Modelica - Language, Libraries, Tools, Workshop and EU-Project

by

Martin Otter and Hilding Elmqvist

DLR Oberpfaffenhofen, Germany and Dynasim AB, Lund, Sweden

Modelica is a new language for convenient modeling of physical systems. In this article an overview about the language features is given, the organisation behind the language development, available Modelica libraries and Modelica simulation environments, the recent, first workshop about Modelica and the EU-Project RealSim to enhance hardware-in-the-loop simulation and design optimization techniques on the basis of Modelica.

1. Modelica - An Overview

Modelica is a freely available, object-oriented language for modeling of large, complex, and heterogeneous physical systems. It is suited for multi-domain modeling, for example, mechatronic models in robotics, automotive and aerospace applications involving mechanical, electrical, hydraulic and control subsystems, process oriented applications and generation and distribution of electric power. Modelica is designed such that it can be utilized in a similiar way as an engineer builds a real system: First trying to find standard components like motors, pumps and valves from manufacturers' catalogues with appropriate specifications and interfaces and only if there does not exist a particular subsystem, a component model would be newly constructed based on standardized interfaces.

Models in Modelica are mathematically described by *differential, algebraic* and *discrete equations*. No particular variable needs to be solved for manually. A Modelica tool will have enough information to decide that automatically. Modelica is designed such that available, specialized algorithms can be utilized to enable efficient handling of large models having more than hundred thousand equations. Modelica is suited (and used) for hardware-in-the-loop simulations and for embedded control systems.

Reuse is a key issue for handling complexity. There have been several attempts to define object-oriented languages for physical modeling. However, the ability to reuse and exchange models relies on a standardized format. It was thus important to bring this expertise together to unify concepts and notations. The Modelica design effort was initiated by Hilding Elmqvist and started in September 1996 within an action of the ESPRIT project "Simulation in Europe Basic Research Working Group (SiE-WG)". The language has been designed by the developers of the object-oriented modeling languages Allan, Dymola, NMF, ObjectMath, Omola, SIDOPS+, Smile und a number of modeling practitioners in different domains. After 19 three-day meetings, during a 3-year period, version 1.3 of the language specification was finished in December 1999. This is the version currently used in actual applications. In December 2000, an update of the language, version 1.4, will be published. Detailed information about Modelica can be downloaded from:

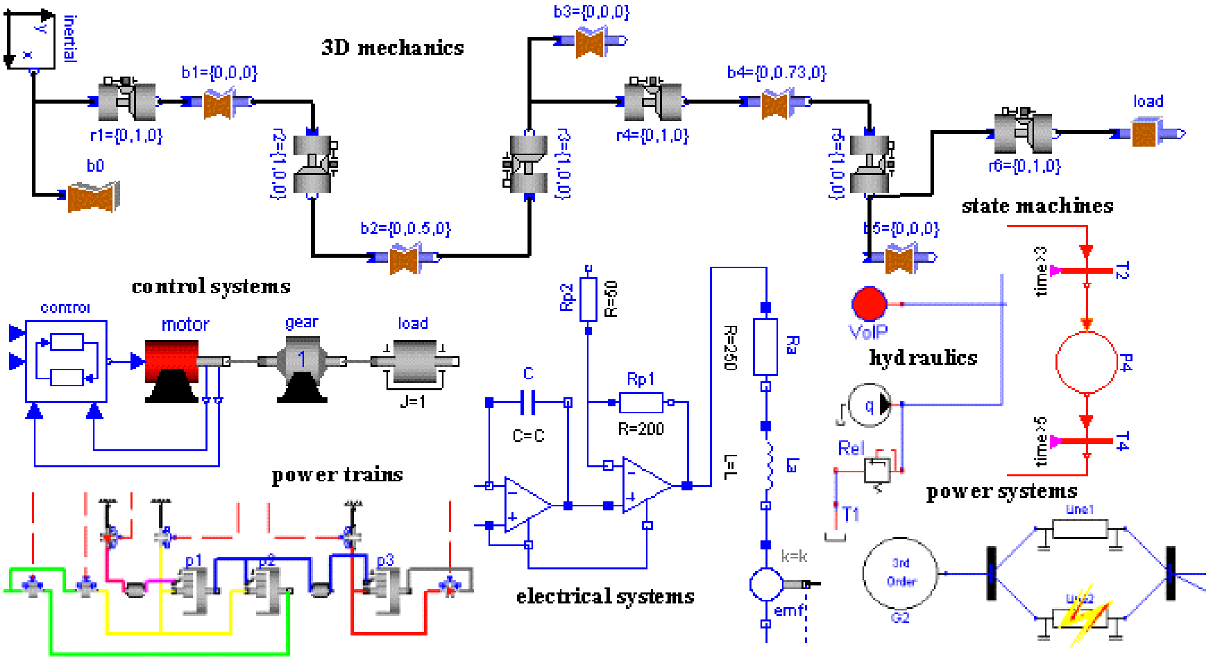
Modelica Homepage:	`http://www.Modelica.org/`
Modelica Tutorial:	`http://www.Modelica.org/current/ModelicaRationale13norev.pdf`
Modelica Specification:	`http://www.Modelica.org/current/ModelicaSpec13norev.pdf`
Modelica Libraries:	`http://www.Modelica.org/library/library.html`

In February 2000, a non-profit, non-governmental organization was founded for the further development, promotion and application of the Modelica language. The name of the organisation is "Modelica Association". The association has its seat in Linköping, Sweden and owns and administrates incorporeal rights related to Modelica, including but not limited to trademarks (such as the Modelica trademark), the Modelica Language Specification, Modelica Standard Libraries, etc., which should be freely available for the promotion of industrial development and research. Membership in the association is open to individual persons and to organizations The current board of the Modelica Association consists of the following persons:

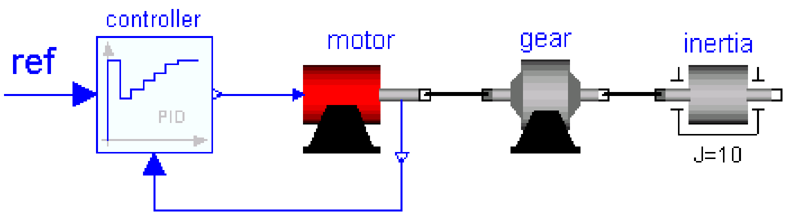
Chairman:	Martin Otter, DLR Oberpfaffenhofen, Germany
Vice-Chairman:	Peter Fritzson, Linköping University, Sweden
Secretary:	Hilding Elmqvist, Dynasim AB, Lund, Sweden (former Chairman)
Treasurer	Michael Tiller, Ford Motor Company, Dearborn, U.S.A.

2. Features of the Modelica Language

Modelica supports both high level modeling by composition and detailed library component modeling by equations. Models of standard components are typically available in model libraries. Using a graphical model editor, a model can be defined by drawing a composition diagram by positioning icons that represent the models of the components, drawing connections and giving parameter values in dialogue boxes. Constructs for including graphical annotations in Modelica make icons and composition diagrams portable between different tools. Typical composition diagrams in different domains are shown in the figure below.



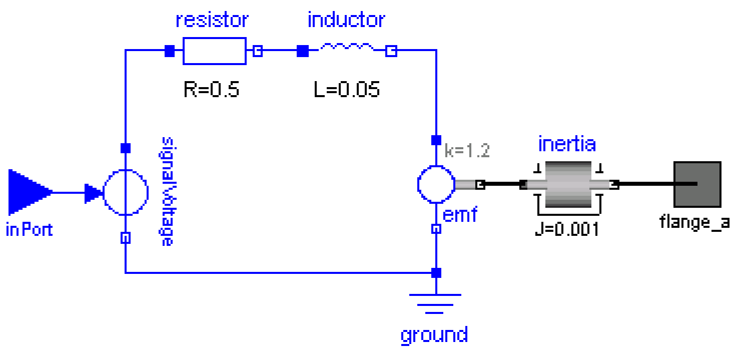
An example of a composition diagram of a simple motor drive system is shown in the figure below.



The system can be broken up into a set of connected components: an electrical motor, a gearbox, a load and a control system. The textual representation of this Modelica model is (graphical annotations are not shown):

```
model MotorDrive
  PID        controller;
  Motor      motor;
  Gearbox    gear    (n=100);
  Inertia    inertia(J=10);
equation
  connect(controller.outPort, motor.inPort);
  connect(controller.inPort2, motor.outPort);
  connect(gear.flange_a     , motor.flange_b);
  connect(gear.flange_b     , inertia.flange_a);
end MotorDrive;
```

It is a composite model which specifies the topology of the system to be modeled in terms of components and connections between the components. The statement "`Gearbox gear(n=100);`" declares a component `gear` of model class `Gearbox` and sets the value of the gear ratio, `n`, to `100`. A component model may be a composite model to support hierarchical modeling. The composition diagram of the model class `Motor` is shown in the next figure.



The meaning of connections will be discussed below as well as the description of behavior on the lowest level using mathematical equations.

Physical modeling deals with the specification of relations between physical quantities. For the drive system, quantities such as angle and torque are of interest. Their types are declared in Modelica as

```
type Angle  = Real(quantity="Angle" , unit="rad", displayUnit="deg");
type Torque = Real(quantity="Torque", unit="N.m");
```

where `Real` is a predefined type, which has a set of attributes such as name of quantity, unit of measure, default display unit for input and output, minimum, maximum, nominal and initial value. The Modelica Standard Library, which is an intrinsic part of Modelica, includes about 450 of such type definitions based on ISO 31-1992.

Connections specify interactions between components and are represented graphically as lines between connectors. A connector should contain all quantities needed to describe the interaction. Voltage and current are needed for electrical components. Angle and torque are needed for drive train elements:

```
connector Pin                connector Flange
   Voltage      v;              Angle        phi;
   flow Current i;              flow Torque tau;
end Pin;                     end Flange;
```

A connection, `connect(Pin1, Pin2)`, with `Pin1` and `Pin2` of connector class `Pin`, connects the two pins such that they form one node. This implies two equations, `Pin1.v = Pin2.v` and

`Pin1.i + Pin2.i = 0`. The first equation indicates that the voltages on both branches connected together are the same, and the second corresponds to Kirchhoff's current law saying that the current sums to zero at a node. Similar laws apply to mass flow rates in piping networks and to forces and torques in mechanical systems. The sum-to-zero equations are generated when the prefix **`flow`** is used in the connector declarations. The Modelica Standard Library includes also connector definitions.

An important feature in order to build reusable descriptions is to define and reuse **`partial`** models. A common property of many electrical components is that they have two pins. This means that it is useful to define an interface model class `OnePort`, that has two pins, `p` and `n`, and a quantity, `v`, that defines the voltage drop across the component.

```
partial model OnePort
   Pin     p, n;
   Voltage v;
equation
   v = p.v - n.v;
   0 = p.i + n.i;
end OnePort;
```

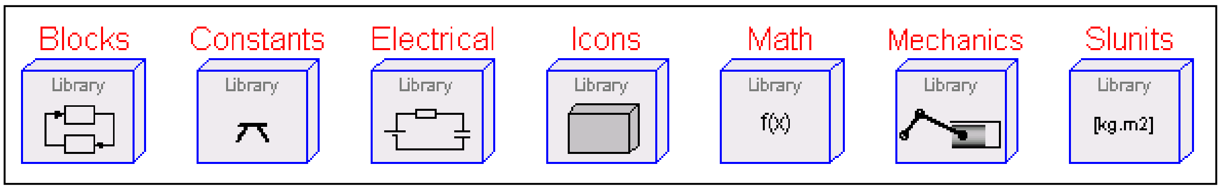
The equations define common relations between quantities of a simple electrical component. The keyword **partial** indicates that the model is incomplete and cannot be instantiated. To be useful, a constitutive equation must be added. A model for a resistor extends `OnePort` by adding a parameter for the resistance and Ohm's law to define the behavior.

```
model Resistor "Ideal resistor"
   extends OnePort;
   parameter Resistance R;
equation
     R*p.i = v;
end Resistor;
```

A string between the name of a class and its body is treated as a comment attribute. Tools may display this documentation in special ways. The keyword parameter specifies that the quantity is constant during a simulation experiment, but can change values between experiments.

The most basic Modelica language elements have been presented. Modelica additionally supports arrays, utilizing a Matlab like syntax. The elements of arrays may be of the basic data types (Real, Integer, Boolean, String) or in general component models. This allows convenient desription of simple discretized partial differential equations. A unique feature of Modelica is the handling of discontinuous and variable structure components such as relays, switches, bearing friction, clutches, brakes, impact, sampled data systems, automatic gearboxes etc. Modelica has introduced special language constructs allowing a simulator to introduce efficient handling of events needed in such cases. Special design emphasis was given to synchronization and propagation of events and the possibility to find consistent restarting conditions. Finally, a powerful package concept is available to structure large model libraries and to find a component in a file system giving its hierarchical Modelica class name.

3. Modelica Libraries



In order that Modelica is useful for *model exchange*, it is important that libraries of the most commonly used components are available, ready to use, and sharable between applications. For this reason, the Modelica Association develops and maintains a growing *Modelica Standard Library*. Furthermore, other people and organizations are developing free and commercial Modelica libraries. For more information and especially for downloading the free libraries, see
http://www.Modelica.org/library/library.html. Currently, component libraries are available in the following domains:

- About 450 type definitions, such as Angle, Voltage, Inertia.
- Mathematical functions such as sin, cos, ln
- Continuous and discrete input/output blocks, such as transfer functions, filters, sources.
- Electric and electronic components such as resistor, diode, MOS and BJT transistor.
- 1-dim. translational components such as mass, spring, stop.
- 1-dim. rotational components such as inertia, gearbox, planetary gear, bearing friction, clutch.
- 3-dim. mechanical components such as joints, bodies and 3-dim. springs.
- Hydraulic components, such as pumps, cylinders, valves.
- Thermo-fluid flow components, such as pipes with multi-phase flow, heat exchangers.
- 1-dim. thermal components, such as heat resistance and heat capacitance.
- Power system components such as generators and lines.
- Power train components such as driver, engine, torque converter, automatic gearboxes.

Component libraries are realized by specialists in the respective area, taking advantage of the new features of Modelica. The Modelica Association is very interested in the development of further libraries. If you would like to contribute, please contact Martin Otter (Martin.Otter@dlr.de).

3. Modelica Simulation Environments

In order that the Modelica language and the Modelica libaries can be utilized, a Modelica translator is needed which transforms a Modelica model into a form which can be efficiently simulated in an appropriate simulation environment. The Modelica language itself is a relatively small language since there is only one basic structuring unit, a *class*, and all other structuring units, such as *model*, *connector, function, package, record* are just specializations of it. Therefore, it is straightforward to develop with reasonable effort an appropriate Modelica translator which reads a Modelica model and transforms it into a set of differential, algebraic and discrete equations. A free Modelica parser which reads a Modelica model and constructs an abstract syntax tree is available at:
http://www.Modelica.org/tools/parser/Parser.shtml.

However, solving this initial set of equations directly with a numerical method is both unreliable and very unefficient. One reason is that many Modelica models, like the mechanical ones, have a DAE index of 2 or 3, i.e., the overall number of states of the model is less than the sum of the states of the sub-components. In such a case, every direct numerical method has the difficulty that the numerical condition becomes worse, if the integrator step size is reduced and that a step size of zero leads to a singularity. Another problem is the handling of idealized elements, such as ideal diodes or Coulomb friction. These elements lead to *mixed* systems of equations having both *Real* and *Boolean unknowns*. Specialized algorithms are needed to solve such systems reliably.

Modelica was designed such that symbolic transformation algorithms can (and have to) be applied to transform the original set of equations into a form which can be integrated with standard methods. For example, the algorithm of Pantelides has to be applied to differentiate certain parts of the equations in order to reduce the index. Realization of such transformation algorithms is difficult and requires a lot of knowledge and implementation work. For this reason, currently no free Modelica simulation environment is available.

However, since June 2000, nearly the complete Modelica language is supported by the modeling and simulation environment *Dymola*. Dymola, commercially available since 1993, has now a Modelica translator which is able to perform all necessary symbolic transformations for large systems (> 100 000 equations). Currently, all Modelica applications are modeled and simulated with this environment. All the figures above are screenshots of Dymolas composition diagram editor. A demo version can be downloaded from http://www.Dynasim.se. For the control community it is important, that Dymola can generate Simulink S-Functions of a Modelica model in CMEX format. This allows to model a plant very conveniently with Modelica (which is tedious or impossible in Simulink for larger systems), utilize it as one input/output block in SIMULINK, realize controllers and filters in Simulink and utilize all the analysis and design capabilities of Matlab/Simulink.

4. Modelica Workshop'2000

On Oct. 23 - 24, 2000, the Modelica Association organized[1] a *workshop* at Lund University, Department of Automatic Control, Lund, Sweden, to bring together Modelica end users with Modelica language designers, Modelica tool vendors and Modelica library developers. Furthermore, a Dymola user's group meeting took place. The organizers were very pleased to welcome 85 participants, more than twice the expected number.

A 3 hour tutorial on Modelica was organized for 35 paricipants. Two rooms with 20 PCs were available, so that 1-2 participants could solve various modeling tasks (for beginners and for experienced users) at one PC. The time of one hour for the "hands-on-excersises" was a little bit short, but the participants enjoyed to solve even quite envolved modeling tasks such as modeling of a 3-shift automatic gearbox.

Most of the about 20 presentations had a high quality and an impressive breadth in the field of modeling with Modelica was presented, such as modeling of electronic systems, electrical power systems, electrical motors, hydraulic systems, thermo-hydraulic systems, vehicle power trains, fuel cell powered drive trains, and 3-dim. vehicle mechanics. New tools have been presented, such as MathModelica and an outlook on future developments have been given, such as modeling of partial differential equations and dimensional analysis of equations. All papers of the workshop are available in pdf-format for download at the Modelica home page.

Highlights of the presentations have been advanced applications of Modelica and Dymola in hardware-in-the-loop simulations and simulations of huge Modelica models:

- Anton Schiela and Hans Olsson presented a new method called "mixed mode integration" which is designed for real-time simulation of Modelica models and demonstrated their method online on a 650 Mhz PC for the real-time simulation of the full model of a robot with control system (78 states), containing stiff and non-stiff equations using a step-size of 1 ms, online tuning of the desired robot paths and CAD-data based real-time visualization. Using an explicit Euler method would have required a step size of 0.05 ms.
- Michael Tiller presented the biggest Modelica model ever simulated (the original model has 260 000 equations and after symbolic processing with Dymola about 25 000 nontrivial algebraic equations and 300 states). This is the result of a feasibility study at Ford Motor Company to evaluate the properties of Modelica and Dymola for detailed vehicle modeling (3D vehicle mechanics, engine, automatic gearbox, hydraulics to control the gearbox).

Due to the great success of this workshop it was decided to repeat it. The date and location will be decided on the next Modelica meeting which will take place on Feb. 7 - 9, 2001.

[1] Workshop and program committee:
Peter Fritzson (program chair), Hilding Elmqvist, Martin Otter, Hubertus Tummescheit (local organizer).

5. EU-Project RealSim

In January 2000, the EU project RealSim (Real-time Simulation for Design of Multi-Physics Systems) started, where Modelica is used as a basis. The primary goals of this project are:

- To develop tools for efficient simulation of *complex*, tightly-coupled *multi-physics* systems, with *real-time constraints*.
- To reduce product development cost and time by *design optimization* and by *hardware-in-the-loop simulation* for assessment, training and test automation purposes.

Especially, the current technological basis for Modelica shall be further improved:

- Symbolic transformation of large scale Modelica models (> 100 000 equations) to generate efficient C-code for real-time applications.
- Parallel and distributed simulation to enhance the performance of hardware-in-the-loop simulation and of multi-criteria optimization.
- Multi-criteria optimization based on distributed simulations.
- Closely integrated Modelica modeling and simulation environment with interactive experimentation, design optimization and visualization.

More information about this project is available under http://www.ida.liu.se/~pelab/realsim/.

6. Outlook

All necessary ingredients for modeling and simulation of industrial applications with Modelica are available: The language, libraries, tools and successful applications. Therefore, it can be expected that Modelica will have a significant impact in the modeling and simulation community in the future. The Modelica Association is very interested to further adapt the Modelica language to the needs of the end-users and to further develop the Modelica libraries. Interested simulation professionals who would like to contribute are encouraged and invited to participate at the design meetings (see the Modelica homepage for details where and when design meetings take place).