# AN INTRODUCTION TO THE PHYSICAL MODELING LANGUAGE MODELICA

**Hilding Elmqvist**

Dynasim AB
Research Park Ideon
SE-223 70 Lund, Sweden
E-mail: Elmqvist@Dynasim.se

**Sven Erik Mattsson**

Department of Automatic Control
Lund Institute of Technology
Box 118, SE-221 00 Lund, Sweden
E-mail: SvenErik@control.LTH.se

## ABSTRACT

A new language called Modelica™ for physical modeling is developed in an international effort. The main objective is to make it easy to exchange models and model libraries. The design approach builds on non-causal modeling with true ordinary differential and algebraic equations and the use of object-oriented constructs to facilitate reuse of modeling knowledge. There are already several modeling language based on these ideas available from universities and small companies. There is also significant experience of using them in various applications. The aim of the Modelica effort is to unify the concepts and to design a new uniform language for model representation. The paper describes the effort and gives an overview of Modelica.

## INTRODUCTION

Mathematical modeling and simulation are emerging as key technologies in engineering. Relevant computerized tools, suitable for integration with traditional design methods are essential to meet future needs of efficient engineering.

In October 1996 an international effort started to design a new language for physical modeling. The language is called Modelica[1]. The main objective is to make it easy to exchange models and model libraries and to allow users to benefit from the advances in object-oriented modeling methodology. This paper presents the status of the Modelica design as of August 1997.

### Today's simulation tools

There is a large amount of simulation software on the market. All languages and model representations are proprietary and developed for certain tools. There are general-purpose tools such as ACSL, SIMULINK, System Build. They are based on the same modeling methodology, input-output blocks, as in the previous standardization effort, CSSL, from 1967 [Strauss (ed.) (1967)]. There are domain-oriented packages: electronic programs (SPICE, Saber), multibody systems (ADAMS, DADS, SIMPACK), chemical processes (ASPEN Plus, SpeedUp) etc. With few exceptions, all simulation packages are only strong in one domain and are not capable of modeling components in other domains reasonably. This is a major disadvantage since technical systems are becoming more and more heterogeneous with components from many engineering domains.

### State-of-the art

Techniques for general-purpose physical modeling have been developed during the last decades, but did not receive much attention from the simulation market. The modern approaches build on non-causal modeling with true equations and the use of object-oriented constructs to facilitate reuse of modeling knowledge. There are already several modeling languages with such a support available from universities and small companies. Examples of such modeling languages are ASCEND [Piela *et al.* (1991)], Dymola [Elmqvist *et al.* (1996)], gPROMS [Barton and Pantelides (1994)], NMF [Sahlin *et al.* (1996)], ObjectMath [Fritzson *et al.* (1995)], Omola [Mattsson *et al.* (1993)], SIDOPS+ [Breunese and Broenink (1997), Smile [Kloas *et al.* (1995)], U.L.M. [Jeandel *et al.* (1996)] and VHDL-AMS [IEEE (1997)]. There is also significant experience of using these languages in various applications. The aim of the Modelica effort is to unify the concepts of these languages in order to introduce common basic syntax and semantics and to design a new unified modeling language for model representation.

### The Modelica effort

The work started in the continuous time domain since there is a common mathematical framework in the form of differential-algebraic equation (DAE) systems and there are several existing modeling languages based on similar ideas. There is also significant experience of using these languages in various applications. It was thus appropriate to collect all knowledge and experience in order to design a new unified modeling language or neutral format for model representation. The short range goal is to design a modeling language based on DAE systems with some discrete-event features to handle discontinuities and sampled systems. The design should allow an evolution to a multi-formalism, multi-domain, general-purpose modeling language.

[1]Modelica is a trade mark of the Modelica Design Group

**Table 1** The active members of the Modelica design group.

| |
|---|
| Fabrice Boudaud, Gaz de France |
| Jan Broenink, Univ. of Twente, Netherlands |
| Dag Brück, Dynasim AB, Lund, Sweden |
| Hilding Elmqvist, Dynasim AB, Lund, Sweden |
| Thilo Ernst, GMD-FIRST, Berlin, Germany |
| Peter Fritzson, Linköping University, Sweden |
| Alexandre Jeandel, Gaz de France |
| Kaj Juslin, VTT, Finland |
| Matthias Klose, Technical Univ. of Berlin |
| Sven Erik Mattsson, Lund University, Sweden |
| Martin Otter, DLR Oberpfaffenhofen, Germany |
| Per Sahlin, BrisData AB, Stockholm, Sweden |
| Hubertus Tummescheit, DLR Cologne, Germany |
| Hans Vangheluwe, University of Gent, Belgium |

The members of the Modelica design group are listed in Table 1. Hilding Elmqvist is the chairman. Information on the Modelica effort is available on WWW at `http://www.Dynasim.se/Modelica/`.

The activity started in October 1996 as an effort within the ESPRIT project "Simulation in Europe Basic Research Working Group (SiE-WG)". Information on SiE-WG can be found in Vangheluwe *et al.* (1996) and the at home page `http://hobbes.rug.ac.be/SiE/`.

In February 1997 the Modelica design effort became a Technical Committee within the Federation of European Simulation Societies, EUROSIM.

## MODELICA FUNDAMENTALS

In order to give an introduction to Modelica we will consider modeling of a simple electrical circuit as defined in Figure 1. The system can be broken up into a set of connected electrical standard components. We have a voltage source, two resistors, an inductor, a capacitor and a ground point. Models of these components are typically available in model libraries. Using a graphical model editor we can define a model by drawing an object diagram as shown in Figure 1, by positioning icons that represent the models of the components and drawing connections.

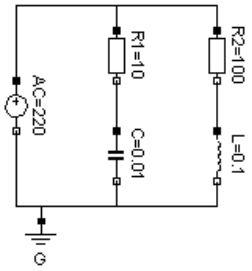


**Figure 1** A simple electrical circuit.

The corresponding Modelica model looks like

```
model circuit
  Resistor  R1 (R=10);
  VsourceAC AC;
  Capacitor C  (C=0.01);
  Ground    G;
  Resistor  R2 (R=100);
  Inductor  L  (L=0.1);
equation
  connect(AC.p, R1.p);  // Capacitor circuit
  connect(R1.n, C.p);
  connect(C.n, AC.n);
  connect(R1.p, R2.p);  // Inductor circuit
  connect(R2.n, L.p);
  connect(L.n, C.n);
  connect(AC.n, G.p);
end circuit;
```

This composite model specifies the topology of the system to be modeled. It specifies the components and the connections between the components.

The statement 'Resistor R1 (R=10);' declares a component R1 of class Resistor and sets the default value of the resistance R to 10.

Connections specify interactions between components. In other modeling languages connectors are referred to as cuts, ports or terminals. A connector must contain all quantities needed to describe the interaction. For electrical components we need the quantities voltage and current. Their types are declared as

```
type Voltage = Real(Unit = "V");
type Current = Real(Unit = "A");
```

where Real is the name of a predefined type. A real variable has a set of attributes such as unit of measure, minimum value, maximum value and initial value.

To simplify the use of Modelica and to support compatibility, there is an extensive standard library of type definitions which always is available with a Modelica translator. The type definitions in this base library are based on ISO 1000 and its naming conventions for physical quantities. Several ISO names are long, which make them awkward in practical modeling work. For this reason, shorter alias-names are provided if necessary. The use of the name "ElectricPotential" repeatedly in a model becomes cumbersome and therefore "Voltage" is also provided as an alternative.

A connector class is defined as

```
connector Pin
  Voltage v;
  flow Current i;
end Pin;
```

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, namely 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 flow rates in a piping network and to forces

and torques in a mechanical system. The sum-to-zero equations are generated when the prefix `flow` is used in the connector declarations. In Modelica it is assumed that the value is positive when the current or the flow is into the component.

Defining a set of connector classes is a good start when developing model libraries for a new application domain. It promotes compatibility of the component models.

A common property of many electrical components is that they have two pins. This means that it is useful to define a "shell" model class

```
partial model TwoPin
  "Shell model with two electrical pins"
  Pin p, n;
  Voltage v;
  Current i;
equation
  v = p.v - n.v;
  p.i + n.i = 0;
  i = p.i;
end TwoPin;
```

that has two pins, `p` and `n`, a quantity, `v`, that defines the voltage drop across the component and a quantity, `i`, that defines the current into the pin `p`, through the component and out from the pin `n`. The equations define common relations between quantities of a simple electrical component. In order to be useful, a constitutive equation must be added. The keyword `partial` indicates that this model class is incomplete. Between the name of a class and its body a string is allowed. It is treated as a comment attribute. Tools may display this documentation in special ways.

To define a model for a resistor we exploit `TwoPin` and add the definition of a parameter for the resistance and Ohm's law to define the behavior:

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

The keyword `parameter` specifies that the quantity is constant during a simulation experiment, but can change values between experiments. A parameter is a quantity which makes it simple for a user to modify the behavior of a model.

A model for an electrical capacitor is defined in a similar way

```
model Capacitor "Ideal capacitor"
  extends TwoPin;
  parameter Capacitance C;
equation
  C*der(v) = i;
end Capacitor;
```

where `der(v)` means the time derivative of `v`.

A model for the voltage source can be defined as

```
model VsourceAC "Sine-wave voltage source"
  extends TwoPin;
  parameter Voltage VA = 220 "Amplitude [V]";
  parameter Frequency f = 50 "Frequency [Hz]";
protected
  constant Real PI=3.141592653589793;
equation
  v = VA*sin(2*PI*f*time);
end VsourceAC;
```

Finally, we must not forget the ground point.

```
model Ground "Ground"
  Pin p;
equation
  p.v = 0;
end Ground;
```

The purpose of the ground model is twofold. First, it defines a reference value for the voltage levels. Secondly, the connections will generate one Kirchhoff's current law too many. The ground model handles this by introducing an extra current quantity `p.i`, which implicitly is defined to be zero by the equations.

## HYBRID MODELING

Realistic physical models typically contain discontinuities, events and changes of structure. Examples of such phenomena are relays, switches, friction, impact, sampled data systems etc. Modelica has introduced special language constructs allowing a simulator to introduce efficient handling of such events. Special design emphasis is given to synchronization and propagation of events and the possibility to find consistent restarting conditions after an event. It is possible to build model libraries allowing the efficient use of finite state machines and Petri nets.

### Modeling of a AC-DC converter

As an example, the discrete behavior which occurs in diodes and thyristors will be studied. Consider the AC-DC converter in Figure 2.

The circuit contains three diodes. The following

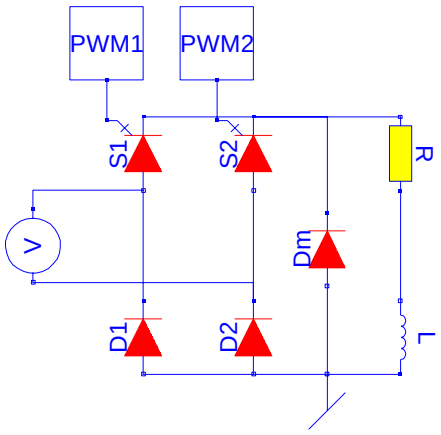


**Figure 2** Circuit diagram of AC-DC converter.

condition is an invariant of an ideal diode

```
(i > 0 and v == 0) or (v <= 0 and i == 0)
```

i.e. either the equation $i = 0$ or $v = 0$ should be active. This can be described as follows

```
0 = if i > 0 or not v <= 0 then v else i;
```

or equivalently

```
0 = if i > 0 or v > 0 then v else i;
```

A simulator supporting Modelica is not required to support such mixing of algebraic equations and boolean conditions since ordinary numerical integration routines cannot be used. Instead the model has to be rewritten using a boolean mode variable, Closed. The complete ideal diode model is given blow.

```
model Diode "Ideal diode"
  extends TwoPin;
  Boolean Closed(Start=false);
equation
  0 = if Closed then v else i;
  new(Closed) = if Closed then i > 0 else v > 0;
end Diode;
```

The first equation states that the voltage across the diode is zero when the diode is `Closed`, otherwise the current is zero. The second equation is boolean stating that if the diode is `Closed` and the current becomes zero or negative, `Closed` is changed to `false`. On the other hand, if `Closed = false` and `v` becomes positive, then `Closed` is changed to `true`. A special operator `new` is used to introduce discrete state variables and break direct dependencies between algebraic and boolean variables. A simulator must perform a fix-point iteration over such boolean variables in order to find consistent restart conditions after a closing or opening event.

A tool supporting Modelica will typically extract information from all relations in order to generate zero-crossing functions allowing the root-finder of the integration algorithm to find the exact time of events. The relation $i > 0$ will use the crossing function $i - 0$ and similarly $v - 0$ will be used for $v > 0$.

A ideal thyristor model is similar and includes the logic of the gate signal for switching. The thyristors are in this case controlled by pulse-width modulation. More details are given in Elmqvist *et al.* (1994).

A typical simulation result is given in Figure 3.

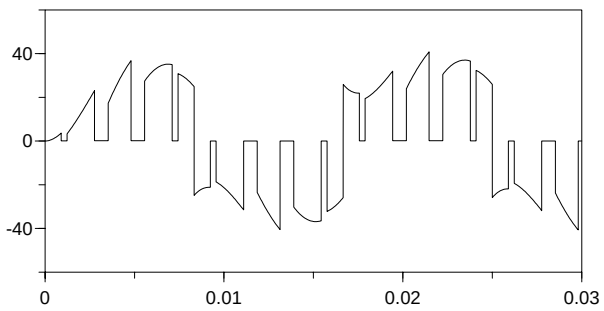


**Figure 3** The current through the voltage source.

## MORE ADVANCED MODELING FEATURES

The Modelica language has been introduced by giving small examples. Model classes and their instantiation form the basis of hierarchical modeling, connectors and connections correspond to physical connections of components. At the lowest level, equations are used to describe the relation between the quantities of the model.

The expressive modeling power of Modelica is large. Some of the more powerful constructs are summarized below.

Modeling of, for example, multi-body systems, control systems and approximations to partial differential equations is done conveniently by utilizing *matrix equations*. Multi-dimensional matrices and the usual matrix operators and matrix functions are thus supported in Modelica. It is also possible to have arrays of components and to define regular connection patterns. A typical usage is the modeling of a distillation column which consists of a set of trays connected in series.

We have so far discussed component parameters like the resistance value. Reuse of model library components is further supported by allowing also *model class parameters*. An example is a controlled plant where some PID controllers are replaced with auto tuning controllers. It is of course possible to just replace those controllers in a graphical user environment, i.e., to create a new model. The problem with this solution is that two models must be maintained. Modelica has the capability to instead just substitute the model class of certain components using a language construct at the highest hierarchical level, so only one version of the rest of the model is needed.

*Algorithms and functions* are supported in Modelica for modeling parts of a system in procedural programming style. Constructs for including *graphical annotations* are available in order that also icons and model diagrams become portable. An extensive *Modelica base library* contains standard variable and connector types promotes reuse by standardizing on interfaces.

## MODELING APPLICATIONS

Modelica has been used to model various kinds of systems. Otter *et al.* (1997) describe modeling of automatic gearboxes for the purpose of real-time simulation. Such models are non-trivial because of the varying structure during gear shift utilizing clutches, free wheels and brakes. Mattsson (1997) discusses modeling of heat exchangers. Class parameters of Modelica are used for medium parametrization and regular component structures are used for discretization in space of the heat exchanger. Ernst *et al.* (1997) discuss thermodynamical and flow oriented models. Broenink (1997) describes a Modelica library with bond graph model classes for supporting the bond graph modeling methodology.

## CONCLUSIONS

The Modelica effort has been described and an overview of Modelica has been given. The design is still evolving (August 1997). A first version of the language definition is scheduled to be available in September 1997.

There is ongoing work to write books on the Modelica language and on Modelica model libraries. Several Modelica tools are also under development. There are discussions to extend the Modelica design into, for example, handling partial differential equations and discrete event models.

More information, including modeling requirements, rationale and definition of the Modelica language and the future developments is available on WWW at

`URL: http://www.Dynasim.se/Modelica/.`


### Acknowledgements

The authors would like to thank the other members of the Modelica Design Group for inspiring discussions and their contributions to the Modelica design.


## REFERENCES


BARTON, P. and C. PANTELIDES (1994): "Modeling of combined discrete/continuous processes." *AIChE J.*, **40**, pp. 966–979.

BREUNESE, A. P. and J. F. BROENINK (1997): "Modeling mechatronic systems using the SIDOPS+ language." In *Proceedings of ICBGM'97, 3rd International Conference on Bond Graph Modeling and Simulation*, Simulation Series, Vol.29, No.1, pp. 301–306. The Society for Computer Simulation International.

BROENINK, J. F. (1997): "Bond-graph modeling in Modelica." In *Proceedings of the 1997 European Simulation Symposium (ESS'97)*. The Society for Computer Simulation, Passau, Germany.

ELMQVIST, H., D. BRÜCK, and M. OTTER (1996): *Dymola — User's Manual*. Dynasim AB, Research Park Ideon, Lund, Sweden.

ELMQVIST, H., F. CELLIER, and M. OTTER (1994): "Object-oriented modeling of power-electronic circuits using Dymola." In *Proceedings of the First Joint Conference of International Simulation Societies (CISS)*. The Society for Computer Simulation, ETH, Zurich, Switzerland.

ERNST, T., M. KLOSE, and H. TUMMESCHEIT (1997): "Modelica and Smile — A case study applying object-oriented concepts to multi-facet modeling." In *Proceedings of the 1997 European Simulation Symposium (ESS'97)*. The Society for Computer Simulation, Passau, Germany.

FRITZSON, P., L. VIKLUND, D. FRITZSON, and J. HERBER (1995): "High-level mathematical modeling and programming." *IEEE Software*, **12:3**.

IEEE (1997): "Standard VHDL Analog and Mixed-Signal Extensions." Technical Report IEEE 1076.1. IEEE.

JEANDEL, A., F. BOUDAUD, P. RAVIER, and A. BUHSING (1996): "U.L.M: Un Langage de Modélisation, a modelling language." In *Proceedings of the CESA'96 IMACS Multiconference*. IMACS, Lille, France.

KLOAS, M., V. FRIESEN, and M. SIMONS (1995): "Smile — A simulation environment for energy systems." In SYDOW, Ed., *Proceedings of the 5th International IMACS-Symposium on Systems Analysis and Simulation (SAS'95)*, vol. 18–19 of *Systems Analysis Modelling Simulation*, pp. 503–506. Gordon and Breach Publishers.

MATTSSON, S. E. (1997): "On modeling of heat exchangers in Modelica." In *Proceedings of the 1997 European Simulation Symposium (ESS'97)*. The Society for Computer Simulation, Passau, Germany.

MATTSSON, S. E., M. ANDERSSON, and K. J. ÅSTRÖM (1993): "Object-oriented modelling and simulation." In LINKENS, Ed., *CAD for Control Systems*, chapter 2, pp. 31–69. Marcel Dekker Inc, New York.

OTTER, M., C. SCHLEGEL, and H. ELMQVIST (1997): "Modeling and realtime simulation of an automatic gearbox using Modelica." In *Proceedings of the 1997 European Simulation Symposium (ESS'97)*. The Society for Computer Simulation, Passau, Germany.

PIELA, P., T. EPPERLY, K. WESTERBERG, and A. WESTERBERG (1991): "ASCEND: An object-oriented computer environment for modeling and analysis: the modeling language." *Computers and Chemical Engineering*, **15:1**, pp. 53–72.

SAHLIN, P., A. BRING, and E.F.SOWELL (1996): "The Neutral Model Format for building simulation, Version 3.02." Technical Report. Department of Building Sciences, The Royal Institute of Technology, Stockholm, Sweden.

STRAUSS (ED.), J. C. (1967): "The SCi continuous system simulation language (CSSL)." *Simulation*, **9**, pp. 281–303.

VANGHELUWE, H. L., E. J. KERCKHOFFS, and G. C. VANSTEENKISTE (1996): "Simulation for the Future: Progress of the ESPRIT Basic Research working group 8467." In BRUZZONE AND KERCKHOFFS, Eds., *Proceedings of the 1996 European Simulation Symposium (Genoa)*, pp. XXIX – XXXIV. Society for Computer Simulation International (SCS).