Modelica

Modelica is an object-oriented, declarative, multi-domain modeling language for component-oriented modeling of complex systems, e.g., systems containing mechanical, electrical, electronic, hydraulic, thermal, control, electric power or process-oriented subcomponents. The free Modelica language[1] is developed by the non-profit Modelica Association.[3] The Modelica Association also develops the free Modelica Standard Library[4] that contains about 1400 generic model components and 1200 functions in various domains, as of version 4.0.0.

Modelica

Paradigm	Declarative language
Developer	Modelica Association Project (MAP) (https://www.modelica.org/projects)
First appeared	1997
Stable release	3.5 / February 18, 2021[1]
OS	Cross-platform
License	CC-BY-SA
Filename extensions	.mo
Website	www.modelica.org (https://www.modelica.org/)
Major implementations	
AMESim, CATIA Systems, Dymola, JModelica.org, MapleSim, Modelon Impact,[2] Wolfram SystemModeler, OpenModelica, Scicos, SimulationX, Xcos	

Contents

Characteristics

While Modelica resembles object-oriented programming languages, such as C++ or Java, it differs in two important respects. First, Modelica is a modeling language rather than a conventional *programming* language. Modelica classes are not compiled in the usual sense, but they are translated into objects which are then exercised by a simulation engine. The simulation engine is not specified by the language, although certainly required capabilities are outlined.

Second, although classes may contain algorithmic components similar to statements or blocks in programming languages, their primary content is a set of *equations*. In contrast to a typical assignment statement, such as

```
x := 2 + y;
```

where the left-hand side of the statement is assigned a value calculated from the expression on the right-hand side, an equation may have expressions on both its right- and left-hand sides, for example,

```
x + y = 3 * z;
```

Equations do not describe assignment but *equality*. In Modelica terms, equations have no pre-defined *causality*. The simulation engine may (and usually must) manipulate the equations symbolically to determine their order of execution and which components in the equation are inputs and which are outputs.

History

The Modelica design effort was initiated in September 1996 by Hilding Elmqvist. The goal was to develop an object-oriented language for modeling of technical systems in order to reuse and exchange dynamic system models in a standardized format. Modelica 1.0 is based on the PhD thesis[5] of Hilding Elmqvist and on the experience with the modeling languages Allan,[6] Dymola, NMF[7] ObjectMath,[8] Omola,[9] SIDOPS+,[10] and Smile.[11] Hilding Elmqvist is the key architect of Modelica, but many other people have contributed as well (see appendix E in the Modelica specification[1]). In September 1997, version 1.0 of the Modelica specification was released which was the basis for a prototype implementation within the commercial Dymola software system. In year 2000, the non-profit Modelica Association was formed to manage the continually evolving Modelica language and the development of the free Modelica Standard Library. In the same year, the usage of Modelica in industrial applications started.

This table presents the timeline of the Modelica specification history:[12]

Release	Release Date	Highlights
1.0	1997, September	First version to model continuous dynamic systems.
1.1	1998, December	Language elements to model discrete systems (pre, when)
1.2	1999, June	Interface to C and Fortran, inner/outer for global variables, refined semantics of event handling
1.3	1999, December	Improved semantics for inner/outer connections, protected elements, array expressions.
1.4	2000, December	Removed declare-before-use rule, refined package concept, refined when-clause
2.0	2002, July	Initialization of models, standardization of graphical appearance, functions with mixed positional and named arguments, record constructor, enumerations
2.1	2004, March	Overdetermined connector to model 3-dim. mechanical systems, enhanced redeclaration of submodels, array and array indices of enumerations
2.2	2005, February	Expandable connector to model signal buses, conditional component declarations, arrays with dynamic size changes in functions
3.0	2007, September	Clean-up version: specification newly written, type system and graphical appearance refined, language flaws fixed, balanced model concept to detect model errors in a much better way
3.1	2009, May	Stream connector to handle bi-directional flow of fluid, operator overloading, mapping model parts to execution environments (for use in embedded systems)
3.2	2010, March	Improved initialization with homotopy method, functions as formal inputs to functions, Unicode support, access control to protect IP, improved support of object libraries
3.3	2012, May	Added language elements to describe periodic and non-periodic synchronous controllers based on clocked equations, as well as synchronous state machines.
3.4	2017, April	Automatic conversion of models. Many minor improvements
3.5	2021, February	Annotations for predefined plots. Change of specification format, with many editorial changes. Clarifications to synchronous language elements and state machines. Many minor clarifications to functions, model conversions, and several other parts of the specification.

Implementations

Commercial front-ends for Modelica include AMESim from the French company Imagine SA (now part of Siemens PLM Software), Dymola from the Swedish company Dynasim AB (now part of Dassault Systemes), Wolfram SystemModeler (formerly *MathModelica*) from the Swedish company Wolfram MathCore AB (now part of Wolfram Research), SimulationX from the German company ESI ITI GmbH, MapleSim from the Canadian company Maplesoft,[13] JModelica.org (open source, discontinued) and Modelon Impact,[14] from the Swedish company Modelon AB, and CATIA Systems [15][16] from Dassault Systemes (CATIA is one of the major CAD systems).

Openmodelica[17] 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 comprehensive Open Source Modelica modeling,[18] compilation and simulation environment based on free software distributed in binary and source code form for research,[19][20] teaching,[21] and industrial usage.

The free simulation environment Scicos uses a subset of Modelica for component modeling. Support for a larger part of the Modelica language is currently under development. Nevertheless, there is still some incompatibility and diverging interpretation between all the different tools concerning the Modelica

language.[22]

Examples

The following code fragment shows a very simple example of a first order system ($\dot{x} = -c \cdot x, x(0) = 10$):

```
model FirstOrder
  parameter Real c=1 "Time constant";
  Real x (start=10) "An unknown";
equation
  der(x) = -c*x "A first order differential equation";
end FirstOrder;
```

Interesting things to note about this example are the 'parameter' qualifier, which indicates that a given variable is time-invariant and the 'der' operator, which represents (symbolically) the time derivative of a variable. Also worth noting are the documentation strings that can be associated with declarations and equations.

The main application area of Modelica is the modeling of physical systems. The most basic structuring concepts are shown at hand of simple examples from the electrical domain:

Built-in and user derived types

Modelica has the four built-in types Real, Integer, Boolean, String. Typically, user-defined types are derived, to associate physical quantity, unit, nominal values, and other attributes:

```
type Voltage = Real(quantity="ElectricalPotential", unit="V");
type Current = Real(quantity="ElectricalCurrent", unit="A");
  ...
```

Connectors describing physical interaction

The interaction of a component to other components is defined by physical ports, called **connectors**, e.g., an electrical pin is defined as

```
connector Pin "Electrical pin"
   Voltage      v "Potential at the pin";
   flow Current i "Current flowing into the component";
end Pin;
```

When drawing connection lines between ports, the meaning is that corresponding connector variables without the "flow" prefix are identical (here: "v") and that corresponding connector variables with the "flow" prefix (here: "i") are defined by a zero-sum equation (the sum of all corresponding "flow" variables is zero). The motivation is to automatically fulfill the relevant balance equations at the infinitesimally small connection point.

Basic model components

A basic model component is defined by a **model** and contains equations that describe the relationship between the connector variables in a declarative form (i.e., without specifying the calculation order):

```
model Capacitor
  parameter Capacitance C;
  Voltage u "Voltage drop between pin_p and pin_n";
  Pin pin_p, pin_n;
equation
  0 = pin_p.i + pin_n.i;
  u = pin_p.v - pin_n.v;
  C * der(u) = pin_p.i;
end Capacitor;
```

The goal is that a connected set of model components leads to a set of differential, algebraic and discrete equations where the number of unknowns and the number of equations is identical. In Modelica, this is achieved by requiring so called **balanced models**.

The full rules for defining balanced models are rather complex, and can be read from [1] in section 4.7.

However, for most cases, a simple rule can be issued, that counts variables and equations the same way as most simulation tools do:

```
A model is balanced when the number of its equations
equals the number of its variables.
```

given that variables and equations must be counted according to the following rule:

```
->Number of model equations                              =
  Number of equations defined in the model               +
  number of flow variables in the outside connectors

->Number of model variables = Number of variables defined in the model
  (including the variables in the physical connectors)
```

Note that standard input connectors (such as RealInput or IntegerInput) do not contribute to the count of variables since no new variables are defined inside them.

The reason for this rule can be understood thinking of the capacitor defined above. Its pins contain a flow variable, i.e. a current, each. When we check it, it is connected to nothing. This corresponds to set an equation pin.i=0 for each pin. That's why we must add an equation for each flow variable.

Obviously the example can be extended to other cases, in which other kinds of flow variables are involved (e.g. forces, torques, etc.).

When our capacitor is connected to another (balanced) model through one of its pins, a connection equation will be generated that will substitute the two i=0 equations of the pins being connected. Since the connection equation corresponds to two scalar equations, the connection operation will leave the balanced larger model (constituted by our Capacitor and the model it is connected to).

The Capacitor model above is **balanced**, since

```
number of equations =   3+2=5   (flow variables: pin_p.i, pin_n.i, u)
number of variables =       5   (u, pin_p.u, pin_p.i, pin_n.u, pi_n.i)
```

Verification using OpenModelica[17] of this model gives, in fact

```
Class Capacitor has 5 equation(s) and 5 variable(s).
3 of these are trivial equation(s).
```

Another example, containing both input connectors and physical connectors is the following component from Modelica Standard Library:

```
model SignalVoltage
  "Generic voltage source using the input signal as source voltage"
  Interfaces.PositivePin p;
  Interfaces.NegativePin n;
  Modelica.Blocks.Interfaces.RealInput v(unit="V")
    "Voltage between pin p and n (= p.v - n.v) as input signal";
  SI.Current i "Current flowing from pin p to pin n";
equation
  v = p.v - n.v;
  0 = p.i + n.i;
  i = p.i;
end SignalVoltage;
```

The component SignalVoltage is balanced since

```
number of equations =   3+2=5  (flow variables: pin_p.i, pin_n.i, u)
number of variables =       5  (i, pin_p.u, pin_p.i, pin_n.u, pi_n.i)
```

Again, checking with OpenModelica[17] gives

```
Class Modelica.Electrical.Analog.Sources.SignalVoltage has 5 equation(s) and 5 variable(s).
4 of these are trivial equation(s).
```

Hierarchical models

A hierarchical model is built-up from basic models, by instantiating basic models, providing suitable values for the model parameters, and by connecting model connectors. A typical example is the following electrical circuit:

```
model Circuit
   Capacitor C1(C=1e-4) "A Capacitor instance from the model above";
   Capacitor C2(C=1e-5) "A Capacitor instance from the model above";
     ...
equation
   connect(C1.pin_p, C2.pin_n);
   ...
end Circuit;
```

Via the language element **annotation(...),** definitions can be added to a model that do not have an influence on a simulation. Annotations are used to define graphical layout, documentation and version information. A basic set of graphical annotations is standardized to ensure that the graphical appearance and layout of models in different Modelica tools is the same.

The freely available book "Modelica by Example (https://mbe.modelica.university/)" contains many more examples like these along with detailed explanations for nearly all the language features in Modelica version 3.3.

Applications

Modelica is designed to be domain neutral and, as a result, is used in a wide variety of applications, such as fluid systems (for example, steam power generation, hydraulics, etc.), automotive applications (especially powertrain)[23] and mechanical systems (for example, multi-body systems, mechatronics, etc.).

In the automotive sector, many of the major automotive OEMs are using Modelica. These include Ford,[24][25][26] General Motors,[27] Toyota,[28] BMW,[29] and Daimler.[30]

Modelica is also being increasingly used for the simulation of thermo-fluid and energy systems.[31]

The characteristics of Modelica (acausal, object-oriented, domain neutral) make it well suited to system-level simulation, a domain where Modelica is now well established.[32]

See also

- AMESim
- AMPL
- APMonitor
- ASCEND
- Domain-Specific Modeling DSM
- Dymola
- EcosimPro: Continuous and Discrete Modelling and Simulation Software
- EMSO
- GAMS
- JModelica.org
- OpenModelica
- MapleSim
- MATLAB
- Modelon Impact (http://www.modelon.com/modelon-impact/)
- MWorks
- Optimica Compiler Toolkit (http://www.modelon.com/products/optimica-compiler-toolkit/)
- SimulationX
- Simulink
- Wolfram SystemModeler
- Scilab/Xcos
- Kepler (Ptolemy)

Notes

1. "Modelica Language Specification, Version 3.5" (https://modelica.org/documents/MLS.pdf) (PDF). Modelica Association. 2021-02-18.
2. "Modelon Impact" (http://www.modelon.com/modelon-impact/). Modelon AB.
3. "Modelica and the Modelica Association" (http://www.modelica.org/).
4. The Modelica Standard Library is available for download here (https://www.modelica.org/libraries)
5. "A Structured Model Language for Large Continuous Systems" (https://lup.lub.lu.se/search/ws/files/4602422/8570492.pdf) (PDF).

6. Jeandel A., Boudaud F.: *Physical System Modelling Languages: from ALLAN to Modelica* (https://www.modelica.org/publications/papers/p303.pdf), Building Simulation'97, IBPSA Conference, Prague, September 8–10, 1997.
7. Per Sahlin (November 1996). "NMF HANDBOOK. An Introduction to the Neutral Model Format. NMF version 3.02" (http://www.equa.se/dncenter/handbook.pdf) (PDF).
8. "ObjectMath Home Page" (http://www.ida.liu.se/labs/pelab/omath/).
9. S.E. Mattsson, M. Andersson and K.J..Aström: Object-oriented modeling and simulation. In: Linkens, ed., CAD for Control Systems (Marcel Dekker, 1993) pp. 31-69.
10. "CiteSeerX — Modeling Mechatronic Systems Using The Sidops+ Language". CiteSeerX 10.1.1.56.4266 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.56.4266).
11. Ernst T., Jähnichen S., Klose M.: *Object-Oriented Physical Systems Modeling, Modelica, and the Smile/M Simulation Environment* (https://www.modelica.org/publications/papers/imacs97.pdf). 15th IMACS World Congress on Scientific Computation, Modelling and Applied Mathematics, Berlin, August 24–29, 1997.
12. "Documents" (http://www.modelica.org/news_items/documents). Modelica Association. Retrieved 2009-10-11.
13. "Supports Modelica standard" (http://www.maplesoft.com/products/maplesim/modelica.aspx). Maplesoft. Retrieved 2009-10-11.
14. "Modelon Impact" (http://www.modelon.com/modelon-impact/). Modelon. Retrieved 2021-04-01.
15. "Modelica in CATIA (module: CATIA Systems Dynamic Behavior)" (http://www.3ds.com/products/v6/v6-plm/portfolio//?no_cache=1&did=75). Dassault Systemes.
16. Announcement of DS' acquisition of Dynasim (http://www.3ds.com/cn/news-events/press-room/release/1220/1/)
17. Administrator. "Welcome to Open Modelica - OpenModelica" (http://www.openmodelica.org).
18. Adrian Pop, David Akhvlediani, Peter Fritzson *Integrated UML and Modelica System Modeling with ModelicaML in Eclipse* (http://www.actapress.com/Abstract.aspx?paperId=32114), In Proceedings of the 11th IASTED International Conference on Software Engineering and Applications (SEA 2007), Cambridge, MA, USA
19. Håkan Lundvall and Peter Fritzson *Automatic Parallelization of Object Oriented Models Executed with Inline Solvers* (https://dx.doi.org/10.1007/978-3-540-75416-9_49), In Proceedings of EuroPvm/Parsim, Springer Verlag LNCS, Volume 4757, 2007
20. EuroPVM/MPI 2007. "EuroPVM/MPI 2007 - PARSIM 2007 - Current Trends in Numerical Simulation for Parallel Engineering Environments New Directions and Work-in-Progress" (http://pvmmpi07.lri.fr/parsim07.html).
21. Anders Fernström, Ingemar Axelsson, Peter Fritzson, Anders Sandholm, Adrian Pop *OMNotebook - Interactive WYSIWYG Book Software for Teaching Programming* (http://www.modelica.org/publications/papers/2006-03-10-Fernstrom-etal-TeachingWorkshop-NotebookTeaching.pdf), In Proc. of the Workshop on Developing Computer Science Education - How Can It Be Done?, 2006. Linköping University, Dept. Computer & Inf. Science, Linköping, Sweden
22. Jörg Frochte *Modelica Simulator Compatibility - Today and in Future* (https://dx.doi.org/10.3384/ecp11063812), The 8th International Modelica Conference, March 20–22, 2011, Technical University, Dresden, Germany
23. Mahmud, Khizir; Town, Graham E. (2016-06-15). "A review of computer tools for modeling electric vehicle energy requirements and their impact on power distribution networks". *Applied Energy*. **172**: 337–359. doi:10.1016/j.apenergy.2016.03.100 (https://doi.org/10.1016%2Fj.apenergy.2016.03.100).
24. Michael Tiller, Paul Bowles, Mike Dempsey *Development of a Vehicle Modeling Architecture in Modelica* (http://www.modelica.org/events/Conference2003/papers/h32_vehicle_Tiller.pdf), 3rd International Modelica Conference

25. Erik Surewaard, Eckhard Karden, Michael Tiller *Advanced Electric Storage System Modeling in Modelica* (http://www.modelica.org/events/Conference2003/papers/h11_Surewaard.pdf), 3rd International Modelica Conference
26. Charles Newman, John Batteh, Michael Tiller *Spark-Ignited Engine Cycle Simulation in Modelica* (http://www.modelica.org/Conference2002/papers/p17_Newman.pdf) Archived (https://web.archive.org/web/20031002230948/http://www.modelica.org/Conference2002/papers/p17_Newman.pdf) 2003-10-02 at the Wayback Machine, 2nd International Modelica Conference
27. E. D. Tate, Michael Sasena, Jesse Gohl, Michael Tiller *Model Embedded Control: A Method to Rapidly Synthesize Controllers in a Modeling Environment* (http://www.modelica.org/events/modelica2008/Proceedings/sessions/session4c4.pdf), 6th International Modelica Conference
28. S. Soejima, T. Matsuba *Application of mixed mode integration and implicit inline integration at Toyota* (http://www.modelica.org/events/Conference2002/papers/p09_Soejima.pdf), 2nd International Modelica Conference
29. Henrik Wigermo, Johannes von Grundherr, Thomas Christ *Implementation of a Modelica Online Optimization for an Operating Strategy of a Hybrid Powertrain* (http://www.modelica.org/events/modelica2008/Proceedings/sessions/session4c3.pdf), 6th International Modelica Conference
30. Brückmann, Strenkert, Keller, Wiesner, Junghanns *Model-based Development of a Dual-Clutch Transmission using Rapid Prototyping and SiL* (http://www.qtronic.de/doc/DCT_2009.pdf), International VDI Congress Transmissions in Vehicles 2009, Friedrichshafen, Germany
31. Michael Wetter, Christoph Haugstetter *Modelica versus TRNSYS - A Comparison Between An Equation-Based and a Procedural Modeling Language for Building Energy Simulation* (http://simulationresearch.lbl.gov/wetter/download/WetterHaugstetter-2006.pdf), 2nd SimBuild Conference, Cambridge, MA, USA, August 2006.
32. Casella, Francesco (2015). *Simulation of Large-Scale Models in Modelica: State of the Art and Future Perspectives* (http://www.ep.liu.se/ecp_article/index.en.aspx?issue=118;article=49). *Proceedings of the 11th International Modelica Conference, Versailles, France, September 21–23, 2015*. Proceedings of the 11th International Modelica Conference, Versailles, France, September 21-23, 2015. **118**. Linköping University Electronic Press. pp. 459–468. doi:10.3384/ecp15118459 (https://doi.org/10.3384%2Fecp15118459). hdl:11311/964804 (https://hdl.handle.net/11311%2F964804). ISBN 978-91-7685-955-1. "Modelica language is well-established for system-level modelling tasks in many domains of engineering, such as automotive, robotics, mechatronics, energy, aerospace, in particular when multi-domain modelling is required."

External links

- Modelica 3.5 language specification (https://modelica.org/documents/MLS.pdf)
- Modelica Association (https://www.modelica.org), the homepage of the non-profit Modelica Association (developing Modelica)
- Modelica by Example (http://book.xogeny.com/) A free interactive HTML book for learning Modelica, by Michael Tiller
- Introduction to Physical Modeling with Modelica (https://www.springer.com/east/home/generic/search/results?SGWID=5-40109-22-33373560-0), book by Michael Tiller
- Fritzson, Peter (February 2004). *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1* (http://www.ida.liu.se/labs/pelab/modelica/OpenModelica/Documents/ModelicaBookExcerpts.pdf) (PDF). Wiley-IEEE Press. ISBN 978-0-471-47163-9.
- Commercial Modelica tools: Dymola (http://www.3ds.com/products/catia/portfolio/dymola), LMS Imagine.Lab AMESim (https://archive.is/20120515225718/http://www.lmsintl.com/imagine-amesim-1-d-multi-domain-system-simulation), CyModelica (http://www.cydesign.com) MapleSim (http://www.maplesoft.com/products/maplesim/index.aspx), Wolfram SystemModeler (http://www.wolfram.com/system-modeler/), Modelica Physical Modeling Toolbox for Matlab (https://web.archi

ve.org/web/20110602031516/http://www.modelon.com/products/modelica-physical-modeling-toolbox-for-matlab/), SimulationX (http://www.simulationx.com/), Vertex (https://web.archive.org/web/20110411154833/http://www.deltatheta.com/products/vertex/), JModelica.org (http://www.jmodelica.org/)

- Open source Modelica tools: OpenModelica (http://www.openmodelica.org/) (GPL or OSMC-PL),
- Modelica Overview (https://modelica.org/education/educational-material/lecture-material/english/ModelicaOverview.pdf)

Retrieved from "https://en.wikipedia.org/w/index.php?title=Modelica&oldid=1015868944"

This page was last edited on 4 April 2021, at 01:20 (UTC).

Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. By using this site, you agree to the Terms of Use and Privacy Policy. Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.