ENABLING THE SIMULATION OF COMPLEX, HETEROGENEOUS SYSTEMS USING THE FUNCTIONAL MOCKUP INTERFACE (FMI)

Hubertus Tummescheit
Modelon Inc & Modelica Association
June 2015
My Background

• MSc Mechanical Engineering Technical University Hamburg-Harburg, 1996
• Modelica Development since 1997
• PhD Automatic Control Lund University 2002 with Karl Åström
• UTRC East Hartford in 2002–2003
• Co-Founded Swedish Company Modelon AB in 2004
• CEO Modelon AB/Inc. Since 2004/2013
• Member of Modelica Association Board and FMI Steering Group
OVERVIEW

- Motivation
- Use Cases
- What is the Functional-Mockup-Interface (FMI)?
- The Business Case for FMI
- Using FMI to improve design flows in the SE ‘V’
- Examples for FMI use in advanced design flows
  - Model Deployment
  - Early Validation for automotive controls
  - Real-time simulation on multiple cores
  - High-Fidelity ABS Brake validation
Modelon Overview & Locations

- Ann Arbor, MI
- Hartford, CT

Modelon Headquarter
- Lund
- Gothenburg

- Munich
- Hamburg

- Global premier provider of Modelica and FMI based solutions for model based systems development
- ~50 engineers (MSc / PhD levels) dedicated to Modelica and FMI
- Global customers mostly among Fortune 500 technology companies in Automotive, Aerospace, Energy and Industrial Equipment
- About 30% annual organic growth
THE COMPLEXITY ISSUE

- System complexity increases while
- Required time to market decreases (most industries)
- Without disruptive changes, an impossible equation to solve.

Source: DARPA AVM pres.
1. WHY FMI?

Problem

- Due to **different applications**, **models** of a system often have to be developed using **different programs** (modeling and simulation environments).

- In order to **simulate** the system, the different programs must somehow interact with each other.

- The system integrator must cope with simulation environments from many suppliers.

- This makes the **model exchange** a necessity. No current standardized interface.

- Even though **Modelica®** is tool independent, it cannot be used as such a standardized interface for model exchange.
USE CASE I:

Combined simulation for system integration

Solution

- As a universal solution to this problem the **Functional Mockup Interface (FMI)** was developed by the EU-project MODELISAR, and is now maintained by the Modelica® Association.
USE CASE II:

• Combine different modeling formats into coherent co-simulation (cyber-physical systems)
  ▪ Physical models, 1D-3D
  ▪ Controls / Software
USE CASE III: FMI FOR SIL AND HIL

- FMI export support from Controls Tools:
  - Simulink through FMIT Coder (Modelon)
  - Scade Suite (safety critical applications)

- FMI supported by most major HIL Vendors
  - DSPACE
  - National Instruments
  - Concurrent
  - IPG
  - Speedgoat

- FMI for ECU virtualization
  - Silver by Qtronic
  - ETAS tools (Bosch)
FUNCTIONAL MOCKUP INTERFACE (FMI)

• Tool independent standard to support both model exchange and co-simulation of dynamic models
• Original development of standard part of EU-funded MODELISAR project led and initiated by Daimler
• First version FMI 1.0 published in 2010
• FMI currently supported by more than 60 tools (see www.fmi-standard.org for most up to date list)
• Active development as Modelica® Association project
• FMI 2.0 released July 2014 and brings additional functionality to FMI standard
FMU: A MODEL WITH STANDARD INTERFACE

• A component which implements the FMI standard is called **Functional Mockup Unit (FMU)**

• Separation of
  ▪ Description of interface data (XML file)
  ▪ Functionality (C code or binary)

• A FMU is a zipped file (*.fmu) containing the XML description file and the implementation in source or binary form

• Additional data and functionality can be included

• Information & Interface specification: [www.fmi-standard.org](http://www.fmi-standard.org)
FMI FLAVORS

- The Functional Mock-up Interface (FMI) is a **tool independent standard** for
  
  - Model Exchange (ME)
  
  - Co-Simulation (CS)
  
- The FMI defines an interface to be implemented by an executable called Functional Mock-up Unit (FMU)
FMI: A BUSINESS MODEL INNOVATION

- FMI-compliant tools often allow liberally licensed export of models for distribution in the organisation and to partners
- Exported FMUs most often don’t require a license from the model authoring tool
- Deployment from few simulation specialists to designers, domain specialists, control engineers
  - One FMU used by many engineers (control design)
  - One FMU run on many cores (robust design)
FMI: A BUSINESS MODEL INNOVATION

1. Separate the model authoring tool from the model execution tool!
2. Free the model unit (FMU) from license restrictions
3. Make the standard widely accepted: https://fmi-standard.org/tools
TYPICAL FMI-BASED WORKFLOWS

Model Authoring Tool(s)

- Additional work flow automation for
  - pre-processing,
  - model calibration,
  - post-processing,
  - analysis,
  - automated reporting
  - automated requirements verification

Export: exported FMU freely licensed

Low-cost Model Execution Platform
May combine FMUs from several tools

- True democratization of simulation
- Greatly improved utilization of models
AUTOMATED REQUIREMENTS VERIFICATION

• Systems Engineering centric FMI-based workflow example: automated requirements verification for hardware and software requirements

Development of a customized workflow to allow rapid iterations of plant & software configuration
MODEL DEPLOYMENT

- FMU deployed (native tool) to support multiple applications
• Engineers in different domains work with FMUs
  ▪ Share models, distributed collaboration, work in tool of choice, reduced license costs, protect IP, couple carefully!!
ENTERPRISE MODEL DEPLOYMENT

“Daimer, QTronic and Vector describe how Mercedes-Benz currently uses virtual ECUs to validate transmission control software for about 200 variants of the Sprinter series in a highly automated way on Windows PC”
# Types and Cost of Testing Platforms

<table>
<thead>
<tr>
<th>Early Cheap Many</th>
<th>Late Expensive Few</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual ECU</td>
<td>Physical ECU</td>
</tr>
<tr>
<td>Virtual Plant</td>
<td>SiL</td>
</tr>
<tr>
<td>Physical Plant</td>
<td>Rapid Control Prototyping</td>
</tr>
<tr>
<td></td>
<td>Prototype</td>
</tr>
</tbody>
</table>
HIL COMPARED TO SIL

Tasks:
- Adaptation
- (pre-)Tuning
- Single-ECU Test
- Multi-ECU Test
- Module Test
- Integration Test
- DEM Test
- Application Layer Test
- CAN Load
- HW Test/Diagnostic
- Power Consumption

- original Basic SW runnable if SiL-MCAL is used

SiL Virtual ECU
Application functions
BSW
SBS
MCAL
XCP

HiL Real ECU
Application functions
Basic SW
XCP
Firmware

Tasks:
- Adaptation
- (pre-)Tuning
- Single-ECU Test
- Multi-ECU Test
- Module Test
- Integration Test
- DEM Test
- Application Layer Test
- CAN Load
- HW Test/Diagnostic
- Power Consumption

<20% Test Effort

Compare:
- Cost
- Availability (when/how often)
- User Training
- Feedback Time

Legend:
BSW: Basic Software
MCAL: MicroController Abstraction Layer
XCP: eXtended Calibration Protocol
SBS: Silver Basic Software
DEM: Diagnostic and Event Management
REUSABILITY

- Reusable models in standard Modelica language as FMUs
  - Compiled models generated internally, from suppliers, from partners, etc.
  - Protect IP as required
  - Many more tools can participate than just Modelica® tools

Mechanical FMU

Control/Actuation FMU

Hydraulics FMU
NEW OPTIONS FOR PROCESS INNOVATION

• Availability of cost-effective interfaces between tools creates new opportunities for integrated work flows
• FMI overcomes the N^2-problem: reduced number and cost of interfacing
• How does this fit into the traditional Systems engineering design view?
• Example from the automotive industry
SYSTEM ENGINEERING WITH FMI-BASED WORKFLOWS

Multi-phase V development with Modelica & FMI

1. Multiple fidelity models of subsystems/components ranging from FEM/CFD to system models

Requirements & Performance Targets
System Design
Platform Based Design
Module Based Design
Component Design
Module Integration & Verification
Component Verification
System Integration & Validation
Finished Product
SYSTEM ENGINEERING WITH FMI-BASED WORKFLOWS

Multi-phase V development with Modelica & FMI

2. Continuous V & V with FMI-based workflow automation
SYSTEM ENGINEERING WITH FMI-BASED WORKFLOWS

Multi-phase V development with Modelica & FMI

3. Seamless integration with Control Design, SIL, HIL and real-time Simulators
SYSTEM ENGINEERING WITH FMI-BASED WORKFLOWS

Multi-phase V development with Modelica & FMI

4. Early Validation with high-fidelity real time simulators

Requirements & Performance Targets
System Design
Module Design
Component Design
Component Verification
Module Integration & Verification
System Integration & Validation
Finished Product
Use Case Examples
DEVELOPMENT TO DEPLOYMENT

Functional Mockup Interface (FMI)

FMU Export via Model Export

PyFMI

Custom GUI

Excel

FMU Simulator

Parameters

Results

SIMULATION OF PHYSICAL MODELS IN PYTHON

CODE EXAMPLE:

```python
# Imports
from pyfmi import FMUModel
import matplotlib.pyplot as plt
# Load model
vm = FMUModel('MyFMU.fmu')
# Set a parameter
vm.set_value('p_1', 1.3)
# Simulate
res = vm.simulate(tfinal=10)
# Get the results
x = res['x']
plt.plot(x)
plt.xlabel('t')
plt.ylabel('x(t)')
plt.show()
```
VTM SYSTEM MODEL ARCHITECTURE

- Thermal
- Mechanical
- Electrical
- Coolant
CUSTOM USER GUI

- Intuitive and highly customizable design
- Runs powerful physical models through FMI
- Batch Runs/DOE support
- Access to component database
- Small model footprint
- Integrated analysis and export of results

Integrates model configuration, parameterization, simulation, and post-processing into custom front end for end-user deployment
CONTROLS DESIGN: ACTIVE SHUTTERS

• DOE results for improved controller design: sample application

Mechanical shutter shuts the grill when speed is above 18 m/sec.

Grill position controlled as functions of vehicle speed, engine torque, coolant temp, and oil temp.

Look-up tables derived from DoE
Real time & use of Multiple cores
Encapsulated in the FMI-interface
TECHNICAL BACKGROUND

• Real-time applications
• Fixed step solvers: assurance of upper limit of computation time
• Explicit is fast but requires smaller step size, heavily limiting system stiffness
• Implicit is slow but stable and handles stiffness
• Leads to large nonlinear systems of equations
• Expensive (typically $O(n^3)$)
• Inlining of implicit solvers can reduce simulation time somewhat
REAL TIME TUNING IN MODELICA TOOLS

• Some Modelica tools use best-in-class and state-of-the-art methods to achieve real time capability for complex models
  ▪ Unique in combining symbolic and numeric methods
• Symbolic Methods
  ▪ Time-discretize equations with most appropriate numeric scheme in as part of symbolic processing “inlining”.
  ▪ Partitioning of equation systems
• Numeric Methods
  ▪ Wide selection of fixed-step, low and high order, explicit and implicit integrators
SCHEDULING - IDEA

• Partitioning needs to be made automatically
• Modelica gives good possibility to automatically partition the model equation execution into separate threads
• Equations are traditionally sorted in sequence, but ‘Some FMI-generators are now able to execute independent sections in parallel

See: http://www.ep.liu.se/ecp/096/038/ecp14096038.pdf

The block lower triangle (BLT) is the result after standard Dymola manipulation, and illustrates the dependencies of the model equations.

It can be compressed vertically to illustrate the parallel execution of blocks that don’t need input from each other.
VEHICLE EXAMPLE

• 300+ states (150+ DOF)
• Defines current state-of-the-art for high accuracy RT vehicle dynamics: **high fidelity real time**
• Inlining with implicit Euler gives one large system \{178\}
• Highly coupled dynamical system
• Dymola RT automation is not enough on its own
• Modification of vehicle model, to allow for separated different subsystems
PARTITIONING OF MODELICA MODEL

Front/rear separation
• Decouple to separate suspensions from body
• Resulting in 3 major sections
  ▪ Front suspension and wheels
  ▪ Rear suspension and wheels
  ▪ Vehicle body
• Many more partitions further down, not shown here
RESULT

• Sizes of manipulated in-lined implicit integration systems:

• \{25, 25, 40, 40, 20, 12, 14, 1, 1, 1, 1, 1, 1, 1, 1\}
POTENTIAL EFFECT (THEORETICAL NUMBER)

• Solving Systems of equations is “O(n^3)”

- \[
\frac{178^3}{\sum\{25, 25, 40, 40, 20, 12, 14, 1, 1, 1, 1, 1, 1, 1\}^3} = 32.84080824550166
\]

• Theoretically, parallelization can speed up by at most that ratio

• Parallelization
  - Depending on number of cores, execution time is just slightly above execution time for the largest / most costly system
MEASURED RESULTS

• <500µs average evaluation time on Concurrent hardware
• Max 1 time step to recover overrun @ 1ms
• Extreme excitation, jumping 360/police turn
• Good accuracy compared to offline simulation
ANIMATED RESULT OF VALIDATION RUNS
ILLUSTRATE A USE CASE

Functional validation of environment model and
- ACC (Adaptive-Cruise-Control and
- EBA (Emergency Brake Assist)
fuctions with Model-in-the-Loop at BMW
ILLUSTRATE A USE CASE

Function meets multiple function behavior
ACC–Adaptive Cruise Control and EBA–Emergency Brake Assistance

- Brake Model: DYMOLA
- Controller: SIMULINK
- Environment Mode with Sensors: ADTF
- Navigation: NAVTEQ
- Software Autosar Builder
- Virtual ECU (VAP)
- Instrument Cluster – Head up Display

**Modelon**

**Modelica**

**Global Automotive Advisory Group**

**IPG Automotive**

**Modelon**
ILLUSTRATE A USE CASE

Development environment for reconstruction the world from raw sensor signal

Dashboard: velocity, engine speed, gear, gas level, clutch, status of ACC/EBA controllers

Simulator: birds eye scene view (CarMaker)

Raw signal of free range sensor
ADAPTIVE CRUISE CONTROL IN ACTION
ILLUSTRATE THE USE CASES

Validation of AUTOSAR HMI Software components with virtual ECU by **Software-in-the-Loop** at BMW
USE CASE II: BRAKING ON WET ROAD

High fidelity antilock brake system simulation

• Hydraulic braking system and ABS logic modeled in Modelica / Dymola
• 3D brake pad and tire/road interaction in Abaqus

Use Case: Modelon & Dassault Systèmes
FMI ADVANTAGE

• Same model – different applications
ALL CONNECTED!
# Controls & Embedded FMI Tools

<table>
<thead>
<tr>
<th>Tool name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ASim - AUTOSAR Simulation</strong></td>
<td>AUTOSAR product from Dassault Systèmes</td>
</tr>
<tr>
<td><strong>ANSYS SCADE Suite</strong></td>
<td>SCADE Suite is a model-based development environment with certified code generation for safety critical embedded applications from ANSYS.</td>
</tr>
<tr>
<td><strong>AVL CRUISE</strong></td>
<td>Vehicle system analysis tool for the optimization of fuel efficiency, emission, performance and driveability, from office to HiL to testbed.</td>
</tr>
<tr>
<td><strong>CarMaker</strong></td>
<td>CarMaker is an open test- and integration-platform for MiL, SiL and HiL.</td>
</tr>
<tr>
<td><strong>ControlBuild</strong></td>
<td>Environment for IEC 61131-3 control applications from Dassault Systèmes</td>
</tr>
<tr>
<td><strong>CosiMate</strong></td>
<td>Co-simulation Environment from ChiasTek</td>
</tr>
<tr>
<td><strong>dSPACE SCALEXIO</strong></td>
<td>dSPACE SCALEXIO is a Hardware-in-the-Loop (HIL) integration and simulation platform from dSPACE. Please also refer to the dSPACE FMI sites for more information about the FMI 1.0 and FMI 2.0 support.</td>
</tr>
<tr>
<td><strong>dSPACE SYNECT</strong></td>
<td>dSPACE SYNECT is a data management tool from dSPACE that enables you to manage FMUs and Simulink models as well as their dependencies, versions and variants throughout the entire software development process. Please also refer to the dSPACE FMI sites for more information about the FMI support.</td>
</tr>
<tr>
<td><strong>dSPACE VEOS</strong></td>
<td>dSPACE VEOS is a PC-based virtual integration and simulation platform from dSPACE. Please also refer to the dSPACE FMI sites for more information about the FMI 1.0 and FMI 2.0 support.</td>
</tr>
</tbody>
</table>
## Controls & Embedded FMI Tools

<table>
<thead>
<tr>
<th>Tool name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dymola</td>
<td>Modelica environment from Dassault Systèmes.</td>
</tr>
<tr>
<td><strong>ETAS - ASCMO</strong></td>
<td>Creation and export of statistical (meta) models using Design of Experiments (DoE) from ETAS.</td>
</tr>
<tr>
<td><strong>ETAS - FMI-based Integration and Simulation Platform</strong></td>
<td>Integration and simulation platform based on FMI 1.0 from ETAS.</td>
</tr>
<tr>
<td><strong>ETAS - FMU Generator for ASCET</strong></td>
<td>FMU Generator for ASCET from ETAS.</td>
</tr>
<tr>
<td><strong>ETAS - FMU Generator for Simulink®</strong></td>
<td>FMU Generator for Simulink® from ETAS.</td>
</tr>
<tr>
<td><strong>ETAS - INCA-FLOW (MiL/SiL Connector)</strong></td>
<td>Guided and automated calibration of FMUs with connection to ETAS INCA.</td>
</tr>
<tr>
<td><strong>ETAS - ISOLAR-EVE (ETAS Virtual ECU)</strong></td>
<td>PC based platform from ETAS for ECU software validation at the component, sub-system or system level; allows for validation of Application Software, Basis Software and complete ECU software in a virtual environment.</td>
</tr>
<tr>
<td><strong>ETAS - LABCAR-OPERATOR</strong></td>
<td>Frontend for ETAS HiL systems LABCAR, operating on the creation of experiments and their subsequent execution.</td>
</tr>
<tr>
<td>FMI add-on for NI VeriStand</td>
<td>NI VeriStand supports FMI through the use of the FMI add-on for NI VeriStand from Dofware</td>
</tr>
<tr>
<td>FMI Library</td>
<td>Open source (BSD) C library for integration of FMI technology in custom applications by Modelon.</td>
</tr>
<tr>
<td>FMI Toolbox for MATLAB/Simulink</td>
<td>The FMI Toolbox for MATLAB/Simulink from Modelon enables FMU import and export for MATLAB/Simulink for both model exchange and co-simulation.</td>
</tr>
<tr>
<td>FMUSDK</td>
<td>FMU Software Development Kit from QTronic.</td>
</tr>
</tbody>
</table>
## Controls & Embedded FMI Tools

<table>
<thead>
<tr>
<th>Tool name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM Rational Rhapsody</td>
<td>IBM® Rational® Rhapsody® family provides a collaborative design, development and test environment for systems engineers and software engineers.</td>
</tr>
<tr>
<td>MESSINA</td>
<td>MESSINA is a test platform for model-based ECU function development.</td>
</tr>
<tr>
<td>NI LabVIEW</td>
<td>Graphical programming environment for measurement, test, and control systems from National Instruments</td>
</tr>
<tr>
<td>PyFMI</td>
<td>For Python via the open source package PyFMI from Modelon. Also available as part of the JModelica.org platform.</td>
</tr>
<tr>
<td>Silver</td>
<td>Generation of virtual ECUs and virtual integration platform for Software in the Loop from QTronic.</td>
</tr>
<tr>
<td>xMOD</td>
<td>Heterogeneous model integration environment &amp; virtual instrumentation and experimentation laboratory from IFPEN distributed by D2T.</td>
</tr>
</tbody>
</table>