Functional Digital Mock-up and the Functional Mock-up Interface – Two Complementary Approaches for a Comprehensive Investigation of Heterogeneous Systems

Olaf Enge-Rosenblatt, Christoph Clauß, André Schneider, Peter Schneider
Fraunhofer Institute for Integrated Circuits, Division Design Automation
Zeunerstraße 38, 01069 Dresden, Germany
Olaf.Enge-Rosenblatt@eas.iis.fraunhofer.de

Abstract

Functional Digital Mock-up (*FDMU*) and *Functional Mock-up Interface* (*FMI*) are two keywords arising in the last years in simulation technology. In this paper, we would like to show that both principles, aiming at a comprehensive investigation of heterogeneous systems, e.g. from mechatronics, are not necessarily competing with each other but may be combined to benefit from the ideas behind. The approaches are based on different ideas and cover different aspects of the interaction of modern simulation tools. For that reason different constraints have to be considered, which do not make things easier. Both principles have advantages and disadvantages. However, by combining both ways, a powerful framework for handling a broad variety of simulation tasks can be formed. In the paper, a possible approach for integrating both technologies will be shown.

Keywords: *FDMU; functional digital mock-up; FMI; functional mock-up interface; co-simulation; heterogeneous system; simulation algorithm*

1 Introduction

In today's industry, the product development process is more and more characterized by intensive usage of simulation. But in all branches, a large variety of languages, simulators, and environments are used which are incompatible to each other in the most cases. Hence, the task of coupling different simulation tools, e.g. by co-simulation concepts, has been moving increasingly into the focus in the last years.

In this context, many activities within the simulation community have been able to be recorded concerning the integration of the principle of *Functional Mock-up* into the world of simulation and simulators. The main target of these attempts is to combine the ideas of the digital mock-up processes with functional aspects. Even though both approaches are open for different modeling languages and simulators, the general purpose modeling language Modelica plays an important role on both sides.

Two "main streams" can be distinguished: the FDMU approach [2][7][9] and the FMI approach [3][5][6]. The FDMU ideas are driven by four German Fraunhofer institutes funded by the Fraunhofer-Gesellschaft. In contrast, the FMI ideas are favored by a network of companies and research institutions spread widely across Europe and working together within the MODELISAR project funded within the ITEA2 framework. Both approaches have advantages and disadvantages and both deal with general ideas of co-simulation. While the FDMU approach focuses on the combination of different simulators with a shared and interactive visualization, the FMI consortium thinks about model exchange and co-simulation ideas. More information about both approaches and a comparison of architectures and data flows is given in the following sections.

The Fraunhofer Institute for Integrated Circuits, Division Design Automation in Dresden has been taken part in both projects. That's why the idea to investigate some opportunities of a combination of both principles is not very far. Taking the differences of both principles into account possible ways to combine both approaches are shown in this paper.

2 FDMU

The main goals of the Fraunhofer-internal project Functional DMU were, first, to develop a principle of a general, tool-independent, and web service-based framework for coupling different simulation tools with each other as well as, second, to provide an integrative (and simultaneously interactive) visualization tool to unify the user's view. The output of the project is a fully implemented ready-to-use framework, the so-called FDMU framework, which can be used in connection with a large variety of simulation tools.

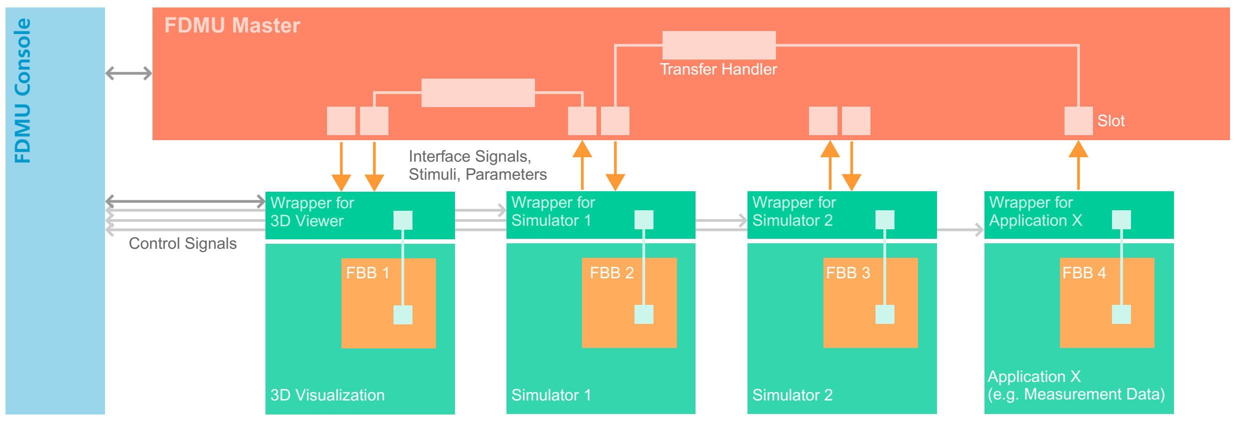

Figure 1: FDMU principle with Master Simulator

2.1 Basic idea

Functional Digital Mock-up (FDMU) is a substantial extension of the widely introduced DMU approach which is used for investigating geometrical and mechanical properties of mechanical systems. The FDMU approach combines traditional digital mock-up (i.e. geometrical information) and aspects of behavioral system description (i.e. functional information). An additional, new and very important point is the possible interaction between visualization and numerical simulation in both directions which is governed by a master simulator. This *FDMU Master Simulator* is also capable to accomplish the communication between different simulation tools. Hence, if dealing with a complex simulation task, more than one simulation tool can be incorporated to solve it.

Finally, the FDMU approach is predestined for multi-physical (or multi-domain) systems like often considered with appropriate modeling languages like Modelica.

2.2 Conceptual properties

In this paper, only the main concepts shall be presented. For more information please see the FDMU references, especially [2] and [7].

FBB and FSM

Within the FDMU approach, the concept of a so-called *Functional Building Block* (FBB) is proposed. An FBB is an envelope summarizing geometric information (CAD models), behavioral models (e.g. described by differential-algebraic equations), and communication interfaces into one basic data module. Geometric information and behavioral information have to be created within their particular modeling tools. These models remain in their associated data files. Pointers to these files as well as all interface information and the mapping between geometrical data and the interface quantities of the behavioral model are collected within the FBB using the modeling language SysML for a unique description. Within every FBB, a simulator tool is defined, too, which is capable to simulate the FBB's behavioral part.

A complete *FDMU Simulation Model* (FSM) consists of one or more FBB. Every input of an FBB must have an appropriate output belonging to another FBB. Furthermore, outputs can be propagated to the visualization to show simulation results using e.g. a geometric 3D model or some kind of plot versus time.

Master Simulator

When simulating an FSM, different simulation tools have generally to interact with each other. This is realized by the concept of a flexible co-simulation. The governing instance, the FDMU Master Simulator, is used to ensure the correct communication between all involved simulators as well as the correct delivering of simulation results to the visualization tool. The Master Simulator is the centralized controlling software and, therefore, the main module of the FDMU approach. It does not contain a model itself. It controls the signal flows between all participating software modules.

Web services

One of the basic concepts of the FDMU approach is the usage of Web services for communication between FDMU Master Simulator, different simulator tools, and the visualization. This provides the opportunity to distribute the tools needed for a particular simulation task among different computers which have only to be connected via Internet. This advantage must be paid with a little loss of coupling performance. But it is planned to enable a replacement of some of the Web services by direct TCP/IP socket connections to improve efficiency.

Wrappers

FDMU Wrappers are used to establish a connection between the different simulation tools and the FDMU Master Simulator. Every wrapper is a software module especially designed for a certain simulation tool to realize the interaction with the tool's environment. To this end, the simulator must provide the capability to include external functions into the simulation code. The wrapper is docked to the simulator like an external function and encapsulates the original simulator. This way a unique interface of every simulator is guaranteed.

Data Transfer

During a simulation run, all simulation tools needed to calculate the behavior of a complete FSM have to run in batch mode. The tools receive simulation control commands via pipe from the wrapper and perform the simulation. The time schemes for communication between the simulation tools and the Master Simulator may completely differ from each other. Hence, the FDMU Master Simulator supports a complex concept of data transfer. There are re-sampling procedures (transfer handlers, *Figure 1*), buffering schemes (slots, *Figure 1*), and an implicit synchronization concept performed by the data flow. The underlying concept is that all data transfers are initiated by the simulator tools.

Visualization

The FDMU framework provides an interactive visualization tool (*Figure 1*) for presenting subsequently a moving 3D scene according to the currently calculated simulation results. It is also possible to interact via the 3D scene with the complete simulation process. This is a manifestation of the close combination of geometrical and functional descriptions. This feature enables the user to control the simulation process like starting, pausing and finishing the simulation as well as to change FBB parameters.

2.3 Realization

The Functional Digital Mock-up approach is implemented within the so-called FDMU framework which is available with different combinations of components. The complete framework consists of a Master Simulator, some simulation tools encapsulated by wrappers, a visualization front end, and some data services. Wrappers for Dymola, Saber, Rhapsody, Simpack, and the multi-purpose tools Matlab and Simulink are currently available. Other wrappers could be realized and provided on request.

3 FMI

The goal of the ITEA2 project MODELISAR was the creation of a model-/ software-/ hardware-in-the-loop standard interface which is both tool and vendor neutral. This interface is called Functional Mock-up Interface (FMI). Via this interface, different tools will be able to communicate and act together to finally reduce both time and costs of development.

3.1 Basic idea

Starting with a technical system the components of which are treated by different simulation environments (simulators), the simulation of the whole system requires interaction of simulators. Taking into account that simulation is the solution of equations (DAE, PDE), the interaction can basically be managed in two ways:

Model exchange: The simulation environment of one component establishes the equations e.g. basing on a description language. The equations are passed over to another simulator which collects all component models and simulates all equations together. The results are then distributed.

Co-Simulation: Each component's simulator solves its own equations but values are exchanged which belong to more than one component. The simulators have to be synchronized.

Of course, hybrid forms of these two ways are possible.

The FMI standard is designed to support a wide variety of coupling possibilities. For each system it is open to decide which simulator should be used, and which way, model exchange or co-simulation, is to prefer. Once the FMI is established, the tool vendors are in charge to support it. Fitted with FMI, the tools should be able to interact.

3.2 Concept

For model exchange, the *Functional Mock-up Interface for Model Exchange* is defined. The intention is that the model exporting environment generates C code of a dynamic system model that can be

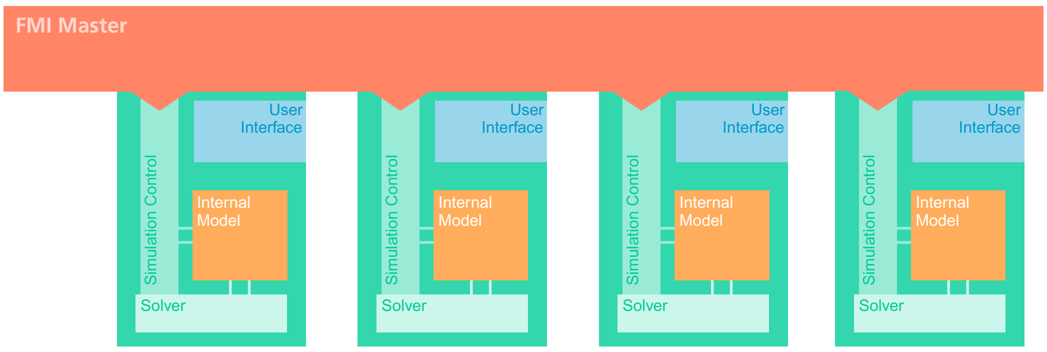

Figure 2: Principle of FMI for Co-Simulation

utilized by other (importing) simulation environments. Since models can describe differential, algebraic and discrete equations with time, state and step events, the interface is designed to be able to exchange all necessary values of variables (time, real, discrete values).

The Functional Mock-up Interface for Model Exchange consists of the *Model Interface* and the *Model Description Schema*. The Model Interface describes data types as well as functions compatible to a common basic mathematical description. All data and functions use the language C. The Model Description Schema contains all data which complete the model, but are not essential during model evaluation, e.g. numbers of variables, icons, documentation. A model specific zip-file contains all this information.

The C functions can either be provided in source form or in binary form, both of which can be added to the zip-file. The models are available by linking to the "importing" simulation environment. The interface is open to establish mechanisms for data transfer e.g. via the web.

The above mentioned ways (model exchange and co-simulation) are straightforward since the co-simulation approach needs the same data exchange like the model exchange approach, but needs additionally data which control the involved simulators. Therefore, the Functional Mock-up Interface for Model Exchange is a base for the Functional Mock-up Interface for Co-Simulation. Additional aspects come from solver coupling issues which are discussed in more detail in the following.

3.3 FMI for Co-Simulation

The *Functional Mock-up Interface for Co-Simulation* is a perspectively standardized interface for coupling two or more simulation tools in a co-simulation environment. Co-simulation is a technique for coupled time-continuous and time-discrete systems that exploits the modular structure of the coupled parts in all stages of simulation.

In co-simulation, different simulation tools have to interact while each of them has different properties concerning coupling algorithms. Important tool properties are:

- the ability to handle variable communication step sizes,
- the capability to discard and repeat communication steps,
- the capability to interpolate continuous inputs,
- the capability to provide information on a communication step (e.g. successfulness or error messages).

Otherwise, depending on the system to be simulated, there are different requirements to the simulation tool.

Therefore, the simulators of the components (slaves) are not directly coupled to each other but to a so-called master. The master's tasks are:

- analyzing the connection graph,
- analyzing the properties of the involved slave simulators,
- choosing a dedicated master algorithm,
- forcing the slaves to initialize,
- forcing the slaves to simulate communication intervals,
- realizing the data transfer according to the connection graph as well as according to the chosen algorithm,
- termination of the slave processes.

The master algorithms will not be standardized. Master algorithms can be developed both as a separate tool as well as an included feature of an existing simulation tool which plays the role of the master.

The FMI for Co-Simulation is designed to support a large variety of master algorithms. Similar defined as for model exchange, the FMI for Co-Simulation consist of the *Co-Simulation Interface* and the

Co-Simulation Description Schema. The *Co-Simulation Interface* is a set of C functions for the exchange of input/output values as well as status information. One of the most important functions is the function

```
fmiDoStep(
  fmiComponent component,
  fmiReal currentCommunicationPoint,
  fmiReal communicationStepSize,
  fmiBoolean newStep);
```

which starts a slave identified by `component` to simulate one communication interval of the length `communicationStepSize` starting from the `currentCommunicationPoint`. Further functions are defined e.g. for retrieving the status information from the slave. The *Co-Simulation Description Schema* contains e.g. capability flags, which characterize the above mentioned properties of the involved slave simulator.

Besides pure technical aspects, issues of protection against unintentional know-how transfer and authorization of models are generally solved.

4 Comparing FDMU and FMI

For comparing the two approaches of FDMU and FMI, the following aspects have to be considered:

- Type of coupling
- Coupling technology
- Programming language bindings
- Co-simulation algorithms
- Implementation

Further aspects may be interesting for certain user communities and will be investigated in the near future. The main focus of this paper is to figure out advantages of both technologies and to make a first proposal for a good combination of them.

There is no doubt that each of the co-simulation frameworks will find their friends in many fields of applications such as Automotive and MEMS[*] design. But there will be situations, too, where a combination of the power of FMI and the flexibility of FDMU will be the most appropriate solution.

4.1 Type of coupling

FMI for Co-Simulation defines a tight coupling between the FMI master and a co-simulation component denoted as FMU acting as slave (*Figure 2*). The interface specification contains a set of C function prototypes. This means all couplings are based on local function calls. The FMI master is the caller, all FMU slaves act as callee. Unfortunately, this clean architecture is slightly broken by a few callback functions e.g. for memory management and notifications.

FDMU is based on a loose coupling between the FDMU Console, the FDMU Master and all FDMU Wrappers (*Figure 1*). The FDMU framework is implemented as a service-oriented architecture (SOA). Each component excepting the FDMU Console (the user's front-end) works as service within a client-server infrastructure. The FDMU API is based on the Web Service standards (WS-I, WS-Attachments, WS-Security, WS-Notification, ...) [11].

4.2 Coupling technology

The main focus of the FMI specification is on efficiency. Using function calls within one process and one memory domain means minimal communication overhead. Currently no FMI benchmarks can be found in the literature. But the authors of this paper guess that the highest performance can be achieved by using the FMI approach for co-simulation in contrast to SOA-based approaches.

The FDMU framework uses HTTP-based SOAP messages for signal data exchange between the co-simulation components as required by the Web Services standards. This approach allows very flexible couplings between different hosts, hardware platforms, operating systems, and IP domains. The main focus of the FDMU approach is on flexibility. FDMU users can couple simulators between different departments or companies via Internet. The Web Services provide secure, encrypted, and standardized communication through firewalls. The drawback of this flexibility is the communication overhead.

Furthermore, the software architecture of the FDMU framework is based on the paradigm of distributed systems. Concurrency and multi-threaded implementations are supported and all communication methods include thread-safe queues for distributed and deadlock-free transmission of data. This architecture ensures scalability for large-scale co-simulation scenarios with more than four or five FBB.

Currently the FMI specification for co-simulation does not contain any details on multi-threaded implementations. Otherwise the specification explicitely supports asynchronous execution of API functions (e.g. `fmiDoStep()`) which means parallel simulation is allowed for FMI slaves and FMI is prepared for concurrency. The reference implementation which will be prepared within the MODELISAR project by the Fraunhofer IIS/EAS is based on a single-threaded approach. The FMI Master controls all

[*] microelectromechanical systems

FMI Slaves using a pure sequential algorithm. More details concerning the asynchronous features and multi-thread aspects will be investigated in further project research.

4.3 Programming language bindings

The FMI specification defines a set of C API functions. For the coupling between the software components (FMI master, FMI slaves), dynamic link libraries (DLL on Windows) or shared objects/libraries (SO on Linux/UNIX) are used. If all the needed DLLs (or SOs) are available, this approach is very simple and user-friendly. But the tight coupling between the different components works properly if binary and/or platform compatibility (32/64 bit, x86, x64, sparc, powerpc) are fulfilled and function calls are used for communication. If two FMUs run in different processes, operating systems or machines, a communication layer is needed. Currently this layer is not prescribed by the FMI specification.

For the FDMU framework, no language binding is required. The coupling interface is completely defined by the WSDL (Web Service Definition Language) specification of the FDMU Master. Each provider of an FDMU Wrapper for a certain simulation tool can use this WSDL specification to generate the code fragments for their own implementation for the needed programming or scripting language like Java, C, C++, C#, Python or Perl. All Web Service frameworks like JAX-WS, .Net WCF, AXIS, CXF, or similar can be used if they are compatible to the W3C standards. Unfortunately, there are some pitfalls and interoperability problems when using the different Web Service frameworks. So the current implementation of the FDMU framework is mostly based on Java and JAX-WS with a small adapter layer for tool-specific components for C, C++, and C#.

4.4 Co-simulation algorithms

The FMI specification for co-simulation defines a sequential computation flow for the master algorithm. The master analyzes the connection graph between all co-simulation components (slaves) and calculates an appropriate invocation order for the signal update and simulating the next time step of all components/simulators. Furthermore, the FMI Master can control in a very fine-grained way the simulation progress of each coupled simulator if the simulator provides the corresponding capabilities. Using these facilities it is possible to provide co-simulation algorithms with step size control or iterations at certain time instants.

The primary focus of the FDMU Master is to provide the concurrent signal flow between all FBB. The simplest FDMU Master algorithm handles all connections independent from each other. This FDMU communication schema prevents dead-locks and allows maximizing the throughput at least for signal connections which are independent from each other. Furthermore, the FDMU approach allows other algorithms e.g. with adapting the simulation step size during a running simulation. But the functionality is more limited here with respect to step size control of FMI.

4.5 Implementation

For FMI the provider of the simulation software has to implement the FMI interface as described in the specification. In this case the main advantage for the user is that he can use FMI-enabled tools without any further investments. For simulation tools without

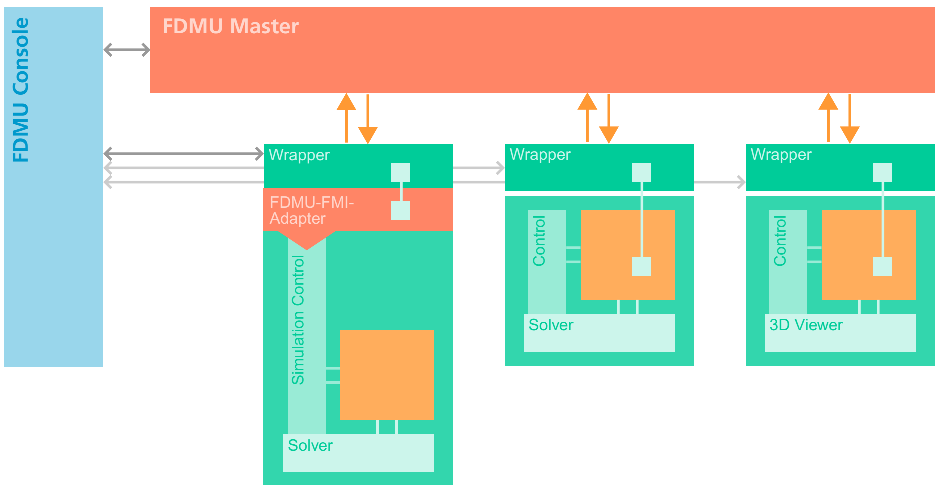

Figure 3: Idea of integrating FMI under FDMU

any FMI support, an FMI-based co-simulation is not possible.

FDMU has another strategy here. A FDMU Wrapper encapsulates a simulation tool using a small software layer around the tool. The advantage is that the simulator remains untouched and open interfaces (external function interface, tool-specific API, file IO …) are used only. The disadvantage is there is needed extra wrapper software. But for simulators without any FMI support this approach could be the only solution for using it within a co-simulation framework.

4.6 Summary

Both approaches, FMI and FDMU, have their benefits and problems. At a first glance FMI and FDMU represent two different perspectives to co-simulation-based system design:

- FMI is focused on efficient co-simulation interfaces between electronical, mechanical, and software models. The primary goal is to simulate and analyze the models.
- FDMU addresses the combination of both the behavioral models and the 3D geometry (CAD) data into one simulation approach. The primary goal is to provide an interactive 3D visualization with functional simulation in background.

Despite this there are similarities too:

- All mathematical aspects of co-simulation, like numerical error introduction by tearing, interpolation at the interface, the need of suitable, intelligent Master algorithms are similar in both appraoches.
- The slave simulators have to be prepared for participating in the coupled simulation.

5 Proposals for combining FDMU and FMI approaches

In this chapter we propose three options for a combination of FDMU and FMI components. A complete integration of the two frameworks into one solution may be useful in the future. Considering the differences in the underlying software architecture, a simpler solution using adapters between FMI and FDMU seems more appropriate in the near future.

In general, there are two options for the implementation of adapters. For FDMU users, FMI slave components can be integrated in the FDMU framework via an FDMU wrapper (*Figure 3*). This wrapper completely encapsulates an FMI component. The wrapper has to emulate all the needed FMI Master function calls and call-back functions.

For FMI users, an adapter between FMI Master and FDMU Wrapper is needed. The adapter can be implemented by an FMI Slave component, which provides the Web service interface to an FDMU Wrapper (*Figure 4*).

The advantage of both solutions is that a user continues to work with his or her favourite framework and has access to additional components of the other framework. The disadvantage of the adapter-based solution is that only a subset of framework functions can be supported. For instance, it is not possible to emulate FMI call-back functions within a Web Service-based framework like FDMU in an efficient way. Further investigation of the details of the adapter-based solutions is needed.

Last but not least a third approach seems reasonable: Both frameworks can be coupled via a bridge (*Figure 5*). In this case a bidirectional communication between FMI Master and FDMU Master is needed. In contrast to the adapter-based solutions this approach is not based on open interfaces. It means a

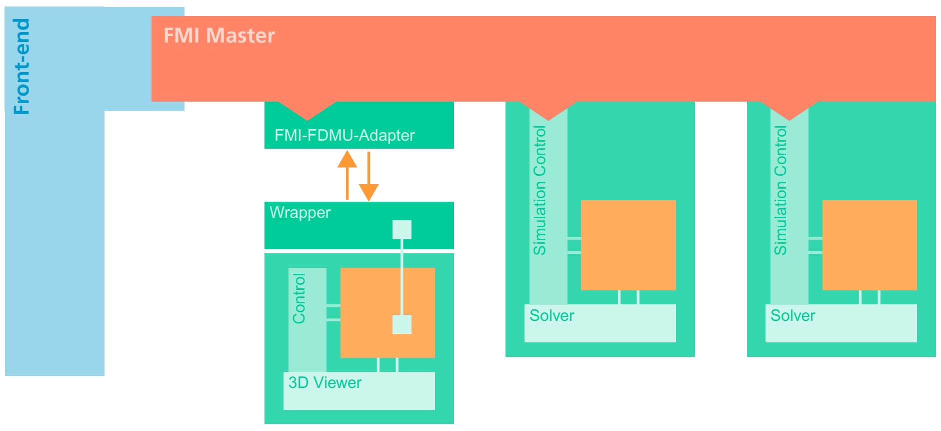

Figure 4: Idea of integrating FDMU under FMI

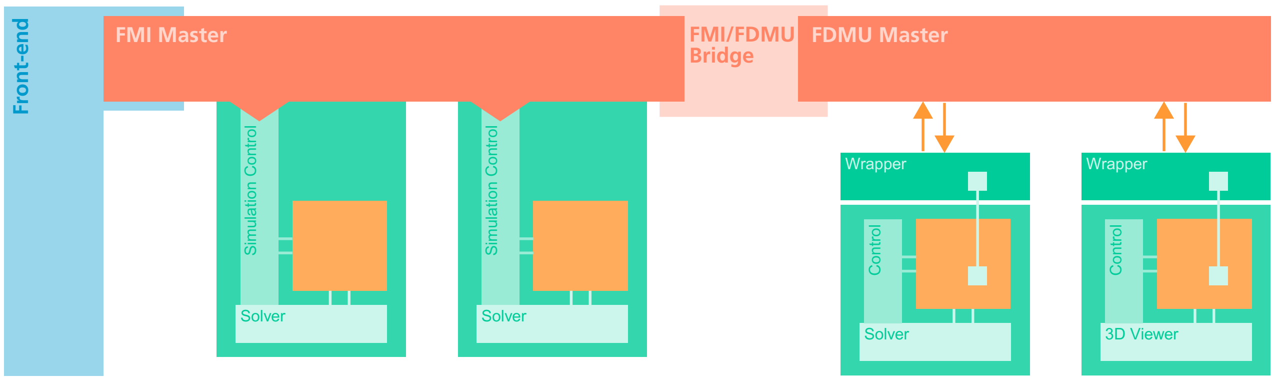

Figure 5: Idea of bridging FMI and FDMU

bridge implementation needs access to the internal interfaces of the two masters. As the authors of this paper are involved in the development of both FMI and FDMU this restriction is less important. Finally, the bridge solution could be a first step forward in unifying FMI approach as well as FDMU approach in the future.

6 Summary

The paper compares the two simulation approaches for heterogeneous systems called *Functional Digital Mock-up* (FDMU) and *Functional Mock-up Interface* (FMI). Both principles are well suited for a comprehensive investigation of multi physical systems. The main ideas of both approaches are shortly presented.

The FDMU approach focuses on a web service-based architecture for a flexible combination of different simulators and the involvement of an interactive visualization. Within this approach, the simulation tools can be used without additional implementations. However, the models must use one-directional connectors which may cause slight modifications.

The main ideas of the FMI specification are characterized by model exchange as well as co-simulation. The first idea realizes a passing over of submodels between different simulation tools. The second way defines a tight coupling between a so-called master simulation tool – acting mainly as caller – and all other simulation tools which are acting as slaves. For establishing the FMI ideas, the vendors have to implement additional code into their tools.

Both approaches have advantages and disadvantages. However, a combination of both principles seems to be possible and promising. Three opportunities for such a combination are shortly presented in the paper. All three ideas ought to be proved more particularly in the future.

References

[1] Bastian, J.; Clauß, C.; Wolf, S.; Schneider, P.: Master for Co-Simulation Using FMI. 8th International Modelica Conference, Dresden, Germany, March 20-22, 2011.

[2] Enge-Rosenblatt, O.; Schneider, P.; Clauß, C.; Schneider, A.: Functional Digital Mock-up – Coupling of Advanced Visualization and Functional Simulation for Mechatronic System Design. Proc. ASIM Workshop, Ulm, March 4-5, 2010.

[3] Modelisar: http://www.modelisar.org

[4] Noll, C.; Blochwitz, T.; Neidhold, T.; Kehrer, C.: Implementation of Modelisar Functional Mock-up Interfaces in SimulationX. 8th International Modelica Conference, Dresden, Germany, March 20-22, 2011.

[5] Otter, M.; Blochwitz, T.; Elmqvist, H.; Junghans, A., Mauss, J., Olsson, H.: Das Functional-Mockup-Interface zum Austausch Dynamischer Modelle. Keynote at ASIM Workshop, Ulm, March 4-5, 2010.

[6] Relovsky, B.: Overview of ITEA2 Project MODELISAR (I). Keynote at the 7th International Modelica Conference, Como, Italy, September 20-22, 2009.

[7] Schneider, P.; Clauß, C.; Enge-Rosenblatt, O.; Schneider, A.; Bruder, T.; Schäfer, C.; Voigt, L.; Stork, A; Farkas, T.: Functional Digital Mock-up – More Insight to Complex Multi-physical Systems. Multiphysics Simulation – Advanced Methods for Industrial Engineering, 1st International Conference, Bonn, Germany, June 22-23, 2010, Proceedings.

[8] Schubert, C.; Thomas Neidhold, Guenter Kunze: Experiences with the new FMI Standard - Selected Applications at Dresden University. 8th International Modelica Conference, Dresden, Germany, March 20-22, 2011.

[9] Stork, A.: FunctionalDMU – Eine Initiative der Fraunhofer Gesellschaft, 2006. http://www.functionaldmu.org

[10] Stork, A.; Thole, C.-A.; Klimenko, S.; Nikitin, I.; Nikitina, L.; Astakhov, Y.: Simulated Reality in Automotive Design. International Conference on Cyberworlds, Hannover, 2007.

[11] W3C: Web Service Standards WS-*, 2010. http://www.w3.org/standards/webofservices/