Extending the IPG CarMaker by FMI Compliant Units

Stephan Ziegler and Robert Höpler
Modelon GmbH München
Agnes-Pockels-Bogen 1, 80992 München, Germany
{stephan.ziegler,robert.hoepler}@modelon.com

Abstract

This paper presents a generic interface which enables exploiting the multi-physics modeling capabilities of Modelica within the virtual test drive simulator CarMaker [1] using the Functional Mock-up Interface (FMI) [2].

Applications ranging from detailed studies of vehicle properties to hardware in the loop test-rigs require models of different complexity. CarMaker is provided an interface which allows for vehicle models being extended by almost arbitrary external component models implementing the FMI. The FMI offers two flavors of units which permit the balancing of computational performance and numerical robustness within CarMaker. Additional model information provided by the FMI is used for automatic integration of the external models as well as for configuring the computations. Involved aspects are shown by an example truck model.

Keywords: FMI, FMU, CarMaker, HIL, Solver, Model export, Interface, Stability, Co-Simulation

1 Introduction

The CarMaker platform [3] is a full-fledged virtual driving environment which offers a wide range of applications from offline operation to hardware in the loop (HIL) tests. CarMaker was designed to support the development process from an early conceptual stage to hardware prototype testing.

Therefore the CarMaker suite is composed of two main components, the CarMaker Interface Toolbox (CIT) and the Virtual Vehicle Environment (VVE), see Fig. 1. The CIT contains a collection of tools for simulation control, parameterization, analysis, visualisation, and file management.

The VVE represents the computer modeled composition of the vehicle with all its components such as powertrain, tires, brakes and chassis as well as road and driver. Vehicle components may be implemented by default generic models, custom code such as MATLAB/Simulink controller models or even real hardware on a test rig. Depending on the desired task

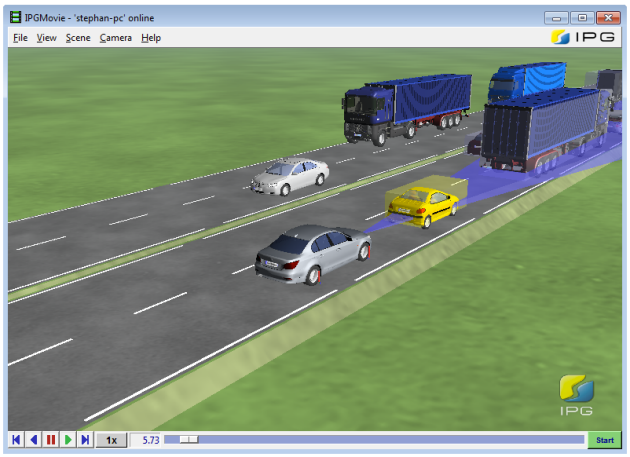

Figure 1: Typical VVE scenery including road and traffic rendered by the CarMaker IPG-Movie animation tool.

the VVE can be operated on a regular office computer or on a real-time system. Real-time operation allows investigation of deterministic behavior, office operation might lack real-time capabilities but is therefore applicable on almost any host computer and allows the simulation to run slower or faster than real-time depending on system performance and model complexity and does not require special hardware.

Simulation techniques are a key enabler in the development of nowadays propulsion systems of electric and hybrid vehicles. These systems challenge the power of vehicle dynamics tools in two ways: On the one hand the number and complexity of topologies of drive-trains is almost combinatorial due to the large number of involved drives, clutches, or gears. On the other hand particularly hybrid vehicles represent an almost classical multi-domain system (electronics,

hydraulics, combustion, chemistry, mechanics) with a dramatically rich dynamics in the interaction between the various technical components.

Modelica forms a perfect basis for the modeling of multi-domain systems especially automotive systems [4]. The benefits of its general equation-based formalism has been shown by numerous publications in the fields of vehicle dynamics, climate control, drivetrain modeling, combustion engines, and hybrid powertrains, see e. g. [5–8].

The increasing complexity in automotive system development especially for hybrid vehicles demands for versatile modeling tools, realistic and reproducible virtual testing as well as seamless test rig integration. An ideal software for modeling and simulation of all vehicle-related scenarios off-line and in real-time would combine these aspects. Augmenting the comprehensive CarMaker VVE suite by the multi-physics capabilities of Modelica is a first step towards this direction.

2 Augmenting CarMaker by Functional Mock-Up Units

Extending CarMaker

The generality of the CarMaker VVE allows for the modifications of the vehicle models in an almost arbitrary number of ways. Generic vehicle components such as powertrain, steering, tires, brake system as well as the complete propulsion system might be configured, as shown in Fig. 2. For several reasons custom models serve as a replacement of the predefined modules in the generic vehicle model or might even replace the vehicle model as a whole: (i) All topologies of hybrid vehicles might not be covered even by an advanced tool specialized in vehicle dynamics. (ii) These multi-physics systems demand for general modeling formalisms and simulation techniques such as Modelica-based approaches. (iii) Many engineering companies and car manufacturers are particularly relying on Modelica during development of components and control units, e.g. [6].

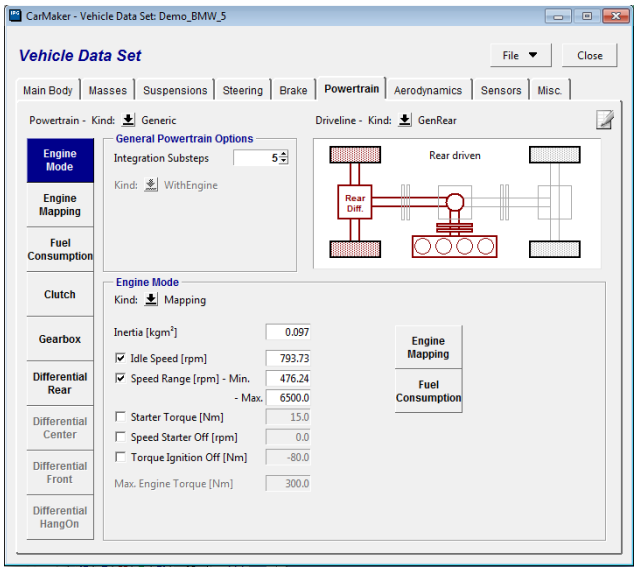

Figure 2: CarMaker interface for vehicle model configuration.

Replacing single vehicle components by so-called custom code is easily done by dynamically linking executable code and registering variables, parameters, and their dimensions and types within the VVE. The requirements when incorporating external code are manifold:

1. Real-time compliance: The code may not rely on specific processing hardware since CarMaker vehicle models should run on arbitrary real-time platforms.

2. Small run-time overhead and memory footprint.

3. Since CarMaker runs its own numerical integration schemes the code of external dynamical models must expose input and output variables. These signals can be physically meaningful, e.g. tire forces, wheel speeds, etc..

4. Implementation details might not be exposed for intellecual property reasons.

5. Standardized, automatic and intuitive inclusion of the external code.

In general, almost all of these requirements can be met by the usage of black-box models, except for the last item. On the other hand these shift problems to different classes: Lack of numerical robustness and/or computational performance.

Functional Mock-Up Interface

The FMI adresses almost all requirements discussed in the previous section. It defines an open standard interface to be implemented by an executable model called Functional Mock-up Unit (FMU). The FMI functions are used (called) by a simulator to create one or more instances of the FMU, and to run these models, typically together with other models. An FMU may either be self-integrating (FMU for co-simulation, FMU-CS)

or require the simulator to perform numerical integration (FMU for model exchange, FMU-ME) [2]. The artifacts which csan be packaged in an FMU-file are C-source code, executable code compiled for one or more platforms, and XML descriptions of variables, parameters, and general solver and model properties.

Interfacing Modelica Models to CarMaker

Choosing the FMI as a basis of the interface is beneficial for several reasons since the exposed C-interface is standardized and XML model description [9] included in any FMU contains excellent structural information about the model. The prerequisite is a modeling tool, e.g. Dymola or SimulationX, which generates this description while exporting the Modelica model as C-code. Alternatively this FMU can be hand-coded or generated from any other modeling tool that supports FMI, e.g. a multibody-package.

When interfacing Modelica models to CarMaker one must consider two key aspects: (i) The executable model or code must represent the modeled dynamical behavior without any restrictions in order to maintain generality and to avoid context dependent model semantics. (ii) The resulting model must fulfill requirements essential for real-time applications, most prominently a fixed integration stepsize.

The Interface consists mainly of two parts. The first part governs static type-checking of the input and output variables and parameters of FMU to assure a smooth setup of the solver, generation of user interfaces for parameter input, and automatic extension of the so-called data dictionary for tracing relevant variables during simulation. Finally the executable model is instantiated and connected to the CarMaker solver. Depending on the specific application each type of FMU can be chosen.

Even if the mean computational performance of two coupled simulators is sufficient it can be difficult to assure definite turn-around times, an aspect critical in e.g. HIL systems. CarMaker provides a simulation engine for robust fixed stepsize time integration so including an FMU-ME is a safe solution for simulating the combined vehicle model in the following cases: When either the model complexity is reasonably low, the eigenvalues of the system are located in the stable region of the integration scheme and no strong non-linearities are in the imported model. For FMU generated from very complex or black-box models the option FMU-CS is advantageous, because the FMU contains the appropriate numerical integration scheme. The interface in combination with CarMaker serves as the master algorithm in the co-simulation of CarMaker and the imported block. This leads to an adaptive oversampling of CarMaker's time grid which might rule out usage on embedded systems or HIL, but leads to superior numerical stability. Compared to the features offered by the full FMU-CS this interface takes advantage only of some features. The setup is the simple one depicted in Fig. 3 with only one single execution engine where the interface is the simulation master which runs sub-system 2 synchronized to the grid the solver of tool 1, i.e. CarMaker.

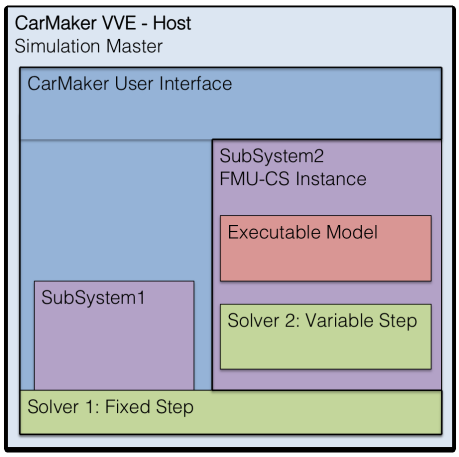

Figure 3: Schematic of run-time architecture.

3 Example Application

The integration of an FMU was investigated for a hybrid truck model based on the Vehicle Dynamics Library [4, 5]. A schematic of the model is depicted in Figure 4. Either the CarMaker standard drive-train or the complete vehicle were replaced within the VVE by an FMU exported from Dymola. In case of the drive-train the driver and environment input signals from the CarMaker model were accelerator pedal, clutch, brake torques, gear number, starter and ignition amongst others. The most prominent model outputs were the wheel speeds.

When using FMU-ME as a replacement the complete vehicle model showed unstable numerical behavior since the fixed solver stepsize did not comply with the dynamics of the new drive-train model. The used stepsize of one millisecond is traditionally used in vehicle dynamics HIL applications. Not even the oversampling feature of CarMaker lead to stable behavior as the constant inputs during the oversampling steps deteriorated the dynamical behaviour. Applying an FMI-CS using an appropriate variable stepsize solver

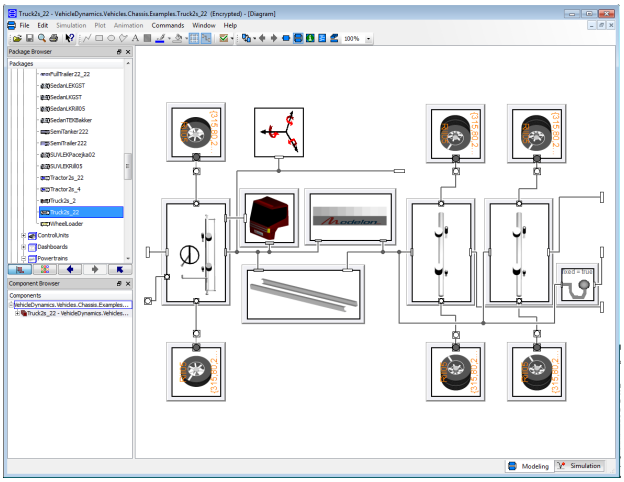

Figure 4: Truck example model in Modelica.

for the drive-train lead to stable numerical simulation within CarMaker. This method allowed the simulation to run in real-time at least in office-mode (soft real-time). Standard techniques to reduce model complexity might help to gain hard real-time capability for HIL applications.

4 Conclusion

This paper presented an interface to augment the CarMaker by functional mock-up interfaces generated from Modelica models. The interface supports both representations of FMI, the one for co-simulation as well as the one for model exchange, exploiting the specific advantages of each approach. The first leads to numerically robust total simulation models where the CarMaker simulation engine is the master algorithm controlling the FMU model containing its own numerical integration scheme and time grid just synchronized on the grid imposed by CarMaker. The latter leads to computaionally efficient executable models ideally suited for real-time applications. In the case where the modelled dynamics does not fit to the fixed stepsize prescribed by CarMaker integration scheme this might lead to poor numerical robustness as has been shown by a simple example application.

The interface is not restricted to executable models stemming from Modelica-based tools. FMUs can be generated from dynamic systems by any modeling and simulation tool which supports the FMI standard or can even be hand-coded. The presented work supports a fraction of the features of the current FMI specification V. 1.0. Possible directions are manifold, e. g. network communication between multiple FMUs in a true co-simulation environment e. g. enabled by Silver [10], parallelization within CarMaker, or the treatment of CarMaker itself as an FMU.

Acknowledgements

The authors would like to thank Christian Schyr, IPG GmbH, and Johan Andreasson, Modelon AB, for valuable comments.

References

[1] Christian Schyr, B. Bernius, U. Haag, and M. Kircher. Engine-in-the-Loop – Efficient Use of Simulation on the Engine Test Rig. In *IPG Technology Conference apply & innovate 2010*, Karlsruhe, Germany, 2010.

[2] MODELISAR consortium. Functional Mock-up Interface for Model Exchange, V. 1.0. http://www.modelisar.org, 2010.

[3] *CarMaker User's Guide Version 3.0*. IPG Automotive GmbH, Karlsruhe, Germany, 2009.

[4] Johan Andreasson. The Vehicle Dynamics Library: New Concepts and New Fields of Application. In *Proceedings of the 8th International Modelica Conference, Dresden, Germany, 20–22 March 2011*.

[5] Johan Andreasson and Mats Jonasson. Vehicle Model for Limit Handling: Implementation and Validation. In *Proceedings of the 6th International Modelica Conference, Bielefeld, Germany, 3–4 March 2008*, pages 327–332.

[6] Henrik Wigermo, Johannes von Grundherr, and Thomas Christ. Implementation of a Modelica Online Optimization for an Operating Strategy of a Hybrid Powertrain. In *Proceedings of the 6th International Modelica Conference, Bielefeld, Germany, 3–4 March 2008*, pages 487–492.

[7] Sanaz Karim and Hubertus Tummescheit. Controller Development for an Automotive Ac-system using R744 as Refrigerant. In *Proceedings of the 6th International Modelica Conference, Bielefeld, Germany, 3–4 March 2008*, pages 477–485.

[8] Christian Schyr and Kurt Gschweitl. Model-based Development and Calibration of Hybrid Powertrains. SAE Technical Paper, 2007.

[9] Francesco Casella, Filippo Donida, and Johan Åkesson. An XML Representation of DAE Systems obtained from Modelica Models. In *Proceedings of the 7th International Modelica Conference, Como, Italy, 20–22 September 2009*, pages 243–250.

[10] Silver. Qtronic GmbH, Berlin, Germany. www.qtronic.de.