"D4.2.15 – Prototype for multi-model simulation with FMI TLM co-simulation in OpenModelica"

"Sub WP 4.2: Prototypes of tool support for multi-mode DAE systems"

"Work Package 4: Systems with multiple operating modes"

Version 1.0

Date 2016-04-22

Authors

Adeel Asghar	SICSEast
Alachew Mengist	LiU
Adrian Pop	LiU

1. Short Summary

The goal of this task is to develop an open source prototype for multi-model simulation with FMI TLM co-simulation in OpenModelica.

In the attached paper we present a general open-source graphical editor and simulation tool for composite modeling and simulation as well as its integration with SKF's TLM-based co-simulation framework for TLM based co-simulation and the OpenModelica FMI co-simulation. In the context of this work a composite model is composed of several sub-models including the interconnections between these sub-models. The graphical editor presented in this paper enables users to create a composite model in XML.

The graphical editor combines a number of features to support end-users with respect to the creation of composite models and co-simulation. These include adding, removing, and connecting components (sub-models) both textually and graphically, as well as integrated co-simulation and visualization of simulation results.

The full graphical functionality of the composite modeling process can be expressed in the following steps:

- Import and add External non-Modelica models such as Matlab/SimuLink, Adams, and BEAST models to the composite model editor,
- External Modelica models such as Dymola and Wolfram SystemModeler models
- Specify startup methods and interfaces of the external model,
- Build the composite models by connecting the external models,
- Set the co-simulation and TLM parameters in the composite model.
- Simulate the composite model using TLM based co-simulation

2. Paper Abstract

A common situation in industry is that a system model (here a composite model) is composed of several sub-models which may have been developed using different tools. FMI is one important technology for exporting/importing models between tools and/or connecting them via co-simulation. TLM based modeling and co-simulation is another important technique for modeling, connecting, and simulation of especially mechanical systems, which is simple, numerically stable, and efficient. A number of tool-specific simulation models, such as Modelica models, SimuLink models, Adams models, BEAST models, etc., have successfully been connected and simulated using TLM based co-simulation. However, previously there was no general open source tool for creation, graphic editing, and simulation of composite models connected via FMI or TLM based co-simulation. In this paper we presented a graphical composite model editor, shown in Figure 1, based on OpenModelica which is integrated with the OpenModelica and the SKF TLM co-simulation frameworks to support both FMI and TLM based composite model editing and simulation.

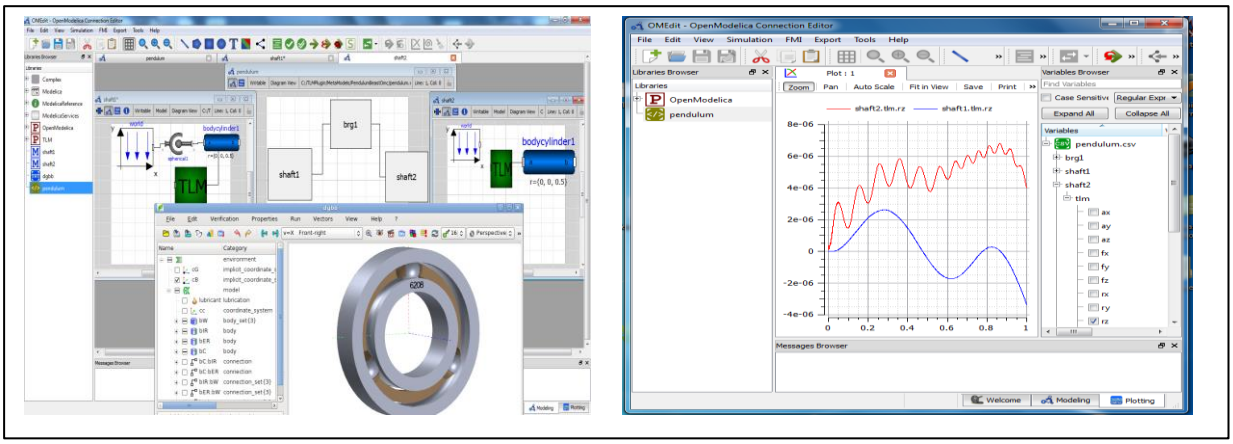

Figure 1. Graphical composite model editor.

The editor supports creating, viewing and editing a composite model both in textual and graphical representation. The system supports simulation of composite models consisting of sub-models created using different tools.

3. References

[1] Alachew Mengist, Adeel Asghar, Adrian Pop, Peter Fritzson, Willi Braun, Alexander Siemers and Dag Fritzson. An Open-Source Graphical Composite Modeling Editor and Simulation Tool Based on FMI and TLM Co-Simulation. In Proceedings of the 11th International Modelica Conference, Versailles, France, September 21-23, 2015.

An Open-Source Graphical Composite Modeling Editor and Simulation Tool Based on FMI and TLM Co-Simulation

Alachew Mengist[1], Adeel Asghar[1], Adrian Pop[1], Peter Fritzson[1], Willi Braun[2], Alexander Siemers[3], Dag Fritzson[3]

[1]PELAB – Programming Environment Lab, Dept. Computer Science, Linköping University, Sweden, {alachew.mengist,adeel.asghar,adrian.pop,peter.fritzson}@liu.se
[2]Dept. Mathematics and Engineering, University of Applied Sciences, Germany, willi.braun@fh-bielefeld.de
[3]SKF, Göteborg, Sweden, {alexander.siemers,dag.fritzson}@skf.com

Abstract

A common situation in industry is that a system model (here a composite model) is composed of several sub-models which may have been developed using different tools. FMI is one important technology for exporting/importing models between tools and/or connecting them via co-simulation. TLM based modeling and co-simulation is another important technique for modeling, connecting, and simulation of especially mechanical systems, which is simple, numerically stable, and efficient. A number of tool-specific simulation models, such as Modelica models, SimuLink models, Adams models, BEAST models, etc., have successfully been connected and simulated using TLM based co-simulation. However, previously there was no general open source tool for creation, graphic editing, and simulation of composite models connected via FMI or TLM based co-simulation. In this paper we present a graphical composite model editor based on OpenModelica which is integrated with the OpenModelica and the SKF TLM co-simulation frameworks to support both FMI and TLM based composite model editing and simulation. The editor supports creating, viewing and editing a composite model both in textual and graphical representation. The system supports simulation of composite models consisting of sub-models created using different tools.

Keywords: Graphic Editing, Composite Modeling, Modelica, FMI, TLM, XML, Simulation, Co-Simulation

1 Introduction

Industrial products often consist of many components which have been developed by different suppliers using different modeling and simulation tools. Modeling and simulation support is needed in order to integrate all the parts of a complex product model. FMI (Blochwitz et al, 2011), (FMI-Standard.org, 2014), both model exchange and co-simulation, is one such important technology. TLM (Transmission Line Modeling) based co-simulation (Siemers et al, 2005), is another important technology, which is simple, numerically stable, and efficient.

This has successfully been demonstrated by integrating and connecting several different simulation models, especially for mechanical applications. Such an integrated model consisting of several model parts is here called a *composite model* since it is composed of several sub-models. Another name used for such a model is *meta-model*, since it is a model of models. In earlier work (Siemers et al, 2005), (Siemers and Fritzson, 2006) Modelica (Fritzson, 2014) with its object oriented modeling capabilities and its standardized graphical notations has demonstrated the possibilities for meta-modeling/composite modeling of mechanical systems using TLM.

The availability of a general XML-based composite modeling language (Siemers et al, 2005) is an important aspect of our FMI and TLM based modeling and co-simulation framework. However, modelers developing composite models are likely to take advantage of the additional availability of tools that assist them with respect to the composite modeling process (i.e., the process of creating and/or editing a composite model, here represented and stored as XML).

We introduce a graphical composite model editor which is an extension and specialization of the OpenModelica connection editor OMEdit (Asghar et al, 2010). In the context of this work a composite model is composed of several sub-models including the interconnections between these sub-models. The graphical editor presented in this paper enables users to create a composite model in XML. It is also integrated with the OpenModelica FMI-based model-exchange and co-simulation and the SKF TLM-based co-simulation framework.

This paper is structured as follows. In Section 2 a brief background of OpenModelica is given along with an overview of the SKF TLM based co-simulation framework. Section 3 defines the different modes of operation. The composite model XML schema is explained in Section 4. In Section 5 the composite model editor and an overview of its interaction with the other system components are discussed. An industrial application is shown in Section 6. Conclusions and future work are presented in Section 7.

2 Background

2.1 OpenModelica

OpenModelica is an open source Modelica-based platform for modeling, simulation, and optimization. The OpenModelica connection editor OMEdit is a graphical Modelica model editing tool. It supports model creation, textual and graphical model editing including connections drawing, simulation and plotting. This editor has been extended to become a composite model editor as described in this paper.

2.2 FMI-Based Model Exchange and Co-Simulation Framework in OpenModelica

OpenModelica currently supports FMI 1.0 and FMI 2.0 for model exchange and most of FMI 2.0 for co-simulation. Other tools can access this functionality e.g., by dynamically linking OMC or by invoking it using a message-passing interface.

The OpenModelica graphic editor and simulator supports connection of several FMUs into a composite model and simulating this model.

Simulation of composite models with algebraic loops involving FMUs is also supported.

2.3 TLM (Transmission Line Modeling)

The TLM (Transmission Line Modeling) technique is based on the fact that propagation of signals takes time in many physical systems, e.g., propagation of pressure waves in hydraulic systems, sound wave propagation, and force propagation in mechanical systems. Thus there is a certain physical communication delay between different parts of the system which can be used to partially de-couple these parts and allow them to be independently simulated and coupled in a numerically stable way via co-simulation techniques.

The TLM approach has been known since a long time (Auslander, 1968), (Burton et al, 1993), (Johns and O'Brien 1980). A general presentation of the theory and methods of TLM-based simulation is given in (Krus et al, 1990).

The HOPSAN (HOPSAN, 1985) software is one of the first general TLM implementations with its own graphical modeling language. A newer version, HOPSAN-NG, (Axin et al, 2010), has recently been developed.

Moreover, a TLM implementation for the Modelica language has recently been developed as part of OpenModelica (Chapter 7, Sjölund, 2015).

2.4 TLM-Based Co-Simulation Framework

As mentioned, a general framework for composite model based co-simulation has previously been designed and implemented (Siemers et al, 2005). The design goals for the simulation part of that framework were portability, simplicity to incorporate additional simulation tools, and computational efficiency. It is also the framework used for TLM-based composite model co-simulation described in this paper.

The TLM composite model co-simulation is primarily handled by the central simulation engine of the framework called the TLM simulation manager. It is a stand-alone program that reads an XML definition of the coupled simulation as defined in (Siemers et al, 2005). It then starts external model simulations and provides the communication bridge between the running simulations using the TLM (Nakhimovski, 2006) method.

The external models only communicate with the TLM simulation manager which acts as a broker and performs communication and marshalling of information between the external models. The simulation manager sees every external model as a black box having one or more external interfaces. The information is then communicated between the external interfaces belonging to the different external models. Additionally the simulation manager opens a network port for monitoring all communicated data.

TLM simulation monitor is another stand-alone program that connects to the TLM simulation manager via the network port. The TLM simulation manager sends the co-simulation status and progress to the TLM simulation monitor via TCP/IP. The simulation monitor receives the data and writes it to an XML file.

3 Modes of Operation

3.1 Textual Format

The user defines the list of sub-models with their corresponding connections as an XML file according to the composite model specification described in Section 4. The user should be able to easily save the file, load a new one, or edit the textual version even while using the GUI. The simulator reads the composite model XML file and performs the simulation.

3.2 Graphical User Interface Format

The user can define the list of sub-models and their connections using the GUI which allows them to drag and drop the sub-models to the editor and make connections between them. The GUI automatically generates the composite model code which is stored in an XML format described in Section 4.

4 Composite Model XML Schema

The composite model XML-Schema for validating the co-simulation composite model is designed according to its specification described in (Siemers et al, 2005). The following is a sample composite model XML representation:

```
<?xml version="1.0" encoding="ISO-8859-
1"?>
<Model Name="Pendulum">
```

```
  <SubModels>
    <SubModel Name="shaft1"
StartCommand="StartTLMOpenModelica"
ExactStep="0" ModelFile="shaft1.mo"
Position="0.0,0.0,0.0"
Angle321="0.0,0.0,0.0">
      <InterfacePoint Name="tlm"
Position="0.0,0.0,0.5"
Angle321="0.0,0.0,0.0"/>
    </SubModel>
    <SubModel Name="shaft2"
StartCommand="StartTLMOpenModelica"
ExactStep="0" ModelFile="shaft2.mo"
Position="0.0,0.0,0.5"
Angle321="0.0,0.0,0.0">
      <InterfacePoint Name="tlm"
Position="0.0,0.0,0.0"
Angle321="0.0,0.0,0.0"/>
    </SubModel>
  </SubModels>
  <Connections>
    <Connection From="shaft1.tlm"
To="shaft2.tlm" Delay="1e-4" Zf="1e4"
Zfr="1e2" alpha="0.2"/>
  </Connections>
  <SimulationParams StartTime="0"
StopTime="5"/>
</Model>
```

In order to use graphical notations in the composite model editor, the composite model XML file needs to describe annotations for each sub-model and connections between them. We propose to extend the composite model specification by including the `Annotation` element in the `SubModel` and `Connection` elements.

```
<Annotation Origin="{-50,54}" Extent="{-
10,-10,10,10}" Rotation="0"
Visible="true"/>
```

The contents of our composite model XML root element, namely `Model` is depicted in Figure 1. Inside the root element there can be a list of connected `SubModels` and TLM `Connections`. `SimulationParams` element is also inside the root element. It has an attribute `Name` representing the name of the composite model.

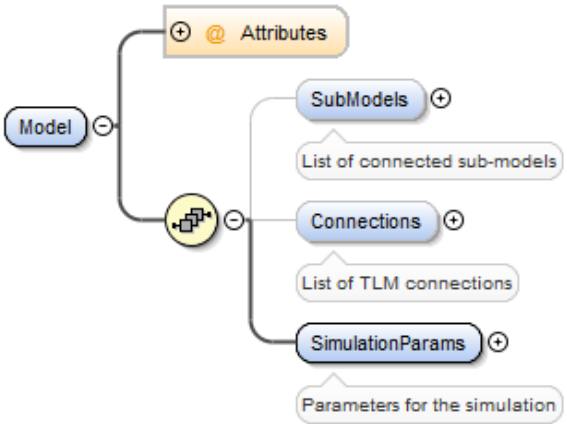

Figure 1. The Model (root) element of the Composite Model Schema.

The `SimulationParams` element specify the start time and end time for the co-simulation

The `SubModel` element, presented in Figure 2, represents the simulation model component that participates in the co-simulation. The required attribute for a `SubModel` are `Name` of the sub-model, `ModelFile` (file name of the submodel) and `StartCommand` (the start method command to participate in the co-simulation). Each `SubModel` also contains a list of interface points. `InterfacePoint` elements are used to specify the TLM interfaces of each simulation component (sub-model).

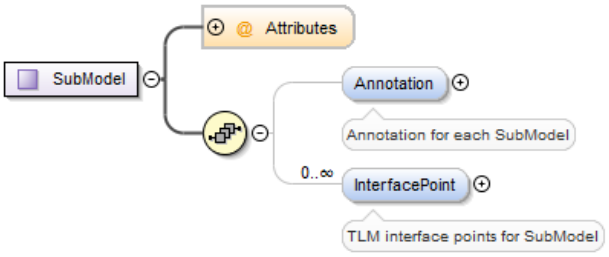

Figure 2. The SubModel element from the Composite Model Schema.

The `Connection` element of the composite model xml schema is shown in Figure 3.

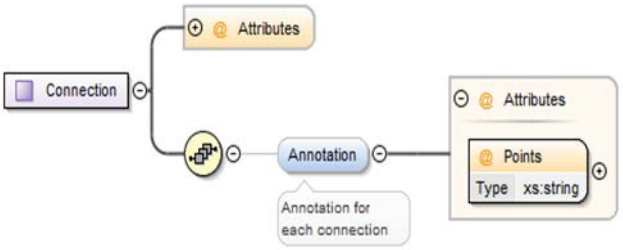

Figure 3. The Connection element from the Composite Model Schema.

The `Connection` element defines connections between two connected interface points, that is, a connection between two TLM interfaces. Its attributes `From` and `To` define which interface of which sub-models are connected. Other attributes of the `Connection` element specify the delay and maximum step size.

5 Composite Model Graphical Editor

One of the primary contributions of this effort is our focus on interoperability in modeling and simulation. Our effort leverage OpenModelica for graphical composite model editing as well as FMI support and SKF's co-simulation framework for TLM Based co-simulation.

As mentioned, the implementation of this graphical composite model editor is an extension of OMEdit (Asghar et al, 2010) which is implemented in C++ using the Qt graphical user interface library.

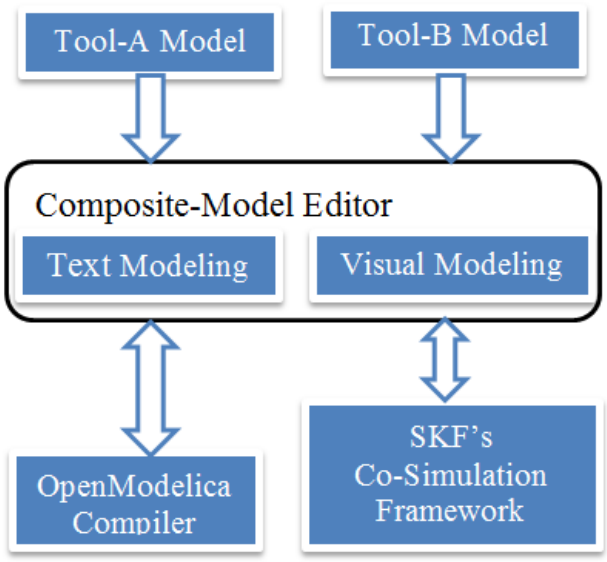

Figure 4. An overview of the interaction between the composite model (meta-model) graphic editor and the other components.

The full graphical functionality of the composite modeling process can be expressed in the following steps:

- Import and add the external models to the composite model editor,
- Specify startup methods and interfaces of the external model,
- Build the composite models by connecting the external models,
- Set the co-simulation and TLM parameters in the composite model.

An overview of the different components that the graphical composite model editor relies on is shown in Figure 4.

The graphical composite model editor communicates with the OpenModelica compiler to retrieve the interface points for the external model and SKF's co-simulation framework to run the TLM simulation manager and simulation monitor. Each tool component is descried in the following subsections.

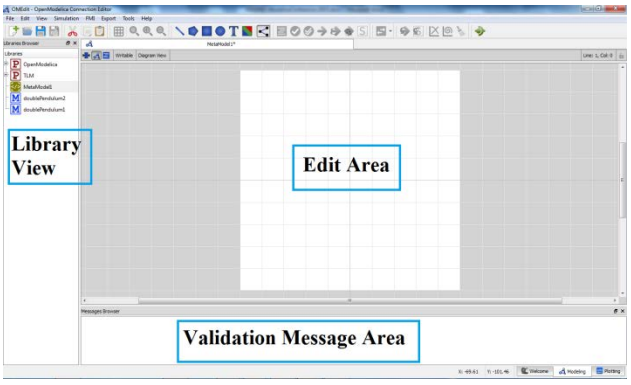

Figure 5. A screenshot of the modeling page area.

In the graphic composite model editor the modeling page area is used for visual composite modeling or text composite modeling. This allows users to create, modify, and delete sub-models. A screenshot of the modeling page area is shown in Figure 5.

5.1 Visual Modeling

Each composite model has two views: a Text view and a Diagram view. In the Diagram view, each simulation model component (sub-model) of the TLM co-simulation can be dragged and dropped from the library browser to this view, and then the sub-model will be automatically translated into a textual form by fetching the interface name for the TLM based co-simulation. The user can complete the composite model (see Figure 6) by graphically connecting components (sub-models).

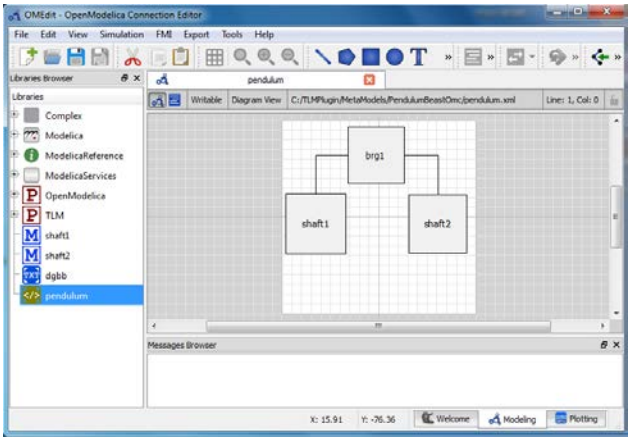

Figure 6. A screenshot of visual composite modeling after a connection has been made.

The test model (see Figure 7) is a multibody system that consists of three sub-models: Two OpenModelica `Shaft` sub-models (`Shaft1` and `Shaft2`) and one SKF/BEAST bearing sub-model that together build a double pendulum. The SKF/BEAST bearing sub-model is a simplified model with only three balls to speed up the simulation.

`Shaft1` is connected with a spherical joint to the world coordinate system. The end of `Shaft1` is connected via a TLM interface to the outer ring of the BEAST bearing model. The inner ring of the bearing model is connected via another TLM interface to `Shaft2`. Together they build the double pendulum with two shafts, one spherical OpenModelica joint, and one BEAST bearing.

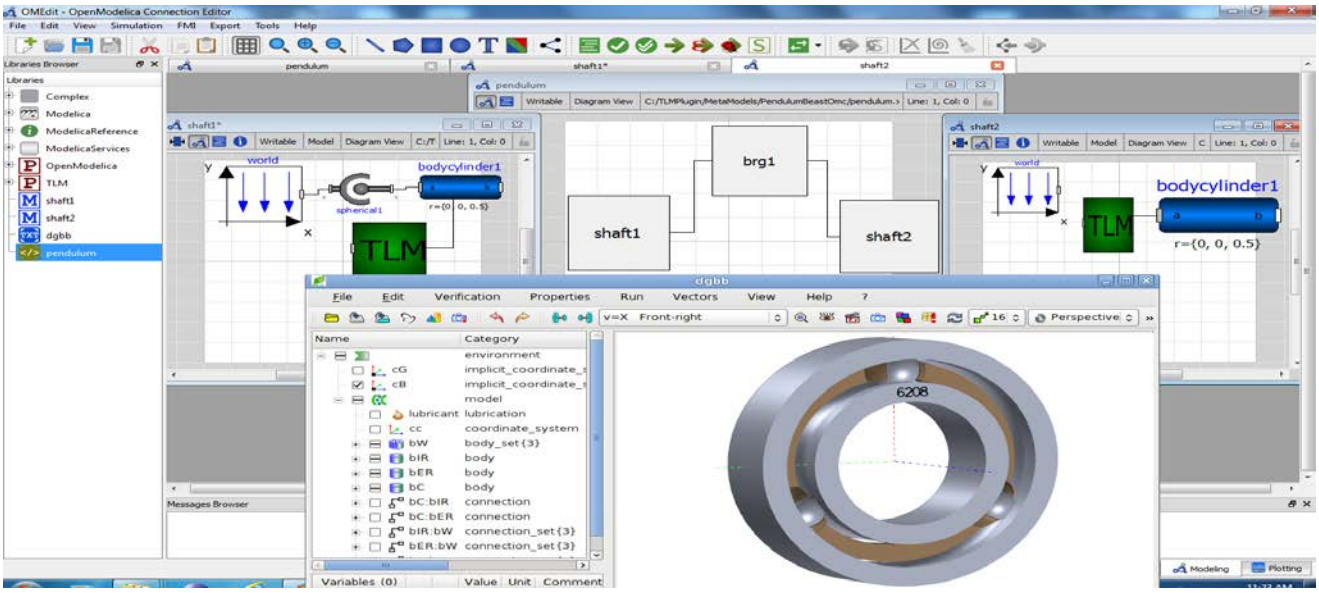

Figure 7. A screenshot of visual composite modeling of double pendulum.

5.2 Text Modeling and Viewing

The text view (see Figure 8) allows users to view the contents (sub-models, connections, and simulation parameters) of any loaded composite model. It also enables users to edit a composite model textually as part of the composite modeling construction process. To facilitate the process of textual composite modeling and to provide users with a starting point, the text view (see Figure 8) includes the composite model XML schema elements and the default simulation parameters.

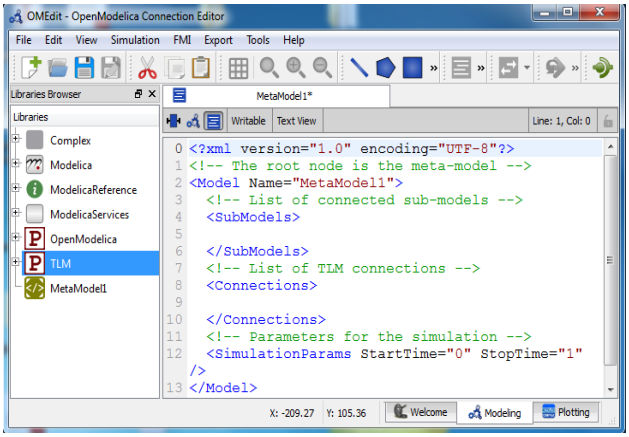

Figure 8. A screenshot of textual composite modeling.

5.3 Composite Model Validation

Since model validation is part of the composite modeling process the composite model editor (see Figure 9) supports users by validating the composite model to ensure that it follows the structure and content rules specified in the composite model schema described in Section 4. In general the composite model editor validation mechanism supports users to verify that:

- The basic structure of the elements and attributes in the composite model matches the composite model schema.
- All information required by the composite model schema is present in the composite model.
- The data conforms to the rules of the composite model schema.

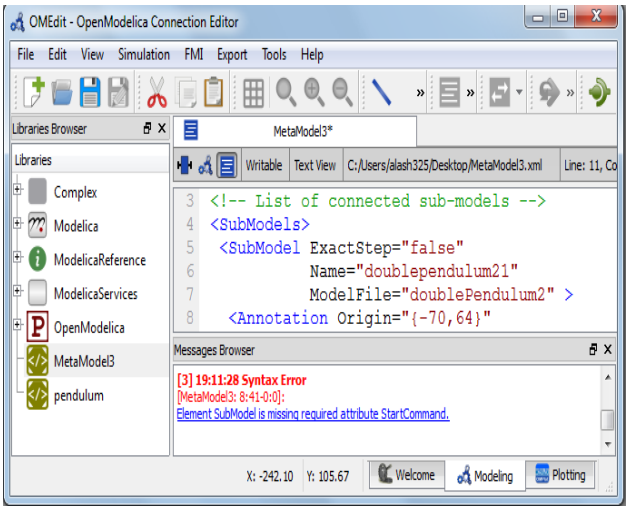

Figure 9. A screenshot of a composite modeling validation message.

5.4 OpenModelica Runtime Enhancement

To support TLM-based co-simulation the OpenModelica runtime has been enhanced. The added functionality supports single solver step simulation so that the executed simulation model can work together with the TLM manager. New flags to enable this functionality in the simulation executable are now available:

- -noEquidistantOutputFrequency
- -noEquidistantOutputTime

The new flags control the output, e.g., the frequency of steps and the time increment.

5.5 Communication with the SKF TLM Based Co-Simulation Framework

The graphic composite model editor in OpenModelica provides a graphical user interface for co-simulation of composite models. It can be launched by clicking the `TLM co-simulation` icon from the toolbar, see Figure 10.

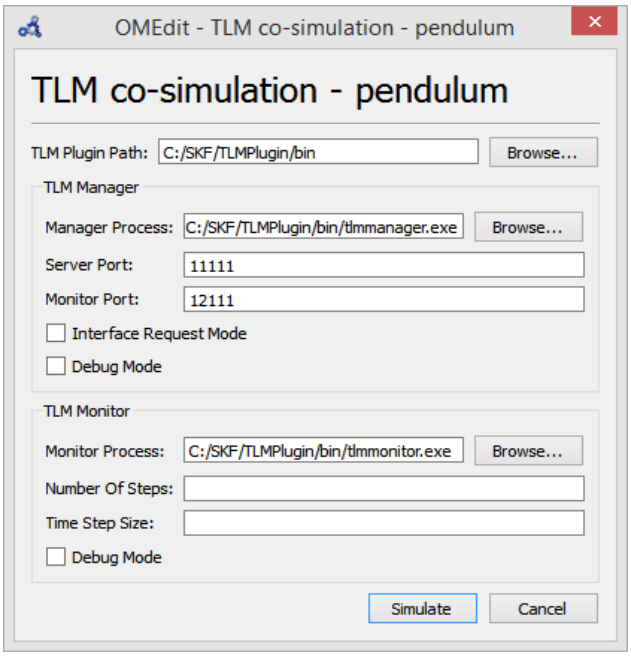

Figure 10. TLM co-simulation setup.

The editor runs the TLM simulation manager and simulation monitor. The simulation manager reads the composite model from the editor, starts the co-simulation, and provides the communication bridge between the running simulations. Figure 11 shows the running status of the TLM co-simulation.

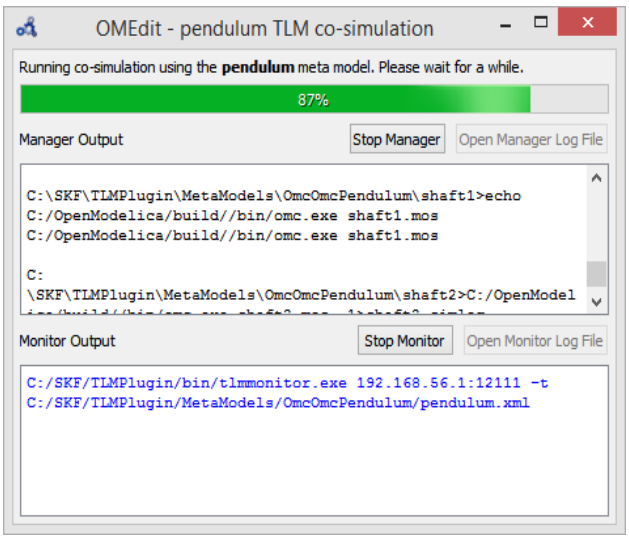

Figure 11. TLM co-simulation.

The simulation monitor communicates with the simulation manager and writes the status and progress of the co-simulation in a file. This file is read by the editor for showing the co-simulation progress bar to the user. The editor also provides the means of reading the log files generated by the simulation manager and monitor.

During the post-processing stage, simulation results are collected and visualized in the OMEdit plotting perspective as shown in Figure 12.

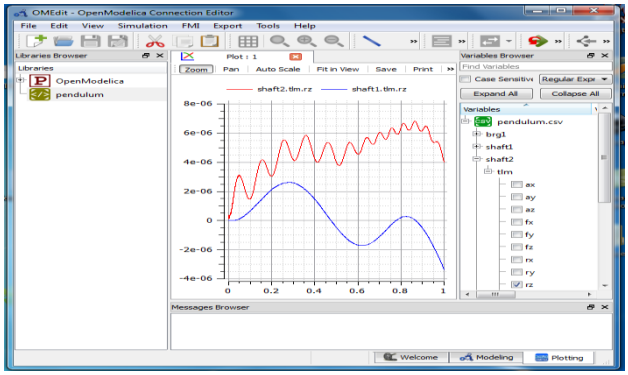

Figure 12. Results of TLM co-simulation.

6 Industrial Application of Composite Modeling with TLM Co-Simulation

SKF has successfully used the TLM co-simulation framework to simulate composite models. For example, Figure 13 shows one such application with an MSC.ADAMS (MSC-Software, 2015) car model containing an integrated SKF BEAST (Stacke, Fritzson, and Nordling, 1999) hub-unit sub-model connected via TLM-connections.

7 Conclusions and Future Work

This paper presents a general open-source graphical editor and simulation tool for composite modeling and simulation as well as its integration with SKF's TLM-based co-simulation framework for TLM based co-simulation and the OpenModelica FMI co-simulation.

The graphical editor combines a number of features to support end-users with respect to the creation of composite models and co-simulation. These include adding, removing, and connecting components (sub-models) both textually and graphically, as well as integrated co-simulation and visualization of simulation results.

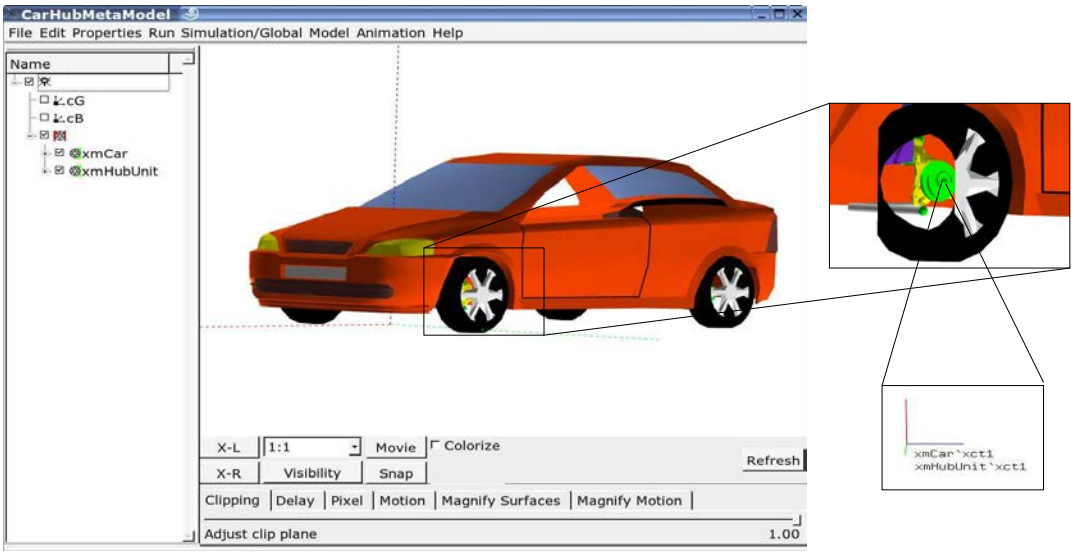

Figure 13. A composite model of an MSC.ADAMS car model with an integrated SKF BEAST hub-unit sub-model (green), connected via TLM connections for co-simulation.

The composite model editor is currently in an early stage of development but already supports external non-Modelica models represented in XML form (essentially black boxes with interfaces) inside the component tree which can be used for composite model composition.

Future development includes 3D visualization in composite modeling as well as being able to use both FMI-connections and TLM-connections in the same composite model, since they currently can only be used separately (either FMI or TLM within a specific composite model).

Future work also involves the development of a proposal for integrating TLM-based co-simulation as an option in the FMI standard, as well as participating in the standardization work in the SSP project in the Modelica Association (System Structure and Parameterization of Components for Virtual System Design abbreviated SSP), hopefully resulting in standardization of and extended version of the composite model XML schema.

A detailed comparison of TLM and FMI co-simulation is outside the scope of this paper. However, especially for mechanical applications TLM provides very simple and easy-to-use interface definitions, as well as in general numerically stable co-simulation. These are important reasons for continued use and improvement of TLM-based tools and future standardization and incorporation into the FMI standard.

Acknowledgements

The work has been supported by Vinnova in the ITEA2 MODRIO project, by EU in the INTO-CPS project, and by the Swedish Government in the Swedish Government in the ELLIIT project. The Open Source Modelica Consortium supports the OpenModelica work. The TLM based co-simulation framework is provided by SKF.

References

Adeel Asghar, Sonia Tariq, Mohsen Torabzadeh-Tari, Peter Fritzson, Adrian Pop, Martin Sjölund, Parham Vasaiely, and Wladimir Schamai. An Open Source Modelica Graphic Editor Integrated with Electronic Notebooks and Interactive Simulation. In *Proc. of the 8th International Modelica Conference 2011*, pp. 739–747. Modelica Association, March 2011.Linköping University, Sweden, 2010.

David M. Auslander. Distributed System Simulation with Bilateral Delay-Line Models. *Journal of Basic Engineering, Trans. ASME*: 195–200, 1968.

Mikael Axin, Robert Braun, Petter Krus, Alessandro dell'Amico, Björn Eriksson, Peter Nordin, Karl Pettersson, and Ingo Staack. Next Generation Simulation Software using Transmission Line Elements. In *Proceedings of the Bath/ASME Symposium on Fluid Power and Motion Control (FPMC)*, September 2010.

Torsten Blochwitz et al. The Functional Mockup Interface for Tool independent Exchange of Simulation Models. *In Proceedings of the 8th International Modelica Conference., Dresden, Mar. 2011.* doi: 10.3384/ecp11063105.

James D. Burton, Kevin A. Edge, and Clifford R. Burrows. Partitioned Simulation of Hydraulic Systems Using Transmission-Line Modelling. In *ASME WAM*, 1993.

FMI-Standard.org. Functional Mock-up Interface for Model Exchange and Co-Simulation Version 2.0, July 25, 2014. https://www.fmi-standard.org/.

Peter Fritzson. Principles of Object Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach. 1250 pages. *ISBN 9781-118-859124, Wiley IEEE Press*, 2014.

HOPSAN. The HOPSAN Simulation Program, User's Manual. *Linköping University, 1985*. LiTH-IKP-R-387.

Peter B. Johns and Mark O'Brien. Use of the transmission line modelling (TLM) method to solve nonlinear lumped networks. *The Radio and Electronic Engineer*, 50(1/2):59–70, 1980.

Petter Krus, Arne Jansson, Jan-Ove Palmberg, and Kenneth Weddfelt. Distributed Simulation of Hydromechanical Systems. In *Proc. of the Third Bath International Fluid Power Workshop*, 1990.

MSC-Software, MSC.ADAMS – interactive motion simulation software, http://www.mscsoftware.com (accessed: 22th of May 2015).

Iakov Nakhimovski. Contributions to the Modeling and Simulation of Mechanical Systems with Detailed Contact Analysis, *Dissertation No. 1009*, Linköpings universitet, Sweden, 2006.

Alexander Siemers, Iakov Nakhimovski, and Dag Fritzson. Meta-modelling of Mechanical Systems with Transmission Line Joints in Modelica. In *Proceedings of the 4th International Modelica Conference*, Hamburg, Germany, 2005.

Alexander Siemers, Peter Fritzson, and Dag Fritzson, Meta-Modeling for Multi-physics Co-simulations applied for OpenModelica. In: *Proc. of ANIPLA 2006 International Congress on 'Methodologies for Emerging Technologies in Automation'*, University of Rome La Sapienza, November 13–14–15, 2006.

Alexander Siemers and Dag Fritzson. A meta-modeling environment for mechanical system co-simulations. In *Proc. of the 48th Scandinavian Conference on Simulation and Modeling* (SIMS 2007), Gothenburg (Särö), Sweden, October 2007.

Alexander Siemers, Contributions to Modelling and Visualisation of Multibody Systems Simulations with Detailed Contact Analysis, *Dissertation No. 1337*, Linköpings universitet, Sweden, 2010

Lars-Erik Stacke, Dag Fritzson, and Patrik Nordling, BEAST—A Rolling Bearing Simulation Tool, Proc. Instn Mech. Engrs, part K, *Journal of Multi-body Dynamics*, 1999.