Composer - A prototype multilingual model composition tool

Erik Billing and Martin Servin

November 10, 2012

Abstract

Facing the task to design, simulate or optimize a complex system it is common to find models and data for the system expressed in different formats, implemented in different simulation software tools. When a new model is developed, a target platform is chosen and existing components implemented with different tools have to be converted. This results in unnecessary work duplication and lead times. The Modelica language initiative [2] partially solves this by allowing developers to move models between different tools following the Modelica standard. Another possibility is to exchange models using the Functional Mockup Interface (FMI) standard that allows computer models to be used as components in other simulations, possibly implemented using other programming languages [1]. With the Modelica and FMI standards entering development, there is need for an easy-to-use tool that supports design, editing and simulation of such multilingual systems, as well as for retracting system information for formulating and solving optimization problems.

A prototype solution for a graphical block diagram tool for design, editing, simulation and optimization of multilingual systems has been created and evaluated for a specific system. The tool is named *Composer* [3].

The block diagram representation should be generic, independent of model implementations, have a standardized format and yet support efficient handling of complex data. It is natural to look for solutions among modern web technologies, specifically HTML5. The format for representing two dimensional vector graphics in HTML5 is Scalable Vector Graphics (SVG). We combine the SVG format with the FMI standard. In a first stage, we take the XML-based model description of FMI as a form for describing the interface for each component, in a language independent way. Simulation parameters can also be expressed on this form, and integrated as metadata into the SVG image.

The prototype, using SVG in conjunction with FMI, is implemented in JavaScript and allow creation and modification of block diagrams directly in the web browser. Generated SVG images are sent to the server where they are translated to program code, allowing the simulation of the dynamical system to be executed using selected implementations. An alternative mode is to generate optimization problem from the system definition and model parameters. The simulation/optimization result is

returned to the web browser where it is plotted or processed using other standard libraries.

The fiber production process at SCA Packaging Obbola [4] is used as an example system and modeled using Composer. The system consists of two fiber production lines that produce fiber going to a storage tank [5]. The paper machine is taking fiber from the tank as needed for production. A lot of power is required during fiber production and the purpose of the model was to investigate weather electricity costs could be reduced by rescheduling fiber production over the day, in accordance with the electricity spot price. Components are implemented for dynamical simulation using OpenModelica and for discrete event using Python. The Python implementation supports constraint propagation between components and optimization over specified variables. Each component is interfaced as a Functional Mock-up Unit (FMU), allowing components to be connected and properties specified in language independent way. From the SVG containing the high-level system information, both Modelica and Python code is generated and executed on the web server, potentially hosted in a high performance data center. More implementations could be added without modifying the SVG system description.

We have shown that it is possible to separate system descriptions on the block diagram level from implementations and interface between the two levels using FMI. In a continuation of this project, we aim to integrate the FMI standard also for co-simulation, such that components implemented in different languages could be used together. One open question is to what extent FMUs of the same component, but implemented with different tools, will have the same model description. For the SVG-based system description to be useful, the FMI model description must remain the same, or at least contain a large overlap, for a single component implemented in different languages. This will be further investigated in future work.

References

[1] Modelica Association. Functional mock-up interface, http://www.fmi-standard.org, November 2012.

[2] Modelica Association. Modelica and the modelica association, http://www.modelica.org, November 2012.

[3] Erik Billing and Martin Servin. Composer, http://imuit.cs.umu.se/composer, November 2012.

[4] SCA Packaging. Sca packaging obbola, http://www.scapackaging.com, November 2012.

[5] Patrik Törmänen and Hussein Jaffal. Reducing electricity cost - case study. Technical report, UMIT Research Lab, Umeå University, 2011.