

Introduction to Modelica modeling and the OpenModelica and MathModelica tools

A Servo Mechanism Model
- a micro example of a full system

1 Introduction
Basic Modelica will be used to model a servo motor, which is a typical example of a mechanical system with electrical control. A block of control software receiving an input signal and controlling a motor is used to illustrate the use of the Modelica language. The use of the Modelica language is illustrated in the following sections.

2 DC Motor
A simple block model of a controlled DC motor is shown. The motor is controlled by a voltage signal. The motor is modeled as a mechanical system with electrical control. The motor is modeled as a mechanical system with electrical control. The motor is modeled as a mechanical system with electrical control.

Figure 1: Block diagram of a servo mechanism.

Figure 2: Plot of the motor's angular velocity over time.

Figure 3: Plot of the motor's angular position over time.

Figure 4: Plot of the motor's current over time.

Figure 5: Plot of the motor's torque over time.

Figure 6: Plot of the motor's angular acceleration over time.

Figure 7: Plot of the motor's angular velocity over time.

Figure 8: Plot of the motor's angular position over time.

Figure 9: Plot of the motor's current over time.

Figure 10: Plot of the motor's torque over time.

Figure 11: Plot of the motor's angular acceleration over time.

6 Conclusions
The control design was using the full set of tools provided by the Modelica language. The design is described in the following sections.

$$\tau_2 = \frac{1}{k_2} \tau_1$$

$$e = \omega_{ref} - \omega_{out}$$

$$u = K \left(e + \frac{1}{T_I} \int_0^t e \, dt \right)$$

$$v = u \quad u_R = R i \quad u_{emf} = k_1 \omega_{emf}$$

$$v = u$$

$$\theta_2 = k_2 \theta_1$$

$$u_L = L \frac{di}{dt}$$

$$u = K \left(e + \frac{1}{T_I} \int_0^t e \, dt \right)$$

$$e = \omega_{ref} - \omega_{out}$$

$$v - u_R - u_L - u_{emf} = 0$$

$$u_{emf} = k_1 \omega_{emf} \quad i = \frac{1}{k_1} \tau_{emf} \quad \tau_2 = \frac{1}{k_2} \tau_1$$

$$\frac{J_1 - J_2 k_2^2}{k_2} \frac{d^2 \theta_2}{dt^2} = \tau_{emf} - k_2 \tau_3$$

$$J_1 \frac{d^2 \theta_1}{dt^2} = \tau_{emf} + \tau_1$$

$$J_2 \frac{d^2 \theta_2}{dt^2} = \tau_2 + \tau_3$$

$$J_3 \frac{d^2 \theta_3}{dt^2} = -\tau_4 - \tau_{load}$$

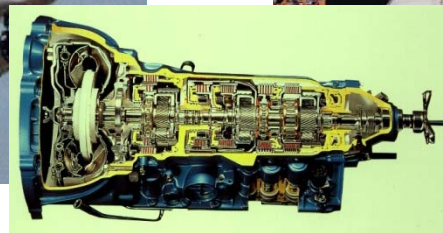
Invited talk to workshop
"Can Systems biology aid personalized medication?"

December 5, 2011

Peter Fritzson

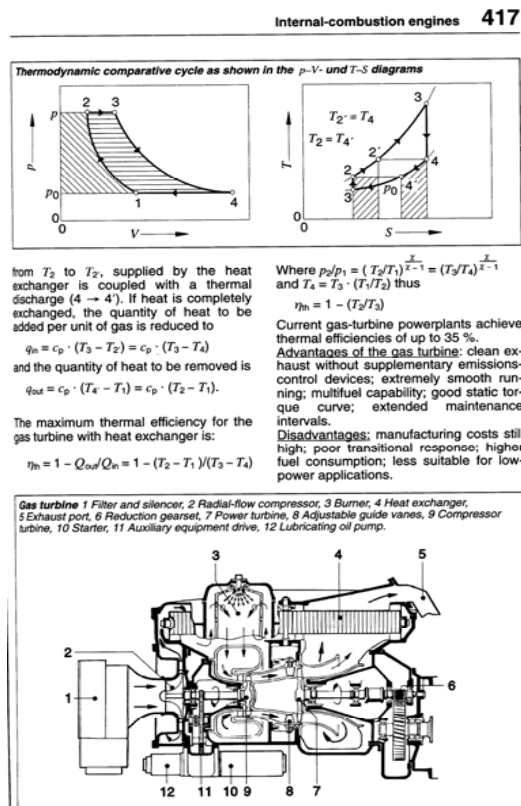
Professor at Linköping University, Sweden
Vice Chairman of Modelica Association
Director of Open Source Modelica Consortium
peter.fritzson@liu.se

Introduction to Modelica



Modelica Background: Stored Knowledge

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



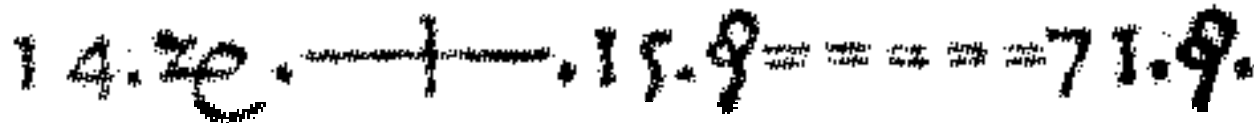
“The change of motion is proportional to the motive force impressed”
– Newton

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

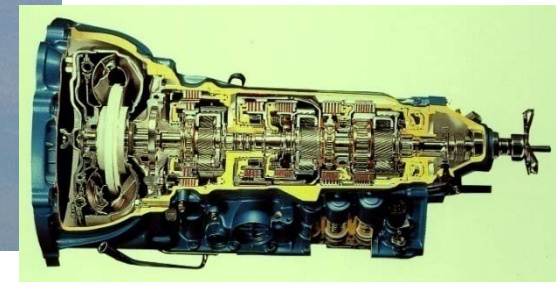
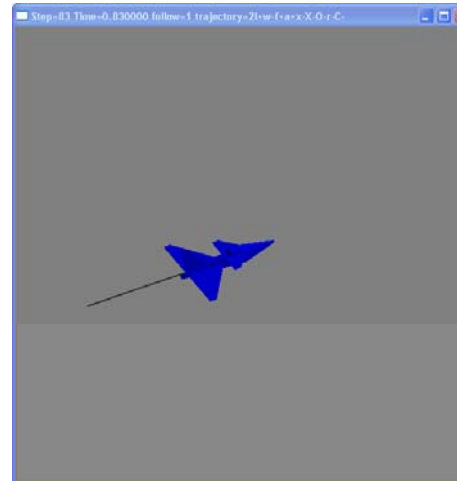
$v = \text{INTEG}(F)/m$

Programming languages usually do not allow equations!

What is Modelica?

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

- Robotics
- Control
- Automotive
- Aircraft
- Satellites
- Power plants
- Systems biology



What is Modelica?

A language for modeling of complex cyber physical systems

i.e., Modelica is not a tool

Free, open language specification:



There exist several free and commercial tools, for example:

- OpenModelica from OSMC
- MathModelica from MathCore
- Dymola from Dassault systems
- SimulationX from ITI
- MapleSim from MapleSoft

Available at: www.modelica.org

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 with Equations

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*:

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

can be used in three ways:

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

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

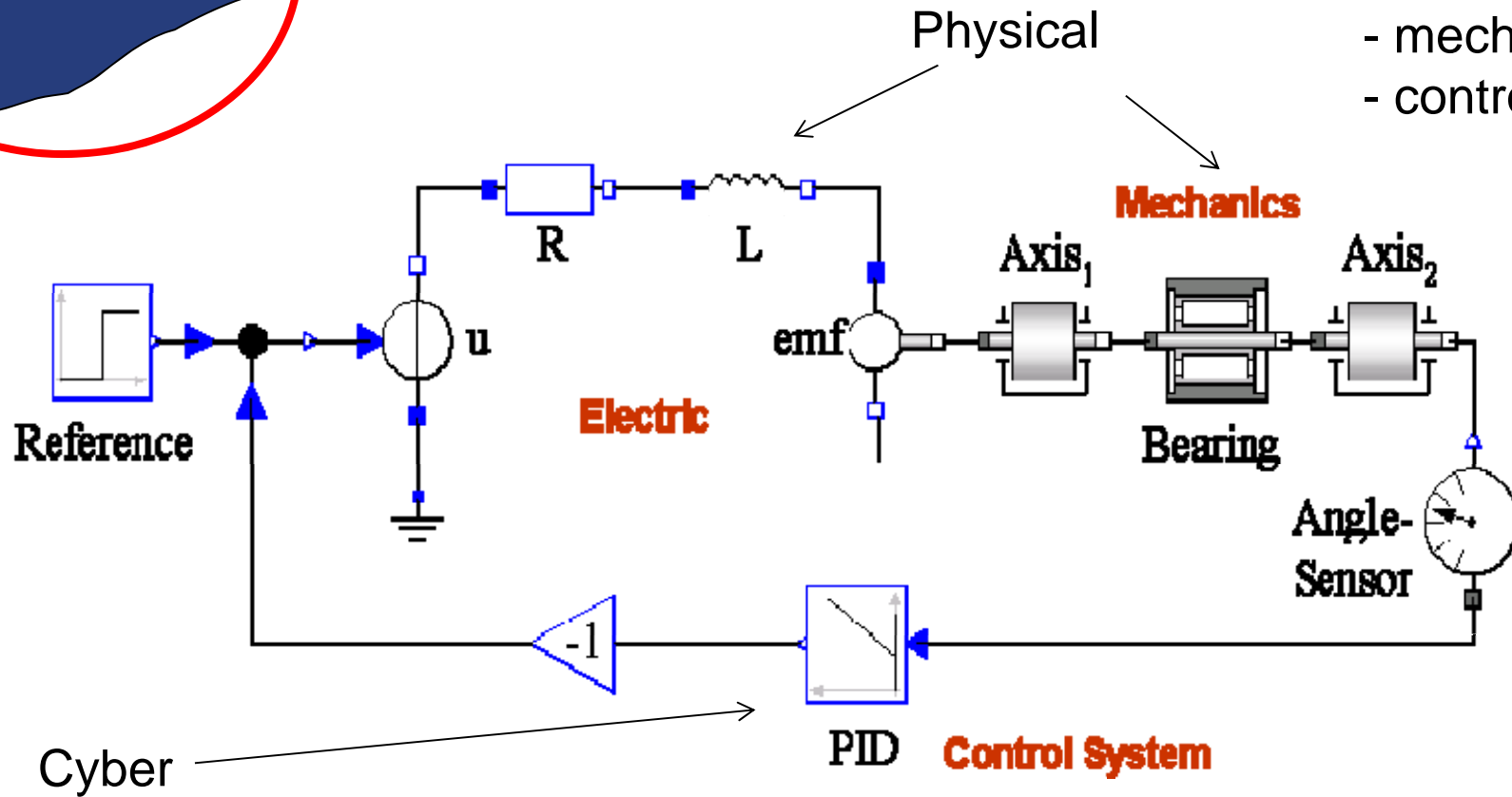
$$\mathbf{R} := \mathbf{v} / \mathbf{i};$$

What is Special about Modelica?

Multi-Domain Modeling

Cyber-Physical Modeling

- 3 domains
- electric
- mechanics
- control



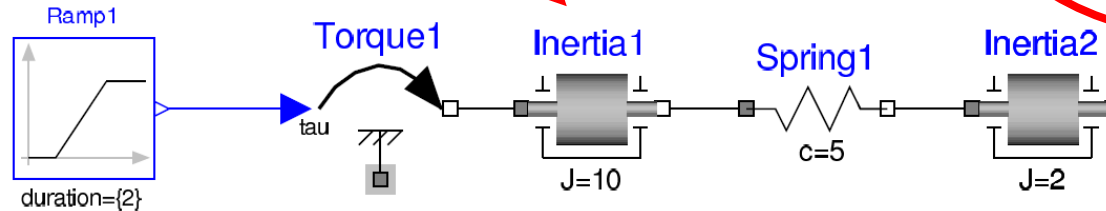
What is Special about Modelica?

Multi-Domain Modeling

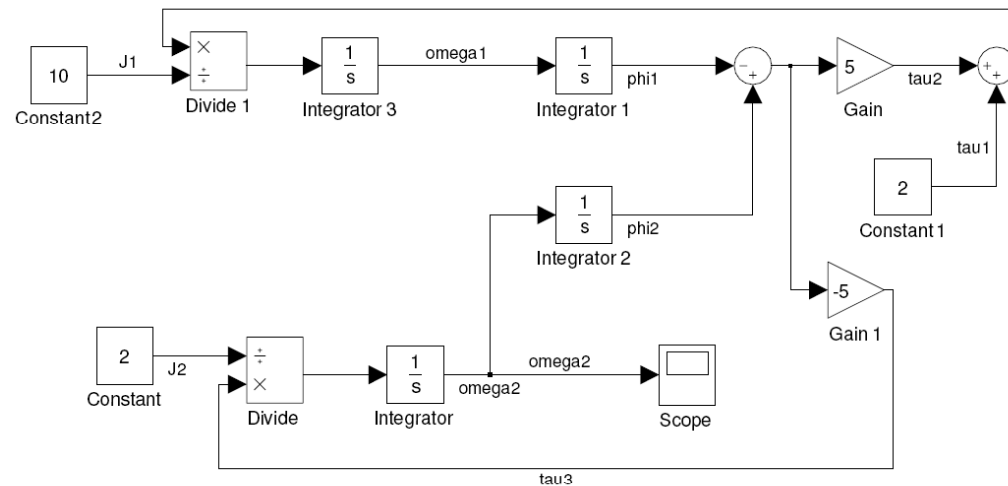
Visual Acausal Hierarchical Component Modeling

Keeps the physical structure

Acausal model (Modelica)



Causal block-based model (Simulink)

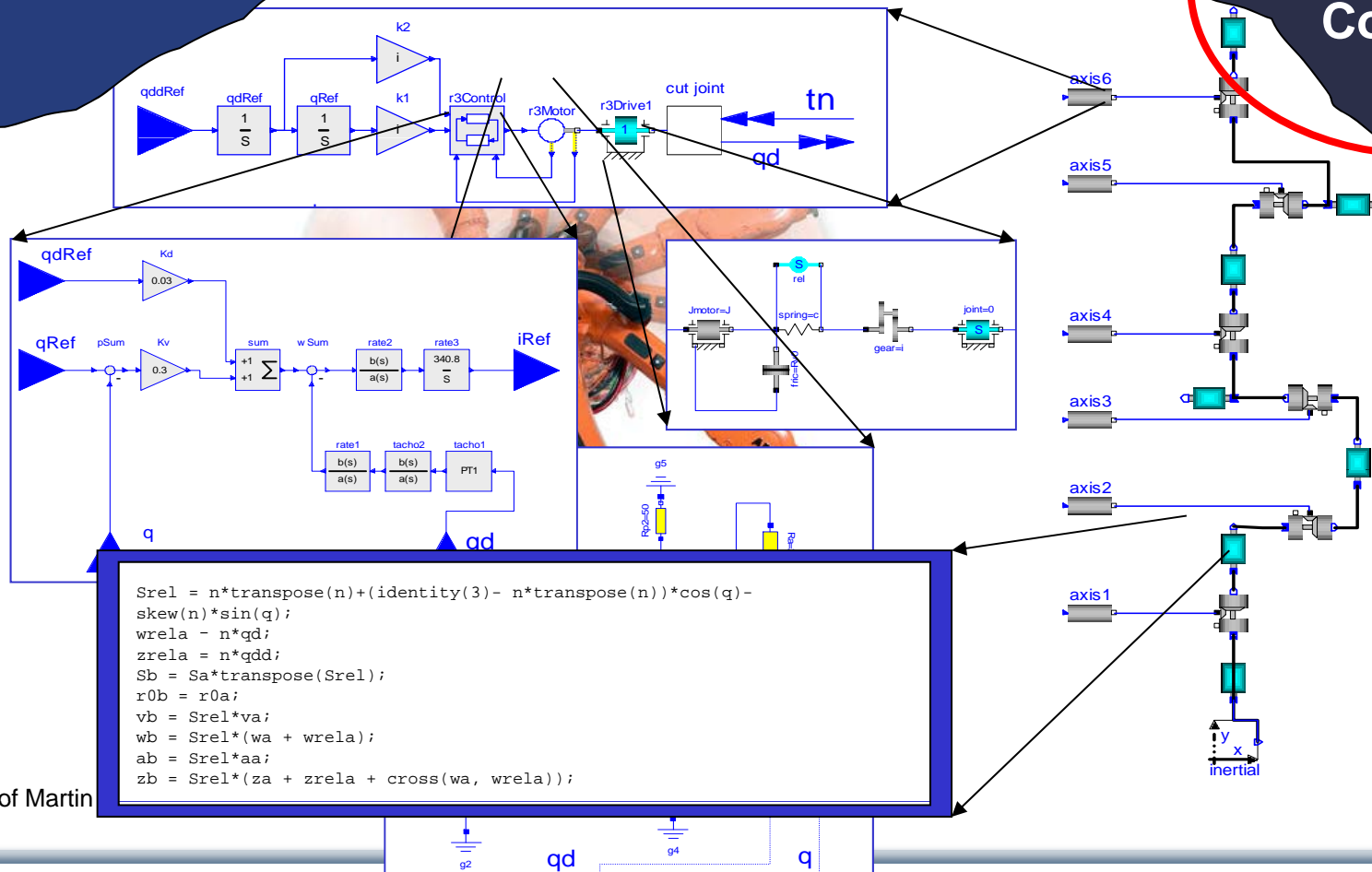


What is Special about Modelica?

Multi-Domain Modeling

Hierarchical system modeling

Visual Acausal Hierarchical Component Modeling



Courtesy of Martin

What is Special about Modelica?

Multi-Domain Modeling

Visual Acausal Hierarchical Component Modeling

A textual *class-based* language
Object-Orientation mainly used as structuring concept

Behaviour described declaratively using

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

Variable declarations

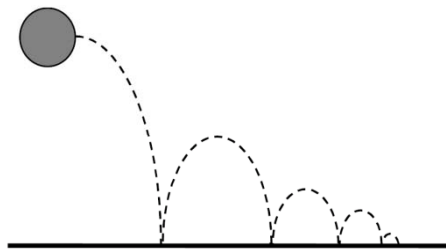
```
class VanDerPol "Van der Pol oscillator model"  
  Real x(start = 1) "Descriptive string for x";  
  Real y(start = 1) "y coordinate";  
  parameter Real lambda = 0.3;  
equation  
  der(x) = y;  
  der(y) = -x + lambda*(1 - x*x)*y;  
end VanDerPol;
```

Differential equations

Typed Declarative Equation-based Textual Language

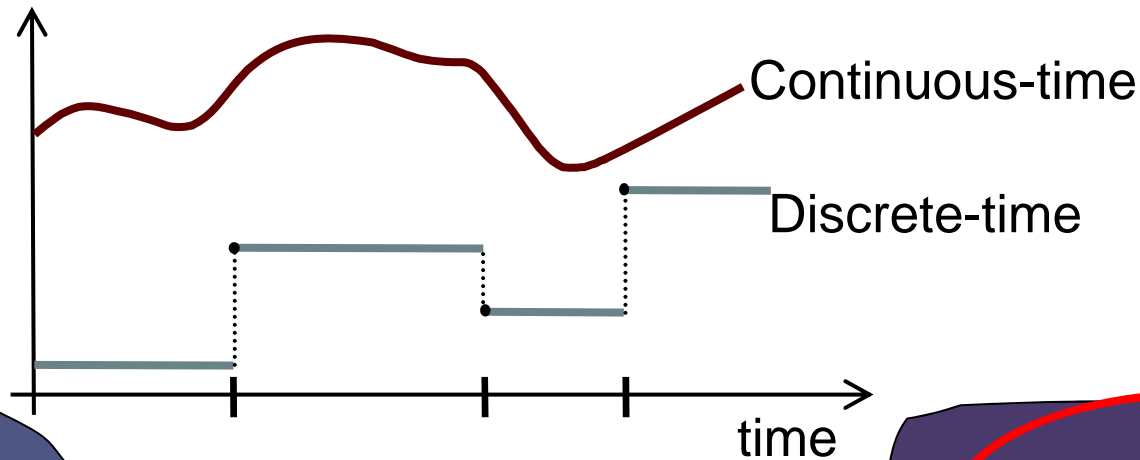
What is Special about Modelica?

Multi-Domain Modeling



Visual Acausal Component Modeling

Hybrid modeling =
continuous-time + discrete-time modeling



Typed Declarative Equation-based Textual Language

Hybrid Modeling

Graphical Modeling - Using Drag and Drop Composition

The screenshot displays the OMEdit - OpenModelica Connection Editor interface. The main workspace shows a graphical circuit model of a DC motor. The circuit includes a step function block labeled 'step1' with the parameter 'startTime=%startTime', connected to a voltage source 'signalvoltage1'. This source is in series with a resistor 'resistor1' and an inductor 'inductor1' with inductance 'L=2'. The circuit is connected to a motor block 'emf1' with a torque constant 'k=' and an inertia block 'inertia1' with inertia 'J='.

The left sidebar shows the 'Modelica Standard Library' with categories like Blocks, Constants, Electrical, Magnetic, Math, and Mechanics. The right sidebar shows the 'Plot Variables' window for 'DCMotor_res.mat', listing variables such as 'emf1', 'der(phi)', 'fixed', 'flange', 'i', 'internalSupport', 'k', 'n', 'p', 'phi', 'useSupport', 'v', 'w', 'ground1', 'inductor1', 'inertia1', 'resistor1', 'signalvoltage1', and 'step1'.

The bottom status bar shows the version information: 'OMEdit, Version: 1.7.0' and 'OpenModelica, Version: "1.7.0"'. The Messages window is currently empty.

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);
```

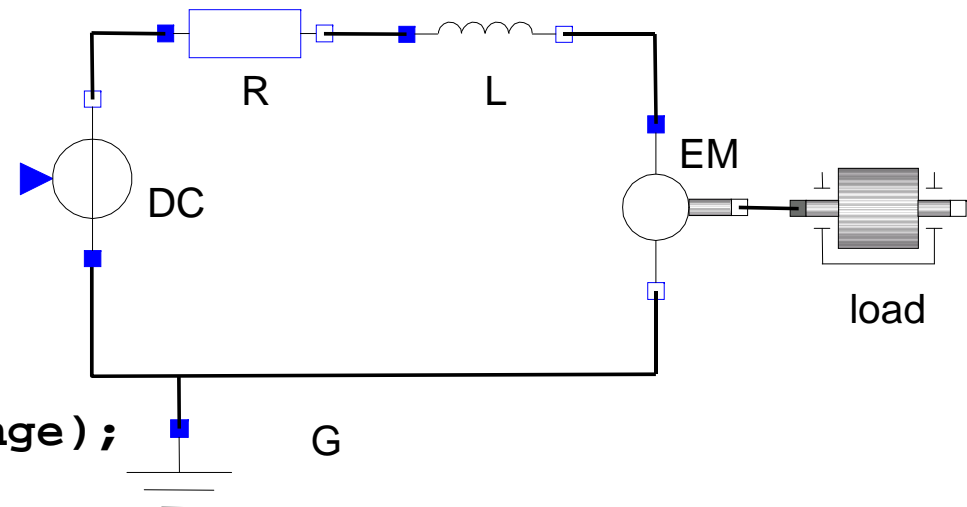
```
connect(L.p, EM.n);
```

```
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:

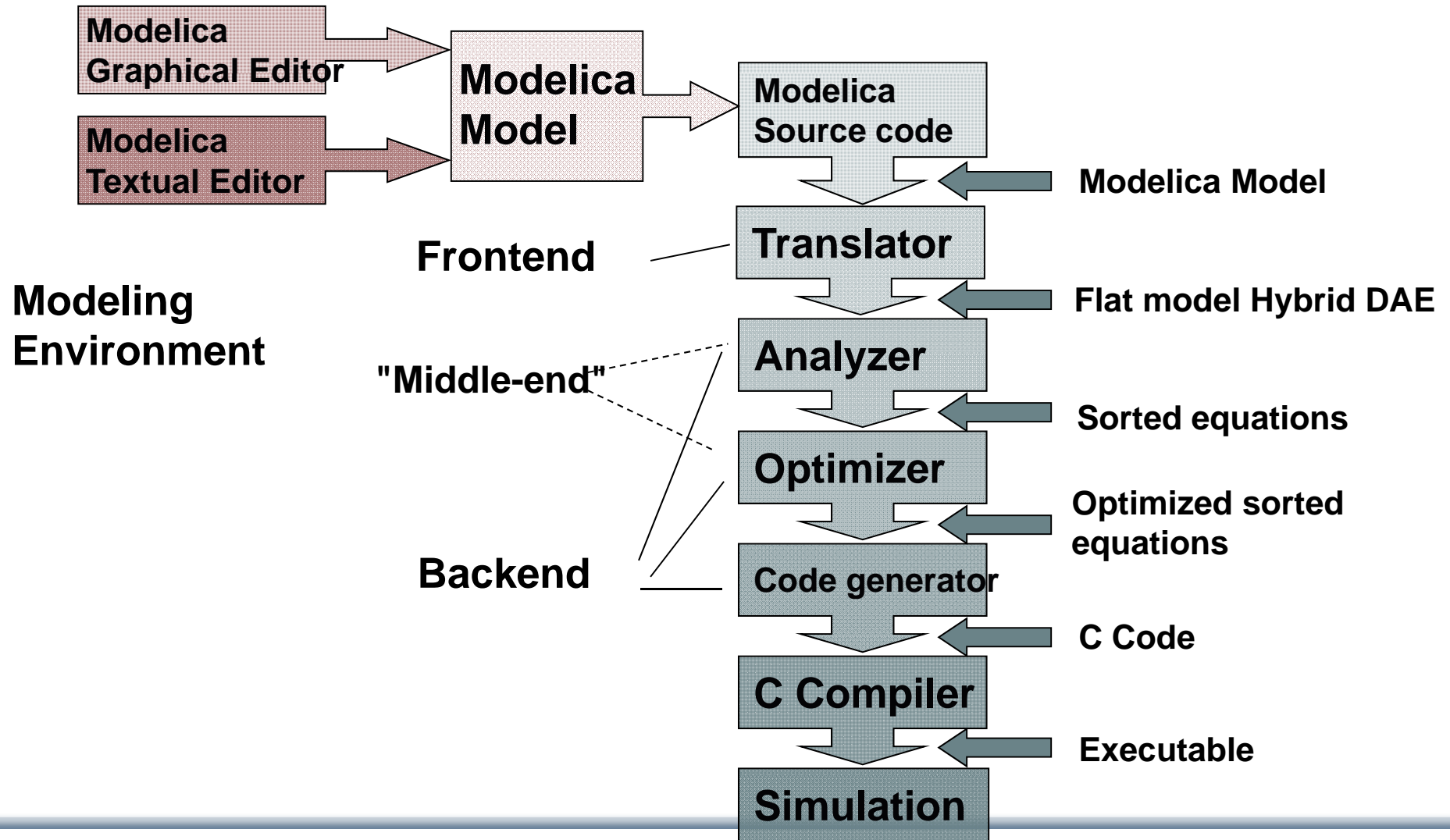
$0 == DC.p.i + R.n.i$	$EM.u == EM.p.v - EM.n.v$	$R.u == R.p.v - R.n.v$
$DC.p.v == R.n.v$	$0 == EM.p.i + EM.n.i$	$0 == R.p.i + R.n.i$
	$EM.i == EM.p.i$	$R.i == R.p.i$
$0 == R.p.i + L.n.i$	$EM.u == EM.k * EM.\omega$	$R.u == R.R * R.i$
$R.p.v == L.n.v$	$EM.i == EM.M / EM.k$	
	$EM.J * EM.\omega == EM.M -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

(load component not included)

Automatic transformation to ODE or DAE for simulation:

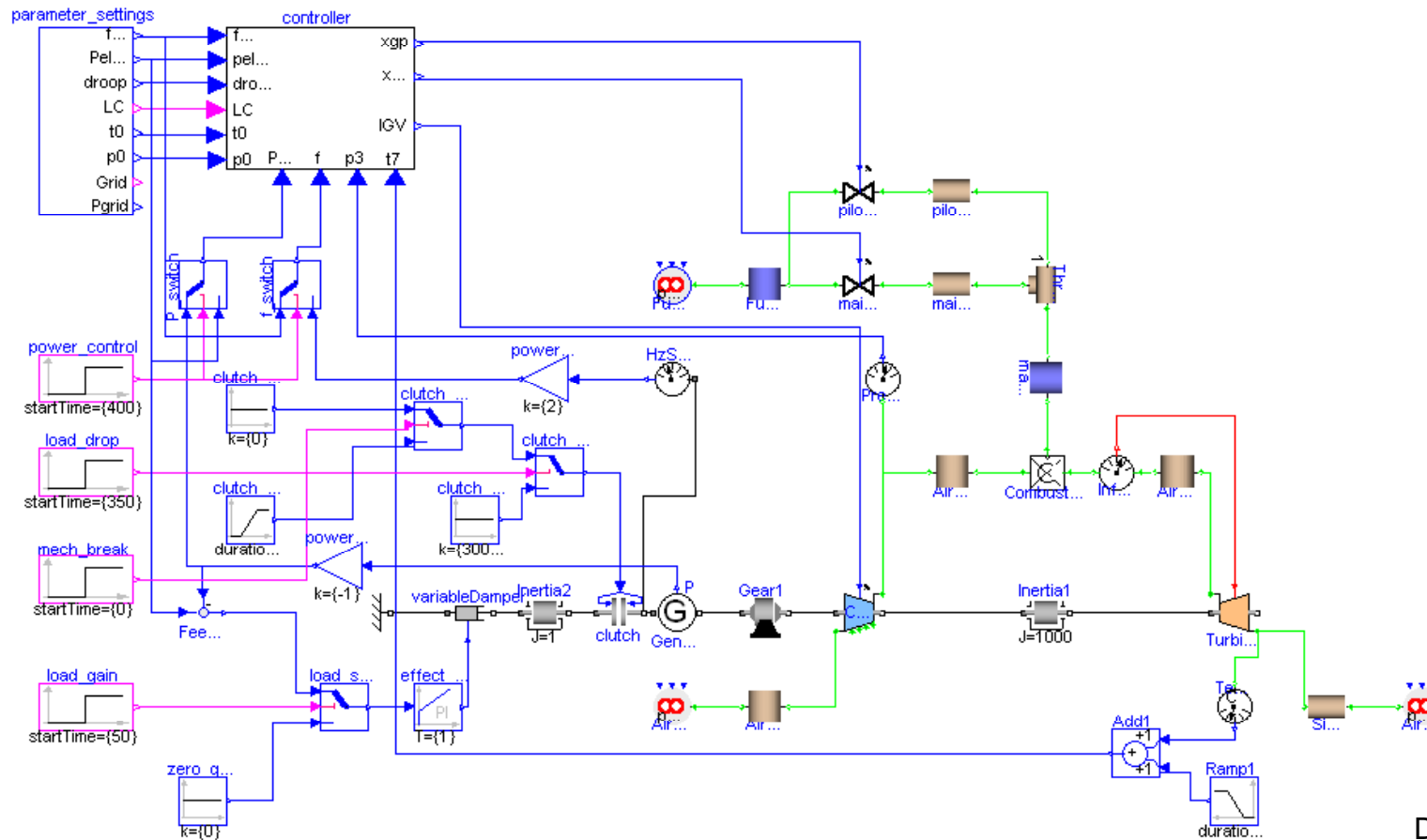
$$\frac{dx}{dt} == f[x, u, t] \quad g\left[\frac{dx}{dt}, x, u, t\right] == 0$$

Model Translation Process to Hybrid DAE to Code



Modelica in Power Generation

GTX Gas Turbine Power Cutoff Mechanism



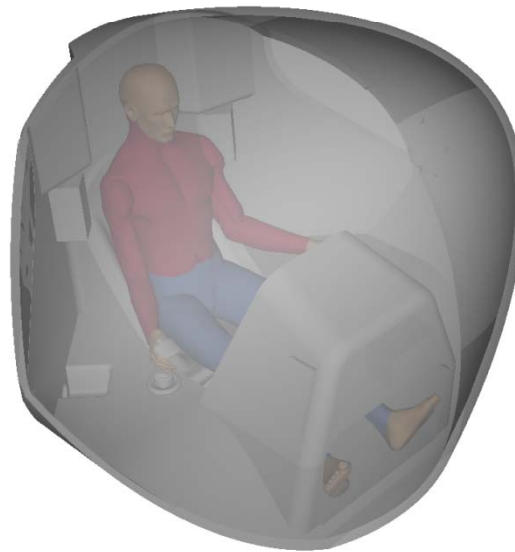
Developed
by MathCore
for Siemens

Courtesy of Siemens Industrial Turbomachinery AB, Finspång, Sweden

Application of Modelica in Robotics Models

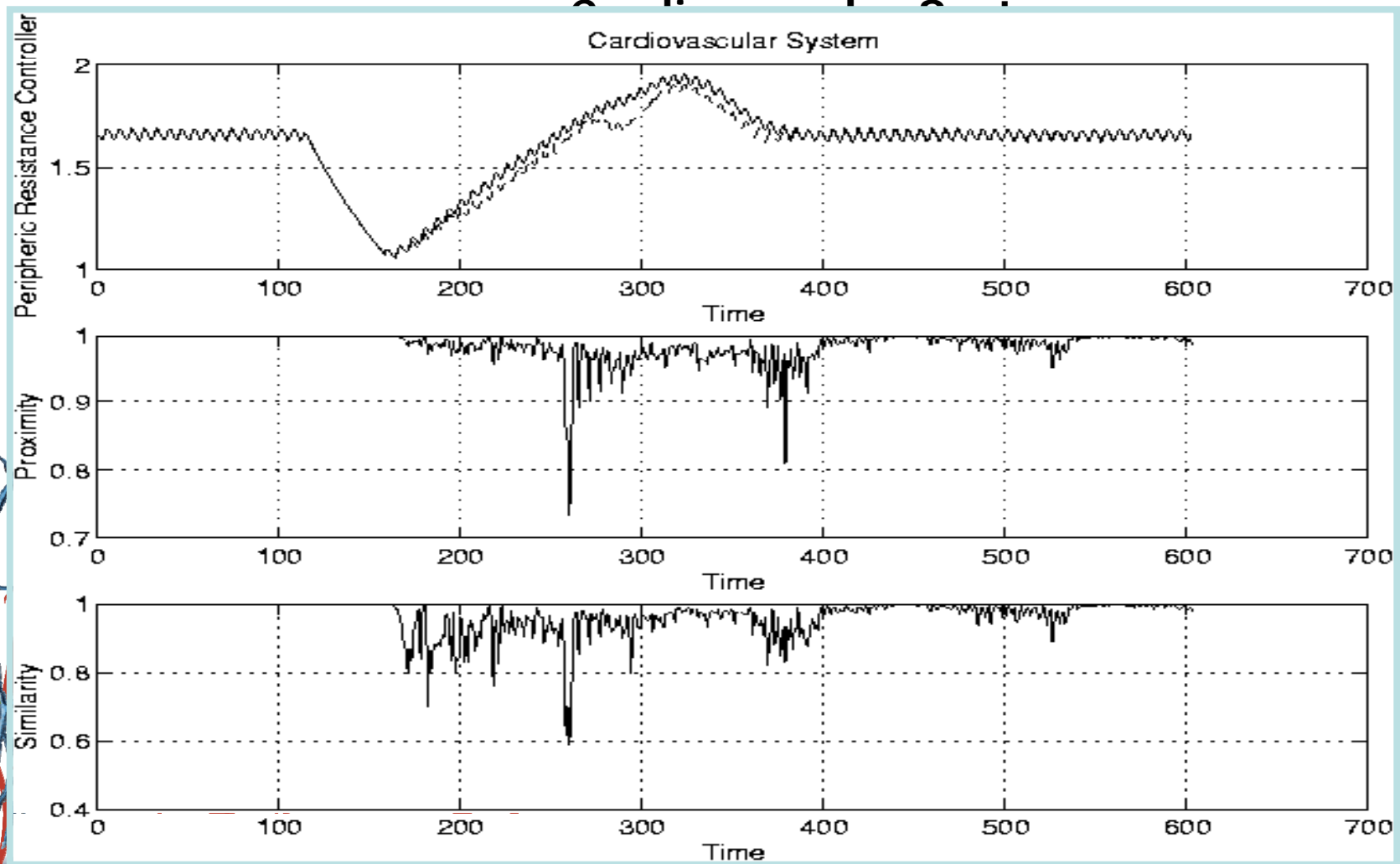
Real-time Training Simulator for Flight, Driving

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



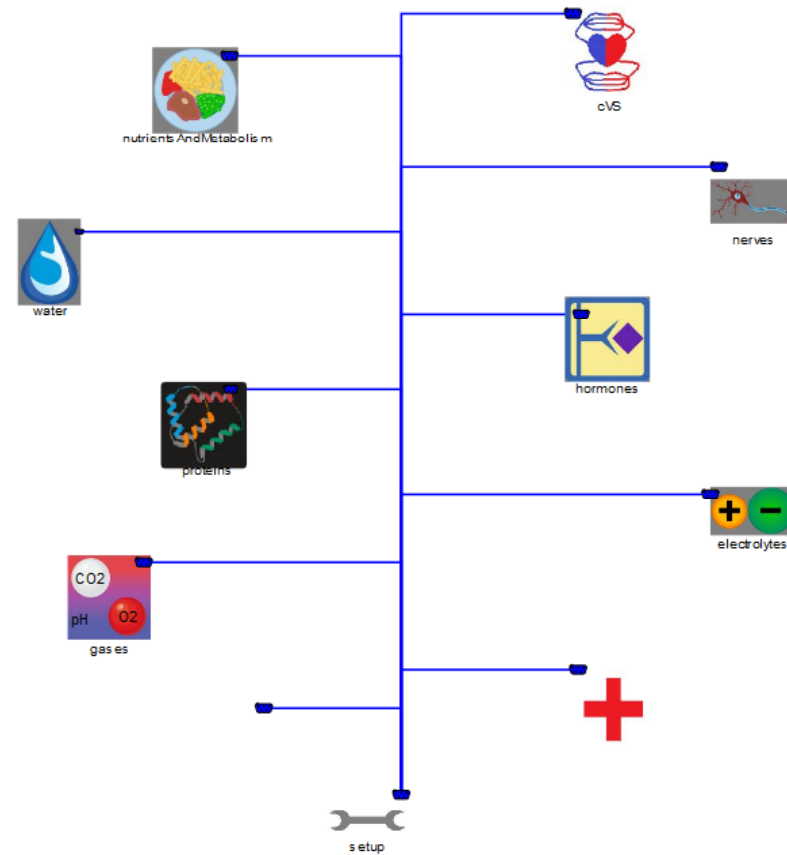
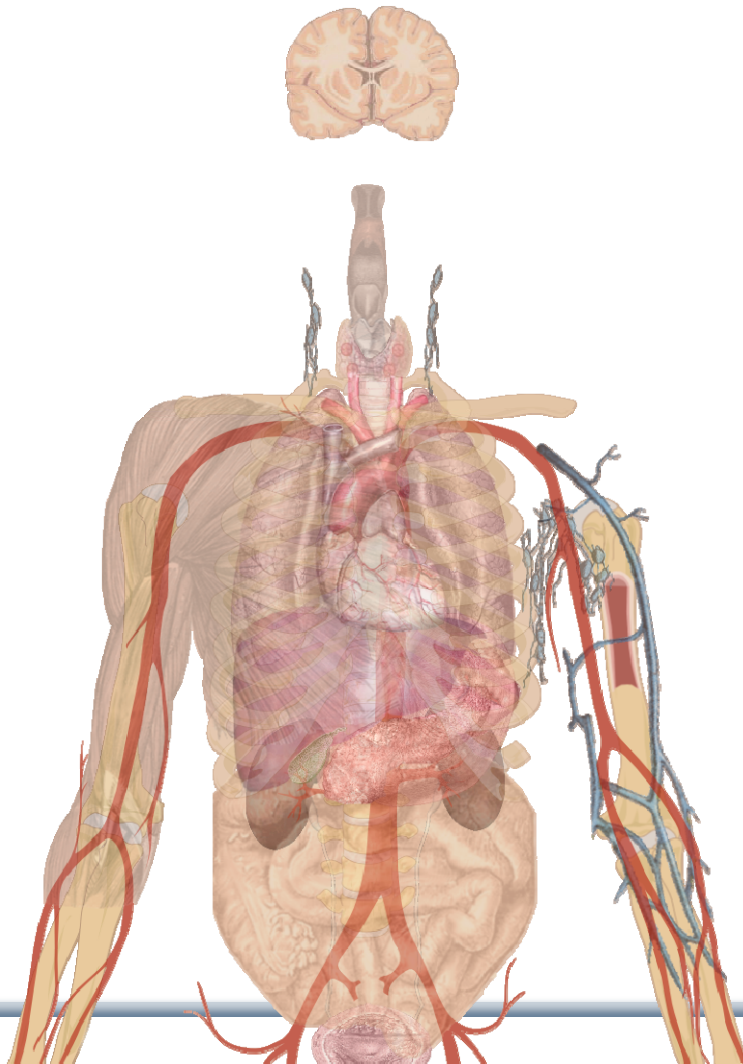
Courtesy of Martin Otter, DLR, Oberpfaffe Germany

Modelica Examples – Systems Biology



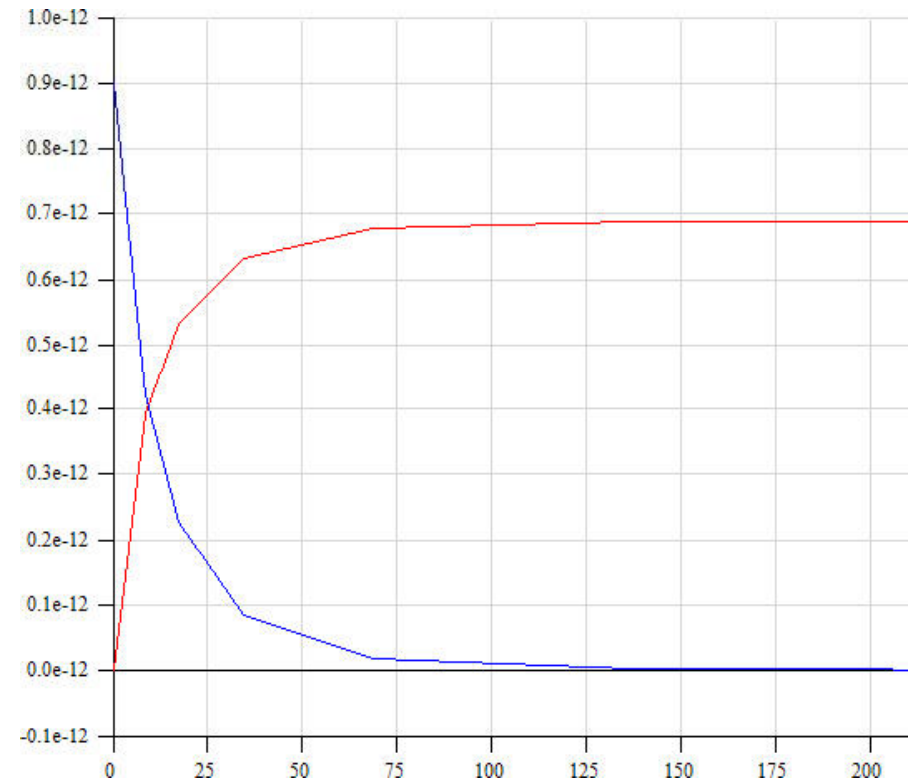
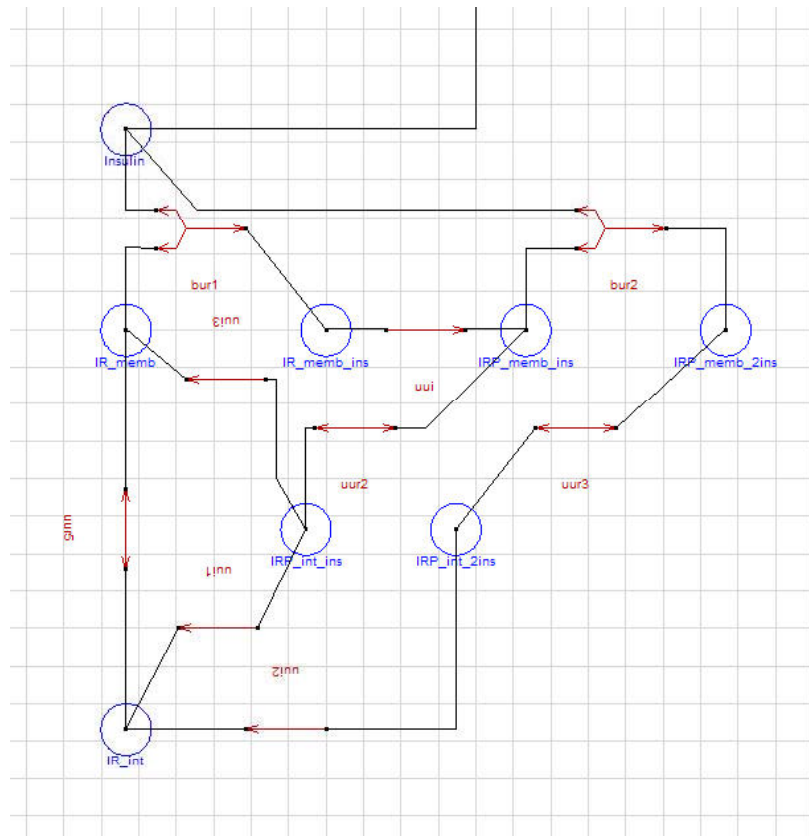
Modelica Examples – Systems Biology

HumMod
Kofranek et Al., Charles University, Prag

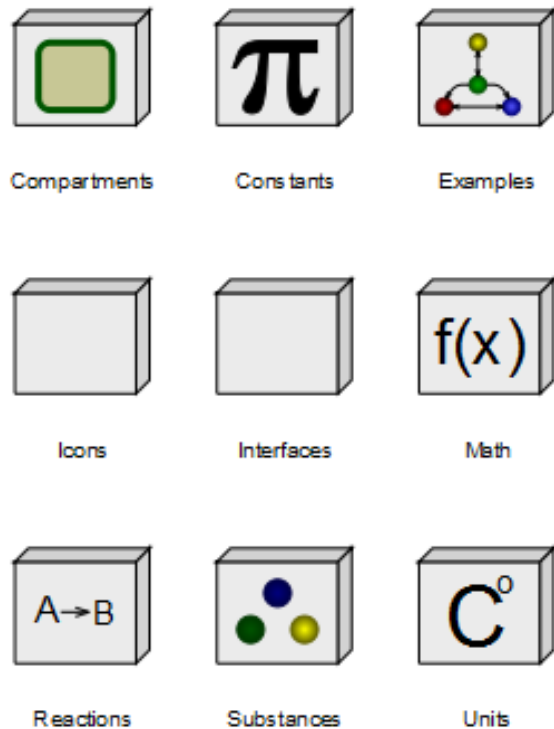


Cell Biology Insulin Receptor Model (Sedgheat et al)

Modelica Modeling Using a free PathWay Library (BioChem)



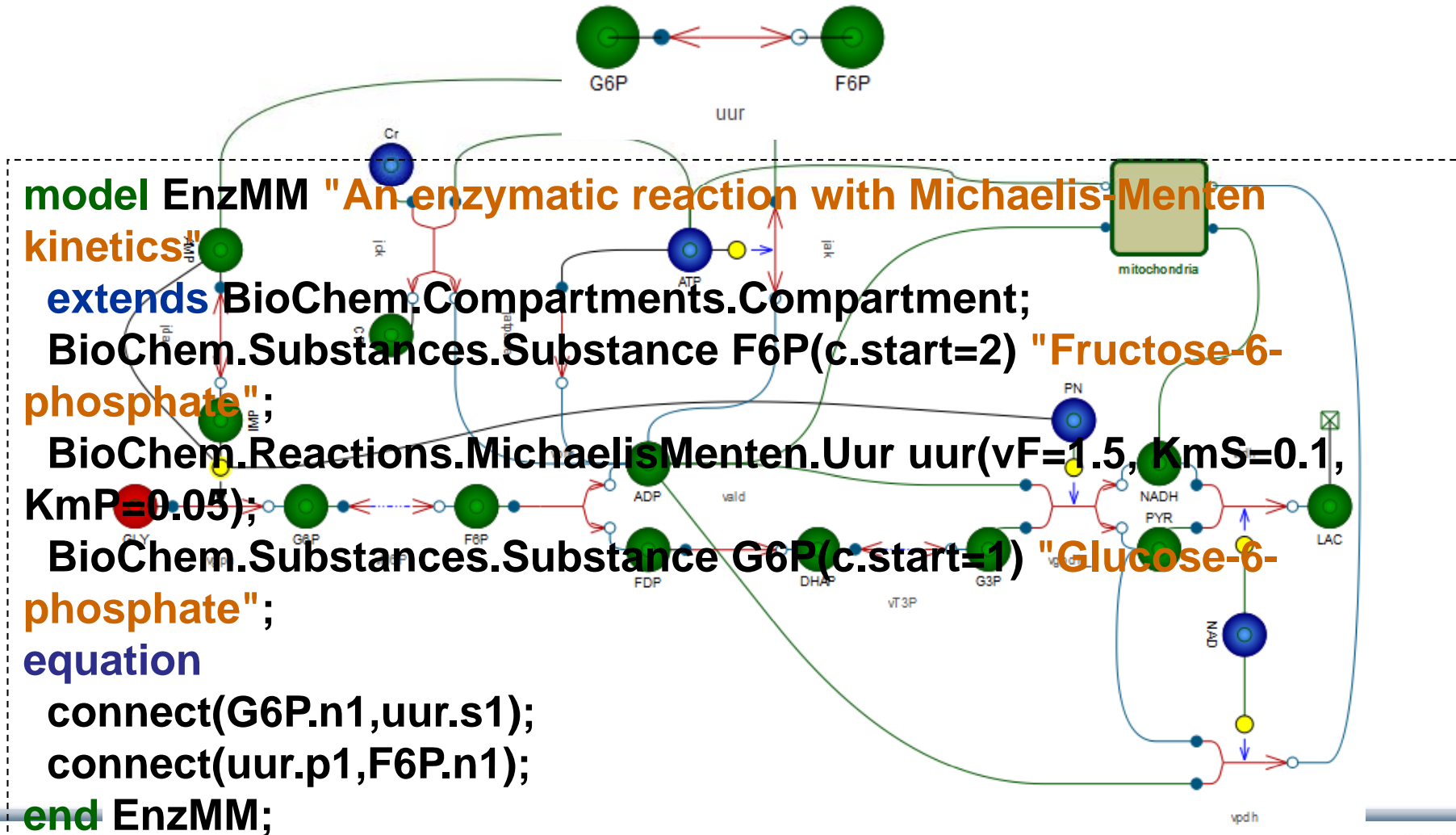
The BioChem Library for PathWay Modeling



- Free Open Source Library
- Originally Developed at PELAB/IDA LIU 2003-2006, continued development at MathCore Engineering AB and LIU
- Also used for SBML to Modelica mapping

Several BioChem Slides, Courtesy of Jan Brugård, MathCore Engineering AB/ Wolfram Research

Introductory Modelica Example using BioChem



Modelica Standard Library

Open Source, Developed by Modelica Association

The Modelica Standard Library contains components from various application areas, including the following sublibraries:

- Blocks Library for basic input/output control blocks
- Constants Mathematical constants and constants of nature
- Electrical Library for electrical models
- Icons Icon definitions
- Fluid 1-dim Flow in networks of vessels, pipes, fluid machines, valves, etc.
- Math Mathematical functions
- Magnetic Magnetic.Fluxtubes – for magnetic applications
- Mechanics Library for mechanical systems
- Media Media models for liquids and gases
- Slunits Type definitions based on SI units according to ISO 31-1992
- Stategraph Hierarchical state machines (analogous to Statecharts)
- Thermal Components for thermal systems
- Utilities Utility functions especially for scripting

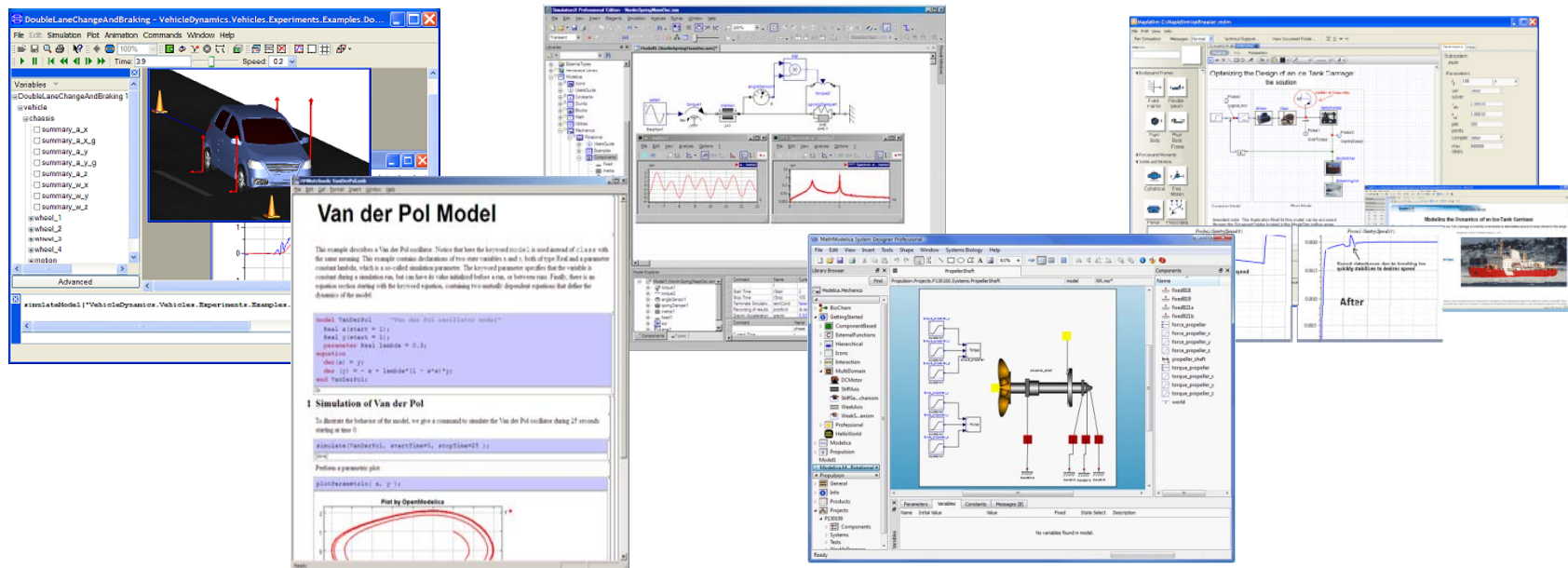
Brief Modelica History

- 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 May 2010
 - 3.3 planned Spring 2012
- Modelica Association established 2000
 - Open, non-profit organization

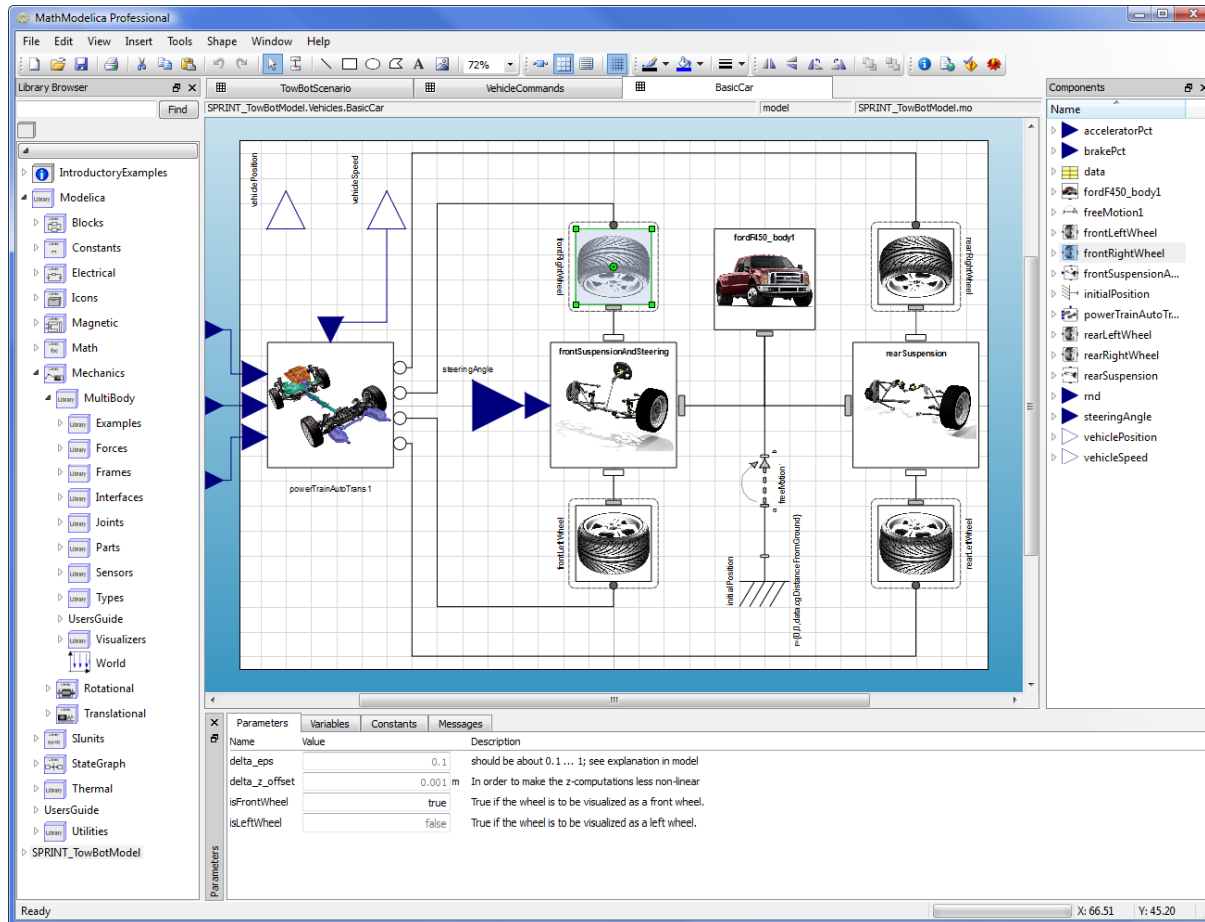
Modelica Conferences

- The 1st International Modelica conference October, 2000
- The 2nd International Modelica conference March 18-19, 2002
- The 3rd International Modelica conference November 5-6, 2003 in Linköping, Sweden
- The 4th International Modelica conference March 6-7, 2005 in Hamburg, Germany
- The 5th International Modelica conference September 4-5, 2006 in Vienna, Austria
- The 6th International Modelica conference March 3-4, 2008 in Bielefeld, Germany
- The 7th International Modelica conference Sept 21-22, 2009, Como, Italy
- The 8th International Modelica conference March 20-22, 2011 in Dresden, Germany
- **The 4th Int. OpenModelica Workshop, Febr 6, 2012, Linköping, Sweden**

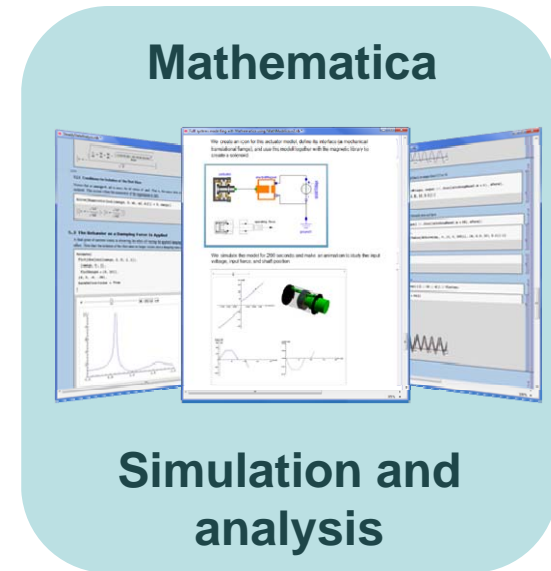
Modelica Environments and OpenModelica



MathModelica – MathCore / Wolfram Research



- Wolfram Research
- USA, Sweden
- General purpose
- Mathematica integration
- www.wolfram.com
- www.mathcore.com



Courtesy
Wolfram
Research

Car model graphical view

MathModelica – Car Model Simulation & Animation

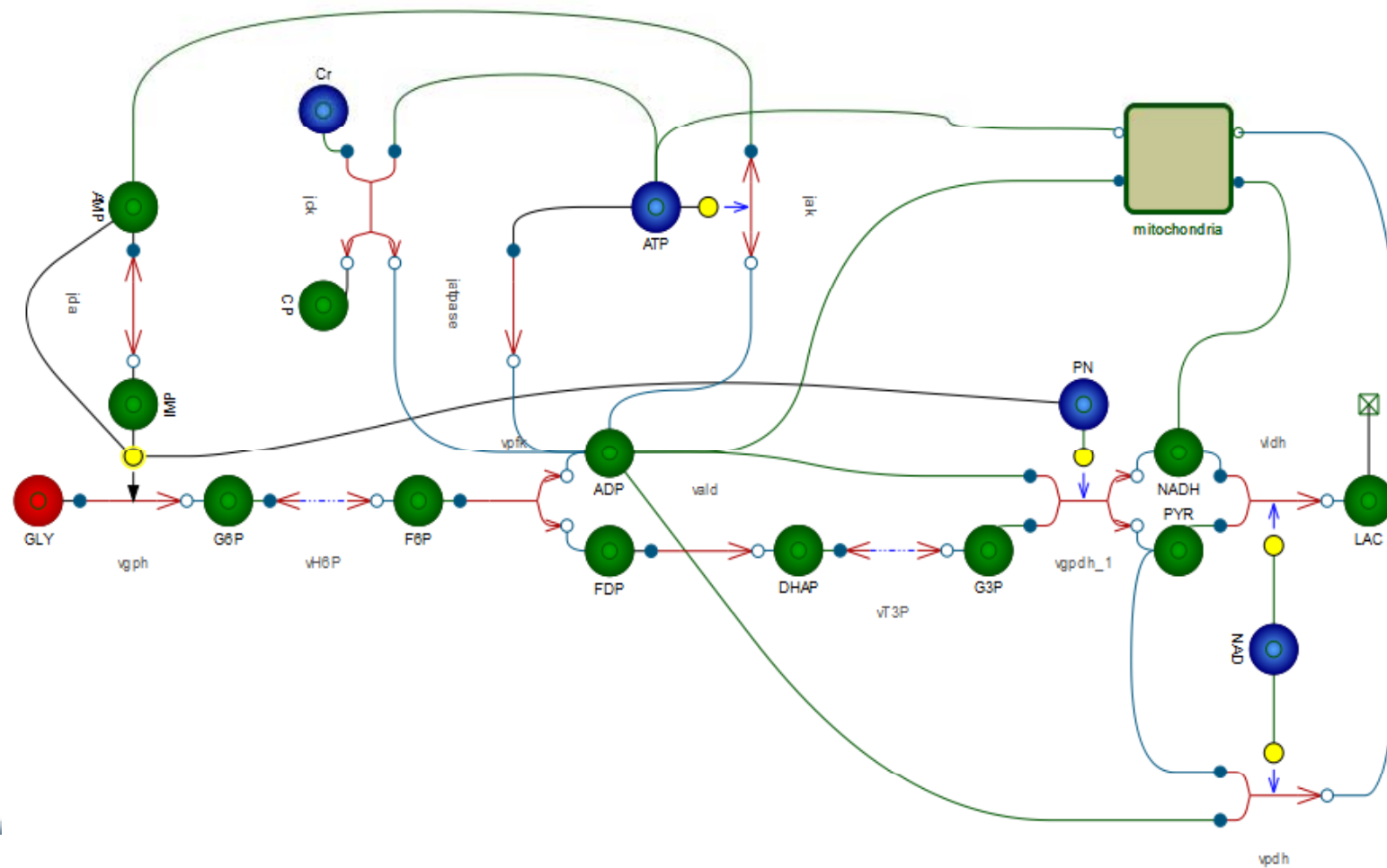
The screenshot displays the MathModelica Professional interface for a car simulation. The Experiment Browser on the left lists components such as `carAutoTrans1`, `fordF450_body1`, and `vehicleCommands1`. The central plots show simulation results:

- Plot Y(T): 1:** `carAutoTrans1.vehicleSpeed` vs `Time [s]`. The speed starts at 0, peaks at approximately 32 m/s at 10s, drops to 0 at 15s, peaks again at 21 m/s at 25s, drops to 0 at 30s, and then reaches a steady state of 14 m/s from 35s to 45s.
- Plot Y(T): 2:** `vehicleCommands1.pedal` (red dashed line) and `vehicleCommands1.brakes` (blue dashed line) vs `Time [s]`. The pedal command is 0 until 5s, then jumps to 0.25 until 10s. The brakes command is 0 until 10s, then jumps to 0.12 until 15s, drops to 0 until 25s, jumps to 0.12 until 30s, drops to 0 until 45s, and finally jumps to 0.13 until 50s.

The Animation window on the right shows a 3D model of a red Ford pickup truck. The status bar at the bottom indicates the simulation time is 7.3611905 seconds.

Courtesy
Wolfram
Research

PathWay Model Using BioChem and MathModelica



Introductory SBML to Modelica Example

model EnzMM "An enzymatic reaction with Michaelis-Menten kinetics"
extends BioChem.Compartments.Compartment;
BioChem.Substances.Substance F6P(c.start=2) "Fructose-6-phosphate";
BioChem.Reactions.MichaelisMenten.Uur1(vF=1.5, KmS=0.1, KmP=0.05);
BioChem.Substances.Substance G6P(c.start=1) "Glucose-6-phosphate";
equation
connect(G6P.n1,uur.s1);
connect(uur.p1,F6P.n1);
end EnzMM;

```

<sbml xmlns="http://www.sbml.org/sbml/level2" level="2">
  <model id="MyExample">
    <listOfCompartments>
      <compartment id="EnzMM" size="1" />
    </listOfCompartments>
    <listOfSpecies>
      <species id="G6P" compartment="EnzMM">
        <notes>
          <html xmlns="http://www.w3.org/1999/xhtml">
            <p>A substance with variable concentration.</p>
          </html>
        </notes>
      </species>
      <species id="F6P" compartment="EnzMM">
        <notes>
          <html xmlns="http://www.w3.org/1999/xhtml">
            <p>A substance with variable concentration.</p>
          </html>
        </notes>
      </species>
    </listOfSpecies>
    <listOfReactions>
      <reaction id="uur1">
        <math xmlns="http://www.w3.org/1999/xhtml">
          G6P \rightleftharpoons F6P
        </math>
        <listOfReactants>
          <speciesReference species="G6P"/>
        </listOfReactants>
        <listOfProducts>
          <speciesReference species="F6P"/>
        </listOfProducts>
        <kineticLaw>
          <math xmlns="http://www.w3.org/1999/xhtml">
            v = \frac{vF \cdot G6P}{KmS + G6P + KmP \cdot F6P}
          </math>
          <listOfParameters>
            <parameter id="vF" name="Forward maximum velocity" value="1.5"/>
            <parameter id="vR" name="Reverse maximum velocity" value="1"/>
            <parameter id="KmS" name="Forward Michaelis-Menten constant" value="1"/>
            <parameter id="KmP" name="Reverse Michaelis-Menten constant" value="1"/>
          </listOfParameters>
        </kineticLaw>
      </reaction>
    </listOfReactions>
  </model>
</sbml>

```


SBML/Modelica Translator

- As the standard modeling language within systems biology applications SBML
 - has a wide range of ready-made models available.
 - a large user base with knowledge about the language and its applications
- By creating a translator to Modelica we can give the users access to a much richer language, offering new possibilities.

The Translation Challenge

4.2 Model

The definition of **Model** is shown in Figure 10 on the following page. Only one instance of a **Model** object is allowed per instance of an SBML Level 2 Version 3 Release 2 document or data stream, and it must be located inside the `<sbml> ... </sbml>` element as described in Section 4.1.

The **Model** object has an optional attribute, `id`, used to give the model an identifier. The identifier must be a text string conforming to the syntax permitted by the `STID` data type described in Section 3.1.7. **Model** also has an optional `name` attribute, of type `string`. The `name` and `id` attributes must be used as described in Section 3.3.

Model serves as a container for components of classes `FunctionDefinition`, `UnitDefinition`, `CompartmentType`, `SpeciesType`, `Compartment`, `Species`, `Parameter`, `InitialAssignment`, `Rule`, `Constraint`, `Reaction` and `Event`. Instances of the classes are placed inside instances of classes `ListOfFunctionDefinitions`, `ListOfUnitDefinitions`, `ListOfCompartmentTypes`, `ListOfSpeciesTypes`, `ListOfCompartments`, `ListOfSpecies`, `ListOfParameters`, `ListOfInitialAssignments`, `ListOfRules`, `ListOfConstraints`, `ListOfReactions`, and `ListOfEvents`. The "list" classes are defined in Figure 10. All of the lists are optional, but if a given list container is present within the model, the list must not be empty; that is, it must have length one or more. The resulting XML data object for a full model containing every possible list would have the following form:

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level2/version3" level="2" version="3">
  <model id="my_model">
    <listOfFunctionDefinitions>
      one or more <functionDefinition> ... </functionDefinition> elements } optional
    </listOfFunctionDefinitions>
    <listOfUnitDefinitions>
      one or more <unitDefinition> ... </unitDefinition> elements } optional
    </listOfUnitDefinitions>
    <listOfCompartmentTypes>
      one or more <compartmentType> ... </compartmentType> elements } optional
    </listOfCompartmentTypes>
    <listOfSpeciesTypes>
      one or more <speciesType> ... </speciesType> elements } optional
    </listOfSpeciesTypes>
    <listOfCompartments>
      one or more <compartment> ... </compartment> elements } optional
    </listOfCompartments>
    <listOfSpecies>
      one or more <species> ... </species> elements } optional
    </listOfSpecies>
    <listOfParameters>
      one or more <parameter> ... </parameter> elements } optional
    </listOfParameters>
    <listOfInitialAssignments>
      one or more <initialAssignment> ... </initialAssignment> elements } optional
    </listOfInitialAssignments>
    <listOfRules>
      one or more elements of subclasses of Rule } optional
    </listOfRules>
    <listOfConstraints>
      one or more <constraint> ... </constraint> elements } optional
    </listOfConstraints>
    <listOfReactions>
      one or more <reaction> ... </reaction> elements } optional
    </listOfReactions>
    <listOfEvents>
      one or more <event> ... </event> elements } optional
    </listOfEvents>
  </model>
</sbml>
```

Although all the lists are optional, there are dependencies between SBML components such that defining some components requires defining others. An example is that defining a species requires defining a compartment, and defining a reaction requires defining a species. The dependencies are explained throughout the text.

31

34 Modelica Language Specification 3.1

```
parameter Integer level=1;
Level1 component1 (J=J) if level==1 "Conditional component";
Level1 component2 if level==2, component3 if level==3;
equation
  connect (component1... ..) "Connection to conditional component";
  component1.u=0; // Illegal
]
```

The expression must be a Boolean scalar expression, and must be a parameter-expression [that can be evaluated at compile time].

If the Boolean expression is false the component is not present in the flattened DAE [its modifier is ignored], and connections to/from the component are removed. Other use of the component is illegal.

4.5 Class Declarations

Essentially everything in Modelica is a class, from the predefined classes `Integer` and `Real`, to large packages such as the Modelica standard library.

[Example: A rather typical structure of a Modelica class is shown below. A class with a name, containing a number of declarations followed by a number of equations in an equation section.]

```
class ClassName
  Declaration1
  Declaration2
  ...
  equation
    equation1
    equation2
    ...
end ClassName;
]
```

The following is the formal syntax of class definitions, including the special variants described in later sections.

```
class_definition :
  [ encapsulated ]
  [ partial ]
  ( class | model | record | block | [ expandable ] connector | type |
    package | function )
  class_specifier

class_specifier :
  IDENT string comment composition end IDENT
  | IDENT "=" base_prefix name [ array_subscripts ]
  [ class_modification ] comment
  | IDENT "=" enumeration "(" ( [ enum_list ] | "(" ) ")" comment
  | IDENT "=" dex "(" name "," IDENT [ "," IDENT ] ")" comment
  | extends IDENT [ class_modification ] string_comment composition
  end IDENT

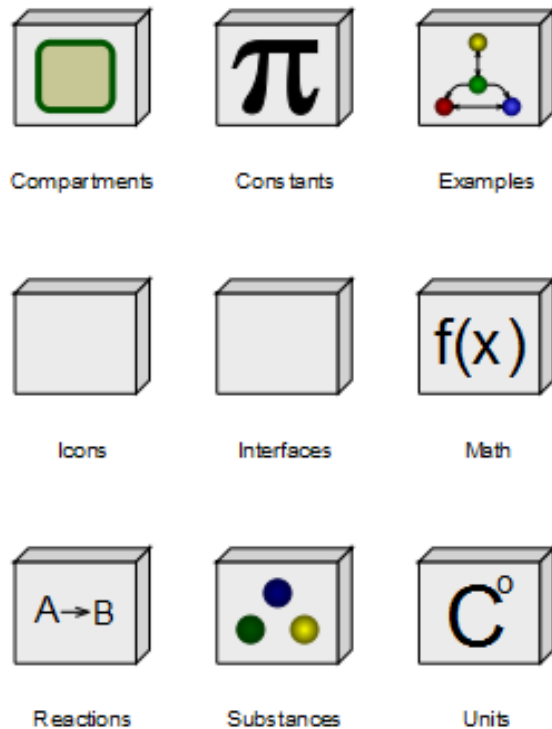
base_prefix :
  type_prefix

enum_list : enumeration_literal { "," enumeration_literal }

enumeration_literal : IDENT comment

composition :
  element_list
  { public_element_list |
  protected_element_list |
  equation_section }
```

Using the BioChem Library and MathModelica for Translation



- Finding a mapping between SBML and the more expressive and general Modelica language
- “Catch” Modelica constructs and map them
- Restrict how the Modelica models are built
 - BioChem library
 - Wizards introduced

Verification of the SBML-Modelica Translator Part of MathModelica

Models from BioModels database have been used to verify if models that are imported to *MathModelica* and exported from give the same result as the simulation published on the database.

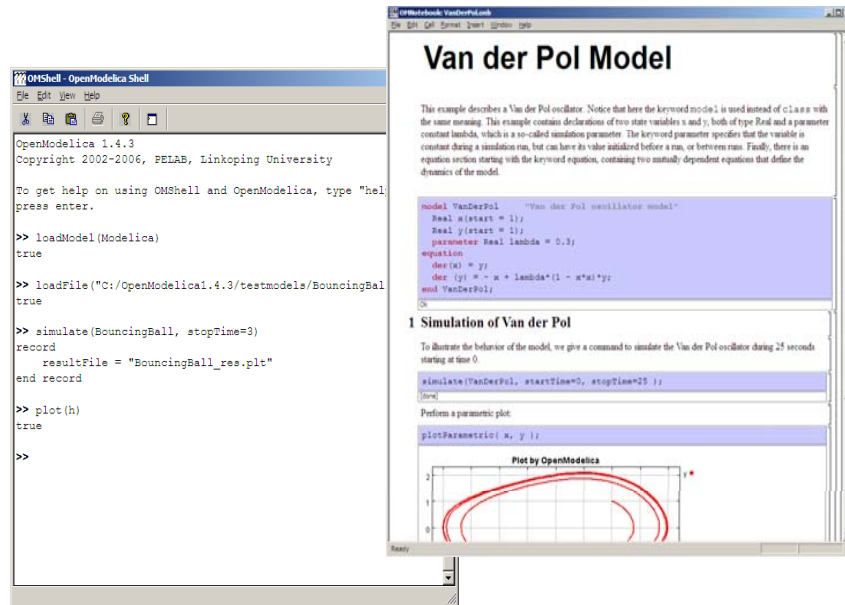
	Test models	Succeeded
Import	216	212 (98%)
Simulation	212	208 (98%)
Export	18	18 (100%)

Comparison With Other Tools

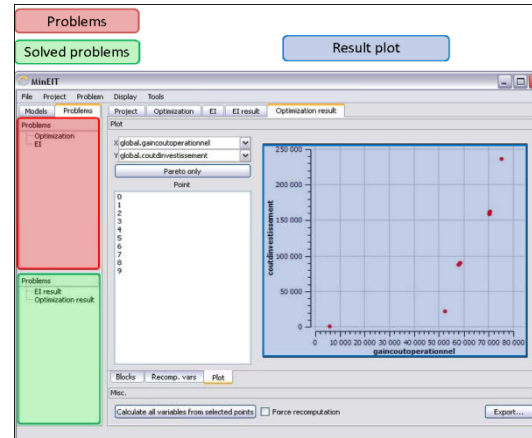
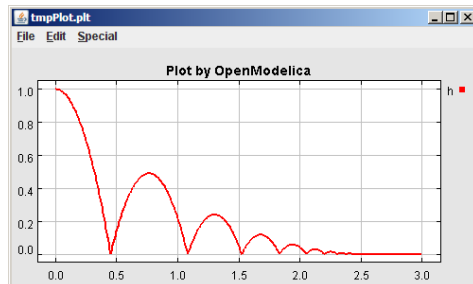
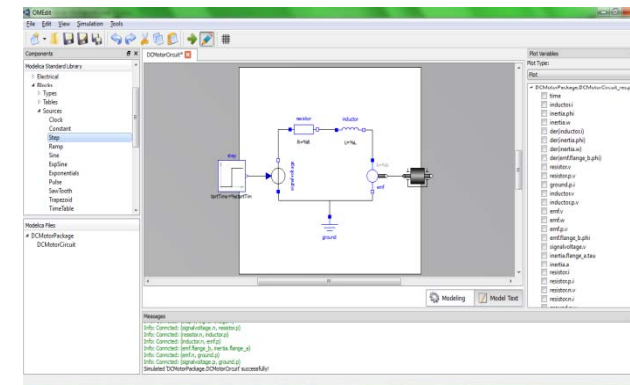
Models from BioModels database have been used to verify if different tools give the same result as the simulation published on the database.

	Test models
<i>MathModelica</i>	98% (42 of 43)

OpenModelica, www.openmodelica.org The Most Complete Open Source Modelica Tool



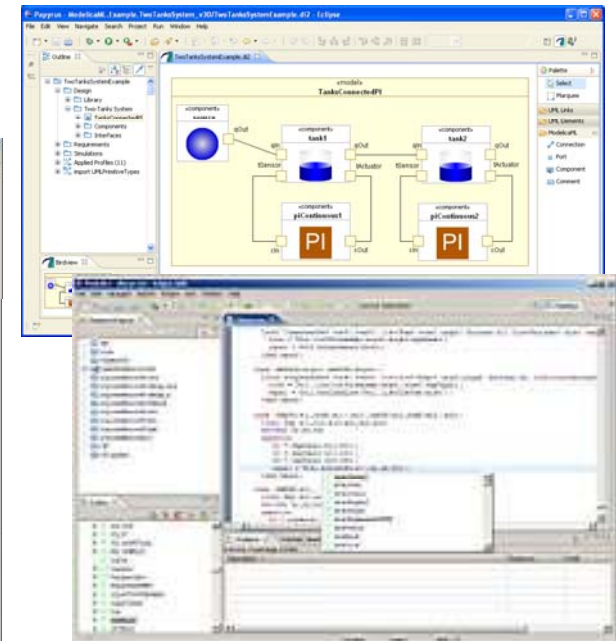
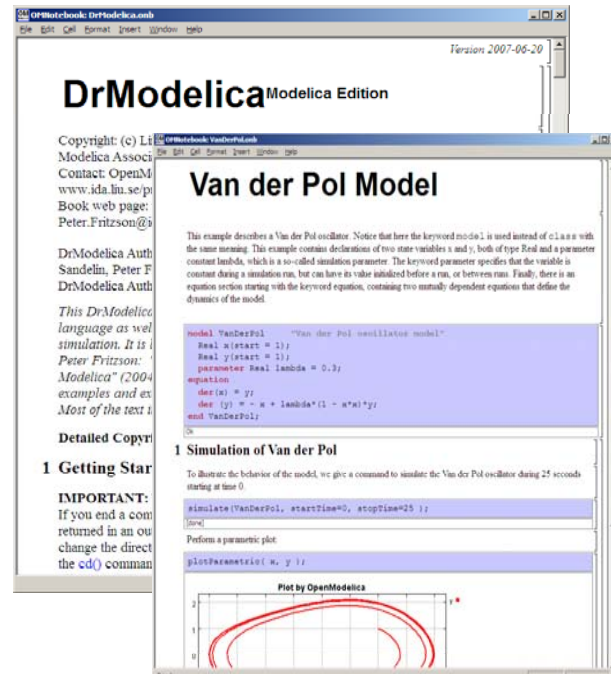
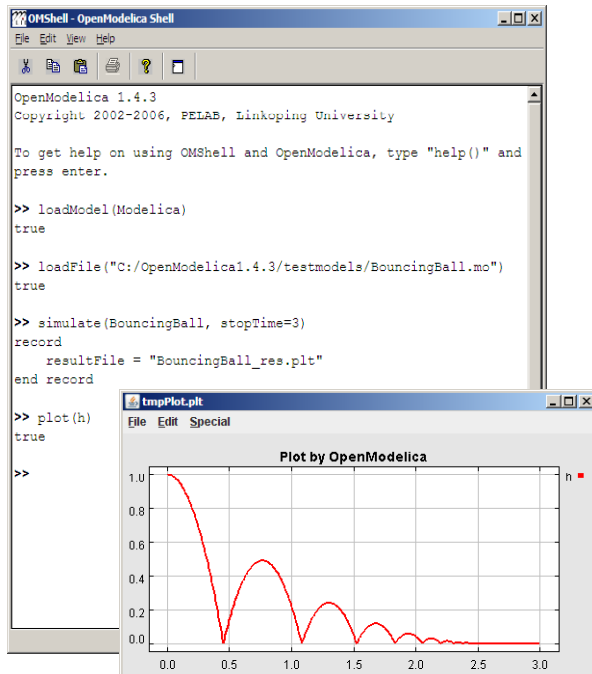
- OpenModelica
- Open Source Modelica Consortium (OSMC)
- International
- Open source
- www.openmodelica.org



- OMEdit, graphical editor
- OMOptim, optimization subsystem

OpenModelica (cont.)

- Advanced Interactive Modelica compiler (OMC)
 - Supports most of the Modelica Language
- Basic environment for creating models
 - **OMShell** – an interactive command handler
 - **OMNotebook** – a literate programming notebook
 - **MDT** – an advanced textual environment in Eclipse
- **ModelicaML** – UML Profile
- **MetaModelica** – symbolic manipulation



OSMC – Open Source Modelica Consortium

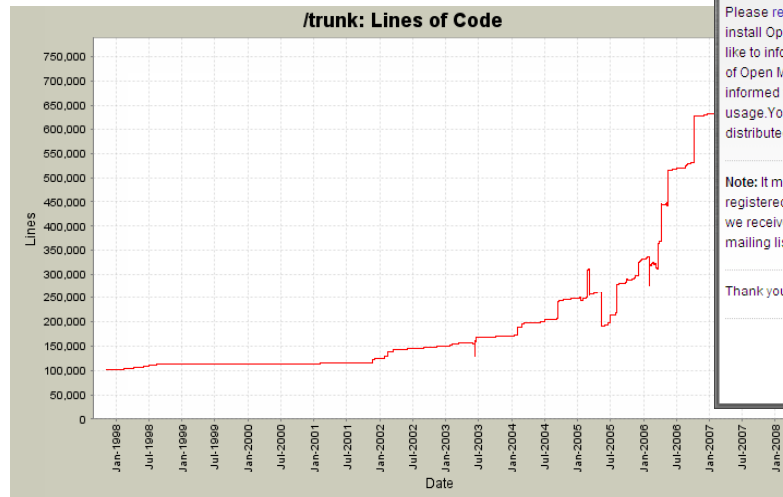
38 organizational members November 2011

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



Welcome to OpenModelica

http://www.openmodelica.org/

OpenModelica

HOME DEVELOPER FORUM DOWNLOAD CONTACT US WORKSHOP RESEARCH

search...

Top information

New OpenModelica website is up.
The new OpenModelica website is up and running.

Registration

Please register if you download and install Open Modelica. Why? We would like to inform you about new releases of Open Modelica! We want be informed who is using it and the kind of usage. Your information will not be distributed to third parties!

Note: It may take a while to be registered as we check the information we receive to fight the spam on our mailing lists.

Thank you for your patience.

Introduction

Tuesday, 15 December 2009 08:58

OPENMODELICA IS AN OPEN-SOURCE Modelica-based modeling and simulation environment intended for industrial and academic usage. Its long-term development is supported by a non-profit organization – the Open Source Modelica Consortium (OSMC).

The goal with the OpenModelica effort is to create a complete Open Source Modelica modeling, compilation and simulation environment based on free software distributed in binary and source code form. We invite researchers and students, or any interested developer to participate in the project.

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®.

Latest news

- Feb 5: OpenModelica Release 1.5.0 RC2
- Jan 28: OMScheme release available for download
- Dec 14: OpenModelica Release 1.5.0 RC1
- Dec 14: Open Master Theses
- Dec 14: Open Positions

Upcoming Events

- OpenModelica Workshop 2010

OSMC 38 Organizational Members, Nov 2011 (initially 7 members, 2007)

Companies and Institutes (17 members)

- ABB Corporate Research, Sweden
- Bosch Rexroth AG, Germany
- Siemens Turbo Machinery AB, Sweden
- CDAC Centre for Advanced Computing, Kerala, India
- CEIT Institute, Spain
- Creative Connections, Prague, Czech Republic
- Fraunhofer FIRST, Berlin, Germany
- Frontway AB, Sweden
- Equa Simulation AB, Sweden
- Evonik Energy Services, Dehli, India
- IFP, Paris, France
- InterCAX, Atlanta, USA
- Wolfram/ MathCore USA, Sweden
- Maplesoft, Canada
- TLK Thermo, Germany
- VI-grade, Italy
- VTT, Finland
- XRG Simulation, Germany

Universities (17 members)

- Linköping University, Sweden
- Hamburg University of Technology/TuTech, Germany
- Technical University of Berlin, Germany
- FH Bielefeld, Bielefeld, Germany
- Technical University of Braunschweig, Institute of Thermodynamics, Germany
- Technical University of Dortmund, Process Dynamics Group, Germany
- Université Laval, modelEAU, Canada
- University of Maryland, USA
- Georgia Tech, Atlanta, USA
- Griffith University, Australia
- Politecnico di Milano, Italy
- Mälardalen University, Sweden
- Technical University Dresden, Germany
- Telemark University College, Norway
- Ghent University, Belgium
- Ecoles des Mines, CEP, Paris, France
- University of Ljubljana, Slovenia

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. 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 often contain noise.

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

Here e denoting a disturbance in the input signal and v denoting a disturbance in the measurement. The error e can be evaluated by the difference between the measured quantity and the estimated quantity:

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

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 $e = y - C\hat{x}$.

The differential error is $\dot{e} = \dot{y} - C\dot{\hat{x}}$.

OMNotebook: Kalman.onb

File Edit Cell Format Insert Window Help

1 Kalman Filter

```

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
          
```

OpenModelica Demo

OMOptim – Optimization (1)

Model structure

Model Variables

Optimized parameters

Optimized Objectives

The screenshot displays the MinEIT software interface with the following components:

- Model Structure (Left Panel):** A list of model components including Pc, Va, Vb, Ia, Ib, Ic, Ea, Eb, Ec, coutinvestissement, gaincoutoperationnel, EmCO2PAC1, Ca, Cb, Cc, Puissae, Puissbe, Puissce, n, na, nb, nc, OCb, OChp, coutdefonctavecPAC, TOSygmaA, TOSygmaB, TOSygmaECS, COPECSsystem, PElecECSMax, EchIAOutCold, Sortieeffluents, echA, Sourcemod, scenarioEchA, scenarioPACA, and echB.
- Variables (Center Panel):** A table listing model variables with their values and descriptions. The table is highlighted with a green border.
- Optimized Variables (Right Panel):** A table showing optimized variables with their descriptions and optimal minimum values. The table is highlighted with a blue border.
- Scanned Variables (Right Panel):** A table for scanned variables with columns for Name, Description, Scan Minimum, and Scan Maximum. It is currently empty.
- Optimization Objectives (Right Panel):** A table showing optimization objectives with their descriptions, directions (Maximize/Minimize), and values. The table is highlighted with a yellow border.

Name	Value	Description
global.sourceeadeville.h	1,18294e+06	[J/kg]
global.sourceeadeville.flowPort.p	100000	
global.sourceInEchColdB.h	1,41347e+06	[J/kg]
global.sourceInEchColdB.flowPort.p	100000	
global.sourceInEchColdB.debit	12,78	[kg/s]
global.sourceEffluentsECS.h	1,35495e+06	[J/kg]
global.sourceEffluentsECS.flowPort.p	100000	
global.sourceEffluentsECS.etat	1	
global.sourceEffluentsECS.debit1	0	
global.sourceEffluentsECS.debit	1	[kg/s]
global.sourceEffluentsB.h	1,35495e+06	[J/kg]
global.sourceEffluentsB.flowPort.p	100000	
global.sourceEffluentsB.etat	1	
global.sourceEffluentsB.debit	1,22612	[kg/s]
global.sourceEffluentsA.h	1,35495e+06	[J/kg]
global.sourceEffluentsA.flowPort.p	100000	
global.sourceEffluentsA.etat	1	
global.sourceEffluentsA.debit	0,601234	[kg/s]
global.scenariosourceEadeville.debit	0,940001	[kg/s]
global.scenariodepartB.z	0	

Name	Description	Opt. Minimum
global.sourceEffluentsB.debit	[kg/s]	0
global.sourceEffluentsA.debit	[kg/s]	0
global.scenarioPACB.MySpecPcomp		0
global.scenarioPACA.MvSpecPcomp		0

Name	Description	Direction	M
global.gaincoutoperationnel		Maximize	0
global.coutinvestissement		Minimize	0

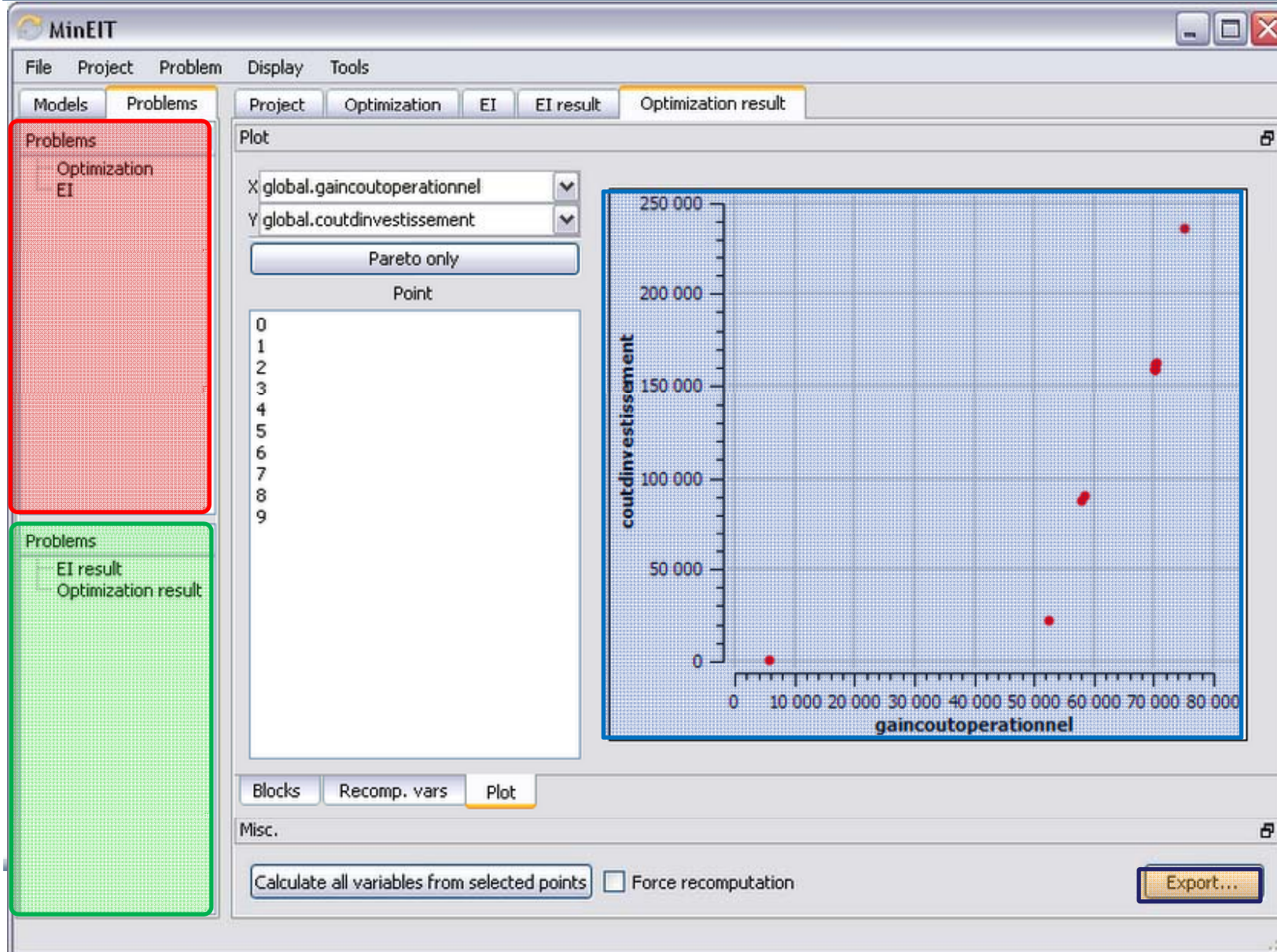
Problems

OMOptim – Optimization (2)

Solved problems

Result plot

Export result data .csv



Pareto Front

Modelica Language Interoperability

External Functions – C, FORTRAN 77

It is possible to call functions defined outside the Modelica language, implemented in C or FORTRAN 77

```
function polynomialMultiply
  input Real a[:], b[:];
  output Real c[:] := zeros(size(a,1)+size(b, 1) - 1);
  external
end polynomialMultiply;
```

The body of an external function is marked with the keyword **external**

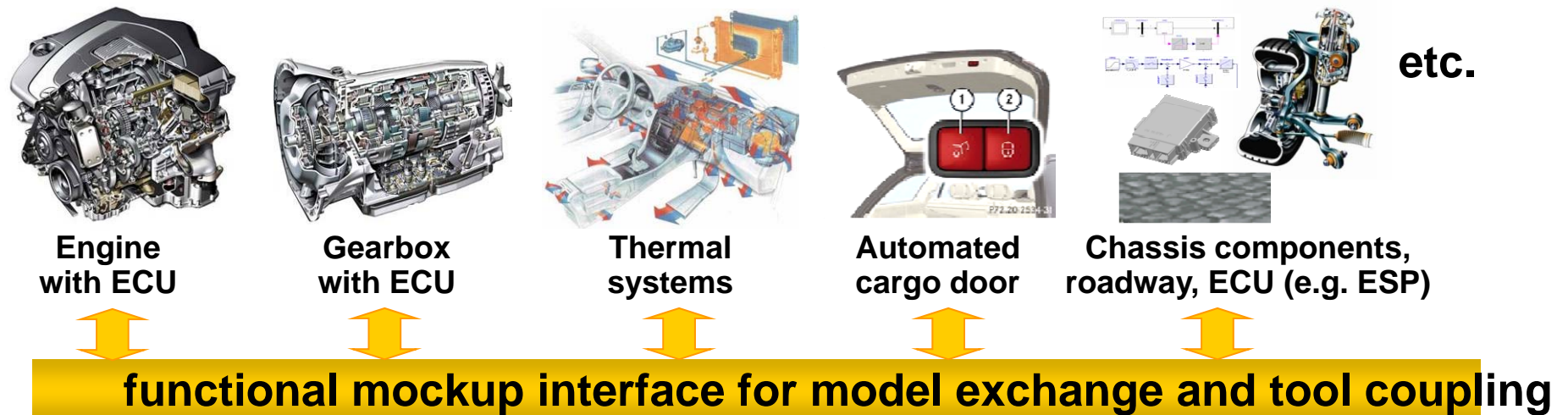
If no language is specified, the implementation language for the external function is assumed to be C. The external function `polynomialMultiply` can also be specified, e.g. via a mapping to a FORTRAN 77 function:

```
function polynomialMultiply
  input Real a[:], b[:];
  output Real c[:] := zeros(size(a,1)+size(b, 1) - 1);
  external "FORTRAN 77"
end polynomialMultiply;
```

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

The FMI development is part of the MODELISAR 29-partner project

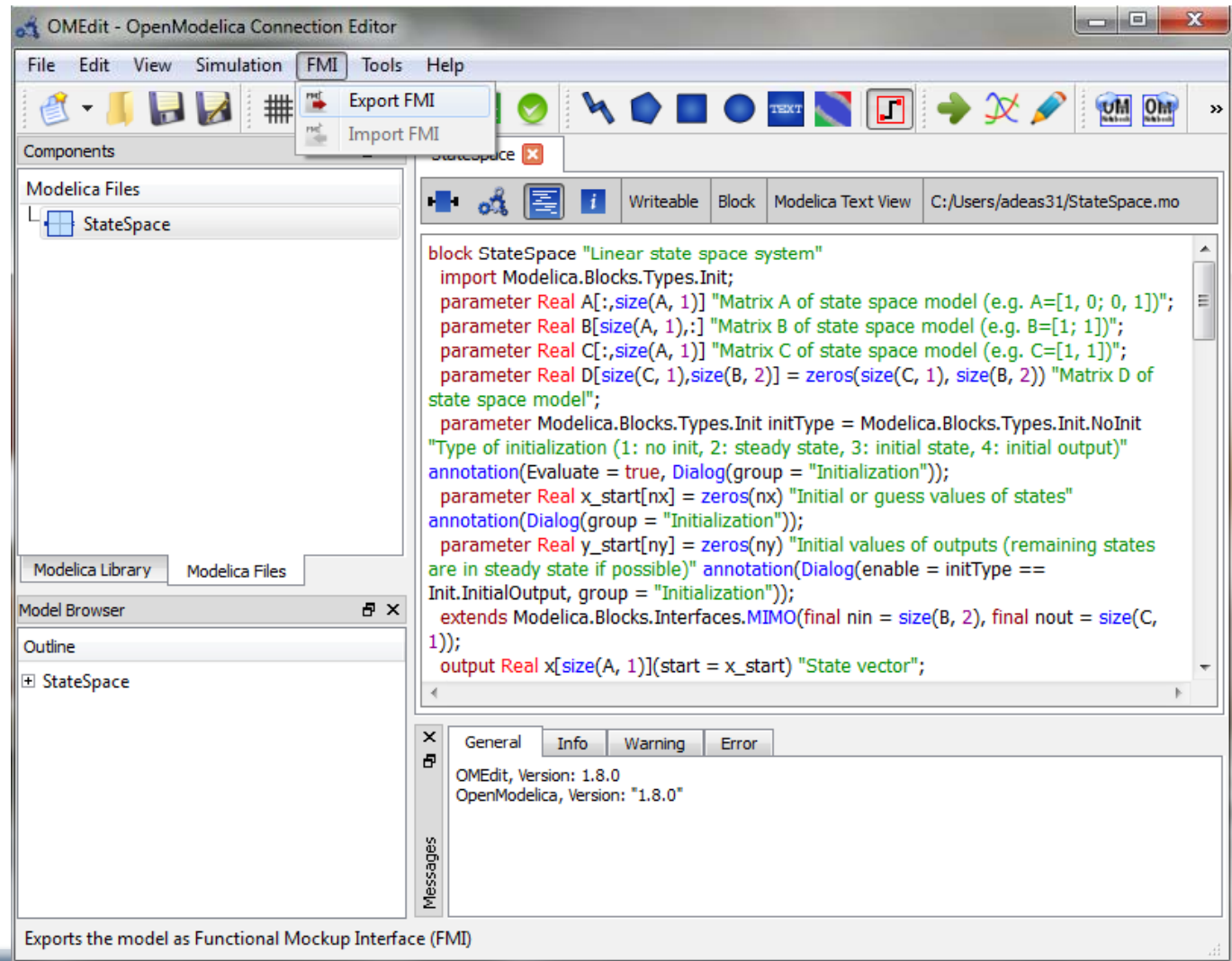
- FMI development initiated by **Daimler**
- Improved Software/Model/Hardware-in-the-Loop Simulation, of **physical** models and of **AUTOSAR** controller models from **different vendors** for automotive applications with **different levels of detail**.
- **Open Standard**
- **14 automotive use cases** for evaluation
- **> 10 tool vendors** are supporting it



courtesy Daimler

OpenModelica FMI Export and Import

- Export:
translateModel
FMU(A)
- importFMU("A.f
mu")



Faster Simulation – Compiling Modelica to Multi-Core

- **Automatic Parallelization of Mathematical Models**
 - Parallelism over the numeric solver method.
 - Parallelism over time.
 - **Parallelism over the model equation system**
 - ... with fine-grained task scheduling
- **Coarse-Grained Explicit Parallelization Using Components**
 - The programmer partitions the application into computational components using strongly-typed communication interfaces.
 - Co-Simulation, Transmission-Line Modeling (TLM)
- **Explicit Parallel Programming**
 - Providing general, easy-to-use explicit parallel programming constructs within the *algorithmic* part of the modeling language.
 - OpenCL, CUDA, ...

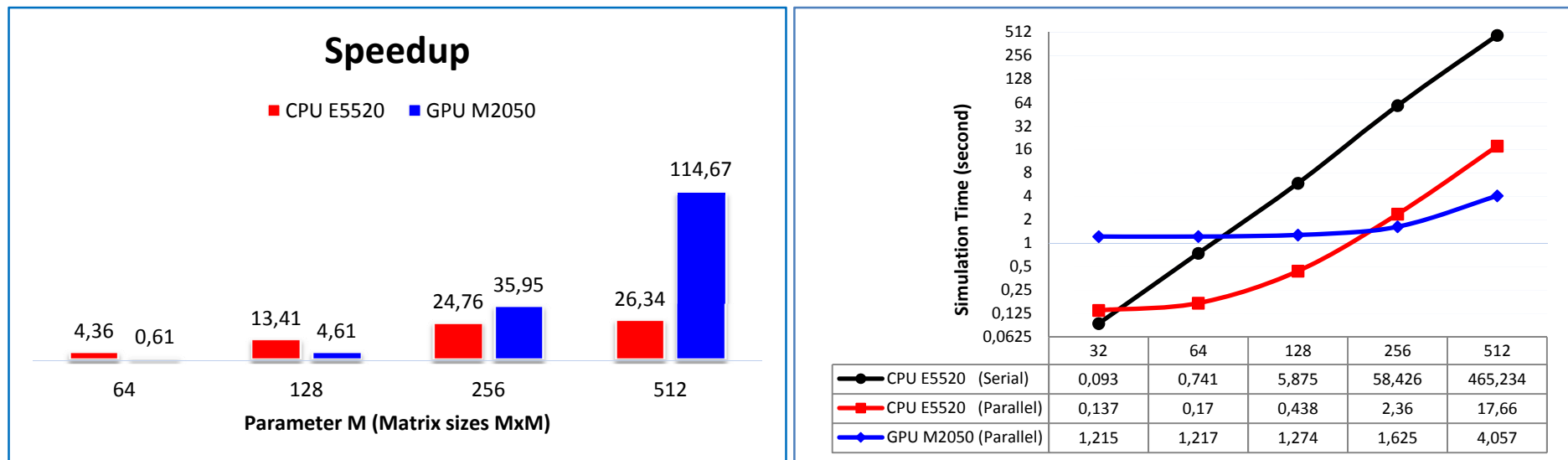
ParModelica – Parallel Modelica Extension

Matrix Multiplication using *Kernel function*

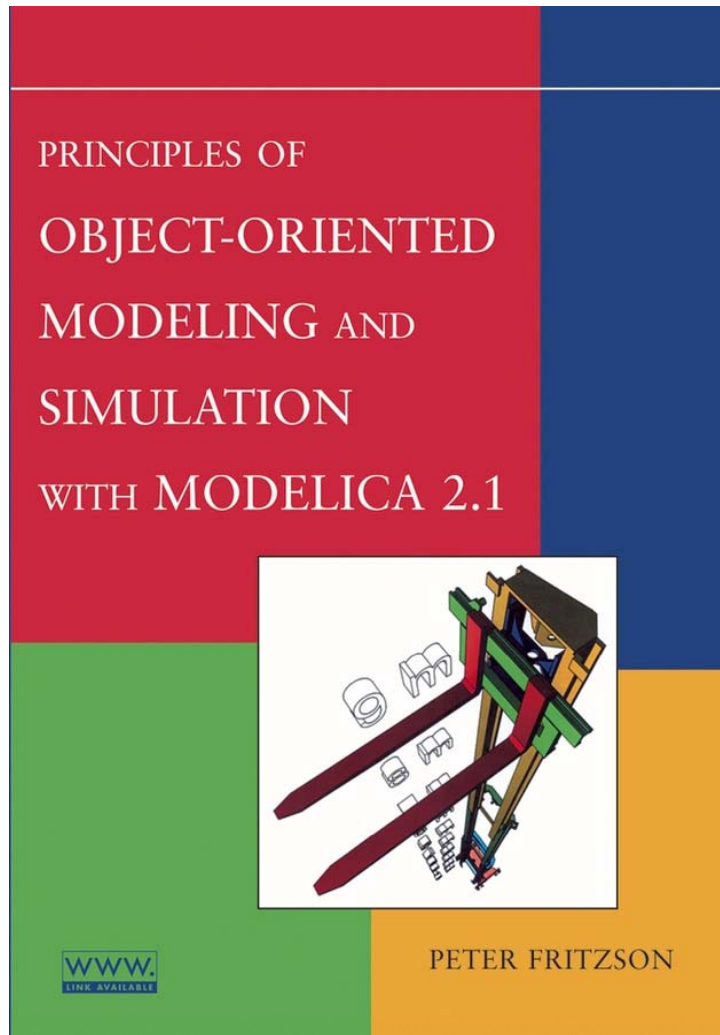
Gained speedup

- Intel Xeon E5520 CPU (16 cores) **26**
- NVIDIA Fermi-Tesla M2050 GPU (448 cores) **115**

Speedup comparison to sequential algorithm on Intel Xeon E5520 CPU



Get More Information, Download Software



Peter Fritzson Principles of Object Oriented Modeling and Simulation with Modelica 2.1

Wiley-IEEE Press, 2004, 940 pages

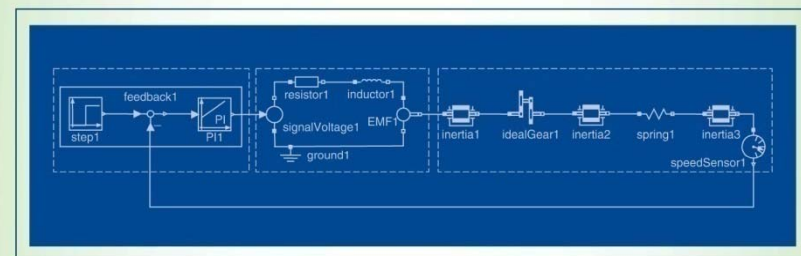
- OpenModelica
 - www.openmodelica.org
- Modelica Association
 - www.modelica.org

New Introductory Book
September 2011
232 pages

Wiley
IEEE Press

For Introductory
Short Courses on
Object Oriented
Mathematical Modeling

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



PETER FRITZSON

Announcements, Coming Workshops

- **Call for Presentations**
- **OpenModelica Workshop**
 - Feb 6, 2012. www.openmodelica.org, Linköping, Sweden
Applications and tool developments in the OpenModelica Open Source Effort.
- **MODPROD Workshop on Model-Based Development**
 - Feb 7-8, 2012, www.modprod.liu.se, Linköping, Sweden

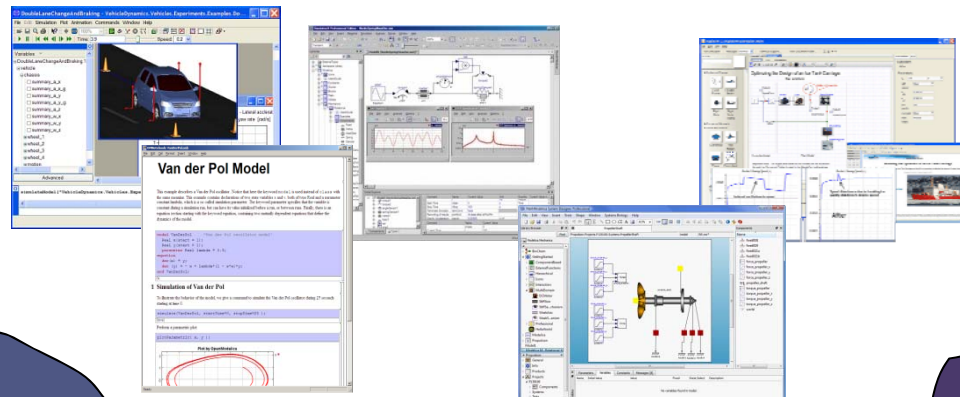
Summary

Multi-Domain
Modeling



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