Experiences with the new FMI Standard Selected Applications at Dresden University

Christian Schubert[a] Thomas Neidhold[b] Günter Kunze[a]
[a]Dresden University of Technology, Chair of Construction Machines and Conveying Technology
Münchner Platz 3, 01187 Dresden, Germany
[b]ITI GmbH, Webergasse 1, 01067 Dresden, Germany

January 31, 2011

Abstract

This paper describes how the new FMI standard is used at Dresden University to provide a universal model exchange. By employing the new standard exchange format a single simulation model can be used for different purposes in different software tools. Specific applications and their advantages will be presented as well as the development effort for their realisation. Special attention is paid to how FMI enables SimulationX models to be run in an interactive simulation on a motion platform. Finally, the benefits and drawbacks of the FMI interface are discussed from a user's perspective.

Keywords: FMI, Modelisar, Virtual Reality, SimulationX, Motion Platform, SARTURIS

1 Introduction

Simulation has become an indispensable tool in the field of machine construction and design. It is especially beneficial for Heavy Mobile Machinery since their low quantities and high production costs render real prototypes uneconomical. During the design process, there are many tasks which can be supported by simulation such as comparing the effect of different parameter combinations and model variations.

In order to achieve maximum efficiency, verified models should be reused as often as possible. However, this can only be possible if there exists a common standardised interface for model exchange between different software tools for different applications.

Modelica has already proven to be a valuable contribution towards the simulation of Mobile Machinery [1, 2]. On the one hand it features an inherent support of multidomain models, on the other hand it features a seamless transition between a mathematical and a graphical model definition allowing for complex but structured models at the same time. Although Modelica is tool independent and can thus be used to exchange models between certain simulation environments, it cannot be used as a general model interface. The reason is, that every tool used for import would have to parse, flatten and optimise the Modelica code for which no general purpose library is available. The new Functional Mockup Interface (FMI), a result of the MODELISAR project [3], overcomes the aforementioned problem.

The next chapter features a brief description of the FMI standard. It is followed by an

overview of the work and the employed software tools at the Chair of Construction Machines and Conveying Technology at Dresden University before FMI has been introduced. Chapter four presents the benefits which have been achieved by implenting and applying FMI to this software environment. Special attention is paid to what has been changed and what had to be implemented. One major achievement of integrating the FMI standard is the option to run SimulationX models on the motion platform at Dresden University and is described in chapter five. It allows the evaluation of different machine configurations and their parameters within an interactive simulation. The paper concludes with a discussion of the advantages, drawbacks and possible solutions when using FMI from a user's perspective.

2 The FMI Standard

FMI, short for Functional Mockup Interface, is a new standard for model exchange [4] as well as co-simulation defined within the MODELISAR project. It describes a set of functions and their parameters which shall be part of a binary .dll (Dynamic Link Libary) file or .so (Shared Object) file under Windows or Linux respectively. The binary file is complemented by an XML file which includes a description of the model and all its variables. The separation of model and description data leads to very efficient code, which may even be run on embedded systems, while maintaining usability for more powerful systems by offering the complementary XML documentation. Using a simple binary format has the advantage that it can be included at minimal effort in almost any application that allows incorporating user code. Thus, the FMI standard fills the gap between abstract Modelica models and vendor dependent co-simulation.

3 Initial Situation at Dresden University

One research topic of the Chair of Construction Machines and Conveying Technology at Dresden University is concerned with studying and optimising Heavy Mobile Machinery and its components by using interactive simulation methods. Therefore a modular software framework called SARTURIS has been developed [2]. It allows real time simulation of technical systems in a virtual reality (VR) environment and features easy and flexible hardware integration in addition to a powerful visualisation. SARTURIS is based on C++, uses freely available libraries and is platform independent. The software framework provides a diversity of applications for models of technical systems. It is suited for both the analysis of machine behaviour on a PC or laptop as well as an interactive simulation on a motion platform (Figure 1).

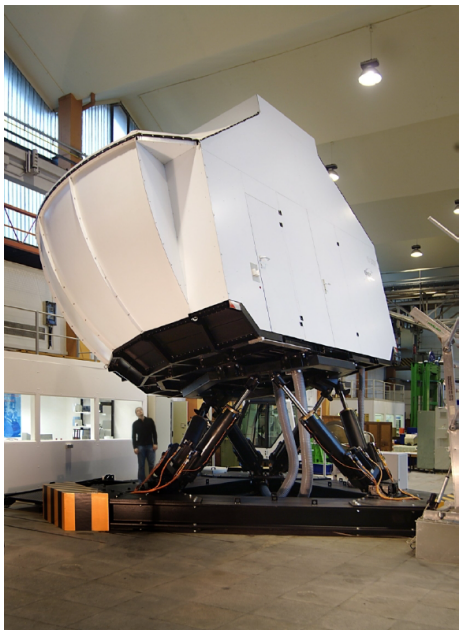

Figure 1: Motion Platform at Dresden University

In order to create simulation models, to run task-specific simulations or to evaluate simulation results different custom-designed software tools are used. Thereby the interaction of two or more tools can be useful and an exchange of data between them is necessary.

Figure 2 depicts the initial situation. Five basic proprietary exchange algorithms already existed or have been implemented for specific tools or tool combinations, providing the basis for different tasks:

1. Simulation tools based on Modelica for instance, are able to exchange models in general.

2. Many commercial tools like SimulationX, Simpack and Dymola support co-simulation with MATLAB/Simulink.

3. Other tools offer a wide range of output formats like PyMbs [2], a software for the realtime simulation of multibody systems.

4. Pynacolada, a software that supports recurring analysing tasks, features a broad range of input formats like MATLAB, SARTURIS and general HDF5 [5] data sets.

5. Specialised auxiliary tools or wrappers had to be developed to accomplish the exchange of required model data from OpenModelica and Simpack for an interactive simulation with SARTURIS.

Such tool couplings are important since experience has shown that general simulation environments are not always the best option for specialised engineering tasks. Next to classical simulations, models of technical systems can be used to solve other problems like inverse kinematics or finding optimal constructive geometries. Software designed for such tasks can only produce reliable results if the technical system regarding the problem is described accuratly enough. Those specialised tools however are rarely convinient modelling environments. If one can establish a comfortable and efficient data and model exchange, both software strategies could be combined with their advantages. Thus an expanded range of applications could be achieved.

Individiual solutions for coupling tools (see option 5) have the great disadvantage that they are very specific and need constant attention. Extending the coupling to another tool usually calls for a different specialised data exchange strategy. This leads to the circumstance that not all possible exchange paths have been realised although many other tool interactions would be benefitial. The new FMI standard is a promising approach to provide a common foundation for universal model exchange and thus increasing flexibility while reducing maintenance efforts.

4 Integration of FMI

The introduction of FMI as the common interface between all tools led to a great simplification, see Figure 3. Now every tool simply exports FMI, imports it or supports both. Thus one single model can be made available to other tools very easily. There is no need for specialized utilities or wrappers anymore to establish data or model exchange between two tools. As soon as a tool features an FMI import or export it can be incorporated in every tool chain without any extra effort. Current versions of Modelica tools, such as SimulationX, JModelica and Dymola already feature FMI exports as well as FMI imports. For the Open Source alternative OpenModelica, the FMI export is currently being developed in a cooperation between Linköping and Dresden University and will be available in future releases. Outside the Modelica community FMI gains in

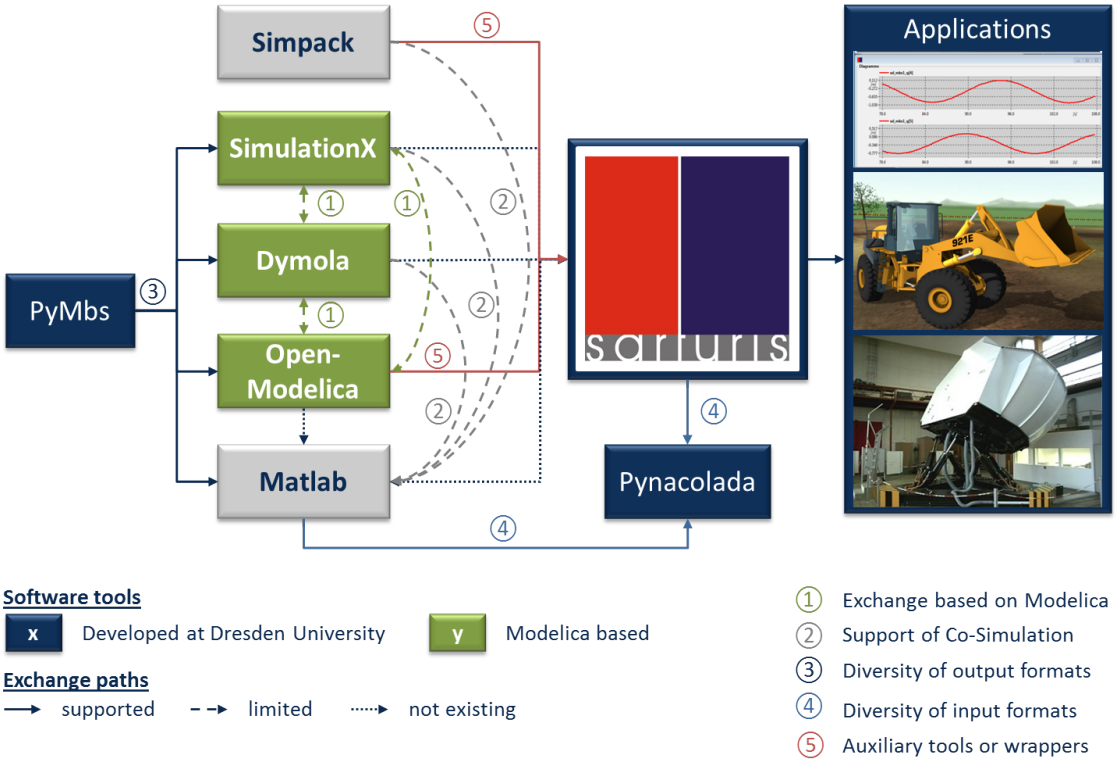

Figure 2: Tool Chains Before FMI

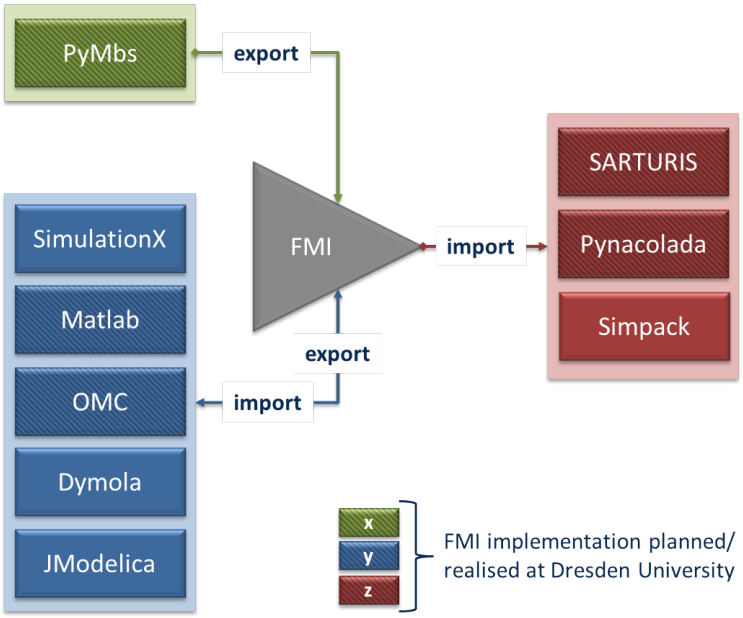

Figure 3: Tool Chains After FMI

importance as well. SIMPACK as a partner in the MODELISAR project for instance is developing an FMI import enabling SIMPACK users to include Hydraulics into their multibody models using specialised software.

Unfortunately, no FMI implementation for MATLAB is known to the authors yet. Hence the authors started developing an import wrapper for both MATLAB and Simulink as well as an export for MATLAB Simulink models based on the C-Code generated by the Real-Time-Workshop (see http://code.google.com/p/fmi2matlab/).

Equipping self developed tools with FMI capabilities, proved to be a reasonable amount of effort. Due to the comprehensive documentation [4] as well as the FMU Software Development Kit provided by QTronic [6] an easy to understand reference implementation is available. Thus PyMbs, Pynacolada and SARTURIS do now support FMI.

5 SimulationX Models in Interactive Simulation via FMI

A particular interesting application of the new FMI based tool chain is the investigation of virtual prototypes using interactive simulation via the motion platform at Dresden University, see Figure 1. Whereas motion platforms are traditionally used to evaluate the influence of the machine and its complexity on the human operator, the aim of the Chair of Construction Machines and Conveying Technologies is the opposite. The motion platform is intended as a tool for studying the influence of the human operator on the machine. Thus it is possible to obtain joint forces and pressure distributions dependent on the behaviour of the operator and generally gain a deeper understanding of the machine and the ways to optimise it while it is in operation.

In order for the results to be significant a detailed model comprising mechanis, hydraulics, drivetrain and control systems has to be simulated. A way of developing such models has been presented by the authors in [2]. Due to the newly implemented FMI functionalities of SARTURIS one can now use SimulationX as well to set up models for the motion platform. Thus the full range of the SimulationX model libraries can be used which can considerably shorten the modeling effort.

To test the FMI based tool chain a model of a wheel loader, see Figure 4, has been developed together with ITI GmbH. In addition to the multibody system it consists of hydraulic cylinders and valves, a reduced drivetrain and Paceijka tire models. Furthermore it considers contact between the bucket and the underground. The model has been developed and thoroughly tested within SimulationX and it is capable of running in realtime.

As the new SimulationX offers FMI Export, the model can now be easily transferred to SARTURIS which in turn controls the motion platform. Within SimulationX all inputs which should be provided by the operator are defined as well as all outputs needed for visualisation and the evaluation of the model. The wheelloader modelled with SimulationX is now running within an interactive simulation on the motion platform and can be driven through a virtual environment.

Along with the described benefits, there are problems that had to be overcome and shall be discussed in the following.

6 Visualisation

Although importing the model via FMI is easy to accomplish, the visualisation of the technical system has to be set up twice. Once in the modeling software for assembling the multibody system and a second time in the simu-

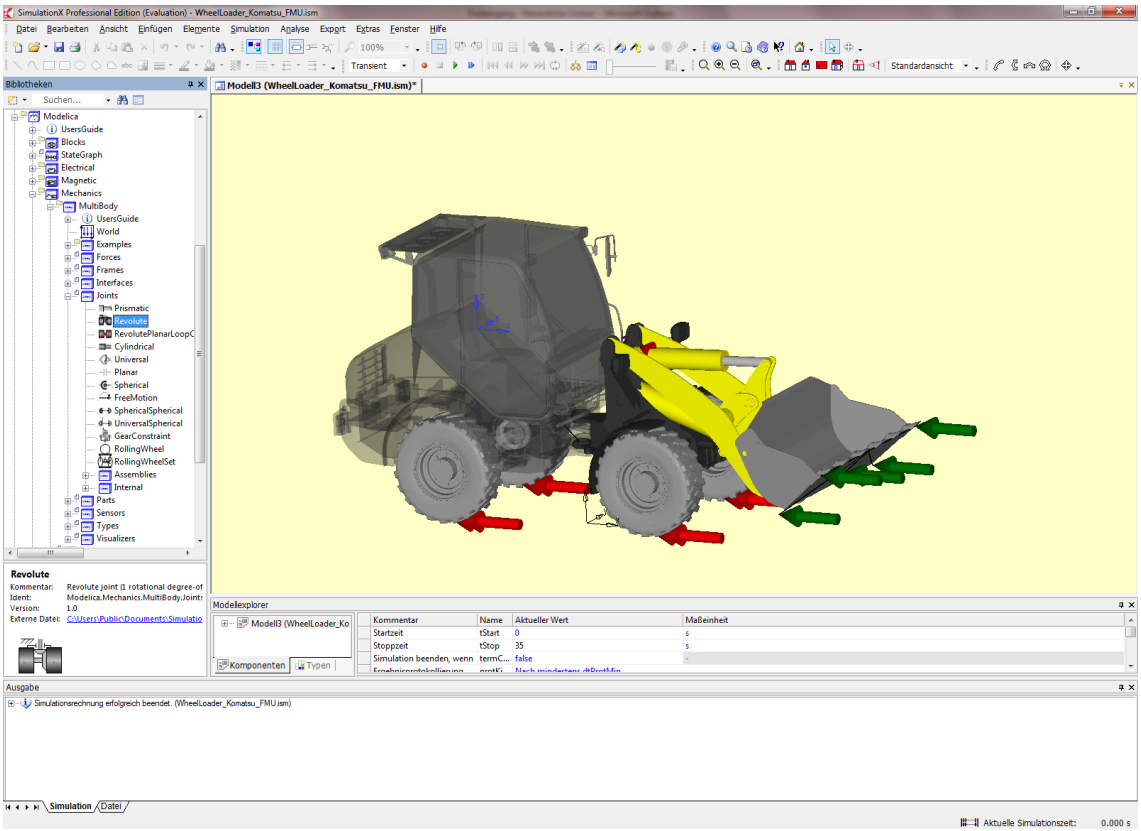

Figure 4: Model of a Wheel Loader in SimulationX

lation software for visualising the results. FMI does not incorporate any additional data concerning the graphics. According to the FMI philosophy, where the binary file is supposed to only contain the vital model information so that it can be run on embedded platforms, these additional information should not be stored in the code but either within the accompanying model description or within an extra file. Possible approaches might involve usage of the standards X3D [7] or COLLADA [8].

7 External Functions

7.1 Problem

It is often required to use the same vehicle model for different simulation tasks on different test tracks. Every vehicle model is therefore stored in a separate FMU. When performing contact detection, as needed for determining the tire contact point or general collisions, an extensive communication between the model and the track is needed.

Using FMI this can either be achieved by using inputs and outputs or linking user defined code statically. Statically linked user defined code is unsuitable since it would be difficult to communicate with the simulation environment. Even if it could be accomplished, the resulting FMU can only be used in combination with that particular environment. Using inputs and outputs are not suitable as well since their number may grow very large the more often a specific function is used. Also iterative methods would be hard to implement.

7.2 Possible Solution

What is needed is a coupling of the FMU and the simulation environment at runtime preferably via callbacks. Unfortunately, the current FMI interface does not support such a feature. That is why for our models we extended the FMI interface by the following function

```
fmiStatus fmiSetExternalFunction(
                fmiComponent c,
                valueReference *vr,
                void *functionPointer);
```

passing a void function pointer to an *fmiComponent*. Therefore the accompanying modeldescription XML has been extended by all external functions, that can be set at runtime with their *valueReference* as well as their input and output parameters. Thus almost no additional code has to be included into the model itself.

7.3 Compatibility

Such an FMU may still be compatible to general purpose simulation tools. When instantiating an FMI component, all external functions and their signature are known from the modeldescription XML file. Thus, a user may choose an external function from all functions that are known to the tool and have the same signature.

7.4 Modelica

In order to create such an FMU from a Modelica tool, these functions have to be marked. This can either be done by introducing a new annotation or by extending the Modelica language. In fact, the Modelica standard 3.2 already offers *Functional Input Arguments To Functions* ([9], section 12.4.2). If this feature was to be carried over from *functions* to general *models* a clear description had been found.

7.5 Example

In order to illustrate this idea, a simple example shall be given. Consider the bouncing ball example:

```
model BouncingBall
  constant Real g = 9.81;
  parameter Real c = 0.9;
```

```
  parameter Real radius = 0.1;
  Real height(start = 1);
  Real velocity(start = 0);
  input HeightFunction heightFun;
equation
  der(height) = velocity;
  der(velocity) = -g;
  when height <= (radius + heightFun(time)
      ) then
    reinit(velocity, -c*pre(velocity));
  end when;
end BouncingBall;
```

The type HeightFunction may be defined as

```
partial function HeightFunction
  input Real t;
  output Real height;
end HeightFunction;
```

returning the height of the underground at a given time t.

A complete test model might look like

```
function SinusHeight
  extends HeightFunction;
algorithm
  height := sin(t);
end SinusHeight;

model SinBounce
  BouncingBall ball;
equation
  ball.heightFun = SinusHeight;
end SinBounce;
```

If the model BouncingBall were translated into an FMU the function pointer heightFun could be treated like any other component mapping it to void* and including it into the modeldescription.xml. In addition the xml description should include the signature of the function (inputs and outputs) as well as its comment which should be used as a short description what this function does.

8 Model Exchange Across Company Borders

One major advantage of FMI is that the model exchange described in this paper does not need to be limited to one institution or company only. Using FMI might also be a valuable tool for cooperation between OEMs and their suppliers. Since it is virtually impossible to deduct information from models in binary format, suppliers can safely provide models to OEMs without revealing too much of their knowledge. However, in order to exchange a model in binary format only, all target platforms have to be known in advance. True platform independence is achieved by exchanging source code allowing more insight into the model.

9 Conclusion

It has been shown that powerful modeling tools are not always the best option for specialised simulation tasks. However, software designed for special tasks can only be as good as their underlying models of the technical system. Thus it is beneficial to exchange models between such tools.

Before FMI it has always been a very specialised solution which was hard to maintain. Using FMI at Dresden University has helped to unify the modeling and simulation software environment. It reduced the need for auxiliary tools establishing the model exchange and the time spent on developing and maintaining them. A major benefit is the option to run multidomain models created in SimulationX within an interactive simulation on the motion platform at Dresden University which can be used to study the influence of the operator on the machine and its components. Open problems like visualisation and calling external functions have been adressed as well as political advantages when using FMI.

References

[1] Beater, P.; Otter, M.: Multi-Domain Simulation: Mechanics and Hydraulics of an

Excavator, Proceedings of the 3rd International Modelica Conference, November 2003, pp. 331-340

[2] Frenkel, J.; Schubert, C.; Guenther, K.: Using Modelica for Interactive Simulations of Technical Systems in a Virtual Reality Environment, Proceedings of the 7th International Modelica Conference, Como, September 2009

[3] MODELISAR consortium: MODELISAR Project Profile, http://www.itea2.org/public/project_leaflets/MODELISAR_profile_oct-08.pdf

[4] MODELISAR consortium: Functional Mock-up Interface for Model Exchange, http://modelisar.org/specifications/FMI_for_ModelExchange_v1.0.pdf

[5] The HDF Group: HDF5, http://www.hdfgroup.org/HDF5/

[6] QTronic GmbH: FMU SDK 1.0.1, http://www.qtronic.de/en/fmusdk.html

[7] web3D Consortium: X3D Specification, http://www.web3d.org/x3d/specifications/

[8] Khronos Group: COLLADA - Digital Asset Exchange Schema for Interactive 3D, http://www.khronos.org/collada/

[9] Modelica Association: Modelica - A Unified Object-Oriented Language for Physical Systems Modeling - Language Specification Version 3.2, March 2010