Object-oriented Modeling of Mechatronics Systems in Modelica Using Wrapped Bond Graphs

François E. Cellier, Ph.D.
Department of Computer Science
ETH Zurich
CH-8092 Zurich
Switzerland

Modelica has established itself as the *de facto* industry standard for modeling physical systems. The language definition is maintained and updated by the *Modelica Association*[1], a loose collaboration of researchers from academia, industry, and government. The Modelica Association also maintains the *Modelica Standard Library*[2], a large library of component models coded in the Modelica language.

The object-oriented modeling paradigm makes it possible to code component models in a completely modular fashion. They can be connected topologically without any restrictions. The Modelica language also supports hierarchical decomposition, i.e., internally connected sets of component models can be re-interpreted as new component models at a hierarchically higher level.

Both the Modelica language definition and the Modelica standard library are in the public domain and can therefore be used freely. Several software developers have implemented Modelica modeling and simulation environments. Some of these implementations are commercial ventures, whereas others are made available through the open-source community. The most advanced among these tools is *Dymola*[3], a commercial software that is mature, implements the full set of Modelica features, and is being routinely used by many modelers from academia, industry, and government agencies for large-scale physical systems modeling. *OpenModelica*[4], on the other hand, is a free open-source implementation maintained by Linköping University.

All implementations offer meanwhile a graphical front-end for composing Modelica models graphically on the screen. They offer a Modelica compiler that makes use of advanced symbolic algorithms for index reduction, partitioning, tearing, and state selection of the differential and algebraic equation systems extracted from the Modelica model. The compiler thereby converts implicit and possibly higher-index DAE models to mostly explicit ODE descriptions that can be simulated using standard numerical ODE solvers. They offer a Modelica simulation run-time system that contains a collection of different numerical ODE solvers, root solvers, and iterative algorithms for dealing with

sets of possibly non-linear tightly-coupled algebraic equation systems. They finally offer graphical back-end software for graphical display of simulation results. Some tools also offer a graphical animation engine, e.g. for animating 3D mechanical systems. In *Dymola*, all of those tools have been integrated into one highly intuitive graphical user interface, i.e., the modeler can access all of these tools without even noticing what is going on behind the interface. Other implementations are less convenient to use (more poorly integrated) and may still lack some of the features.

Mechatronics systems place high demands on the modeling and simulation environment. They usually call for large-scale models operating in multiple energy domains. They contain at least an electronic and a mechanical subsystem, but they may also contain hydraulic or pneumatic components, and it may be desired to investigate their thermodynamic properties as well.

Mechatronics systems place high demands on the model compiler, because they almost invariably lead to higher-index DAE systems with large sets of algebraically coupled equations. The model compiler must therefore support symbolic index reduction, offer efficient symbolic partitioning and tearing algorithms, and also support symbolic state selection, in order to generate a compact set of simulation equations. Of all Modelica implementations, only *Dymola* is currently capable of dealing with large mechatronics systems effectively and efficiently.

Mechatronics systems also place high demands on the simulation run-time system, as the resulting ODEs are almost invariably stiff containing nasty discontinuities. Hence the simulation engine must feature sturdy stiff system solvers with root solving capabilities. While tearing will break large loosely-coupled algebraic equation systems down into smaller tightly-coupled algebraic equation systems, those still need to be handled numerically by the run-time system. Hence the simulation run-time system must either offer a stiff numerical index-1 DAE solver with root solving capabilities, or alternatively a stiff numerical ODE solver with root solving capabilities plus a Newton solver to deal with the remaining algebraic loops.

Mechatronics systems also place high demands on the graphical back-end software, as the users of the software usually ask for animation. For example, a flight simulator ought to be able to animate the aircraft itself (3D mechanical animation), and animate the cockpit (instrument animation and ground animation).

None of the current Modelica implementations provides very advanced animation capabilities. *Dymola* offers the most complete set of tools also in this respect. It features a convenient to use automatic 3D mechanical animation. However, if the modeler wishes his bodies to assume more realistic shapes, the programming becomes cumbersome. There is no full graphics support yet for doing so. *Dymola* doesn't offer instrument animation, although this shortcoming can be circumvented fairly easily by re-interpreting each instrument as a simple 3D mechanical system. *Dymola* doesn't offer a link to a 3D database yet for realistic ground animation.

In spite of the impressively large standard library available with Modelica, a mechatronics systems modeler will invariably face the situation that some component models that he or she needs are missing. Thus, modelers must be prepared to create their own component models.

To this end, the modelers need tools that enable them to create component models safely and reliably. In particular, the modeler should be able to model his new component models down as far as possible using graphical tools. Equations are to be used as the tool of last resort.

This is where bond graphs fit in. Bond graphs are an object-oriented graphical modeling methodology representing power flows through a physical system. They are the most primitive graphical modeling methodology that is still fully object oriented. Any graphical representation that is more primitive than a bond graph, such as a block diagram or a signal flow graph, is no longer object oriented. For example, when I exchange a capacitor by a resistor in an electrical circuit, I can exchange the capacitor by a resistor also in the corresponding bond graph. Yet, two block diagrams representing the two circuits will look entirely different. They don't preserve the topological information of the circuit.

Bond graphs are generic, i.e., they can be used to represent power flows in any physical system. All physical systems with lumped parameters can be described by bond graphs. Furthermore, there are only a small number of graphical bond graph modeling elements needed to describe all physical systems. For this reason, most component models can be composed from bond graphs without need to resorting to equations at all. The equations are all hidden underneath the graphical bond graph primitives.

However, bond graphs are not convenient for the description of complex mechanical systems. When we wish to animate a multi-body system, the animation must be associated with the articulations (such as prismatic, revolute, and spherical joints), and not with individual bonds. A bond graph is at the wrong granularity level for associating an animation model with it.

For this reason, it is important to be able to hide bond graphs inside higher-level graphical representations, such as a multi-body library. To this end, we need to employ wrapping techniques to wrap the bond graph tightly inside a higher-level component model located at a coarser granularity level.

In this presentation, I shall demonstrate how complex mechatronics systems can be composed from wrapped bond graphs using *Modelica* and, more explicitly, using *Dymola*.

[1] http://www.modelica.org/
[2] http://www.modelica.org/news_items/libraries/Modelica/releases/3.0.1
[3] http://www.dynasim.se/index.htm
[4] http://www.ida.liu.se/labs/pelab/modelica/OpenModelica.html

François E. Cellier received his BS degree in electrical engineering in 1972, his MS degree in automatic control in 1973, and his PhD degree in technical sciences in 1979, all from the Swiss Federal Institute of Technology (ETH) Zurich. Dr. Cellier worked at the University of Arizona as professor of Electrical and Computer Engineering from 1984 until 2005. He recently returned to his home country of Switzerland. Dr. Cellier's main scientific interests concern modeling and simulation methodologies, and the design of advanced software systems for simulation, computer aided modeling, and computer-aided design. Dr. Cellier has authored or co-authored more than 200 technical publications, and he has edited several books. He published a textbook on Continuous System Modeling in 1991 and a second textbook on Continuous System Simulation in 2006, both with Springer-Verlag, New York.