The Modelica Object-Oriented Equation-Based Language and Its OpenModelica Environment with MetaModeling, Interoperability, and Parallel Execution

Peter Fritzson

PELAB Programming Environments Lab, Department of Computer Science
Linköping University, SE-581 83 Linköping, Sweden
`peter.fritzson@liu.se`

Abstract. Modelica is a modern, strongly typed, declarative, equation-based, and object-oriented language for modeling and simulation of complex systems. Major features are: ease of use, visual design of models with combination of lego-like predefined model building blocks, ability to define model libraries with reusable components, support for modeling and simulation of complex applications involving parts from several application domains, and many more useful facilities. This paper gives an overview of some aspects of the Modelica language and the OpenModelica environment – an open-source environment for modeling, simulation, and development of Modelica applications. Special features are MetaModeling for efficient model transformations and prototype generation of parallel code for multi-core architectures. Ongoing work also include UML-Modelica integration and interoperability support through the new FMI standard.

1 Introduction

A large number of modeling and simulation tools are available on the market. Most tools are very specialized and only address a specific application domain. These tools usually have incompatible model formats.

However, certain tools are of a more general nature. For modeling and simulation of dynamic systems, we have e.g. the following de-facto standards:

- Continuous: Matlab/Simulink, [28], MatrixX/ SystemBuild, Scilab/Scicos, ACSL,... for general systems, SPICE and its derivatives for electrical circuits, ADAMS, DADS/Motion, SimPack, SimMechanics,... for multi-body mechanical systems, ANSYS, FEMLAB, etc. for finite-element analysis, ...
- Discrete: general-purpose simulators based on the discrete-event GPSS line, VHDL- and Verilog simulators in digital electronics, etc.
- Hybrid (discrete + continuous): Modelica, gPROMS [19], AnyLogic, VHDL-AMS and Verilog-AMS simulators (not only for electronics but also for multi-physics problems), etc.

The insufficient power and generality of the former modeling languages has stimulated the development of Modelica (as a true object-oriented, multi-physics language) and VHDL-AMS/Verilog-AMS (multi-physics but primarily for electronics, and therefore often not general enough).

1.1 Modelica Background

In 1996 several first generation object-oriented mathematical modeling languages and simulation systems (ObjectMath [25], Dymola [23] [24], Omola [29], NMF [32], gPROMS [19], Allan, Smile, etc.) had been developed.

However, the situation with a number of different incompatible object-oriented modeling and simulation languages was not satisfactory. Therefore, in the fall of 1996, a group of researchers from universities and industry started work towards standardization and making this object-oriented modeling technology widely available. This language is called Modelica [2][1][33] and designed primarily for modeling dynamic behavior of engineering systems; moreover, meta-modeling extensions have been developed [26]. The language is intended to become a de facto standard.

The language allows defining models in a declarative manner, modularly and hierarchically and combining various formalisms expressible in the more general Modelica formalism. The multidomain capability of Modelica allows combining electrical, mechanical, hydraulic, thermodynamic, etc., model components within the same application model.

Compared to most widespread simulation languages available today this language offers several important advantages:

- *Object-oriented mathematical modeling.* This technique makes it possible to create model components, which are employed to support hierarchical structuring, reuse, and evolution of large and complex models covering multiple technology domains.
- *Acausal modeling.* Modeling is based on equations instead of assignment statements as in traditional input/output block abstractions. Direct use of equations significantly increases re-usability of model components, since components adapt to the data flow context in which they are used. Interfacing with traditional software is also available in Modelica.
- *Physical modeling of multiple application domains.* Model components can correspond to physical objects in the real world, in contrast to established techniques that require conversion to “signal” blocks with fixed input/output causality. In Modelica the structure of the model naturally correspond to the structure of the physical system in contrast to block-oriented modeling tools such as Simulink. For application engineers, such “physical” components are particularly easy to combine into simulation models using a graphical editor.
- *A general type system* that unifies object-orientation, multiple inheritance, components/connectors, and templates/generics within a single class construct.

Hierarchical system architectures can easily be described with Modelica thanks to its powerful component model. Components are connected via the connection mechanism realized by the Modelica system, which can be visualized in connection diagrams. The component framework realizes components and connections, and ensures that communication works over the connections.

For systems composed of acausal components with behavior specified by equations, the direction of data flow, i.e., the causality is initially unspecified for connections between those components. Instead the causality is automatically deduced by the compiler when needed. Components have well-defined interfaces consisting of ports, also known as connectors, to the external world. A component may internally consist of other connected components, i.e., hierarchical modeling. Fig. 1 shows hierarchical component-based modeling of an industry robot. More information on Modelica robot models can for example be found in [35], [36], [37], and [38]. Methods for real-time simulation, e.g. in [39].

Fig. 1. Hierarchical model of an industrial robot, including components such as motors, bearings, control software, etc. At the lowest (class) level, equations are typically found.

The language design and model library development has proceeded through a number of design meetings, a nonprofit Modelica Association was started in Linköping, Sweden year 2000, and a conference series was started the same year, with the 5th conference in Vienna Sept. 2006. Several commercial implementations of Modelica (i.e., subsets thereof) are available, including Dymola [22], MathModelica [27], and IDA [20]. This presentation primarily focuses on the OpenModelica, which is currently the major Modelica open-source tool effort and most complete open source environment.

1.2 An Example Modelica Model

The following is an example Lotka Volterra Modelica model containing two differential equations relating the sizes of rabbit and fox populations which are represented by the variables *rabbits* and *foxes*: The rabbits multiply; the foxes eat rabbits. Eventually there are enough foxes eating rabbits causing a decrease in the rabbit population, etc., causing cyclic population sizes. The model is simulated and the sizes of the rabbit and fox populations as a function of time are plotted in Fig. 2.

Fig. 2. Number of rabbits – prey animals, and foxes - predators, as a function of time simulated from the PredatorPrey model

The notation `der(rabbits)` means time derivative of the rabbits (population) variable.

```

class LotkaVolterra
  parameter Real g_r = 0.04      "Natural growth rate for rabbits";
  parameter Real d_rf = 0.0005  "Death rate of rabbits due to foxes";
  parameter Real d_f = 0.09     "Natural deathrate for foxes";
  parameter Real g_fr = 0.1     "Efficiency in growing foxes fr rabbits";
  Real rabbits(start=700) "Rabbits, (R) with start population 700";
  Real foxes(start=10)      "Foxes, (F) with start population 10";
equation
  der(rabbits) = g_r*rabbits - d_rf*rabbits*foxes;
  der(foxes)   = g_fr*d_rf*rabbits*foxes - d_f*foxes;
end LotkaVolterra;

```

2 The OpenModelica Environment

The OpenModelica open source environment [4], [5] project supported by the Open Source Modelica Consortium has several goals, including, but not limited to the following:

- Providing an efficient interactive computational environment for the Modelica language, for both academic and industrial usage.
- Development of a complete reference implementation of Modelica in MetaModelica, an extended version of Modelica itself.
- Providing an environment for teaching modeling and simulation. It turns out that with support of appropriate tools and libraries, Modelica is very well suited as a computational language for development and execution of both low level and high level numerical algorithms, e.g. for control system design, solving nonlinear equation systems, or to develop optimization algorithms for complex applications.
- *Language design*, e.g. to further *extend the scope* of the language, e.g. for use in diagnosis, structural analysis, system identification, etc., as well as modeling

problems that require extensions such as partial differential equations, enlarged scope for discrete modeling and simulation, etc.

- *Language design to improve abstract properties* such as expressiveness, orthogonality, declarativity, reuse, configurability, architectural properties, etc.
- *Improved implementation techniques*, e.g. to enhance the performance of compiled Modelica code by generating code for parallel hardware.
- *Improved debugging* support for equation based languages such as Modelica, for improved ease of use.
- *Easy-to-use* specialized high-level (graphical) *user interfaces* for certain application domains.
- *Application usage* and model library development by researchers in various application areas.

3 MetaModelica – A Unified Equation-Based Modeling Language

The idea to define a unified equation-based mathematical and semantic modeling language started from the development of the OpenModelica compiler [5]. The entire compiler is generated from a Natural Semantics specification written in RML [34], a compiler generator for Natural Semantics. The open source OpenModelica compiler has its users in the Modelica community which have detailed knowledge of Modelica but very little knowledge of RML and Natural Semantics. In order to allow people from the Modelica community to contribute to the OpenModelica compiler we retargeted the development language from RML to MetaModelica, which is based on the Modelica language with several extensions. We already translated the OpenModelica compiler from RML to the MetaModelica using an automated translator. We also developed a compiler which can handle the entire OpenModelica compiler specification (~160 000 lines of code) defined in MetaModelica.

The basic idea behind the unified language is to use equations as the unifying feature. Most declarative formalisms, including functional languages, support some kind of limited equations even though people often do not regard these as equations, e.g. single-assignment equations.

Using the meta-programming facilities, usual tasks like generation, composition and querying of Modelica models can be automated. MetaModelica supports e.g. efficient pattern matching and transformation of models represented as abstract syntax trees.

4 Integrated UML-Modelica for Whole-Product Modeling

Complex products are increasingly made of both software and hardware components which are closely interacting. Thus, modeling tools and processes need to support co-design of software and hardware in an integrated way. This is the main goal addressed by the OPENPROD project [3].

Currently, UML is the dominant graphical modeling notation for software, whereas Modelica is the major object-oriented mathematical modeling language for component-oriented modeling of complex physical systems, e.g., systems containing

mechanical, electrical, electronic, hydraulic, thermal, control, electric power or process-oriented subcomponents.

For these reasons we developed the first generation prototype UML-Modelica-SysML integrated modeling environment [16] as a ModelicaML UML profile integrated in Eclipse as a plug-in. The profile reuses some artifacts from the System Modeling Language (SysML) profile, and combines the major UML diagrams with Modelica graphic connection diagrams. Requirement, equation, and simulation diagrams are also supported in an integrated way.

The second generation ModelicaML profile prototype [6][7] is more comprehensive and more precisely defined. It is based directly on UML, implemented using the Papyrus UML tool, generates executable Modelica code, and also supports activity diagrams and requirements diagrams. It is possible to model integrated software/hardware models, specify requirements, generate executable Modelica code, and simulate the model while checking if the requirements are fulfilled or violated. A recent further development in ModelicaML [8] supports generation of executable Modelica from UML state charts, e.g. for use in embedded systems modeling.

A related ongoing effort [9] by the OMG SysML-Modelica task force is to define an executable SysML with precise semantics by mapping it to Modelica

5 Parallel Execution on Multi-core Platforms

A common problem in simulation is too long computing time due to increasingly complex models. The new multi-core platforms provide more computing power, that however typically is not that easy to use since these platforms usually require significant restructuring and re-programming, making the programming task even more complex and increasing the risk of introducing errors. Automatic or semi-automatic parallelization is needed, but this a challenging problem to solve for realistic applications.

However, starting from a mathematical model at a high level of an abstraction gives more freedom for the compiler to perform parallelization and avoids many low-level programming details that may hinder parallelization. This approach has been investigated by us, e.g. in [10], and involves extracting a sufficient amount of parallelism from equations represented in the form of a data dependence graph (task graph), requiring analysis of the code at a detailed level. Efficient clustering and merging of fine-grained tasks is essential to avoid communication overhead. Speedup results from the preliminary OpenModelica prototype on fast Linux clusters and Nvidia GPUs are encouraging.

A recent development [10] uses a combination of software pipelining and solver inlining to achieve good speedups for compiling a nonlinear mechanical Modelica model to an Intel multi-core platform. We are currently re-designing and re-implementing this prototype in order to increase scalability and support parallelization of larger models.

Work is also ongoing to extend the automatic parallelization to support efficient event-handling in parallelized code in order to handle hybrid continuous-discrete models, aiming at several target real-time execution platforms available in the OPENPROD project. Coarse-grained manual hints using transmission line modeling connections [13] can partition the simulation model for simulation on different

processors. Thus, both coarse-grained manual partitioning and automatic fine-grained parallelization can be combined.

6 Tool Interoperability through Functional Mockup Interface

A new open standard for model interchange and tool interoperability which simplifies whole product modeling has recently been released [14]¹, developed in the ITEA2 MODELISAR project [15]. This open standard is called FMI – Functional Mockup Interface. The current release is FMI for model exchange 1.0.

The intention of FMI is that a modeling environment can generate C-Code from a dynamic system model that can be exported to other modeling and simulation environments either in source or binary form.

Models are described by differential, algebraic and discrete equations with time-, state- and step-events. In particular, all Modelica 3.2 models are supported and all Modelica variable attributes (like units and display units) as well as description texts can be exchanged. The models to be treated by this interface can be large for usage in offline or online simulation or can be used in embedded control systems on micro-processors. It is possible to utilize several instances of a model and to connect models hierarchically.

The MODELISAR consortium plans to publish additional parts of the FMI, especially for coupling of different simulation tools via co-simulation and for coupling of simulators with applications like testing, optimization, and product lifecycle management. An open-source implementation of FMI has been started in the OPENPROD project.

7 Conclusions

This paper has given a short overview of the Modelica language, its background and the most important features of the language, as well as the OpenModelica open source environment. This environment includes many valuable features for engineers and researchers, and it is the only Modelica environment so far with good support for debugging Modelica algorithmic code as well as support for meta-programming integrated in the language. Prototype parallel code generation was also mentioned. We believe that this open source platform can be part of forming the foundation of the next generation of the Modelica language and environment development efforts, both from a research perspective and a system engineering usage point of view

We have also presented ongoing work in integrating UML and Modelica in ModelicaML for model-based product development in the OPENPROD project, as well as the FMI standard on tool and model interoperability developed in the MODELISAR project. By basing an executable subset of UML on Modelica as an action language in ModelicaML, with translation of state charts or activity diagrams into Modelica, executable modeling of whole products with both software and hardware components is possible.

---

¹ Most of the text in this section is cited from [14].

Acknowledgements

Vinnova, the Swedish Governmental Agency for Innovation Systems, supports Linköping University in the ITEA2 OPENPROD project. Support is also received from VR and SSF.

References

- [1] Fritzson, P.: Principles of Object-Oriented Modeling and Simulation with Modelica 2.1., p. 940. Wiley-IEEE Press, Chichester (2004)
- [2] The Modelica Association. The Modelica Language Specification Version 3.2 (May 2010), <http://www.modelica.org>
- [3] OPENPROD project web page, <http://www.openprod.org> and http://www.itea2.org/public/project_leaflets/OPENPROD_profile_oct-09.pdf (accessed May 2010)
- [4] OSMC. The Open Source Modelica Consortium, <http://www.openmodelica.org> (accessed September 2010)
- [5] Fritzson, P., Aronsson, P., Lundvall, H., Nyström, K., Pop, A., Saldamli, L., Broman, D.: The OpenModelica Modeling, Simulation, and Software Development Environment. In: Simulation News Europe (SNE), vol. 44 (January 2006), <http://www.openmodelica.org>
- [6] Schamai, W.: Modelica Modeling Language (ModelicaML) A UML Profile for Modelica, Technical report 2009:5, Linköping University, Sweden (2009), <http://www.ep.liu.se>
- [7] Schamai, W., Fritzson, P., Paredis, C., Pop, A.: Towards Unified System Modeling and Simulation with ModelicaML Modeling of Executable Behavior Using Graphical Notations. In: Proceedings of the 7th International Modelica Conference (Modelica 2009), Como, Italy (September 20-22, 2009), <http://www.modelica.org>
- [8] Schamai, W., Pohlmann, U., Fritzson, P., Paredis, C., Helle, P., Strobel, C.: Execution of UML State Machines Using Modelica. Accepted to EOOLT 2010 (June 2010), <http://www.eoolt.org>
- [9] Bernard, Y., Burkhart, R.M., de Koning, H.-P., Friedenthal, S., Fritzson, P., Paredis, C., Rouquette, N.F., Schamai, W.: An Overview of the SysML-Modelica Transformation Specification. In: Proc. of INCOSE 2010, 20th Annual INCOSE Int. Symposium, Chicago, USA, July 11-15 (2010), <http://www.incose.org>
- [10] Aronsson, P.: Automatic Parallelization of Equation-Based Simulation Programs. PhD thesis, Dissertation No. 1022, Dept, Computer and Information Science, Linköping University, Linköping, Sweden
- [11] Lundvall, H., Fritzson, P.: Automatic Parallelization using Pipelining for Equation-Based Simulation Languages. In: Proc. of the 14th Workshop on Compilers for Parallel Computing (CPC 2009), Zurich, Switzerland, (January 7-9, 2009)
- [12] Fritzson, P., Aronsson, P., Lundvall, H., Nyström, K., Pop, A., Saldamli, L., Broman, D.: The OpenModelica Modeling, Simulation, and Software Development Environment. In: Simulation News Europe, vol. 44(45) (December 2005), <http://www.openmodelica.org>
- [13] Nyström, K., Fritzson, P.: Parallel Simulation with Transmission Lines in Modelica. In: Proceedings of the 5th International Modelica Conference (Modelica 2006), Vienna, Austria, (September 4-5, 2006)

- [14] Otter, M.: Functional Mockup Interface (FMI) for Model Exchange. In: Modelica Newsletter 2010-1 (April 2010), <http://www.modelica.org>, <http://www.functional-mockup-interface.org>
- [15] MODELISAR itea2 project, http://www.itea2.org/public/project_leaflets/MODELISAR_project_profile_oct-08.pdf (accessed May 2010)
- [16] Pop, A., Akhvlediani, D., Fritzson, P.: Integrated UML and Modelica System Modeling with ModelicaML in Eclipse. In: Proceedings of the 11th IASTED International Conference on Software Engineering and Applications (SEA 2007), Cambridge, MA, USA (November 19-21, 2007)
- [17] Papyrus UML, <http://www.papyrusuml.org>
- [18] OMG. OMG Unified Modeling Language™ (OMG UML). Superstructure Version 2.2 (February 2009)
- [19] Barton, P., Pantelides, C.: The Modelling of Combined Discrete/Continuous Processes. *AIChE Journal* 40, 966–979 (1994)
- [20] Equa, A.B.: The IDA simulation tool, <http://www.equa.se> (accessed 2006)
- [21] Christen, E., Bakalar, K.: VHDL-AMS—A Hardware Description Language for Analog and Mixed-Signal Applications. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing* 46(10), 1263–1272 (1999)
- [22] Dynasim AB. Dymola—Dynamic Modeling Laboratory with Modelica, Users Manual, Version 6.0. Dynasim AB, Research Park Ideon, SE-223 70, Lund, Sweden (2006)
- [23] Elmqvist, H.: A Structured Model Language for Large Continuous Systems. Ph.D. thesis, TFRT-1015, Dept. of Automatic Control, Lund Institute of Technology, Lund, Sweden (1978)
- [24] Elmqvist, H., Bruck, D., Otter, M.: Dymola—User’s Manual. Dynasim AB, Research Park Ideon, SE-223 70, Lund, Sweden (1996)
- [25] Fritzson, P., Viklund, L., Fritzson, D., Herber, J.: High-Level Mathematical Modelling and Programming. *IEEE Software* 12(4), 77–87 (1995), <http://www.ida.liu.se/labs/pelab/omath>
- [26] Fritzson, P., Pop, A., Aronsson, P.: Towards Comprehensive Meta-Modeling and Meta-Programming Capabilities in Modelica. In: Proceedings of the 4th International Modelica Conference, Hamburg, Germany (March 7-8, 2005)
- [27] MathCore Engineering AB. MathModelica User’s Guide (2010), <http://www.mathcore.com>
- [28] MathWorks. The Mathworks - Simulink - Simulation and Model-Based Design, <http://www.mathworks.com/products/simulink/> (last accessed: May 15, 2006)
- [29] Mattsson, S.-E., Andersson, M.: The Ideas Behind Omola. In: Proceedings of the 1992 IEEE Symposium on Computer-Aided Control System Design (CADCS 1992), Napa, California (March 1992)
- [30] Pop, A., Fritzson, P.: A Portable Debugger for Algorithmic Modelica Code. In: Proceedings of the 4th International Modelica Conference, Hamburg, Germany (March 7-8, 2005)
- [31] Pop, A., Fritzson, P., Remar, A., Jagudin, E., Akhvlediani, D.: Open-Modelica Development Environment with Eclipse Integration for Browsing, Modeling, and Debugging. In: Proc. of Modelica 2006, the 5th Int. Modelica Conf., Vienna, September 4-5 (2006)
- [32] Sahlin, P.: Modelling and Simulation Methods for Modular Continuous Systems in Buildings. Ph.D. thesis, Dept. of Building Science, Royal Inst. of Technology Stockholm, Sweden (May 1996)

- [33] Tiller, M.: Introduction to Physical Modeling with Modelica, p. 366. Kluwer Academic Publishers, Dordrecht (2001)
- [34] Pettersson, M.: Compiling Natural Semantics. LNCS, vol. 1549. Springer, Heidelberg (1999)
- [35] Hirzinger, G., Bals, J., Otter, M., Stelter, J.: The DLR-KUKA Success Story: Robotics Research Improves Industrial Robots. IEEE Robotics & Automation Magazine 12(3), 16–23 (2005)
- [36] Ferretti, G., Gritti, M., Magnani, G., Rocco, P.: A Remote User Interface to Modelica Robot Models. In: Fritzson, P. (ed.) Proceedings of the 3rd International Modelica Conference, Linköping (November 3–4, 2003), http://www.modelica.org/events/Conference2003/papers/h23_Gritti.pdf
- [37] Frenkel, J., Schubert, C., Kunze, G., Jankov, K.: Using Modelica for Interactive Simulations of Technical Systems in a Virtual Reality Environment. In: Proceedings of the 7th International Modelica Conference, Como, Italy. Linköping Univ. Electronic Press (September 20–22, 2009), <http://www.ep.liu.se/ecp/043/061/ecp09430080.pdf>
- [38] Reiner, M., Heckmann, A., Otter, M.: Inversion based control of flexible body systems. In: Proceedings of the 9th Conference on Motion and Vibration Control, MOVIC 2008, Munich (September 15, 2008)
- [39] Sjölund, M., Braun, R., Fritzson, P., Krus, P.: Towards Efficient Distributed Simulation in Modelica using Transmission Line Modeling. In: Proceedings of the 3rd International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools, EOOLT 2010, In Conjunction with MODELS 2010, Oslo, Norway. Linköping University Electronic Press (October 3, 2010), <http://www.ep.liu.se>