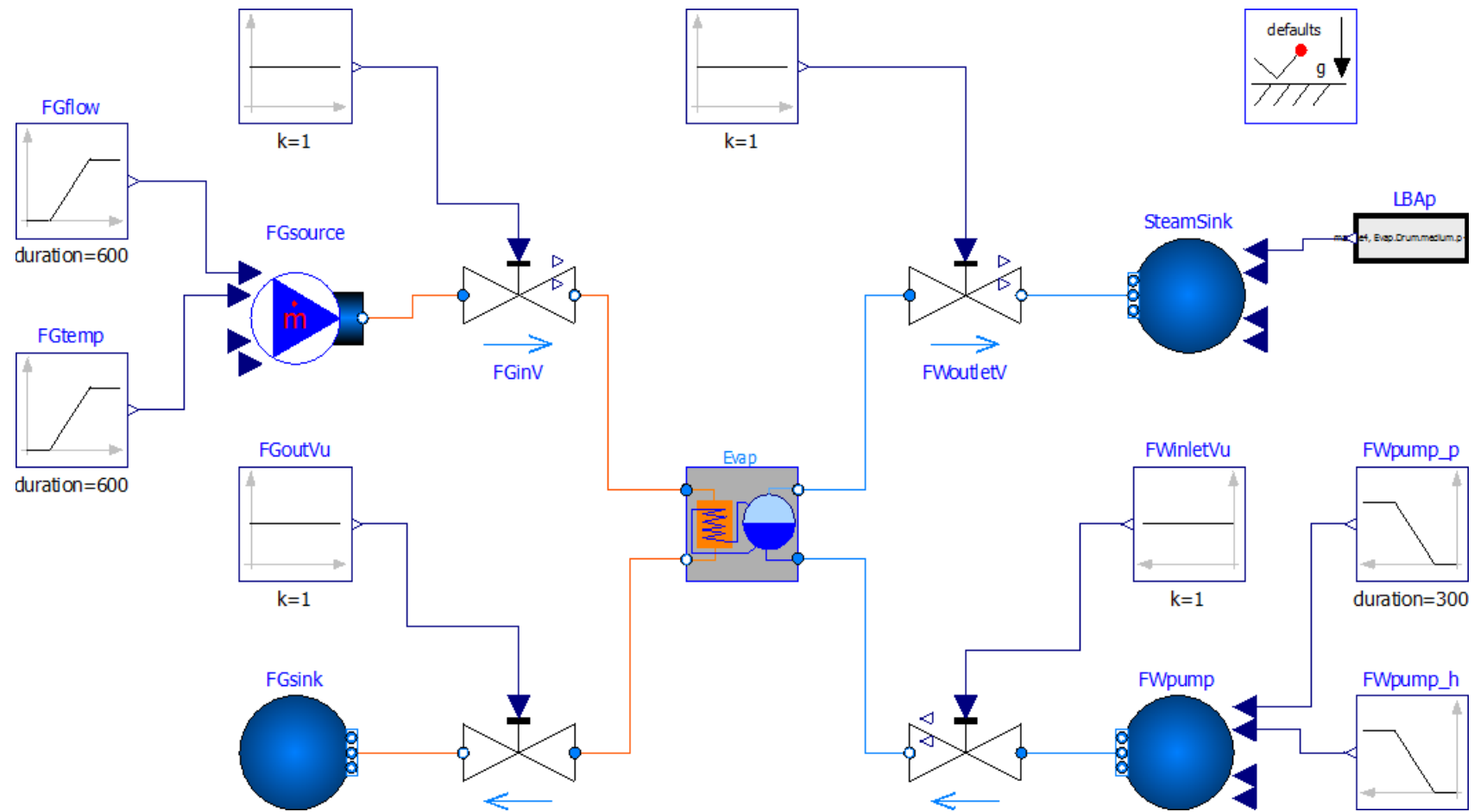
ABB Commercial Application Use of Debugger

- 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)

The screenshot displays the OMEdit - Transformational Debugger interface. The main window is divided into several panes:

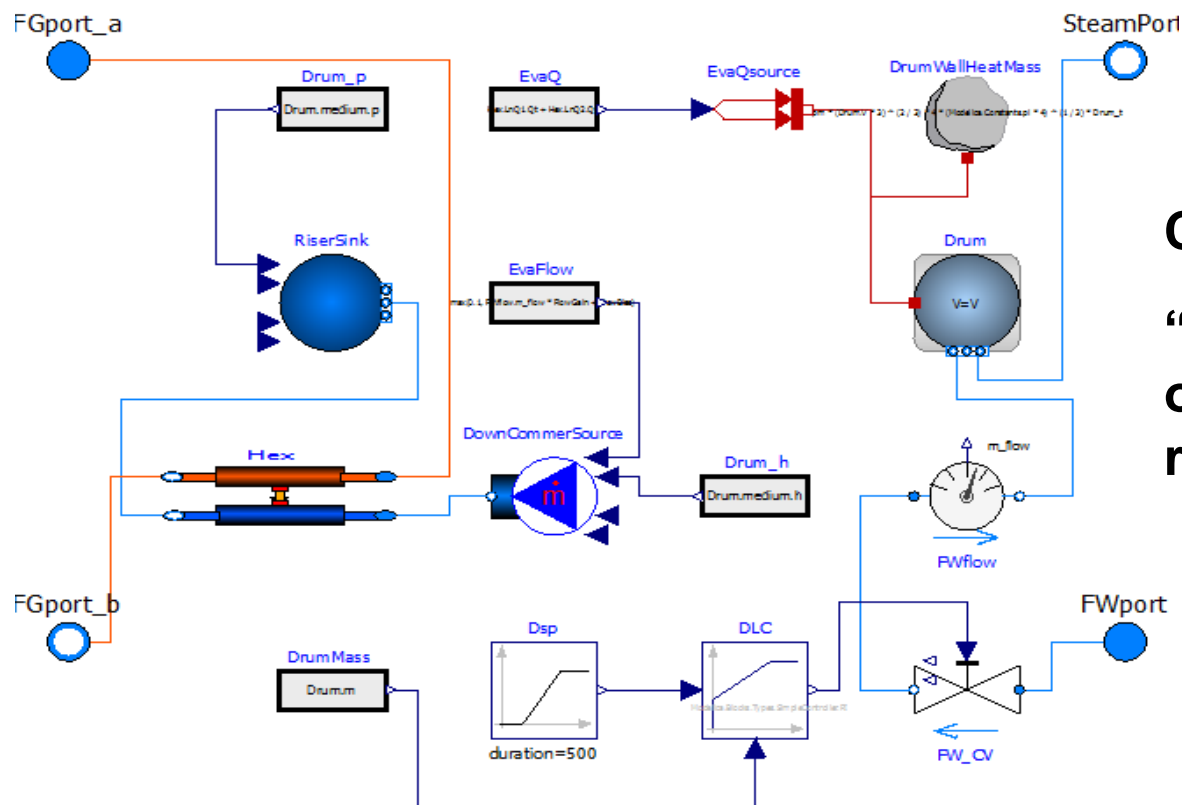
- Variables Browser:** Lists variables such as Scse1, Scse2, Scse3, Scse4, Scse5, Scse6, Evap, Fgflow, FginV, FginVu, FgoutVu, FgoutlVu, Fgsink, Fgsource, Fgtemp, FwinletV, FwinletVu, FwoutletV, Fwpump, Fwpump_h, Fwpump_p, SteamSink, and system.
- Equations Browser:** Lists equations with indices from 1461 to 1488, including regular equations like Evap.Hex.LnC.us[1], Evap.Hex.LnC.flowModel.mus[1], and Evap.Hex.LnC.H.flows[2].
- Defined In Equations:** Shows the source location for each equation, such as C:/OpenMod...package.mo.
- Used In Equations:** Shows the equations that use the variables from the left pane.
- Source Browser:** Displays the source code for the selected equation, showing a block division operation: `y = u1/u2;` with an annotation for documentation.

A callout box with a black border and white background points to the equation `y = u1/u2;` in the source code. The text inside the callout box reads: "Pointing out the buggy equation $y = u1/u2;$ that gives division by zero".

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.”

Part IV

Dynamic Verification/Testing of Requirements vs Usage Scenario Models

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

Van der Pol Model

This example describes a Van der Pol oscillator. Notice that here the keyword `model` is used instead of `class` with the same meaning. This example contains declarations of two state variables `x` and `y`, both of type `Real` and a parameter `lambda`, which is a so-called simulation parameter. The keyword `parameter` specifies that the variable is constant during a simulation run, but can have its value initialized before a run, or between runs. Finally, there is an equation section starting with the keyword `equation`, containing two mutually dependent equations that define the dynamics of the model.

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

1 Simulation of Van der Pol

To illustrate the behavior of the model, we give a command to simulate the Van der Pol oscillator during 25 seconds starting at time 0:

```
simulate(VanDerPol, startime=0, stopTime=25);
```

Some

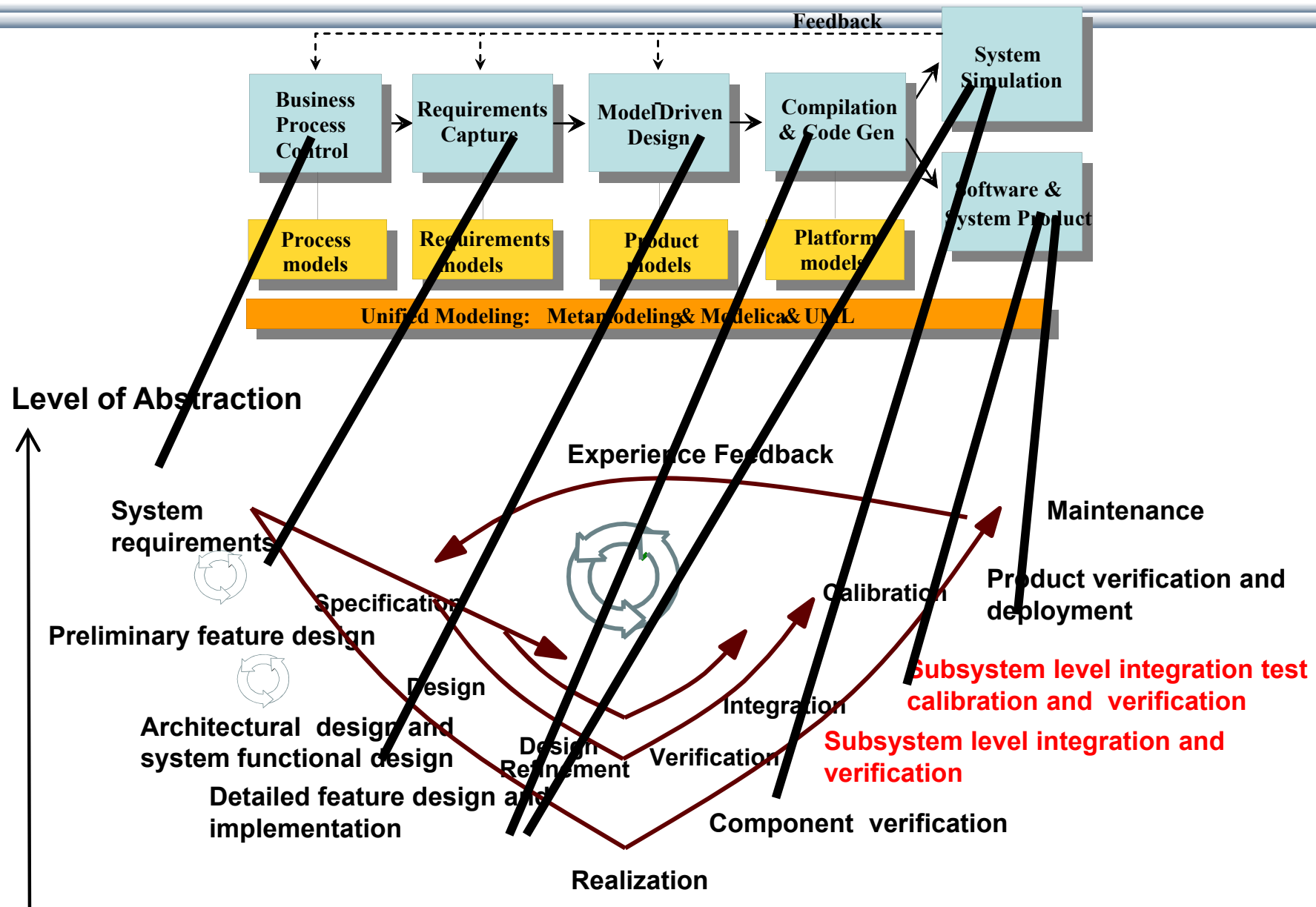
Performs a parametric plot

```
plotparametric(x, y);
```

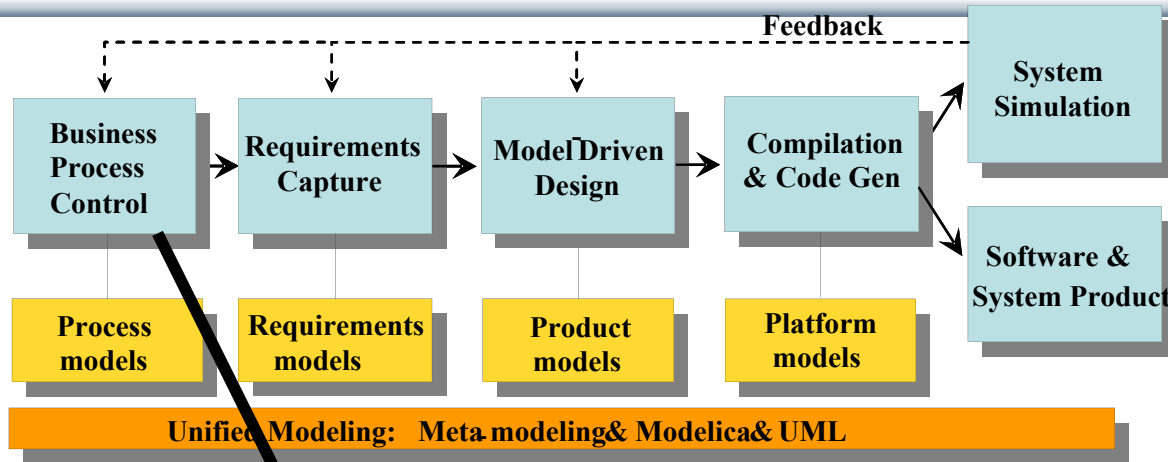
Plot by OpenModelica

Time	x	y
0	1.0	1.0
1	1.0	1.0
2	1.0	1.0
3	1.0	1.0
4	1.0	1.0
5	1.0	1.0
6	1.0	1.0
7	1.0	1.0
8	1.0	1.0
9	1.0	1.0
10	1.0	1.0
11	1.0	1.0
12	1.0	1.0
13	1.0	1.0
14	1.0	1.0
15	1.0	1.0
16	1.0	1.0
17	1.0	1.0
18	1.0	1.0
19	1.0	1.0
20	1.0	1.0
21	1.0	1.0
22	1.0	1.0
23	1.0	1.0
24	1.0	1.0
25	1.0	1.0

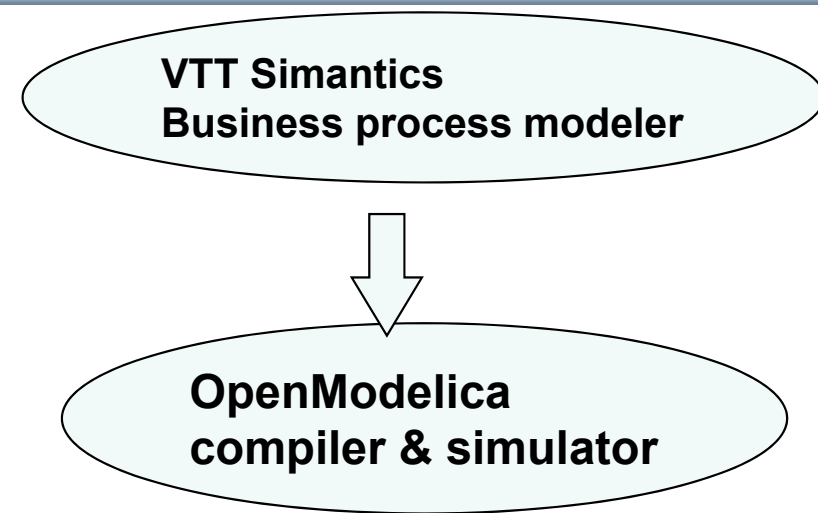
OpenModelica and Papyrus Based Model-Based Development Environment to Cover Product-Design V



Business Process Control and Modeling

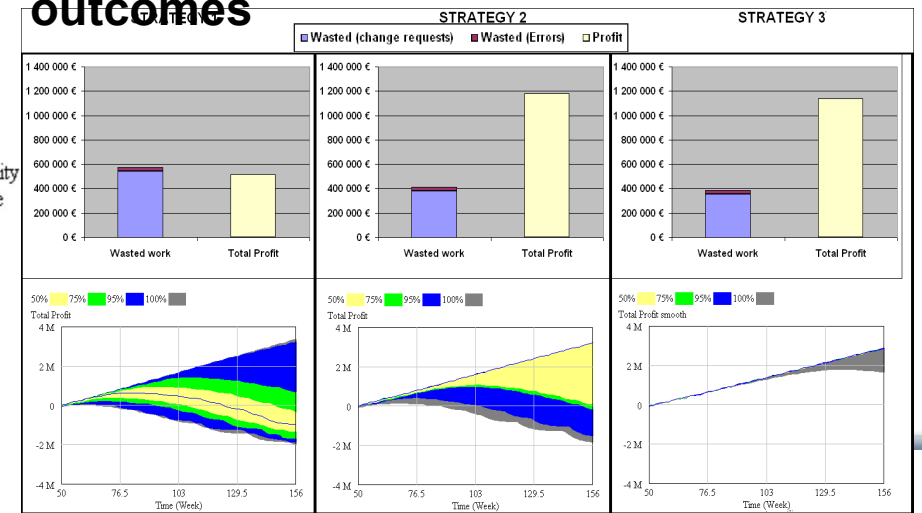
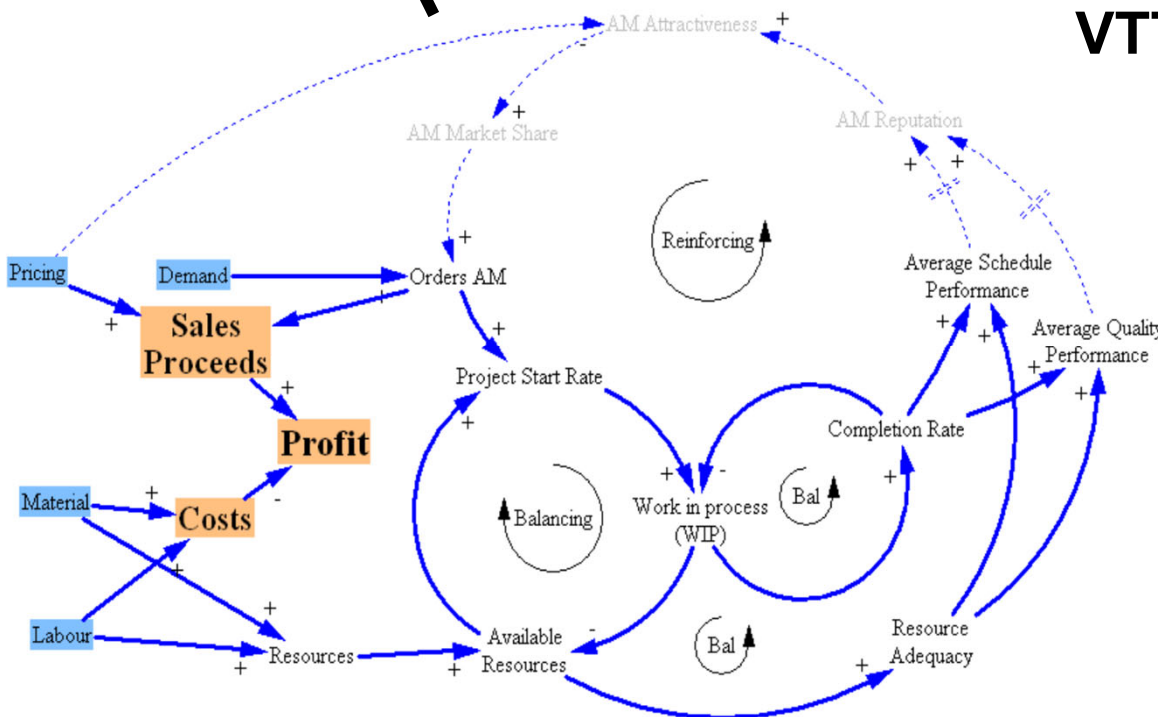


OpenModelica based simulation

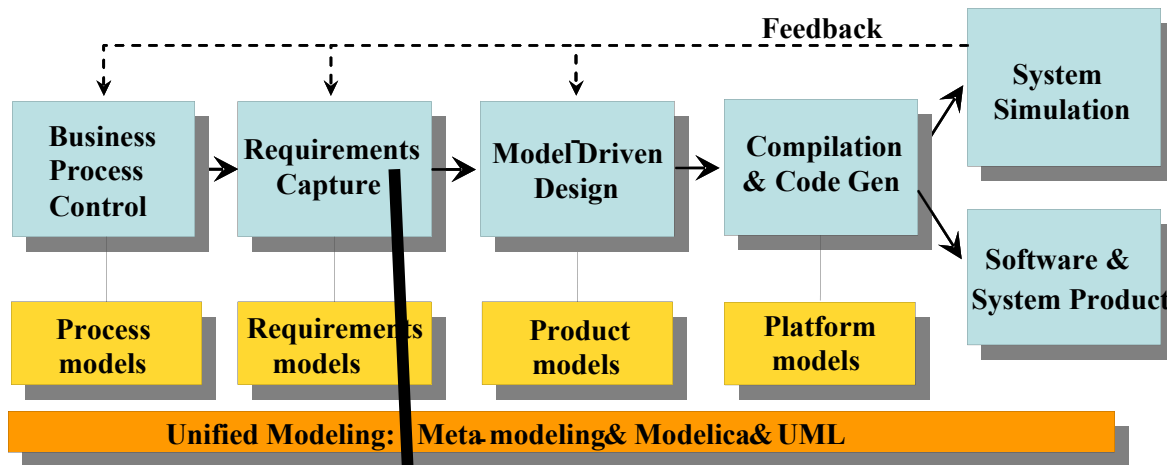


Metso Business model & simulation VTT Simantics Graphic Modeling Tool

Simulation of 3 strategies with outcomes



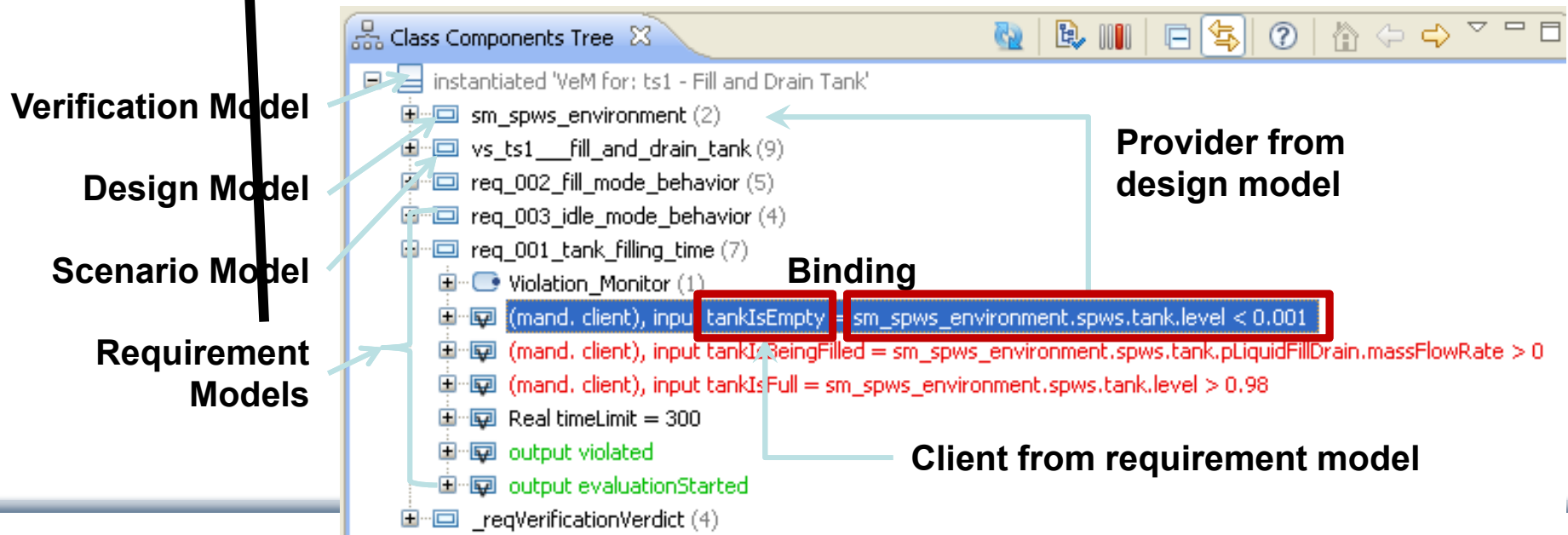
Requirement Capture



vVDR (virtual Verification of Designs against Requirements)

in ModelicaML UML/Modelica Profile, part of OpenModelica

OpenModelica based simulation

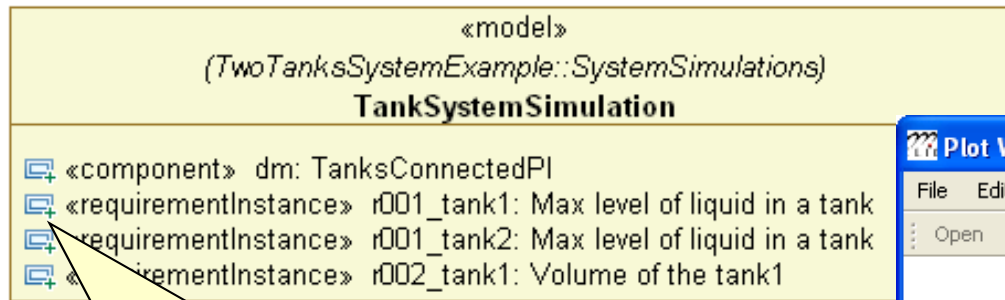


OpenModelica – ModelicaML UML Profile

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

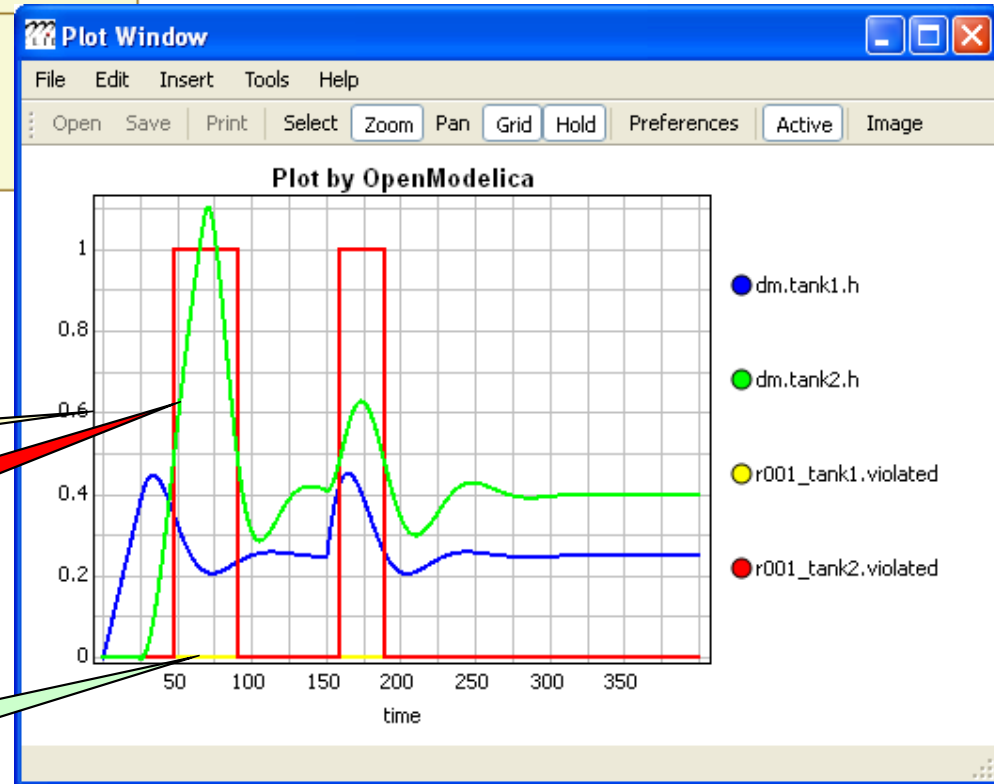


**Req. 001 is instantiated 2 times
(there are 2 tanks in the system)**

tank-height is 0.6m

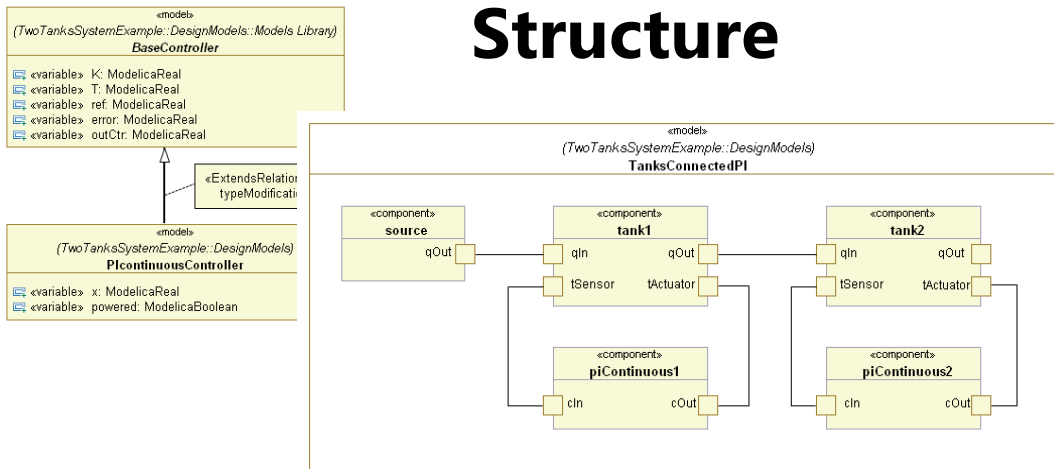
Req. 001 for the tank2 is violated

Req. 001 for the tank1 is not violated

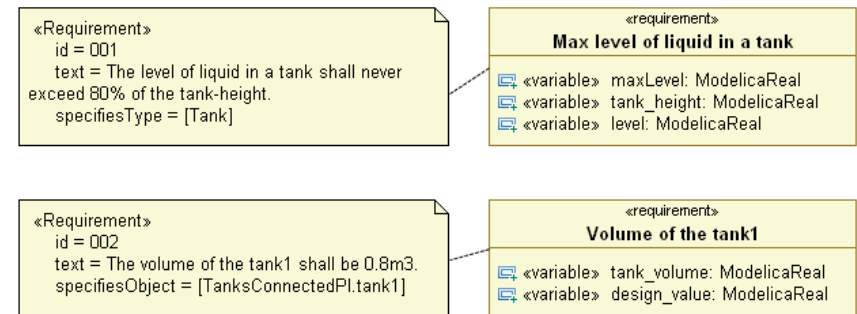


ModelicaML: Graphical Notation

Structure

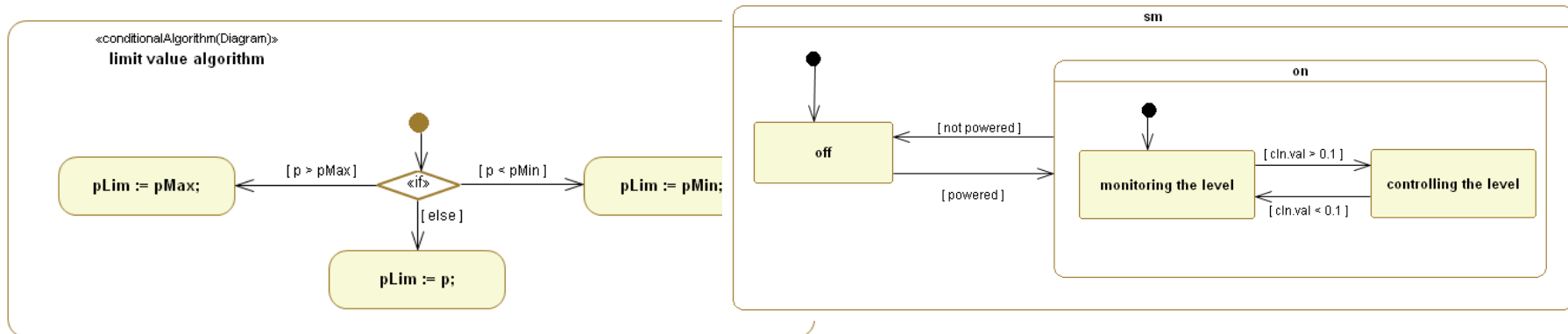


Requirements

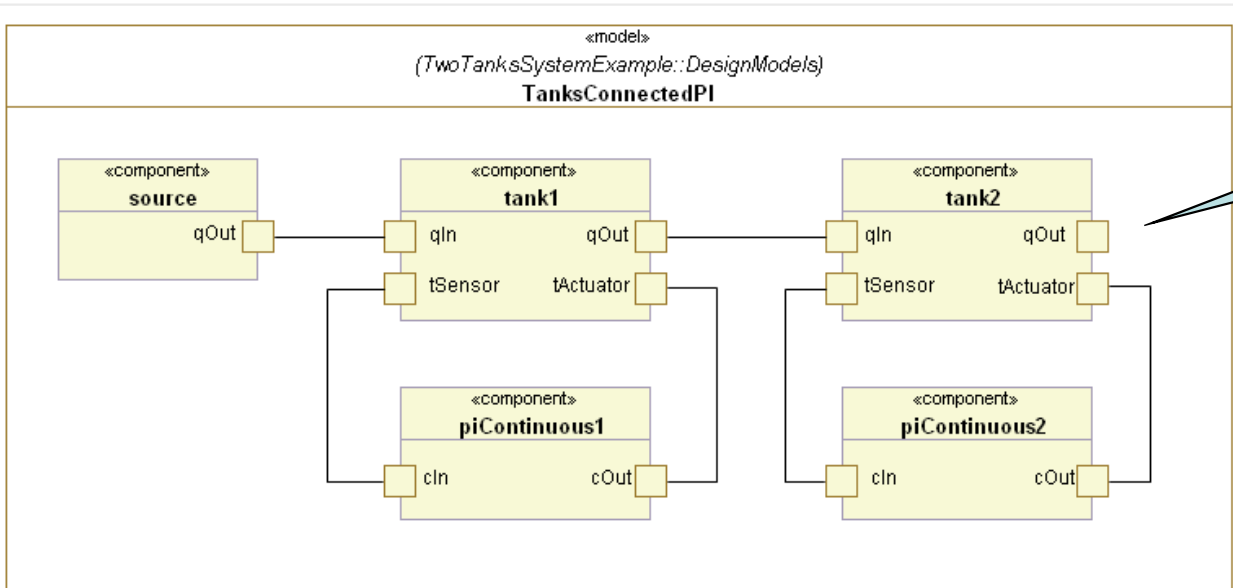


a

Behavior

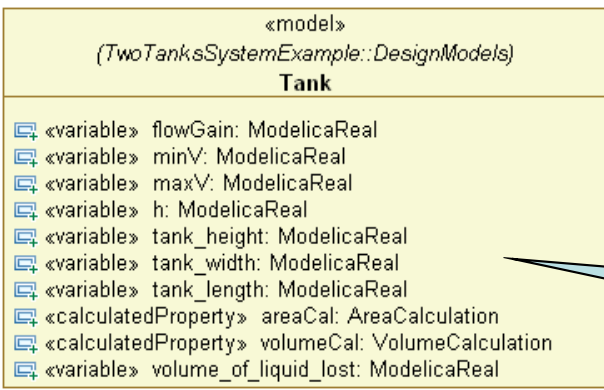


Example: Representation of System Structure



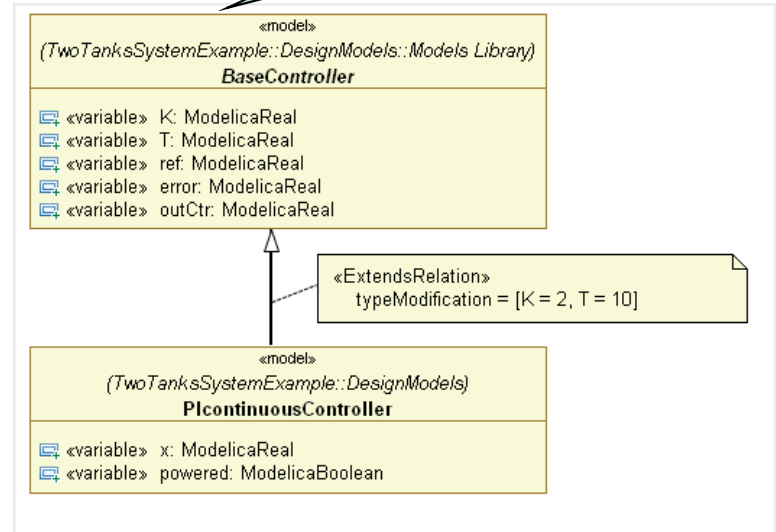
Interconnections

Inheritance

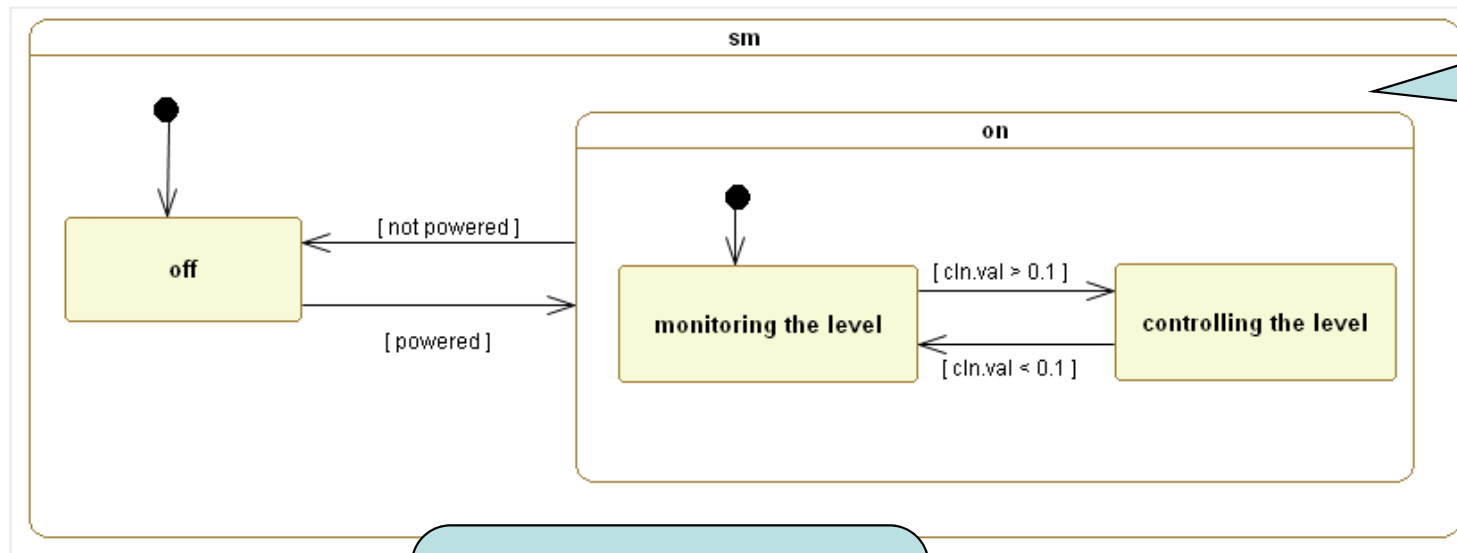


«Assert»
 condition = `minV >= 0`
 message = `minV - minimum Valve level must be >= 0`
 level = warning

Components



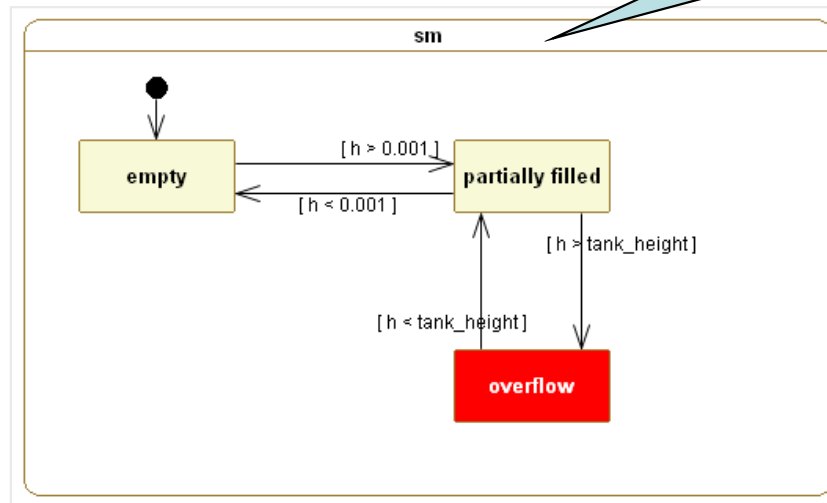
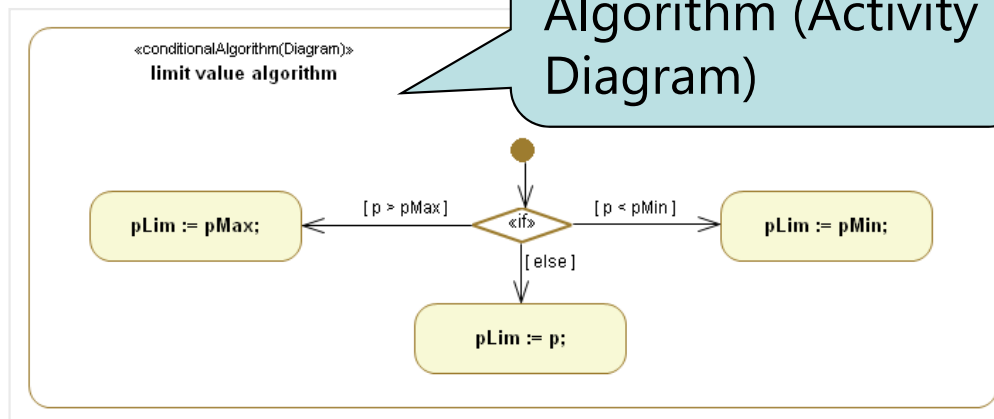
Example: Representation of System Behavior



State Machine of the Controller

State Machine of the Tank

Conditional Algorithm (Activity Diagram)



Example: Representation of System Requirements

Textual Requirement

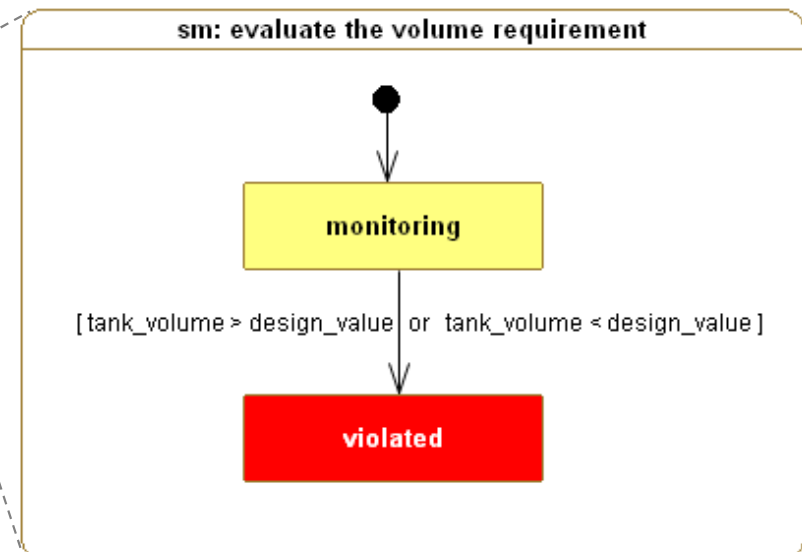
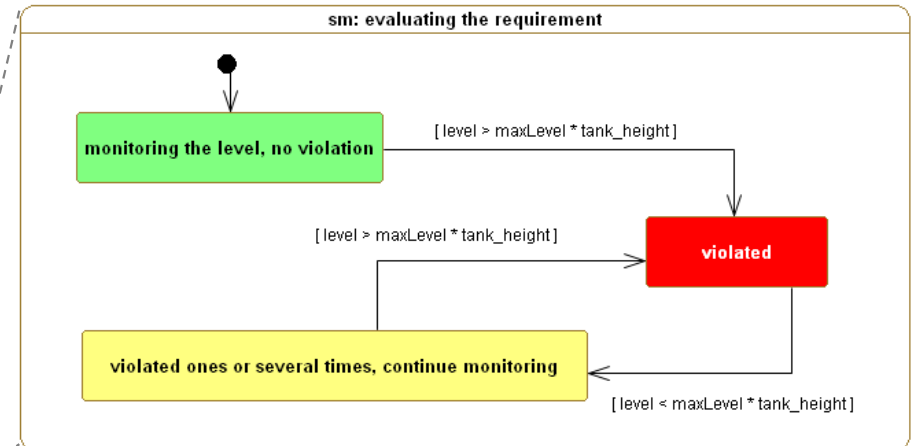
«Requirement»
id = 001
text = The level of liquid in a tank shall never exceed 80% of the tank-height.
specifiesType = [Tank]

«Requirement»
id = 002
text = The volume of the tank1 shall be 0.8m3.
specifiesObject = [TanksConnectedPl.tank1]

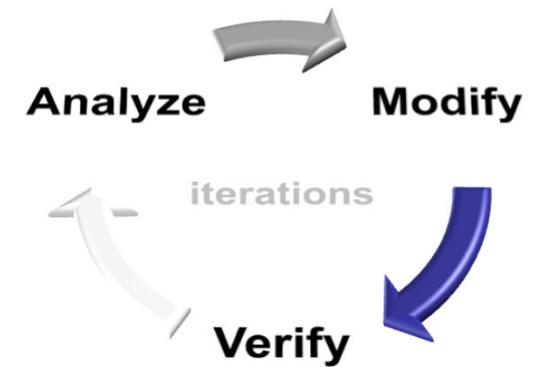
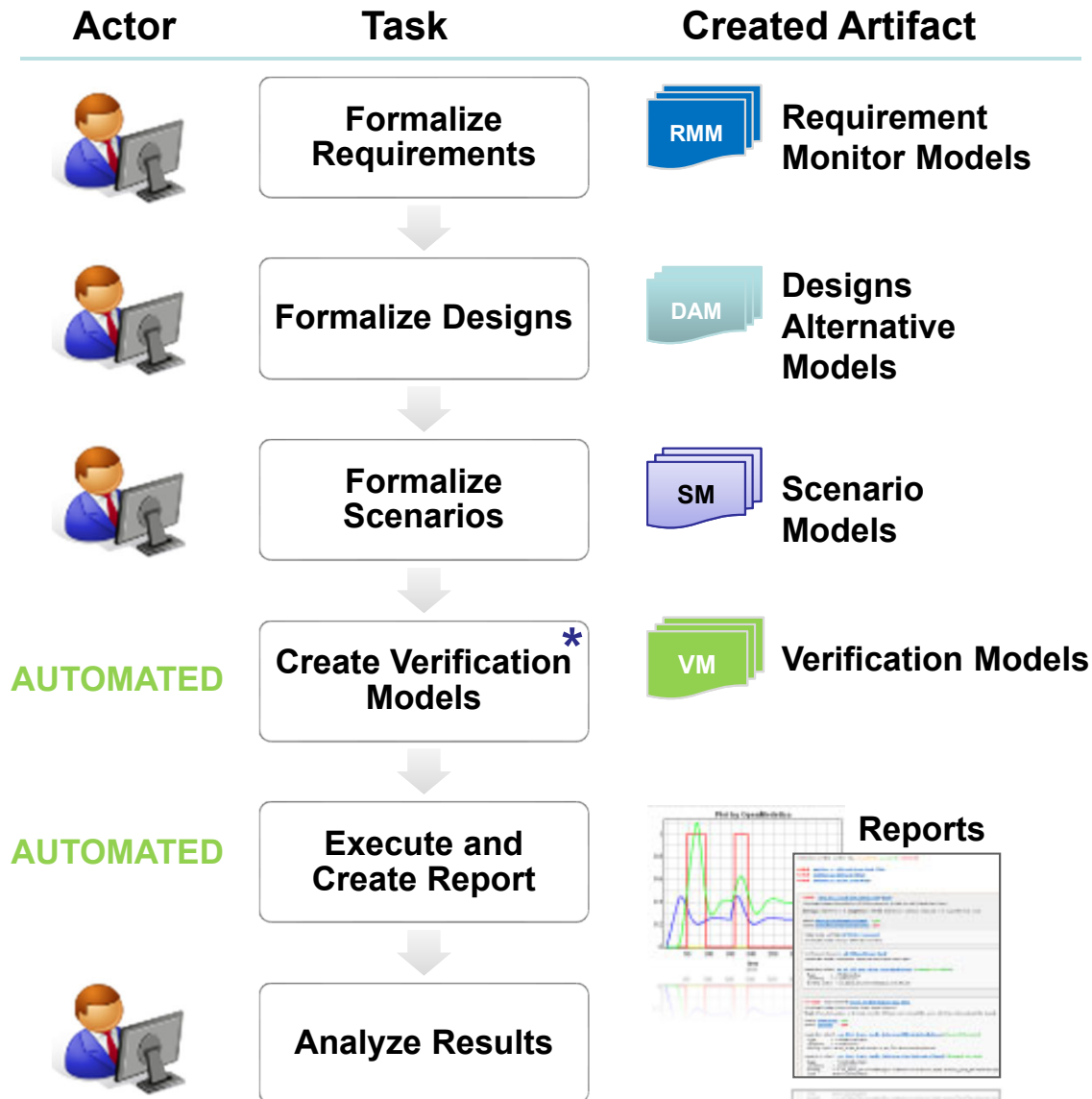
Formalized Requirement

«requirement»
Max level of liquid in a tank
«variable» maxLevel: ModelicaReal
«variable» tank_height: ModelicaReal
«variable» level: ModelicaReal

«requirement»
Volume of the tank1
«variable» tank_volume: ModelicaReal
«variable» design_value: ModelicaReal



vVDR Method – virtual Verification of Designs vs Requirements

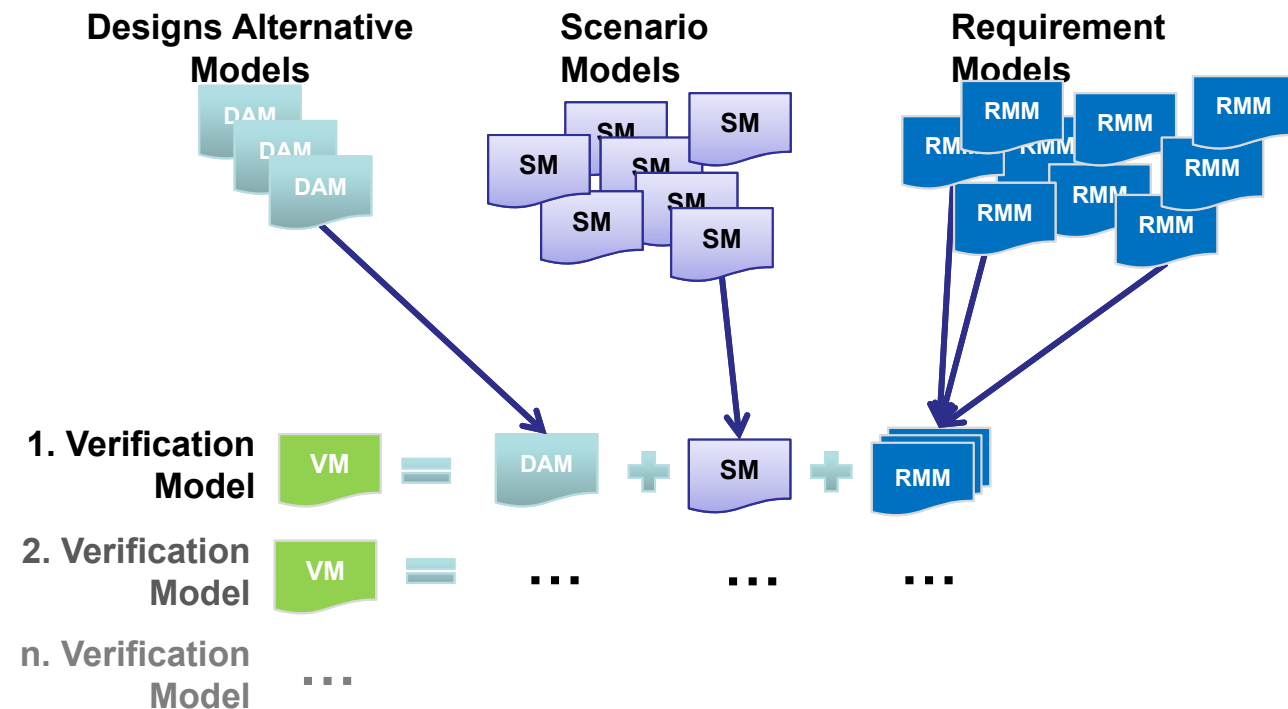
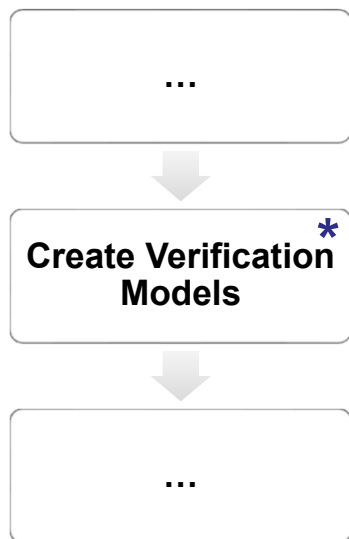


Goal: Enable on-demand verification of designs against requirements using automated model composition at any time during development.

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**?

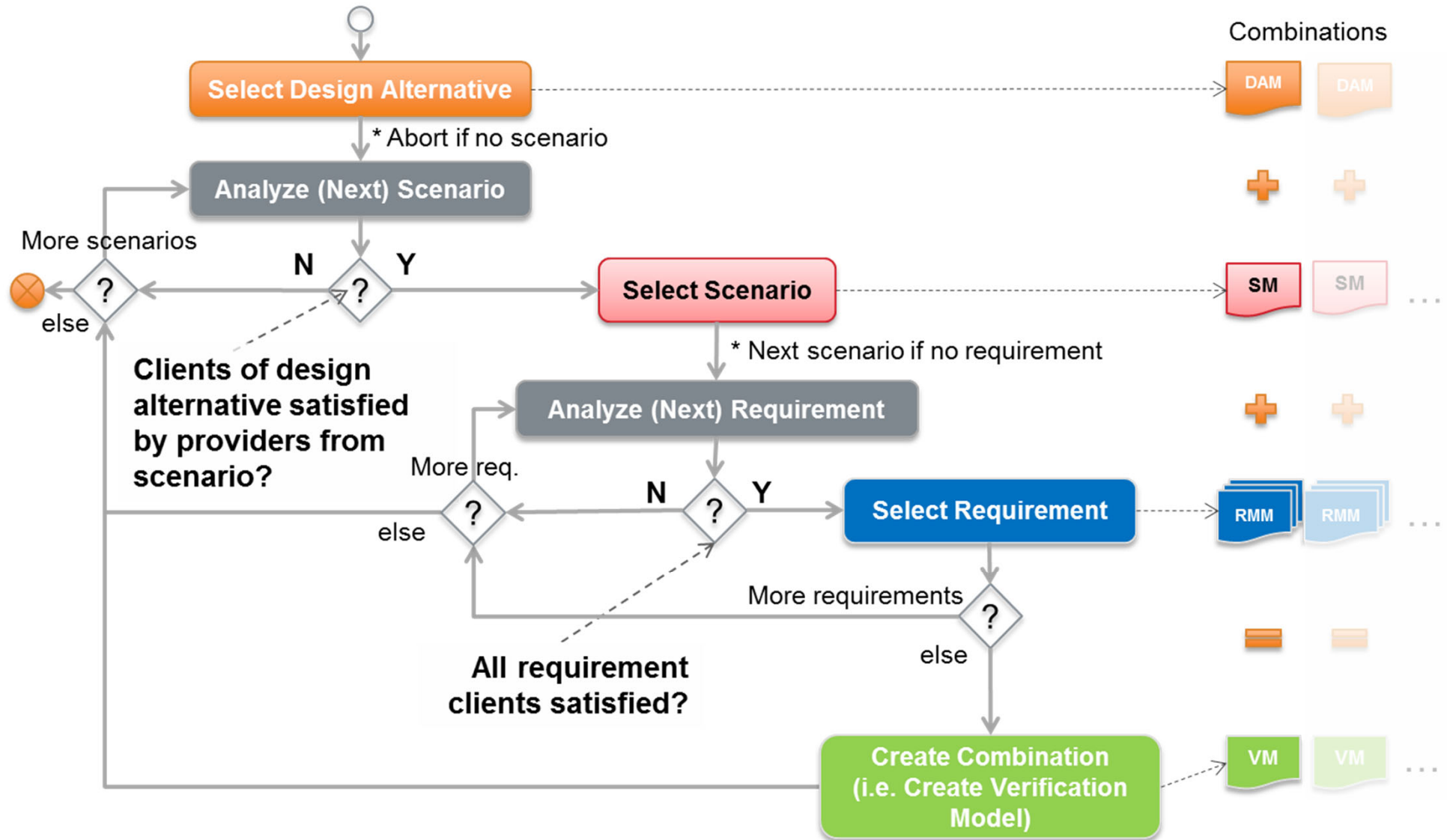


Composing Verification Models

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 of scenarios or requirements he/she wishes to consider

Generating/Composing Verification Models algorithm

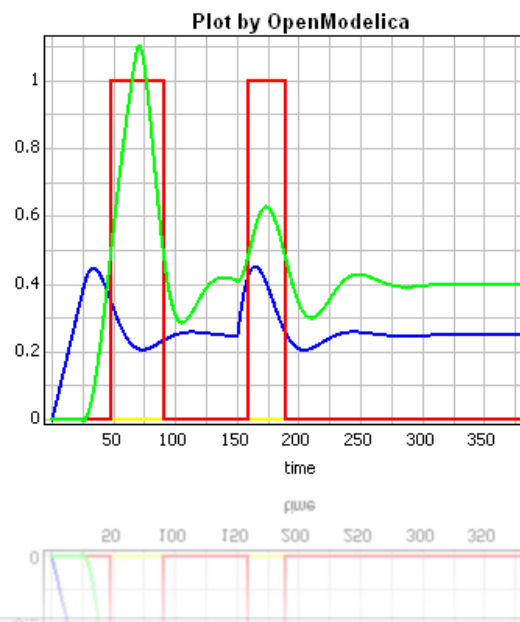


Simulation and Report Generation in ModelicaML

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)

mandatory 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**

mandatory client: [req 004 drain mode behavior.fillDrainValveIsOpen](#) (changed its value)

Type : = ModelicaBoolean

Variability : = continuous

Binding code : = sm_spws_environment.spws.fillDrainValve.isFullyOpen

mandatory client: [req 004 drain mode behavior.otherValvesAreClosed](#) (changed its value)

Type : = ModelicaBoolean

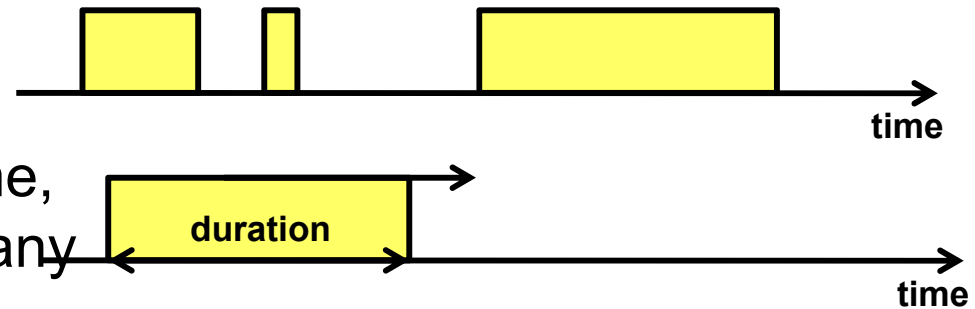
Variability : = continuous

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

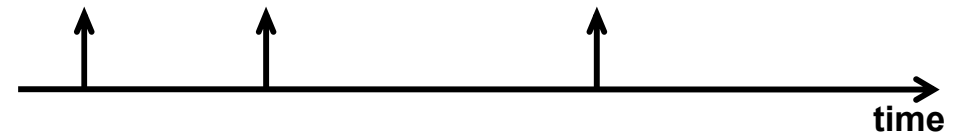
- A Continuous Time Locator(CTL) specifies one or more time intervals

- 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
<i>duringAny</i> duration	duringAny (duration)
<i>after</i> event	after (event)
<i>after</i> event1 <i>untilNext</i> event2	afterUntil (event1, event2)
<i>after</i> event <i>for</i> duration	afterFor (event, duration)
<i>after</i> event <i>within</i> duration	afterWithin (event, duration)
<i>until</i> event	until (event)
<i>every</i> duration1 <i>for</i> duration2	everyFor (duration1, duration2)
<i>when</i> condition <i>changes</i>	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;

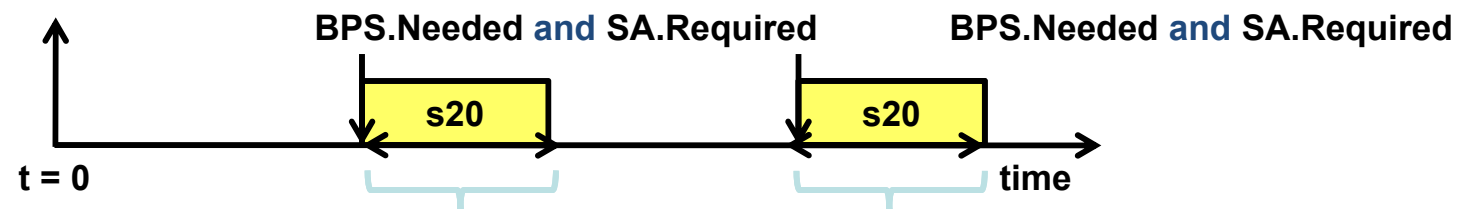
equation

status = **if** afterWithin (bPSNeeded == notViolated **and**
sARequired == notViolated, 20) **then**

if set1Powered == notViolated **then**

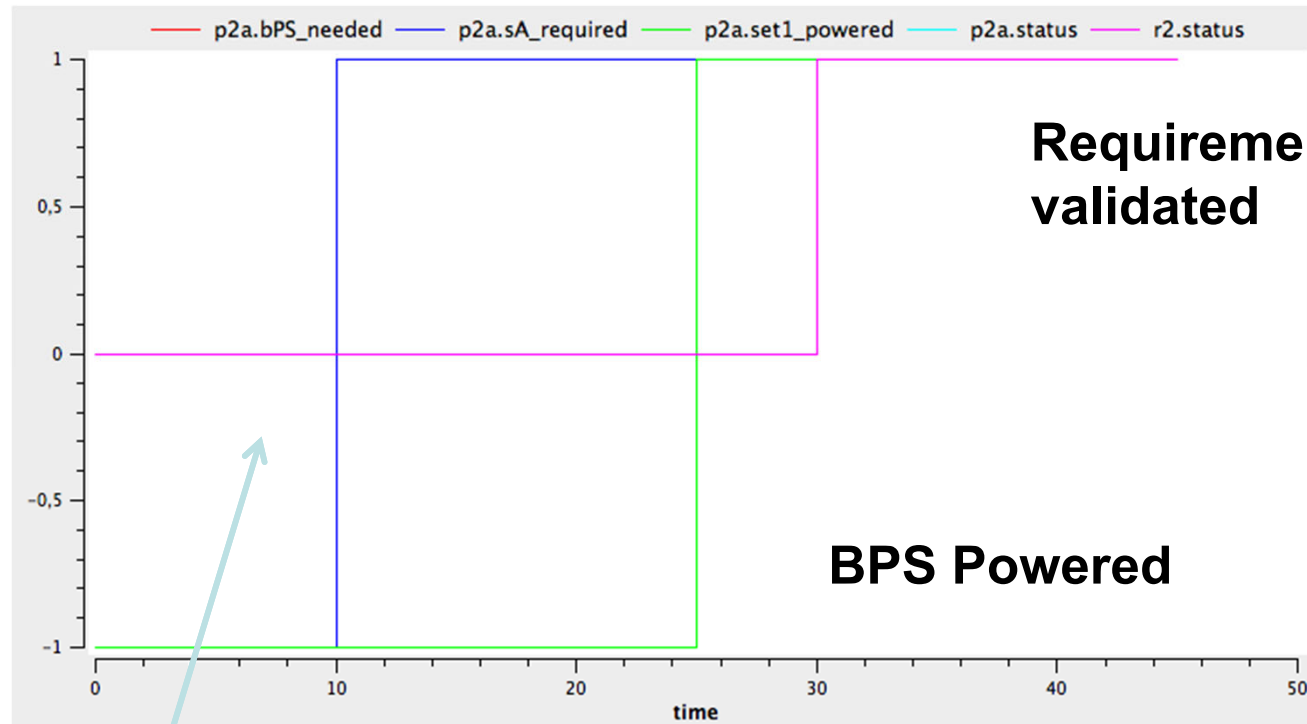
notViolated **else** violated **else** undefined;

end P2a;



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

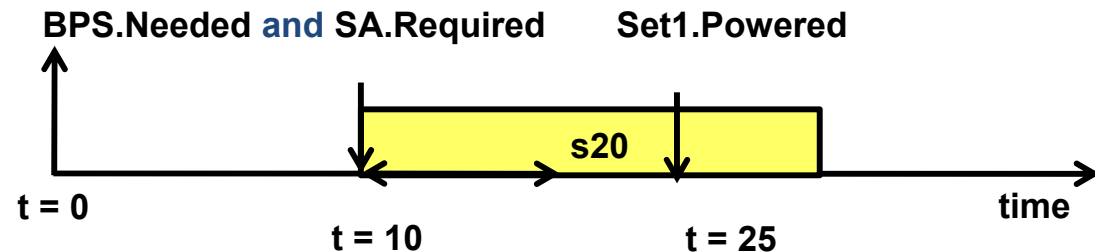
From Text to Simulated Requirement – Requirement not Violated – OpenModelica Simulation



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

Requirement undefined outside the specified time window

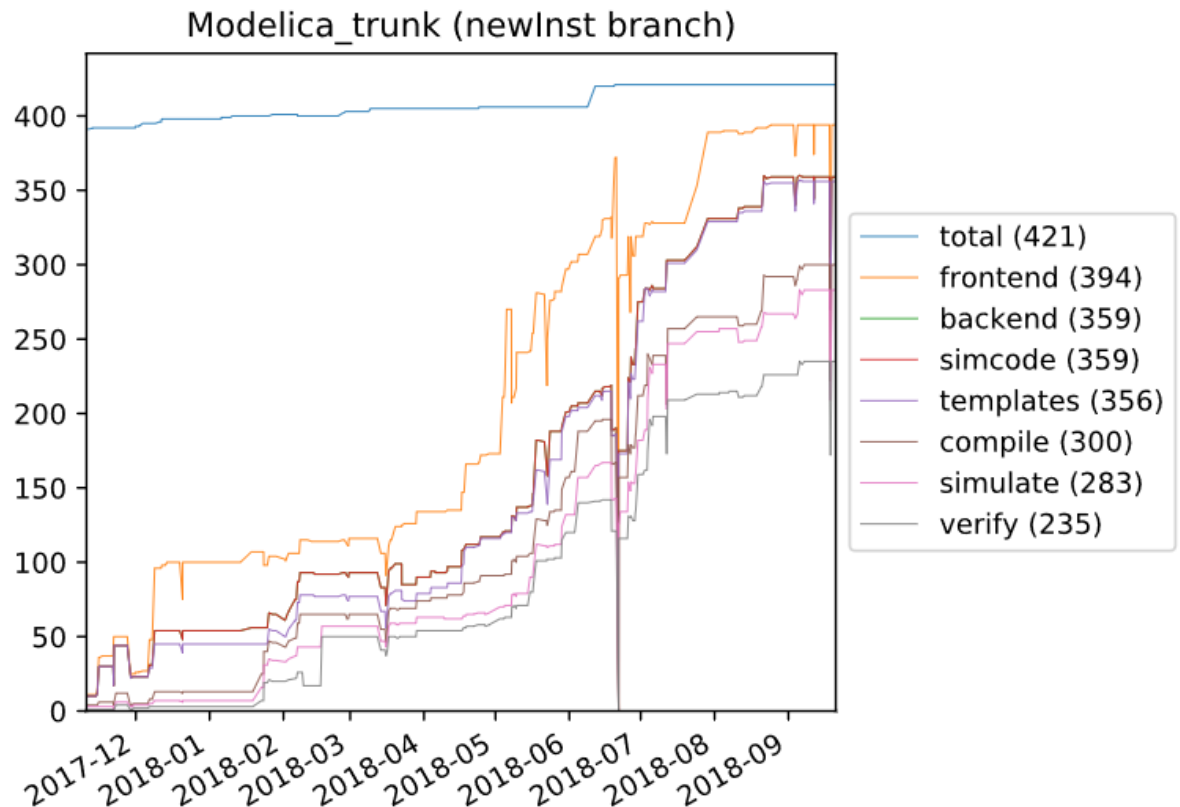
Within 20s



Outlook:

New OpenModelica Frontend for Large-Scale models

- Soon: New OMC Compiler frontend for fast compilation and large-scale models
- Been under development the past 2-3 years
- Now (sept 24) simulates 67% of MSL models, coverage increases about 6% per month
- About 10-200 times faster than the old frontend, depending on model



OpenModelica DAEMode for Large-Scale models

- Goal – to handle hundreds of thousands to millions of equations
- Introduced sparse solvers in the solution chain:
 - KLU for linear algebraic equations,
 - Kinsol for nonlinear algebraic equations, and
 - IDA for causalized differential equations.
- Largest system so far: electro-mechanical power system model with about 600.000 differential-algebraic equations
- Under development for even larger systems

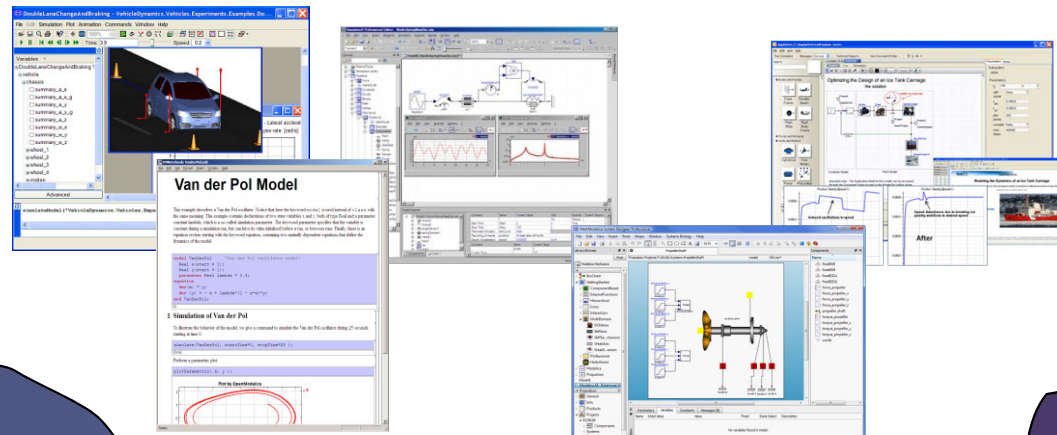
Summary and Questions

Multi-Domain
Modeling


M O D E L I C A

Visual Acausal
Component
Modeling

www.modelica.org – Language, Standard Library
www.openmodelica.org – Open Source Tool



Typed
Declarative
Textual Language

Thanks for listening!

Hybrid
Modeling