An Easy-to-use Generic Model Configurator for Models in Multiple Tool Formats

Vadim Engelson[1], Peter Fritzson[1], Ulf Sellgren[2]

[1]Department of Computer and Information Science
Linköping University
SE-58183 Linköping, Sweden
{petfr,vaden}@ida.liu.se
http://www.ida.liu.se/~pelab

[2]KTH - Department of Machine Design
Royal Institute of Technology
SE-10044 Stockholm, Sweden
ulfs@md.kth.se

ABSTRACT

Application models for simulation are usually built by combining and configuring a selection of component models into an application model. This can be done by manually editing the model text representations. However, it is usually much easier to use a graphic editor for selecting och connecting components. The MathModelica tool has such a graphic editor, adapted for components in Modelica libraries. However, since Modelica is a generic model description language, it has been possible to extend the editor to support configuration of components in other modeling tool formats, e.g. ADAMS or ANSYS. When the editor is used in this generic way, the Modelica connections and interfaces are translated into the format of the specific tool. The editor does not need to know anything about the internals of model components that are combined, just the interfaces. The configurator has successfully been used to configure and simulate mechanical bearing models as well as a wheel loader model, currently supporting three different formats: Modelica MBS, ADAMS, and ANSYS, with ongoing development for the SKF BEAST MBS model format

KEYWORDS

Modelica, Configuration, Multi-Body Dynamics, MathModelica, Equation-based, FEM

1 INTRODUCTION

This paper describes generic configuration of simulation applications based on model components using a generalized GUI tool as extensions of MathModelica [3] based on the Modelica standard [2], [5], extended to handle domain-specific models not expressed in Modelica. For non-Modelica models, configuration and connection are done at the meta level, using a drag-and-drop GUI to specify model components and connections. How this can work for non-Modelica simulation tools? The reason is that the graphical model design and topology (in the Modelica standard format) is interpreted differently. Components are interpreted as components in another tool; connections are interpreted as data communication between those components in the other tool. However, such interpretation is only possible for coarse-grained components, at the “meta-level”.

2 THE MATHMODELICA GRAPHIC CONFIGURATION EDITOR

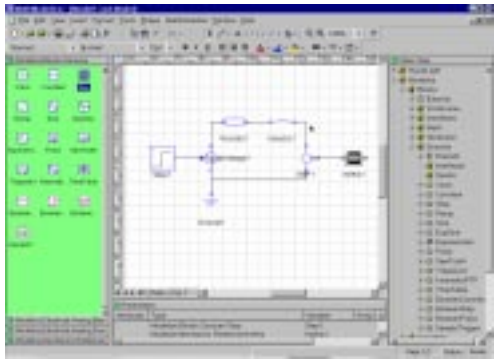

Figure 1. The MathModelica graphic model editor showing a simple electro-mechanical DC-motor model.

The standard MathModelica graphic model editor allows picking components from the library windows to the left, dragging these components icons into the drawing area in the middle, and connecting these by lines that represent communication or attachment between the components. The configuration editor we have developed as en extension works in a similar manner.

3 CONNECTING WHEEL LOADER COMPONENTS

A special library of wheel loader components was created, with model components either in Modelica, in ADAMS, or in ANSYS. A general graphic configuration editing tool has been developed, for configuration of components in either of the three model formats with connections representing physical interactions.

The graphic configuration editor maps icons and graphic connections to a Modelica model representation which is already standardized according to [5] or [2], and defines strongly typed, semantically well-specified connections. This Modelica connection model format is subsequently translated to ADAMS or ANSYS script file representation. To summarize:

- For Modelica, the mapping from icons and graphic connections to a Modelica model representation is already standardized according to [5] and [2]. The wheel loader lifting mechanism components are mapped to primitive components in the Modelica mechanical Multi-Body System (MBS) library [5] for simulation.
- For ADAMS, the editor maps connections to small pieces of ADAMS command script code, and each component to a corresponding file containing a single ADAMS part component.
- For ANSYS, the editor maps each connection to an ANSYS connection model that properly connects the related groups on nodes, i.e., the mating features. Each component is mapped to a corresponding file containing a single ANSYS submodel corresponding to a component. The total aggregated FEM model resulting from connecting the component models is a higher order ANSYS script file that sequentially loads and orients the configured component models.

Note that configuring components based on Modelica MBS or ADAMS model components results in a dynamic multi-body simulation, whereas in the third case a completely different kind of simulation is obtained, i.e., a FEM-based simulation computing distributed physical variable such as tension, etc.

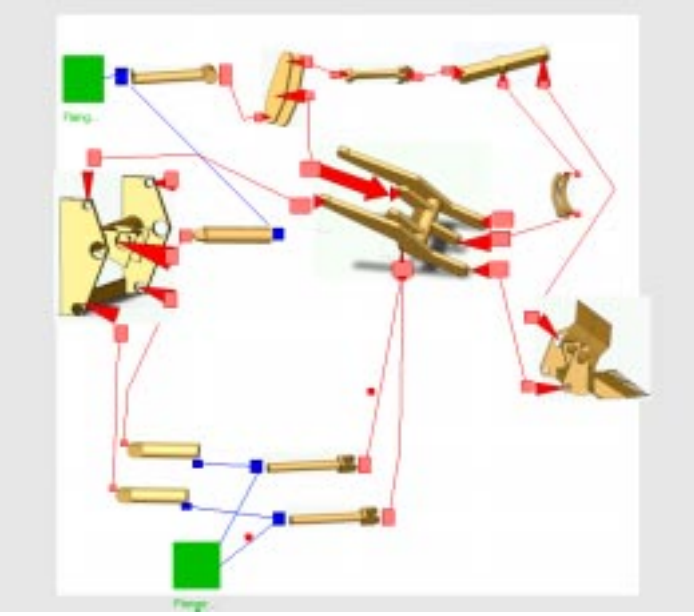

Figure 2. Connected components from the wheel-loader library, forming a wheel-loader lifting unit model.

In Figure 3 below the complete wheel loader lifting mechanism is shown in a 3D visualization. The obtained simulation and visualization generated from the complete configured wheel loader lifting unit model can be run and controlled in real-time.

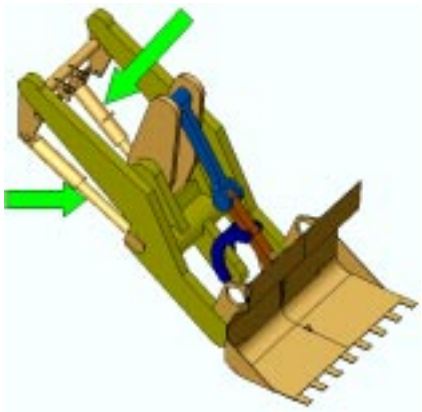

Figure 3. Graphic 3D view of the configured wheel loader lifting unit mechanism.

4 CONFIGURING COMPONENTS VIA CONNECTIONS

In the following we will primarily discuss components, connections, and connectors in some more detail, with emphasis on connections. Modelica is used as a general system model configuration language.

Components in Modelica models are simply instances of Modelica classes. Those classes should have well-defined interfaces, sometimes called ports, in Modelica called connectors, for coupling between a component and the rest of a system – e.g. the wheel loader lifting unit. Each rectangle (or other shape) in a connection diagram, e.g. depicted in Figure 2 and Figure 4, represents a physical component such as a body, a bar, a joint, etc.

The connections represented by *lines* in the diagram correspond to real, physical connections. For example, connections can be realized by mechanical connections, by pipes for fluids in flow systems, by heat exchange between components, etc.

The connectors, i.e., interface points, are shown as small *square dots* on the shapes in the diagrams of Figure 2 and Figure 4. Variables at such interface points represent the interaction between the component and the rest of the system.

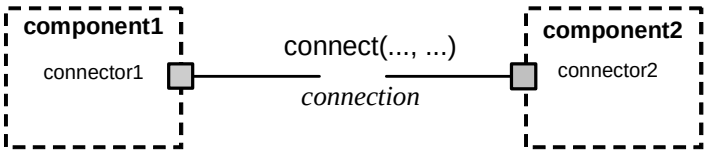

Figure 4. Connection between two connectors, i.e., ports, on components via a connection depicted as a line and realized as a connect statement: `connect(component1.connector1, component2.connector2)`.

Connections between components can be established between connectors of equivalent type. Modelica supports equation-based acausal connections, which means that connections are realized as equations. For acausal connections, the direction of data flow in the connection need not be known. If needed, causal connections with a flow direction can be established by connecting a connector with an `output` attribute to a connector declared as `input`.

Two types of coupling can be established by connections depending on whether the variables in the connected connectors are nonflow (default), or declared using the `flow` prefix:

1. Equality coupling, for nonflow variables, according to Kirchhoff's first law.
2. Sum-to-zero coupling, for flow variables, according to Kirchhoff's current law.

For mechanical systems such as the wheel loader lifting unit, the flow variables are typically forces and nonflow variables are positions.

Not all connections between connectors are legal. The types of the connectors to be connected must agree. Moreover, certain tool or library-specific conventions apply to the way mate couplings, frame connections, etc., can be legally connected in models based on components from a particular library or tool such as Modelica MBS, ADAMS, or ANSYS.

5 CREATING COMPONENT ICONS WITH CONNECTOR MARKERS

One issue concerns the creation of pictorial component representations, i.e. icons, for use in the connection diagrams.

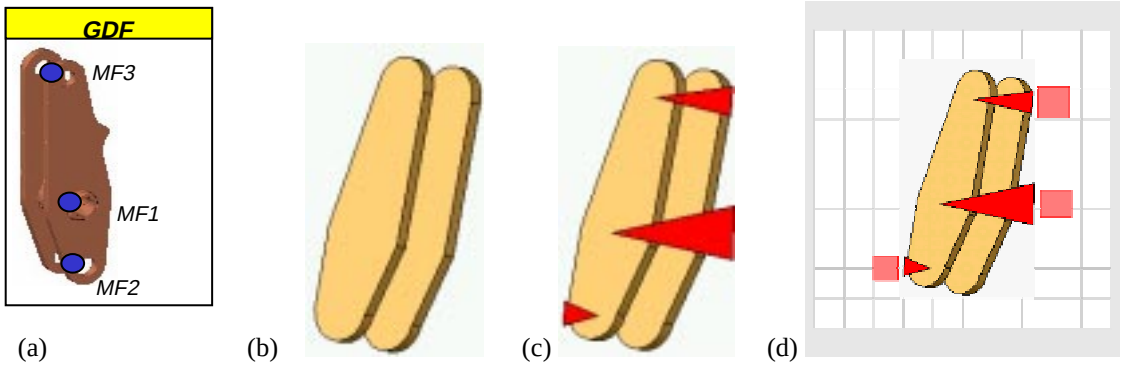

Figure 5. (a): A 3D image from a CAD tool with joints marked. (b): The 3D image icon exported from CAD. (c): Same image with joint positions marked at tips of arrows. (d) Image with rectangular connector icons.

An icon needs to be defined for each component. For mechanical components of a known mechanism the icon can be produced as a screen-dump from a 3D visualization or using a CAD tool.

Joint positions are *points*. The connectors are represented by rectangles. In order to emphasize joint positions for the body in Figure 5, red arrows have been added. This represents a single graphical entity (icon) that can be used in the diagram editor, and is specified in Modelica as follows:

```
partial model GDF_icon
  annotation(Diagram(Bitmap(extent=[-80, 80; 80, -80], name="GDF_icon.gif")),
             Icon(Bitmap(extent=[-80, 80; 80, -80], name="GDF_icon.gif")) );
end GDF_icon;
```

The three connectors used with these components are described with their positions specified via the extent attribute. Here coordinates are presented as pairs [x1,y1; x2,y2] where x is measured left-to right in an [-100,100] interval, Y is measured top-to-down in the same interval.

```
model GDF_comp
  extends GDF_icon;
  matingconnector mc1  annotation(extent=[-62, -70; -42, -50]);
  matingconnector mc2  annotation(extent=[ 50, -28;  76, -6]);
  matingconnector mc3  annotation(extent=[ 48,  38;  74, 64]);
end GDF_comp;
```

6 SIMULATION TOOL ATTRIBUTES FOR COMPONENTS

The kind of simulation tool associated with the component when it appears in the diagram is indicated by inheriting partial classes `AdamsPart`, `AnsysPart`, `ModelicaPart` with associated attributes and icons.

Each simulation code (with simulation code filename) is represented as a separate Modelica model. It has its own graphical icon (which can be generated automatically from extra information about the model). In addition to filename, the model can store arbitrary additional numerical and textual attributes that can be useful when the "complete", "top-level code" simulation file is generated.

```
model GDF_Ansys_Part
  extends AnsysPart(
    filename="GDFBeams.ans",
    orientation="PHMGDFTM.txt");
  extends GDF_comp;
  matingfeature mf1(…);
  matingfeature mf2(…);
  ...
end GDF_Ansys_Part;
```

```
model GDF_Adams_Part
  extends AdamsPart(
    filename="GDF.cmd",
    part="GDF");
  extends GDF_comp;
  matingfeature mf1(…);
  matingfeature mf2(…);
  ...
end GDF_Adams_Part;
```

```
model GDF_Modelica_Part
  extends ModelicaPart(
    filename="GDF.mo");
  extends GDF_comp;
  matingfeature mf1(…);
  matingfeature mf2(…);
  ...
end GDF_Modelica_Part;
```

7 ANSYS

An FEM submodel is a discretized representation of a product model component with a specified orientation in space that is stored in the ANSYS proprietary script format. The interactions between two FEM submodels is modeled with an ANSYS connection model.

An ANSYS connection model is a self-contained ANSYS model, e.g. a group of contact elements or sets of linear equations that connects pairs of higher order mating features on two different submodels.

Mating features as currently defined as groups of nodal entities that are defined within each FEM submodel. Node numbers, property ranges and the like for an ANSYS submodel or interface feature are stored as source file metadata, e.g. [6].

Parts of the generated ANSYS script files for the wheel loader lifting unit configuration are shown below:

```
! load submodels
   *use,orient,'PHMGDFTM','txt'
   *use,load,'GDFBeams','ans'
   *use,orient,'PHMGHTM','txt'
   *use,load,'GHBeams','ans'
   *use,orient,'PHMBITM','txt'
   *use,load,'BIBeams','ans'
   *use,orient,'PHMHIJTM','txt'
   *use,load,'HIJBeams','ans'
```

```
! connect submodels
   *use,connect,'revolute','gdfmfgh','ghmfgdf'  ! gdf-gh
   *use,connect,'revolute','ghmfhij','hijmfgh'  ! gh-hij
   *use,connect,'revolute','bimfhij','hijmfbi'  ! bi-hij
```

8 ADAMS

When used in ADAMS connectors need to have several special attributes, e.g. as follows:

```
matingfeature mf3(matetype="rotate", altmatetype="spherical")
```

Here `matetype` and/or `altmatetype` can be "`rotate`", "`fixed`" or "`spherical`". The type of connect used when resulting code is emitted is computed internally partly based on such information. Some generated ADAMS script code is shown below:

```
file command read file="C:/Demonstrator/init_ground.cmd"
```

```
file command read file="C:/Demonstrator/Lift_frame.cmd"
file command read file="C:/Demonstrator/tvarbalk.cmd"
file command read file="C:/Demonstrator/Liftcyl.cmd"
...
! Read marco files
macro read macro_name="connect spherical" file_name="C:/Demonstrator/connect_sph.mac"
macro read macro_name="connect revolute" file_name="C:/Demonstrator/connect_rev.mac"
...
!Connect Lift Frame to ground
connect revolute jname=REV1 Part1=ground Part2=Lift_Frame Part1marker=MARKER_MF2 Part2marker=MARKER_MF2
connect revolute jname=REV2 Part1=ground Part2=Lift_Frame Part1marker=MARKER_MF1 Part2marker=MARKER_MF1
!Connect Lift cylinder
connect revolute jname=REV3 Part1=ground Part2=Lift_piston Part1marker=MARKER_MF4 Part2marker=MARKER_MF1
...
```

9 CONCLUSIONS

This general configuration approach to modeling has demonstrated that it is indeed possible to have an easy-to-use general model configurator based on a general and well specified model representation format such as Modelica or ModelicaXML, where the configurator can translate the configuration information to other formats and tools such as ANSYS and ADAMS. In this case the translation was from a graphic configuration to Modelica, and for ANSYS and ADAMS via ModelicaXML to the respective proprietary format.

This configuration approach to FEM modeling, here with ANSYS, assisted by the presented generic model configurator, enables most practical finite element models of technical systems to be created in a time linear to the number of FEM submodels and interface features [6].

The fact that the time to configure a FEM model of a system is almost independent of the size of the submodels indicates that the presented modeling method and tool reduces FEM modeling complexity.

Similar conclusions can be drawn for other kinds models, such as the multi-body system models configured for simulation with Modelica MBS or ADAMS.

10 ACKNOWLEDGEMENTS

Kjell Andersson contributed the ADAMS demonstration example with script code. Daniel Hedberg and Peter Aronsson made important contributions to the design and implementation of the graphic configuration tool, with input from Dag Fritzson and Alexander Siemers at SKF. Partial support for this work was received from Vinnova in the VISP project. The development of the graphic configuration tool has also been partially supported by SKF and by MathCore Engineering AB.

REFERENCES

[1] Engelson, Vadim, Peter Bunus, Lucian Popescu, and Peter Fritzson. Mechanical CAD with Multibody Dynamic Analysis Based on Modelica Simulation. In *Proceedings of the 44th Scandinavian Conference on Simulation and Modeling* (SIMS'2003), available at www.scan-sims.org. Västerås, Sweden. September 18-19, 2003.

[2] Fritzson, Peter. *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*. Wiley-IEEE Press, ISBN 0-471-471631, 2003.

[3] MathCore Engineering AB. *MathModelica User's Guide*. www.mathcore.com, 2003.

[4] MathCore Engineering AB. Internal document draft. *MathModelica CAD Integrator Users Guide*. www.mathcore.com, 2003.

[5] Modelica Association. *A Unified Object-Oriented Language for Physical Systems Modeling: Language Specification Version 2.1*. Available at http://www.modelica.org, April, 2004.

[6] Sellgren, Ulf. Architecting models of technical systems for non-routine simulations, In *Proceedings of the 14th International Conference on Engineering Design* (ICED 03), ISBN : 1-904670-00-8, Stockholm, Sweden, August 19-21, 2003.

[7] Tiller, Michael. *Introduction to Physical Modeling with Modelica*. Kluwer Academic Publishers 2001, ISBN 0-7923-7367-7.