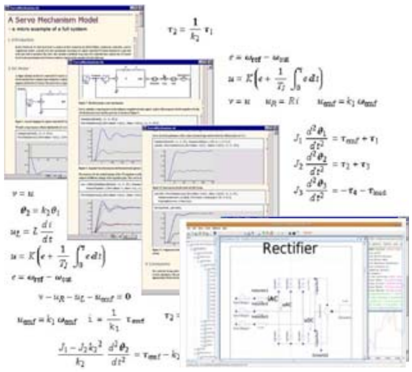
Principles of Object-Oriented Modeling and Simulation with Modelica



Peter Fritzson
Linköping University, petfr@ida.liu.se

Mohsen Torabzadeh-Tari
Linköping University, mohto@ida.liu.se

Martin Sjölund
Linköping University, marsj@ida.liu.se

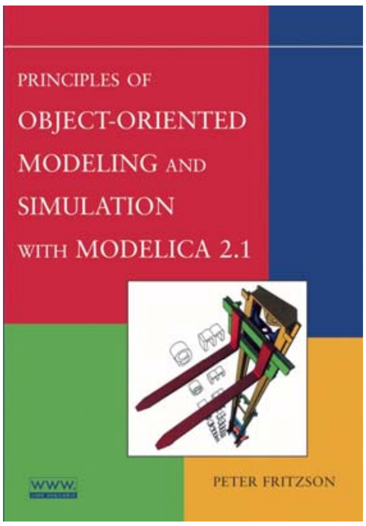
Slides
Based on book and lecture notes by Peter Fritzson
Contributions 2004-2005 by Emma Larsdotter Nilsson, Peter Bunus
Contributions 2007-2008 by Adrian Pop, Peter Fritzson
Contributions 2009 by David Broman, Jan Brugård, Mohsen Torabzadeh-Tari, Peter Fritzson
Contributions 2010 by Mohsen Torabzadeh-Tari, Peter Fritzson

Linköpings universitet

2010-10-13 Course at Linköping University

MODELICA

Course Based on Book, 2004



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

Wiley-IEEE Press

940 pages

MODELICA pelab

Outline Day 1

Part I

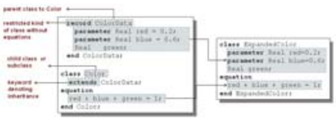
Introduction to Modelica and a demo example

Part II

Modelica environments

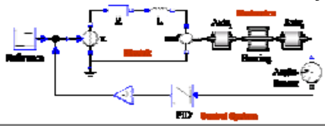
Part III

Modelica language concepts and textual modeling



Part IV

Graphical modeling and the Modelica standard library



Acknowledgements, Usage, Copyrights

- If you want to use the Powerpoint version of these slides in your own course, send an email to: peter.fritzson@liu.se
- Thanks to Emma Larsdotter Nilsson for contributions to the layout of these slides, to Peter Bunus, Adrian Pop, David Broman, Jan Brugård, Mohsen Torabzadeh-Tari for contributions.
- Most examples, figures and much text in this course are adapted with permission from Peter Fritzson’s book ”Principles of Object Oriented Modeling and Simulation with Modelica 2.1”, copyright Wiley-IEEE Press
- Some examples and figures reproduced with permission from Modelica Association, Martin Otter, Hilding Elmqvist, and MathCore
- Modelica Association: www.modelica.org
- OpenModelica: www.openmodelica.org

Software Installation

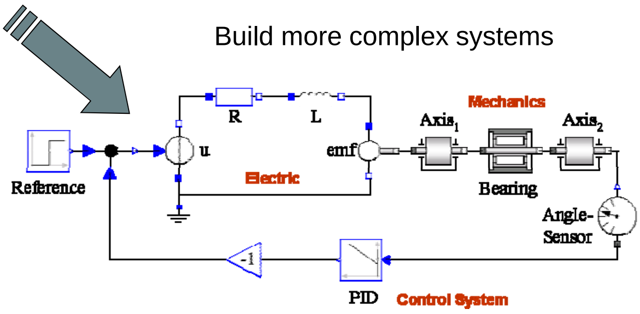
- Start the software installation
- Install OpenModelica-1.5.msi and simForge (e.g. SimForge-0.9.RC2.jar) from the USB Stick
- (If you have a Mac or Linux computer, install OpenModelica-1.5.0)

Outline

- Introduction to Modeling and Simulation
- Modelica - The next generation modeling and Simulation Language
- Modeling and Simulation Environments and OpenModelica
- Classes
- Components, Connectors and Connections
- Equations
- Discrete Events and Hybrid Systems
- Algorithms and Functions
- Demonstrations

Why Modeling & Simulation ?

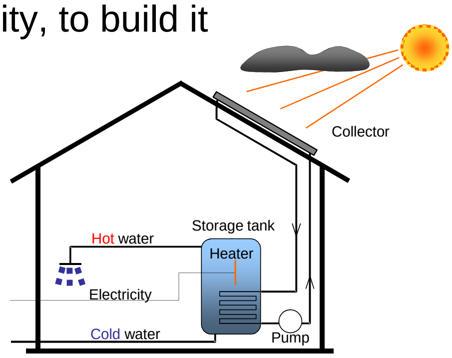
- Increase understanding of complex systems
- Design and optimization
- Virtual prototyping
- Verification



 Peter Fritzson MODELICA pelab

What is a system?

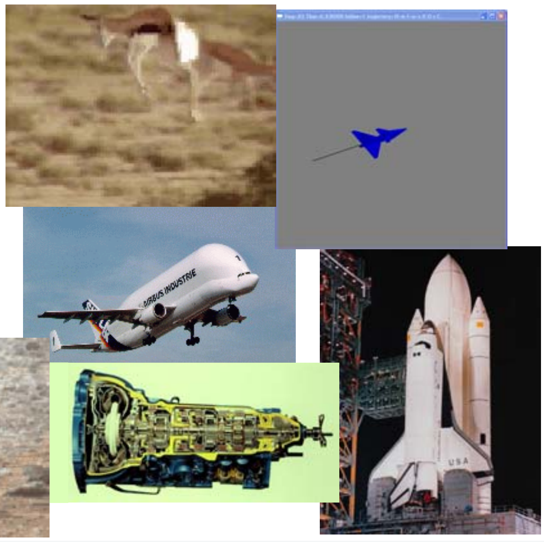
- A system is an object or collection of objects whose properties we want to study
- Natural and artificial systems
- Reasons to study: curiosity, to build it



 Peter Fritzson MODELICA pelab

Examples of Complex Systems

- Robotics
- Automotive
- Aircrafts
- Satellites
- Biomechanics
- Power plants
- Hardware-in-the-loop, real-time simulation



pelab

Experiments

An *experiment* is the process of extracting information from a system by exercising its inputs

Problems

- Experiment might be too *expensive*
- Experiment might be too *dangerous*
- System needed for the experiment might *not yet exist*

pelab

Model concept

A *model* of a system is anything an *experiment* can be applied to in order to answer questions about that *system*

Kinds of models:

- **Mental model** – statement like "a person is reliable"
- **Verbal model** – model expressed in words
- **Physical model** – a physical object that mimics the system
- **Mathematical model** – a description of a system where the relationships are expressed in mathematical form – a *virtual prototype*
- **Physical modeling** – also used for mathematical models built/structured in the same way as physical models

Simulation

A *simulation* is an *experiment* performed on a *model*

Examples of simulations:

- **Industrial process** – such as steel or pulp manufacturing, study the behaviour under different operating conditions in order to improve the process
- **Vehicle behaviour** – e.g. of a car or an airplane, for operator training
- **Packet switched computer network** – study behaviour under different loads to improve performance

Reasons for Simulation

- Suppression of *second-order effects*
- Experiments are too *expensive*, too *dangerous*, or the system to be investigated does *not* yet *exist*
- The *time scale* is not compatible with experimenter (Universe, million years, …)
- Variables may be *inaccessible*.
- Easy *manipulation* of models
- Suppression of *disturbances*

Dangers of Simulation

Falling in love with a model

The Pygmalion effect (forgetting that model is not the real world, e.g. introduction of foxes to hunt rabbits in Australia)

Forcing reality into the constraints of a model

The Procrustes effect (e.g. economic theories)

Forgetting the model's level of accuracy

Simplifying assumptions

Building Models Based on Knowledge

System knowledge

- The collected *general experience* in relevant domains
- The *system* itself

Specific or generic knowledge

- E.g. software engineering knowledge

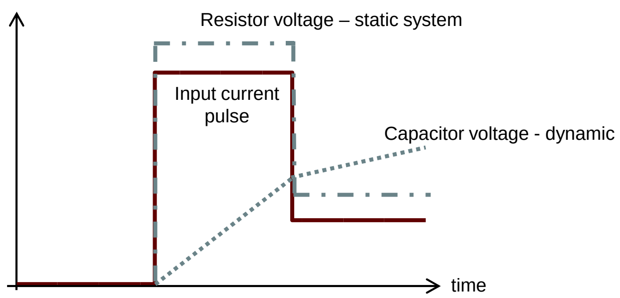
Kinds of Mathematical Models

- Dynamic *vs.* Static models
- Continuous-time *vs.* Discrete-time dynamic models
- Quantitative *vs.* Qualitative models

Dynamic vs. Static Models

A **dynamic** model includes *time* in the model

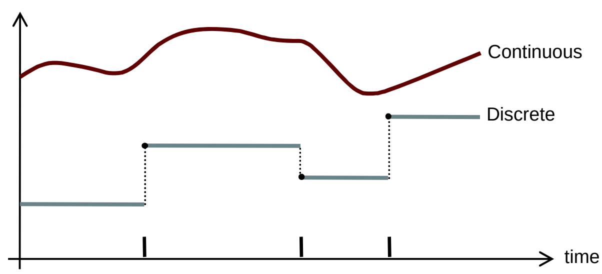
A **static** model can be defined *without* involving *time*



Continuous-Time vs. Discrete-Time Dynamic Models

Continuous-time models may evolve their variable values *continuously* during a time period

Discrete-time variables change values a *finite* number of times during a time period

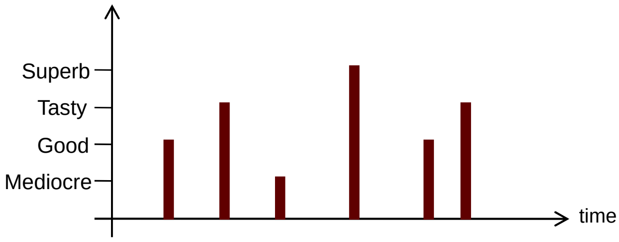


Quantitative vs. Qualitative Models

Results in qualitative data

Variable values cannot be represented numerically

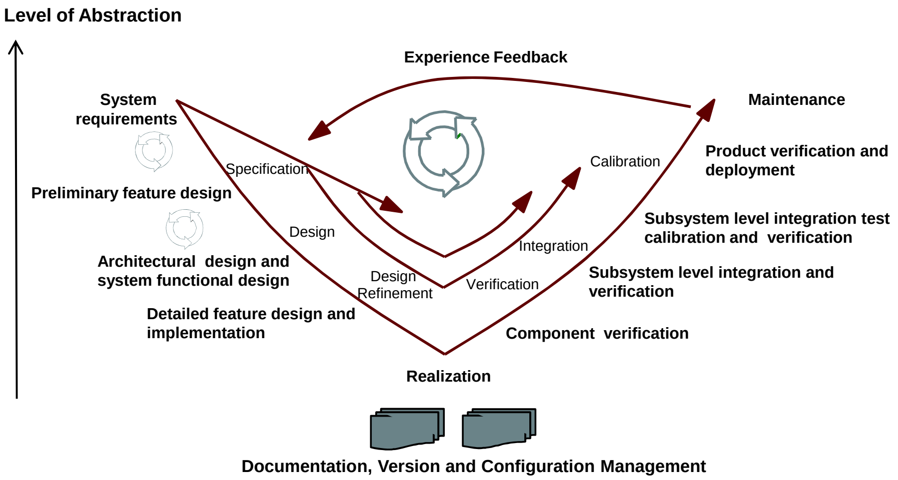
Mediocre = 1, Good = 2, Tasty = 3, Superb = 4



Quality of food in a restaurant according to inspections at irregular points in time

  pelab

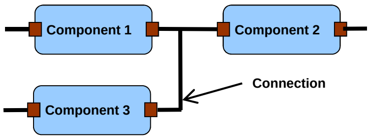
Using Modeling and Simulation within the Product Design-V



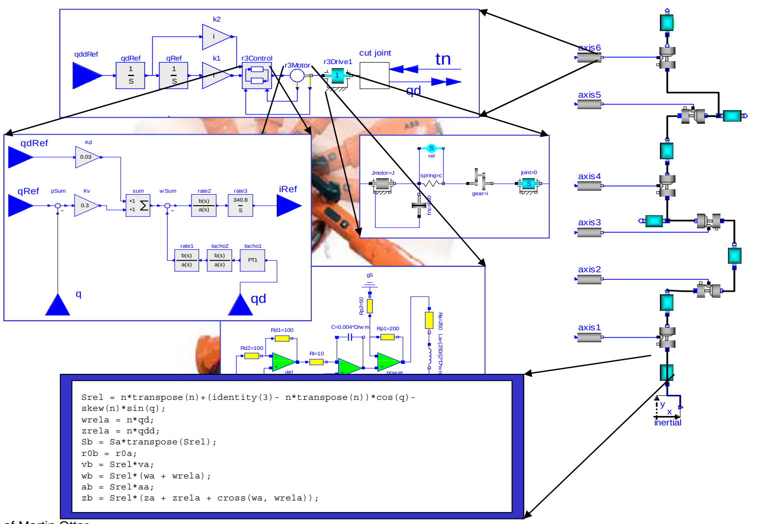
  pelab

Principles of Graphical Equation-Based Modeling

- Each icon represents a physical component i.e. Resistor, mechanical Gear Box, Pump
- Composition lines represent the actual physical connections i.e. electrical line, mechanical connection, heat flow
- Variables at the interfaces describe interaction with other component
- Physical behavior of a component is described by equations
- Hierarchical decomposition of components

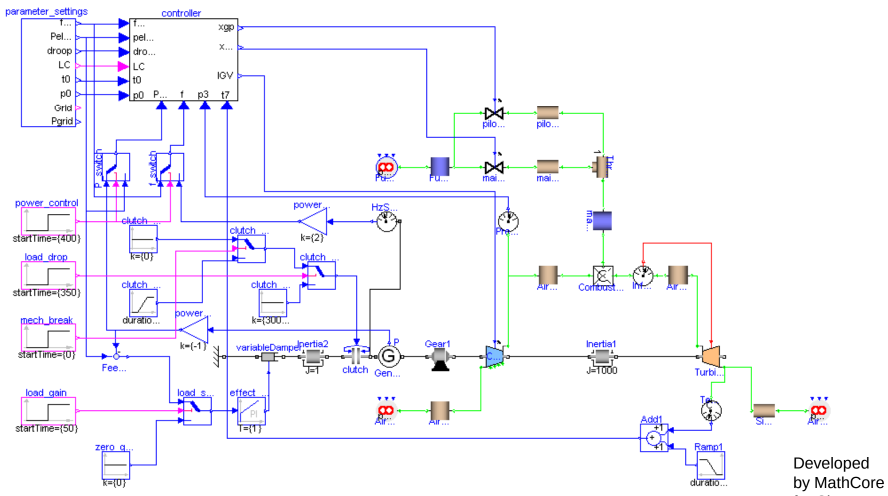


Application Example – Industry Robot



Courtesy of Martin Otter

GTX Gas Turbine Power Cutoff Mechanism

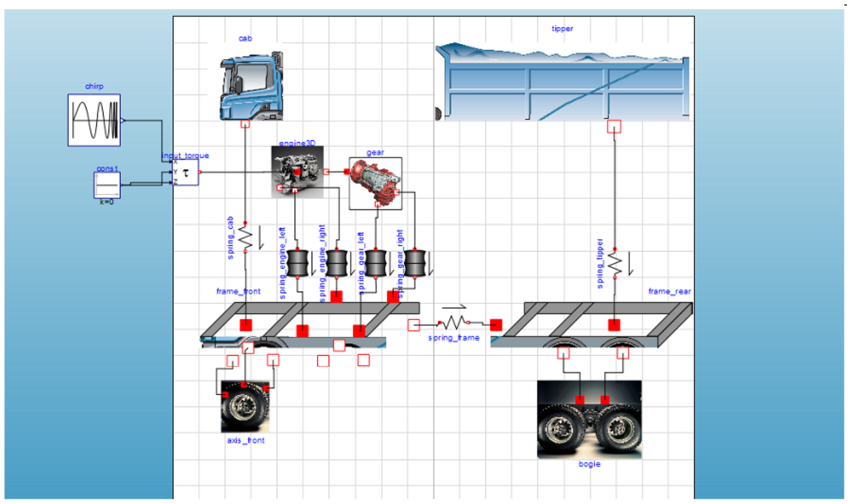


Developed by MathCore for Siemens

Courtesy of Siemens Industrial Turbomachinery AB

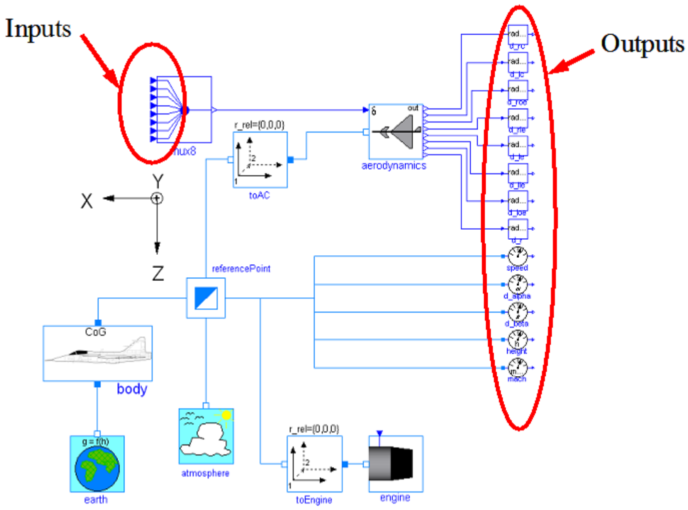
MathCore MODELICA pelab

Modelica in Automotive Industry

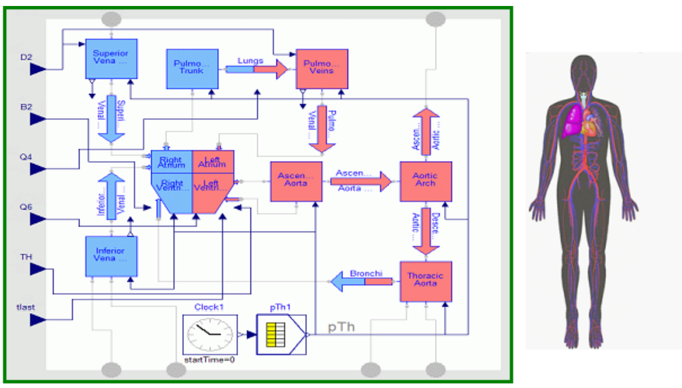


MODELICA pelab

Modelica in Avionics



Modelica in Biomechanics

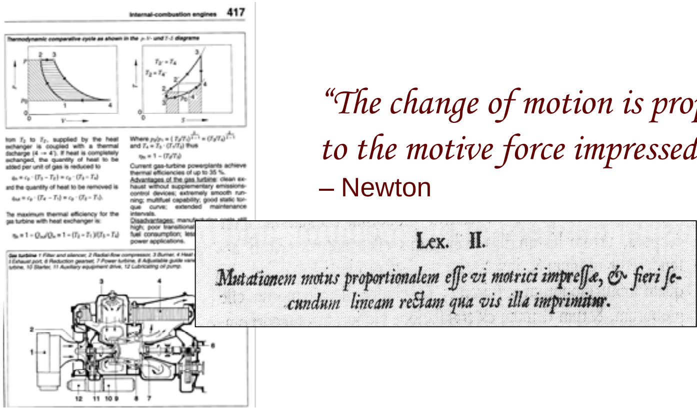


Modelica –
The Next Generation
Modeling Language

Stored Knowledge

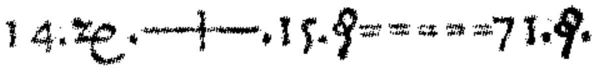
Model knowledge is stored in books and human minds which computers cannot access

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



The Form – Equations

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



Newton still wrote text (Principia, vol. 1, 1686)
"The change of motion is proportional to the motive force impressed "

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

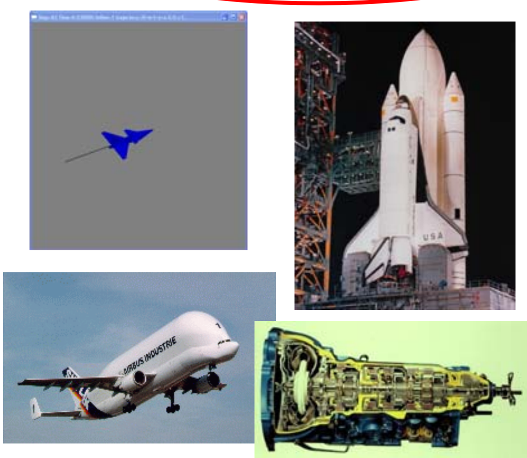
```
variable = expression
v = INTEG(F)/m
```

Programming languages usually do not allow equations!

What is Modelica?

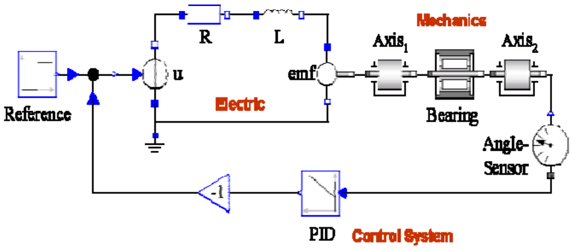
A language for modeling of complex physical systems

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



What is Modelica?

A language for modeling of complex 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 physical systems

i.e., Modelica is <u>not</u> a tool

Free, open language specification:

Available at: www.modelica.org

There exist several free and commercial tools, for example:

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

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 – The Next Generation Modeling Language

High level language

MATLAB-style array operations; Functional style; iterators, constructors, object orientation, equations, etc.

MATLAB similarities

MATLAB-like array and scalar arithmetic, but strongly typed and efficiency comparable to C.

Non-Proprietary

- Open Language Standard
- Both Open-Source and Commercial implementations

Flexible and powerful external function facility

- LAPACK interface effort started

Modelica Language Properties

- **Declarative** and **Object-Oriented**
- **Equation-based**; continuous and discrete equations
- **Parallel** process modeling of real-time applications, according to synchronous data flow principle
- **Functions** with algorithms without global side-effects (but local data updates allowed)
- **Type system** inspired by Abadi/Cardelli
- **Everything is a class** – Real, Integer, models, functions, packages, parameterized classes....

Object Oriented Mathematical Modeling with Modelica

- The static *declarative structure* of a mathematical model is emphasized
- OO is primarily used as a *structuring concept*
- OO *is not* viewed as dynamic object creation and sending messages
- *Dynamic model* properties are expressed in a *declarative way* through equations.
- Acausal classes supports *better reuse of modeling and design knowledge* than traditional classes

Brief Modelica History

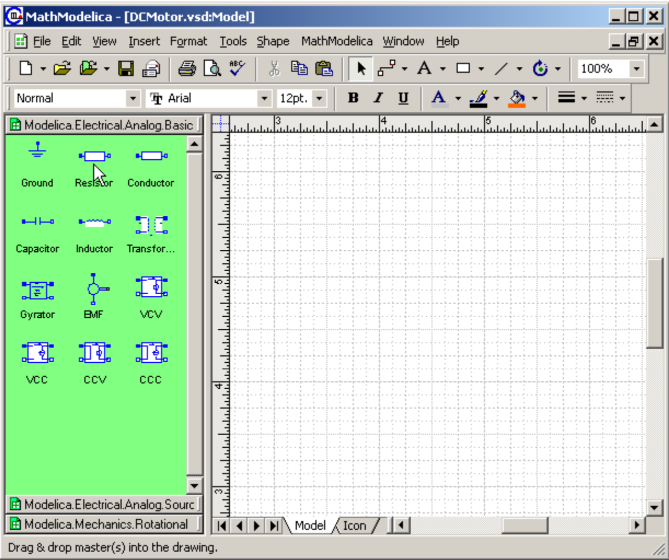
- First Modelica design group meeting in fall 1996
  - International group of people with expert knowledge in both language design and physical modeling
  - Industry and academia
- Modelica Versions
  - 1.0 released September 1997
  - 2.0 released March 2002
  - 2.2 released March 2005
  - 3.0 released September 2007
  - 3.1 released May 2009
- 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 in Como, Italy

Exercises Part I
Hands-on graphical modeling
(20 minutes)

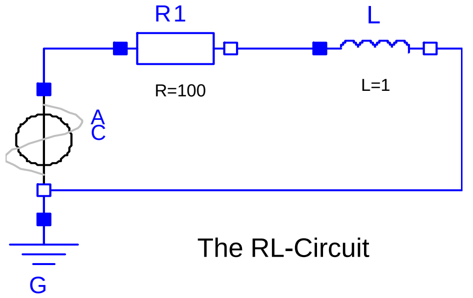
Graphical Modeling
Using Drag and Drop Composition



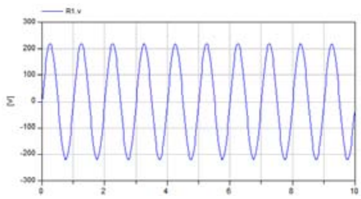
Courtesy
MathCore
Engineering AB

Exercises Part I – Basic Graphical Modeling

- (*See instructions on next two pages*)
- Start the simForge editor
- Draw the RL-Circuit
- Simulate



The RL-Circuit



Simulation

Exercises Part I – simForge Instructions Page 1

- Start simForge, (e.g. SimForge-0.9.RC2.jar).
- Go to **File** menu and choose **New Project**.
- Write *RL_Circuit* and click on the **Browse** button for choosing the destination folder.
- Press **OK.**
- In the navigation bar in the left, there should be three items, **Modelica, IEC61131-3** and **Simulation result**. Double-click on the **Modelica**.

• Under the **Modelica :**

- The standard Modelica library components are listed in the **Used external package.**
- The **Modelica classes** and **Modelica files** are the places where your models will end up under. The first folder is for the graphical models and the latter is for the texual form.

Exercises Part I – simForge Instructions Page 2

- Go to **File** menu and choose **New File.** Write *RL_circuit* and press **OK.**
- In the **Add Class** pop-up dialog box change the **Type** from **package** to **class** and press **OK.**
- Double click on the *RL_circuit* under the **Modelica classes** and the graphical window will appear.
- **Drag and Drop** components from the standard Modelica library to your model.
- For connecting components, move the cursor to the target pin and press shift+click ***once*** and just move the cursor with the mouse to the destination pin and press again shift+click.
- Start the simulation with **simulation** button.
- In the simulation pop-up you can leave out some fields like the **Stop time**, which will result in a default value of 1 sec. will be used.
- The result will appear under the **Simulation result.**

• Under the **Edit** menu -> **Advanced properties** you can tick the **visible legend** bar.

