FunctionalDMU: Co-Simulation of Mechatronic Systems in a DMU Environment

André Stork, Mathias Wagner, Fraunhofer IGD; Peter Schneider, Fraunhofer IIS/EAS; Andreas Hinnerichs, Fraunhofer FOKUS; Thomas Bruder, Fraunhofer LBF

Introduction

DMU systems such as the TC Vis products from Siemens UGS and Dassault Systèmes' DMU Navigator are in widespread use in industry. They are characterized by the ability to process multiple CAD formats (multi-CAD) and possess functionality which permits the geometrical analysis of models, in particular for collision detection. However, what is largely missing from DMU is the ability to experience the behavior and functioning of the model.

In a way similar to the modeling of geometries in a variety of software systems, behavior and function models in the virtual product development process used for mechatronic components are also created and run in a range of different systems. Familiar representatives of this type of system include Matlab and Simulink, Modelica-based systems (Dymola, SimulationX), VHDL-AMS-compatible systems (e.g. Saber), as well as Adams, SimPack, Rhapsody, etc.

All these systems focus on different areas and, if used in isolation, are unable to provide satisfactory answers to all the questions involved in the global simulation of mechatronic systems. At the same time, industry demands for early, integration of systems engineering models are growing. This both forces system suppliers to be more open and stimulates the demand for innovative solution concepts.

Instead of implementing yet another one-to-one coupling between simulation tools, the Fraunhofer FunctionalDMU (FDMU) project has set itself the aim of providing an open, vendor-independent platform for the integration of a variety of simulators from the software, electrical/electronic and mechanical domains. Its vision is to map the demanded/desired functioning of software which does not yet exist on as yet non-existent hardware (electronic/mechanical), simulate the global functionality and, in this way, control digital (geometrical) models.

The FDMU framework has been evaluated on the basis of three application scenarios. The present article describes our methodological approach and the technical implementation at system level. We also provide an illustration of a selected application scenario.

Challenges, heterogeneity and solution concepts

As mentioned above, a large number of different systems are used in the various areas in which mechatronics play a part. These systems are based on different behavior description or behavior modeling languages. These behavior models are run in dedicated simulators. Although certain simulators benefit from the possibility of exporting or transferring models, this type of approach would not seem to represent a feasible overall solution. For this reason, the solution we wish to develop consists of a distributed, vendor-independent co-simulation environment with an extended

3D-DMU-visualization component which acts a uniform user interface for system simulation. Figure 1 depicts a sample configuration of the FDMU runtime environment together with the information that is supplied to the individual components.

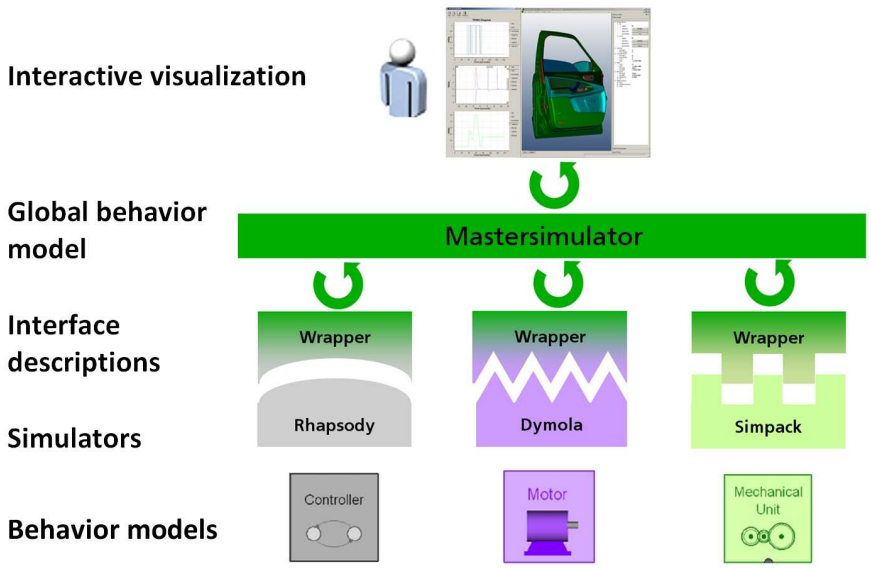

Figure 1: Sample configuration of the FDMU runtime environment

The FDMU runtime environment is structured as follows, from bottom to top:

- Native behavior models which are run
- by means of the corresponding simulators.
- Wrappers which perform a standardization of the various simulator interfaces and connect these to a Mastersimulator. At runtime, the wrappers receive a description of the behavior models' simulation values and map incoming or outgoing signals to these or convert outgoing values if necessary. In addition, the wrappers control the encapsulated simulator.
- The internal interfaces of the global behavior model are known to the Mastersimulator. It is responsible for simulator communication and coordination.
- The interactive visualization componet makes it possible to run and configure global simulations.

How are native behavior models integrated in a global simulation?

The FDMU approach assumes that behavior models are present in different modeling languages. These modeling languages may sometimes use different syntaxes, units, solvers etc. Similarly, the names used for the simulation values in the individual models may differ – the behavior models might, for example, have been developed and made available by a supplier. This makes it necessary to harmonize the simulation values.

We used the universal interface description language SysML in FDMU. SysML is internationally standardized and, as an XML-based language, is supported by an increasing number of XML-tools. SysML permits a standardized, tool-independent description of the interfaces of the native behavior models.

The user describes the interfaces of the native behavior models graphically and interactively using the FDMU editor. The user orients all the required native simulation values 'outwards' and maps these to standardized types in SysML. Users can assign appropriate geometry models to the behavior models in order to create functional building blocks (FBB) which encapsulate the geometry and function in accordance with an object-oriented template, e.g. the user can assign an electromechanical behavior model and a corresponding geometry model to an FBB 'Electric Motor'. When multiple behavior models described with SysML are linked together, a system model is created. By way of example, Figure 2 presents an extract from a simple system model for a window regulator:

- the native behavior models (Controller, Motor, Mechanical Unit) are each encapsulated in a SysML wrapper,
- important internal simulation values from the individual behavior models are directed to the SysML level,
- corresponding geometry models are assigned.

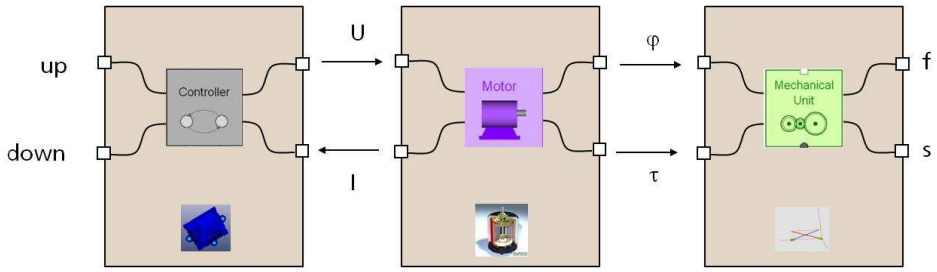

Figure 2: Extract from a simple system for a window regulator

It should be noted that so-called connectors have to be added to the native behavior models for the values that are to be derived – consequently, a library was developed during the FDMU project.

Constructing the global simulation model in SysML makes it possible to test the consistency of the model at an abstract level. In addition, it is possible to specify permitted ranges of values for the parameters. These can then be checked while the simulation is running.

How are different simulators integrated in the FDMU framework?

There are just as many different simulators as there are behavior modeling languages. The simulators differ in terms of their programming interfaces and the way in which they can be integrated in 'external' environments. Integration in the FDMU framework is achieved by means of wrappers which map the simulators' individual programming interfaces to a uniform interface. This is achieved by using the external function integration mechanism which is provided by many simulators.

The wrappers perform two tasks:

- They control the underlying simulator, i.e. during the initialization of the global simulation, they supply the behavior model, start the simulation, interrupt it if necessary, continue it or terminate it – all in response to the user's commands or to a control file.
- They ensure communication with the Mastersimulator, receive incoming data and return results. As part of this task, they map the native simulation values to the standardized parameters and *vice versa*.

How do the simulators communicate in the FDMU framework?

The Mastersimulator is the communication and synchronization hub in the FDMU framework. This Mastersimulator is the result of development conducted within the FDMU project. The Mastersimulator coordinates the execution of the global simulation model. It uses the wrappers to modify the individual simulations and ensures the routing of data between the wrappers and other connected services such as the FDMU visualization service. The Mastersimulator orchestrates the global simulation and does not contain any simulators itself. We have prepared a number of interfaces which permit the integration of a variety of Mastersimulator algorithms which can be used to modify the sequence of the individual simulations. The aim here is to achieve the dynamic adaptation of step sizes and – insofar as is permitted by the individual simulators – determine solutions, if necessary, on an iterative basis at any given time.

One important component of the Mastersimulator is the TransferHandler, which ensures correctly adapted data communications between the connected simulators. Figure 3 presents a diagram in which the user can operate a switch (here a window regulator) at any given time, whereas the controller expects a signal with equidistant sampling points. The TransferHandler can be configured to adjust the signals automatically. It possesses additional protocols such as upsampling, downsampling, etc.

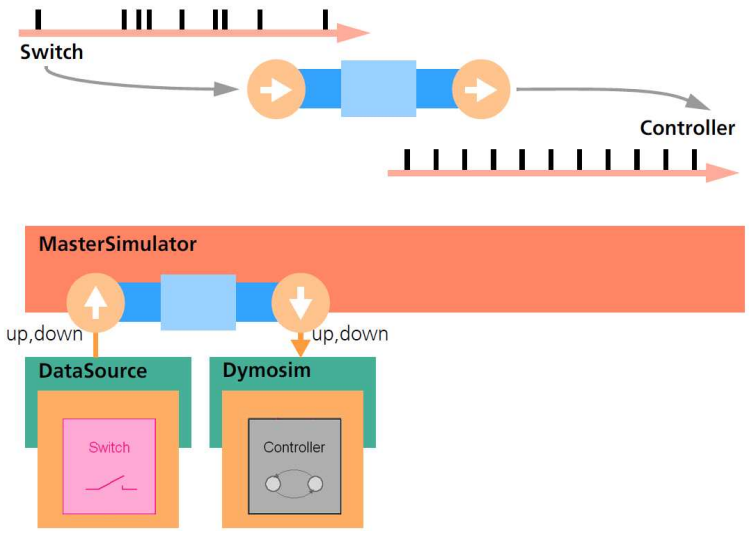

Figure 3: Example of the TransferHandler in the Mastersimulator

How do users interact with the FDMU framework?

In general, DMU makes high demands in terms of the need to interact with large volumes of data which are traditionally geometrical in nature. It therefore makes sense to encourage interactive capabilities in FDMU, thus leading to increased requirements being placed on the simulators, which are not usually designed for use in interactive situations. Interactive simulations are characterized by the fact that before the simulation starts, it is not known what result will occur and when. However, not all FDMU application scenarios require these interactive capabilities. It is also imaginable that FDMU could be used for the automated sequential processing of test scenarios. To this end, the processing of input files is provided for as an alternative to user interaction.

The FDMU visualization component allows users to explore global simulations interactively. It comprises:

- the 3-dimensional display and navigation through the (DMU) geometry data,
- the generation of simulation results in the 3D scene or by means of buttons
- the modification of simulation parameters which are directly passed to the simulators,
- the observation of the progress of simulations in the 3D scene, and
- the analysis of simulation values in the form of graphs.

To this end, all the selected data that are communicated via the Mastersimulator are visualized in a standardized user interface.

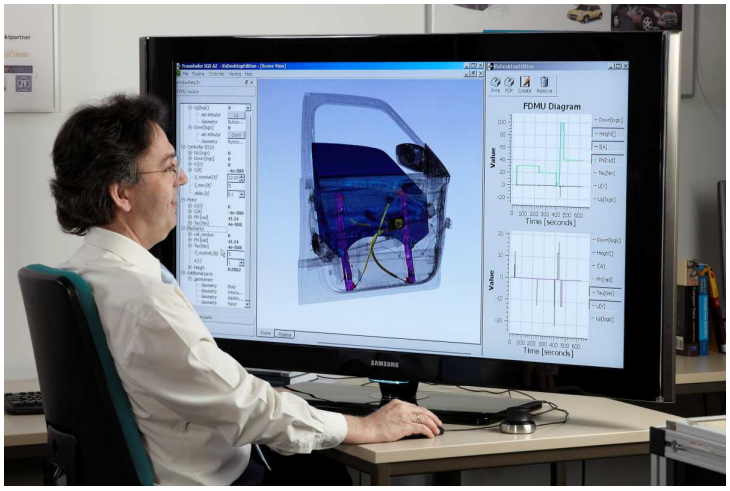

Figure 4: FDMU visualization component

Figure 4 illustrates the FDMU visualization component including the display of geometry data, a navigation and interaction bar on the right and functionality permitting the graphical presentation of signal forms on the left. It is possible to combine multiple plots and users are free to choose the simulation values they want to view. All the values are depicted on a common time axis.

In the center is the 3D graphics area which makes it possible to display even large DMU models. Here it is also possible to display and observe the movements that are induced by the global system

simulation. Additional functionalities include the display of signal forms (source/sink), the highlighting of geometries, collision detection etc.

On the right-hand side, the user can see the representations of the FBBs. Here it is possible to set and modify parameter values, read current simulation values and perform other similar operations. The control elements for the FBBs are automatically generated from the global system description together with boundary constraints for the parameters as previously specified, e.g. discrete values or permitted ranges.

Selected application scenario: thermodynamic simulation of electric motor and controller

An electric motor is equipped with control electronics including a set-point adjuster. The electronic components are located on a printed circuit board which is fixed externally to the front of the motor. The electrical behavior of the components on the board undergoes thermal influences due to the heat generated when the motor is running.

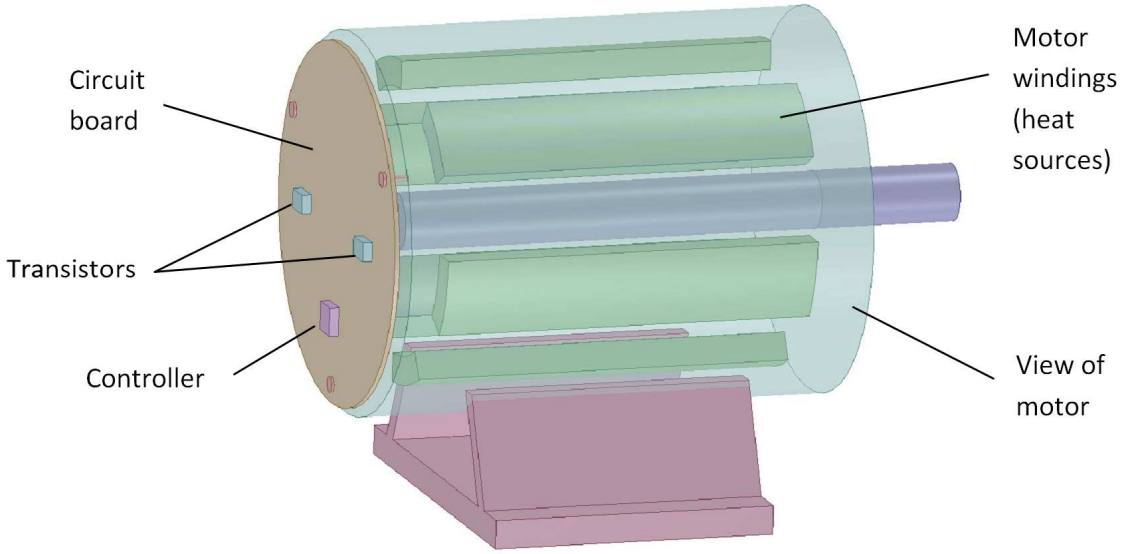

Figure 5: Electric motor with controller on rear

Typical design questions that need to be answered are as follows:

- Is the motor layout correct?
- How much do the power transistors heat up?
- How great is the additional influence of motor heating on the transistors?
- Does the heating of the motor affect other components (e.g. controller)?
- What happens if larger heat sinks are used for the transistors?
- Would increasing the distance between the motor and the circuit bord help?

The breakdown of the model is depicted in Figure 6. The electrical unit comprises the power amplifiers. The mechanical unit contains a damping element and transmits the load profile to the motor. The electrical unit and motor have reciprocal electrical and thermal influences on one another. The simulation values are also listed in Figure 6.

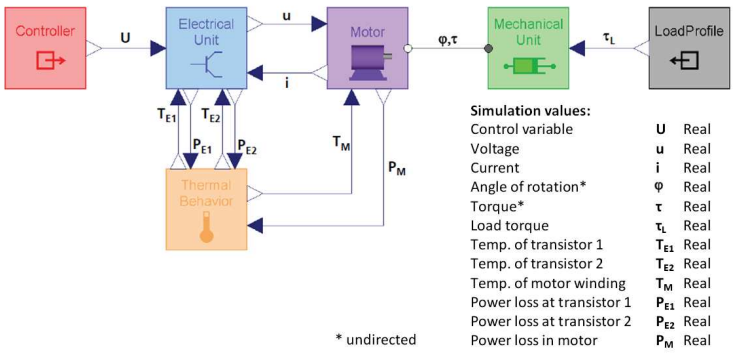

Figure 6: Breakdown of the global system

This application scenario integrates a thermodynamic simulation based on the FE method in the global simulation and emphasizes the potentially wide range of applications for the FDMU framework. The heat propagation model was created using Ansys in a 50,000-node network. Model order reduction procedures were used to shorten the computation times and make the design of the simulation more interactive. Consequently, two models (one with 100 nodes and one with 7 nodes) were created as a test. The drastic reduction in the number of equations involved in the model resulted in a significant improvement in the performance of the global system while preserving the key characteristics of the thermodynamic submodel.

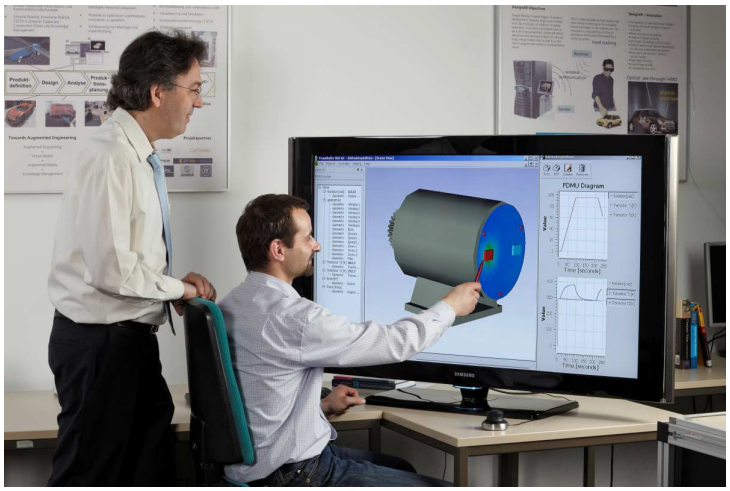

Figure 7: Viewing the results using the FDMU visualization component.

Figure 7 presents a snapshot of a possible visualization of the thermal results. The current temperatures at the measuring points are displayed as colors at the nodes of a triangular network. An interpolation is performed between these points. In this way, it is an easy task to track the temperature distribution through the simulation.

This example of a thermodynamic electric motor is an impressive example of the range of interconnections that the FDMU framework currently makes possible. Comprehensive solutions already exist for simulators such as Dymola, Saber, Simpack and Matlab and Simulink which cover, among others, the behavior description languages Modelica, MAST, VHDL-AMS and Simulink S-functions. A prototype solution currently exists for FE models and this will be extended in the future.

The uses and advantages of FunctionalDMU

FunctionalDMU represents the creation of a new design review tool with integrated visualization component and the ability to influence multidomain simulations. The standardized user interface permits centralized access to linked simulations. The visualization component supports the communication of proposed solutions and problems occurring in the global system across domain boundaries.

The FDMU framework permits the rapid construction of coupled simulations without requiring users to concern themselves with the specific connections between simulators. Overall, the flexible composition of the simulators and behavior models opens up new opportunities for coupled, multi-domain simulations.

Problems relating to the interactions between the behavior models can be identified early on, thus making it possible to draw conclusions about the expected system behavior. Both global systems and subsystems can be designed on a context-specific basis. Consistency checks can be performed at model level and possible conflicts between the behavior models in the subsystems or in the assumptions made concerning the input and output parameters can be identified during the simulation. In the future, it is conceivable that FDMU may be embedded in optimization solutions.

The ability of the system to run in distributed mode satisfies the requirement relating to the use of existing resources, in particular in terms of licenses and computer capacity.

Summary

With the FunctionalDMU framework, it is now possible to couple mechatronic domains at a level that goes beyond the limitations of commercially available simulation tools. This makes it possible to reduce the time required to develop mechatronic systems and exercise greater control over the time and effort involved in conducting functional design reviews.

The FunctionalDMU framework is based on vendor- and platform-independent technologies and can therefore be used in a variety of environments. The framework is also remarkable for its open, easily extendable architecture. Consequently, in addition to the existing interfaces to the simulators Rhapsody, SimPack, Saber, Dymola (Modelica), Matlab and Simulink, it is also possible to connect other simulators. The use of a service-oriented architecture makes it possible to distribute the computation loads across different systems.

A modeling, integration and utilization methodology has been developed to assist in the structured introduction and use of FDMU. This permits users, for example, to use simulators and hardware more flexibly, gain a rapid description of the global behavior and identify problems at an earlier phase of product development.

The feasibility of implementing cross-domain simulations has been proven on the basis of the scenario described here as well as two further scenarios (electric window regulator and steering test rig).

Contact:

Mathias Wagner
Fraunhofer Institute for Computer Graphics Research (IGD)
Darmstadt

Tel.: +49 6151 155 470
E-mail: mathias.wagner@igd.fraunhofer.de

Bildlegenden

Abb. 1

Interaktive Visualisierung	Interactive visualization
Gesamtverhaltensmodell	Global behavior model
Schnittstellenbeschreibungen	Interface descriptions
Simulatoren	Simulators
Verhaltensmodelle	Behavior models

Abb. 5

Platine	End plate
Transistoren	Transistors
Controller	Controller
Motorwicklungen (Wärmequellen)	Motor windings (heat sources)
Blick in den Motor	View of motor

Abb. 6

Schnittstellen	Interfaces
Stellgrösse	Control variable
Spannung	Voltage
Strom	Current

Drehwinkel	Angle of rotation
Drehmoment	Torque
Lastmoment	Load torque
Temp. Transistor 1	Temp. of transistor 1
Temp. Transistor 2	Temp. of transistor 2
Temp. Motorwicklung	Temp. of motor winding
Verlustleistung Transistor 1	Power loss at transistor 1
Verlustleistung Transistor 2	Power loss at transistor 2
Verlustleistung Motor	Power loss in motor
* ungerichtet	* undirected