10 Years of FMI
Where are we now?
Where do we go?

A. Junghanns (QTronic GmbH)  FMI Project Leader
T. Blochwitz (ESI ITI GmbH)  FMI Deputy Project Leader
Contents

▪ Where are we now? How did we get there?
  ▪ Motivation
  ▪ Main Idea
  ▪ FMI for Co-Simulation, FMI for Model Exchange
  ▪ Releases
  ▪ Tools supporting FMI

▪ Where do we go?
  ▪ Roadmap towards the next version

▪ FMI is great but not magic

▪ Some Use Case

▪ Organizational Structure

▪ Related Research Projects

▪ Conclusions
Where are we now?
## Motivation

### Concept
- Architecture
- Requirements
- Executable Specification
- UML
- SysML

### Design and Optimization
- Component
  - FEM, CFD
  - MBS
  - Modelica
  - VHDL-AMS
  - Signal based

### Implementation
- Model
  - Predictive Control
  - Observers
  - Diagnostics

### Test and Verification
- MiL
- SiL
- HiL
- VHiL

### Operation
- Operator Training
- Monitoring
- Predictive Maintenance

---

<table>
<thead>
<tr>
<th>Concept</th>
<th>Design and Optimization</th>
<th>Implementation</th>
<th>Test and Verification</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture</td>
<td>Component</td>
<td>Model</td>
<td>Model</td>
<td>Model</td>
</tr>
<tr>
<td>Requirements</td>
<td>FEM, CFD</td>
<td>Predictive Control</td>
<td>MiL</td>
<td>Operator Training</td>
</tr>
<tr>
<td>Executable Specification</td>
<td>MBS</td>
<td>Observers</td>
<td>SiL</td>
<td>Monitoring</td>
</tr>
<tr>
<td>UML</td>
<td>Modelica</td>
<td>Diagnostics</td>
<td>HiL</td>
<td>Predictive Maintenance</td>
</tr>
<tr>
<td>SysML</td>
<td>VHDL-AMS</td>
<td></td>
<td>VHiL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Signal based</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Motivation

Challenges for Functional Mock-up:
- Different tools and languages are involved
- No standards for model interface and co-simulation available
- Protection of model IP and know-how of supplier

Modelisar project:
- Functional Mock-up Interface for Model Exchange and Co-Simulation
Functional Mock-up Interface

EU project Modelisar (2008 – 2011, 26 Mill. €, 178 my)

- Initiated by Daimler AG, 28 European partners
  - Tool vendors
  - Users
  - Research organizations
- Proof of concept in industrial use cases

After 2011

- Continuation as Modelica Association Project
- Modelica Association changed its bylaws to become an umbrella organization for projects related to model based system design
FMI – Main Design Idea

- **FMI for Model Exchange**

  ![Diagram of FMI for Model Exchange]

- **FMI for Co-Simulation**

  ![Diagram of FMI for Co-Simulation]
FMI – Main Design Idea

- A component which implements the interface is called a Functional Mockup Unit (FMU)

- Separation of:
  - Description of interface data: XML file
  - Functionality: Application Programming Interface (API) in C

- An FMU is a zipped file (*.fmu) containing:
  - modelDescription.xml
  - Implementation in source and/or binary form
  - Additional data and functionality
XML Model Description

Interface definition is stored in an xml-file:

- Implementation and capability flags
- Definition of physical units
- Definition of variable types
- Variables and their attributes
- Dependency information
C-Interface

- Instantiation:
  
  \[
  \text{fmiComponent } \text{fmiInstantiate}(\text{fmiString } \text{instanceName}, \ldots)
  \]

  Returns an instance of the FMU. Returned \text{fmiComponent} is an argument of the other interface functions.

- Functions for initialization, termination, destruction

- Support of real, integer, boolean, and string inputs, outputs, parameters

- Set and Get functions for each type:

  \[
  \begin{align*}
  \text{fmiStatus } \text{fmiSetReal} & \quad (\text{fmiComponent } \text{c}, \\
  & \quad \text{const fmiValueReference } \text{vr}[], \text{ size_t } \text{nvr}, \\
  & \quad \text{const fmiReal } \text{value}[])
  \\
  \text{fmiStatus } \text{fmiSetInteger} & \quad (\text{fmiComponent } \text{c}, \\
  & \quad \text{const fmiValueReference } \text{vr}[], \text{ size_t } \text{nvr}, \\
  & \quad \text{const fmiInteger } \text{value}[])
  \end{align*}
  \]

- Identification by \text{valueReference}, defined in the XML description file for each variable
FMI for Model Exchange

- Functionality of state of the art modeling methods can be expressed
- Support of continuous-time and discrete-time systems
- Model is described by differential, algebraic, discrete equations

- Interface for solution of hybrid Ordinary Differential Equations (ODE)
- Handling of time, state and step events, event iteration

- Discarding of invalid inputs, state variables

- No explicit function call for computation of model algorithm
  - FMU decides which part is to be computed, when a `fmi2GetXXX` function is called
  - Allows for efficient caching algorithms
FMI for Model Exchange

Exchanged data:

- $t_0$, initial values (a subset of $v(t_0)$)
- $v$
- $u$
- $y$
- $x$
- $\dot{x}$, $z$

Enclosing Model

External Model (FMU Instance)

Solver

- $t$ time
- $m$ discrete states (constant between events)
- $p$ parameters of type Real, Integer, Boolean, String
- $u$ inputs of type Real, Integer, Boolean, String
- $v$ all exposed variables
- $x$ continuous states (continuous between events)
- $y$ outputs of type Real, Integer, Boolean, String
- $z$ event indicators

© 2018 FMI Modelica Association Project | www.fmi-standard.org
Co-Simulation

Motivation
- Simulation of heterogeneous systems
- Partitioning and parallelization of large systems
- Multirate integration
- Hardware-in-the-loop simulation

Definition:
- Coupling of several simulation tools
- Each tool treats one part of a modular coupled problem
- Data exchange is restricted to discrete communication points
- Subsystems are solved independently between communication points
FMI for Co-Simulation

- FMI enables sophisticated Co-Simulation Master Algorithms:
  - Optional variable communication step size
  - Optional higher-order approximation of inputs and outputs
  - Optional repetition of communication steps

- Capabilities of the slave are contained in the XML-file

- Master can decide which coupling algorithm is applicable

Tools which do not support all features are not excluded

- Asynchronous execution (allows for parallel execution)
FMI for Co-Simulation

Exchanged data:

- Status information
- Derivatives of inputs, outputs w.r.t. time for support of higher order approximation between communication steps
FMI for Model Exchange and Co-Simulation

- **Model Exchange:**
  (One model evaluation)

  
  | t_{Start} | t_C | time |

  /* Set inputs*/
  fmiSetReal(m, id_u, u, nu);
  fmiSetTime(m, t_C);
  fmiSetContinuousStates(m, x, nx);
  /* Get outputs*/
  fmiGetReal(m, id_y, y, ny);
  fmiGetDerivatives(m, derx, nx);
  fmiGetEventIndicators(m, z, nz);

- **Co-Simulation:**
  (One communication step)

  
  | t_{Start} | t_C | t_{C+1} |

  /* Set inputs*/
  fmiSetReal(s, id_u, u, nu);
  /* Do computation*/
  fmiDoStep(s, t_C, h_C, fmiTrue);
  /* Get outputs*/
  fmiGetReal(s, id_y, y, ny);
FMI Releases

FMI 1.0
- FMI for Model Exchange: January 2010
- FMI for Co-Simulation: October 2010

FMI 2.0
- Unification and harmonization of Model Exchange and Co-Simulation
- Clarification and improvement of specification document
- Improvement of usability
- Performance improvement for large models
- Release: July 2014

FMI 1.0.1
- No new features, only corrections and clarifications
- Release: July 2017
Tools supporting FMI

See http://fmi-standard.org/tools/

- All fields of modelling and simulation
- All development stages
- Test and verification
- Optimization

Some statistics:

<table>
<thead>
<tr>
<th>FMI Tools</th>
<th>2011</th>
<th>2012</th>
<th>2013</th>
<th>2014</th>
<th>2015</th>
<th>2017</th>
<th>2018</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support</td>
<td>15</td>
<td>32</td>
<td>37</td>
<td>47</td>
<td>80/42</td>
<td>87/52</td>
<td>108</td>
</tr>
<tr>
<td>Planned</td>
<td>9</td>
<td>7</td>
<td>10</td>
<td>6</td>
<td>8</td>
<td>8</td>
<td>6</td>
</tr>
</tbody>
</table>

During MODELISAR project, partners and non-partners
Where do we go?
Current Activities

Maintenance Release:
- No new features, but corrections and clarifications
- FMI 2.0.1: in development

FMI 3.0:
- 6 FMI Working Groups develop new features
- Alpha Features List was published in December 2017
- FMI Change Proposals are harmonized after FMI Design Meeting (April 4th/5th at ESI ITI in Dresden, Germany)
- Test implementations can start
FMI 3.0 Alpha Feature List

Ports and Icons:

- Help the user to build consistent systems from FMUs and render the systems more intuitively with better representation of structured ports (for instance busses and physical connectors) in the `modelDescription.xml`.
- Group inputs and outputs to connectors
- Define some semantics for support of flow and stream variables
- Allow the definition of a graphical representation for FMUs and connectors

Array variables:

- Allow FMUs to communicate multi-dimensional variables and change their sizes using structural parameters.
- New `fmi2Get/Set` functions for multidimensional variables
FMI 3.0 Alpha Feature List

Clocks and Hybrid Co-Simulation

- Introduces clocks for synchronization of variables changes across FMUs
- Co-Simulation with events:
  - Early return from fmi2DoStep in case of an event,
  - Introduction of Event mode similar to Model Exchange to support event iteration and super dense time

Binary Data Type:

- Adds an opaque binary data type to FMU variables to allow, for instance, efficiently exchanging of complex sensor data.
- Data is exchanged via an array of char
- Semantics of the data is defined by a MIME type
FMI 3.0 Alpha Feature List

Intermediate Output Values (FMI for Co-Simulation):

- Allow access to intermediate output values between communication time points from the FMU to disclose relevant subsystem behavior for analysis or advanced co-simulation master algorithms.
- Every time a co-simulation slave finishes an internal time step it calls a callback function. In this function variables can be retrieved via `fmi2GetXXX` function calls.

Source code FMUs:

- Adding more information to the `modelDescription.xml` file to improve automatic import and compilation of source code FMUs.
FMI is great, but not magic!
FMI is great: Why we love it!

- Standardized, open, vendor-neutral API
- Convenient container for handling simulation artefacts: storing, sharing, archiving...
- Free simulation users from modeling/generation tool knowledge
- Reduce IP sharing

- A new quality of simulation is attainable now, because:
  - Producing, sharing and using simulation components is simpler than ever
  - Coupling multi-disciplinary simulations is now more efficient than ever

Before FMI  |  With FMI  |  What we want
FMI is great: But not magic!

- Implementation quality is a continuous effort: new tools, new standard versions
- License issues: license-free generation? Digital rights managements?
- Numeric challenges remain:
  - Splitting systems into components increases numeric problems
  - Pressure towards Co-Simulation: simpler handling, but time delays introduce errors
  - Still based on floating point numeric: \((a + b) + c \neq a + (b + c)\)
  - Stability and speed of simulations still depend on solver technology
- We need to educate users and management
  - Experts are still needed
  - Goals need to stay realistic
Some Use Cases
FMI Use Cases: Automotive Industry

FMI usage is state of the art in automotive industry:

- OEMs and suppliers exchange FMUs (power train and chassis components) for integration of components to system models
- FMUs are integrated in engine test benches for real drive emission tests
- FMUs are integrated in Software-in-the-Loop and Hardware-in-the-Loop application for test and verification
FMI Use Cases: Development in Automotive Control Software

Engineers develop controllers with virtual systems combined of FMUs from different sources, e.g. Silver, SimulationX, Dymola,…

- Read more: E. Chrisofakis et. al.: Simulation-based development of automotive control software with Modelica. 8th International Modelica Conference, 20-22.03.2011, Dresden, Germany
  
  [Link topaper](http://qtronic.de/doc/SiL_at_Daimler_2011.pdf)
FMI Use Cases: Aerospace Industry

Simcenter Amesim FMUs are integrated with dSPACE SCALEXIO for real-time simulation and test:

FMI Use Cases: Aerospace Industry

Real-time capable model of electro hydraulic actuator in Simcenter Amesim:
FMI Use Cases: Aerospace Industry

The whole system in dSPACE SCALEXIO:

- Combination of FPGA based models and processor based models created from various model sources
- Models created via dSPACE XSG Electric Component Library, MathWorks Simulink, Siemens Simcenter Amesim
FMI Use Cases: Industry

Model-in-the-loop strategy for control development:

Courtesy of Thyssenkrupp Elevator
FMI Use Cases: Industry

Hardware-in-the-loop strategy for elevator systems:

- Read more: “Efficient safeguarding of elevator functionalities through virtual commissioning” ProSTEP iViP Symposium 2017
  http://www.prostep-ivip-symposium.org/fileadmin/Veranstaltungen/symposium17/Presentations/Presentation_Aloth_ESI-ITI.pdf
FMI Use Cases: FEM and system simulation

Coupling of SimulationX actuators and VPS FE model for loading of a crane to capture realistic and dynamic stress strain effects:
Related Research Projects
Research Projects, related to Modelica and FMI

ACOSAR (2015-2018):
- Advanced Co-Simulation Open Software Architecture
- https://itea3.org/project/acosar.html

EMPHYSIS (2017-2020):
- Standard for integration of code (in different levels) in ECUs

EMBrACE (in preparation):
- Environment for model-based rigorous adaptive co-design and operation of CPS (https://itea3.org/project/embrace.html)
- Specification of a common requirements modelling language, so that requirements can easily be understood by all stakeholders whatever their domain of expertise
Research Projects, related to Modelica and FMI

PEGASUS (2016-2019):

- Establishment of Generally Accepted Quality Criteria, Tools and Methods as well as Scenarios and Situations for the Release of Highly-automated Driving Functions

- Standardized Interface for Sensor Simulation
  => Open Simulation Interface (OSI)
  https://github.com/OpenSimulationInterface/open-simulation-interface

- Packaging of Sensor models into FMUs
  => OSI Sensor Model Packaging (OSMP) – Binary Variables
  https://github.com/OpenSimulationInterface/osi-sensor-model-packaging

ACOSAR

Objective:
▪ Tool independent standard to simplify integration of RT and non-RT systems

Motivation:
▪ Efficient integration of heterogeneous test systems
▪ Tool neutral integration of distributed co-simulation
▪ 9 Automotive use-cases

Plan:
▪ Transfer the result to the Modelica Association and maintain the standard within a Modelica Association Project in parallel to FMI

Facts
Framework: ITEA3 (Call1)
Duration: 09/2015 – 08/2018
Overall Budget: 7.9 M€
Countries: AT, DE, FR; 16 Partners
Coordinator: VIRTUAL VEHICLE (AT)
Website: www.acosar.eu
FMI Architectures

- **Standalone:**

- **Tool Based:**

- **Distributed:**

  ACOSAR
ACOSAR Approach

Distributed Co-Simulation Protocol (DCP):

- Network protocol is defined for different media (UDP, TCP/IP, EtherCAT, USB, Bluetooth)
- A DCP-Slave is described by an XML file (similar to FMI modelDescription)
- Same data types as in FMI (Real, Integer, Boolean, String, Binary, multi-dimensional) are supported
- Semantics of co-simulation is consistent to FMI for Co-Simulation
- Realtime and non-realtime co-simulation is supported

Enables:

- Distributed tool based interactive Co-Simulation
- Connection of test benches and simulation models
ACOSAR Use Cases

Non-real time Co-Simulation:

Simulator1

- Combustion engine
- Torque
- Accelerator pedal and Speed

Simulator2

- FMU
- Wheel speed
- Driving shaft torque
- Vehicle & Environment
- Torque
- Speed
- Accelerator pedal

DCP over UDP/IP
ACOSAR Use Cases

Real time Co-Simulation:

**Test bench**
- Combustion engine
- Dynamometer
- Accelerator pedal
- ECU
- Torque
- Speed
- Automation system

**HIL system**
- FMU
- Wheel speed
- Driving shaft torque
- Vehicle & Environment
- Torque
- Speed

**DCP via EtherCAT**
Organizational Structure
Project Rules

General conditions

- Results are owned by the Modelica Association (MA)
- Results are freely available under a copyleft license
- Contributors must sign Corporate Contributor License Agreement (CCLA)
- FMI MAP members need not to be MA members
- Meetings are open to the public

FMI Steering Committee

- Defines FMI policy, strategy, feature roadmap, releases
- Voting rights
- Bosch, Dassault Systèmes, dSPACE, ESI ITI, IFP EN, MapleSoft, Modelon, QTronic, Siemens

FMI Advisory Committee

- Contribute to FMI design
- Access to FMI infrastructure (repository, trac, meeting minutes)
- AVL, Armines, DLR, IBM, ETAS, Fraunhofer (IIS/EAS, First, SCAI), Open Modelica Consortium, Synopsys, TWT, University of Halle
Current Activities

Quality of tool implementations
- **FMI Cross Check Rules**
- Continuous maintenance of **FMI Compliance Checker**

Improvement of processes
- Adaption of FMI Project Rules to the current status
- **Definition of FMI Development Process**
- **Coordination of FMI Working Groups**

Public relations
- FMI is a registered trademark in Europe
- FMI logo is publicly available
- Rearrangement and new content of website (download, FAQ)
Resources

Website: fmi-standard.org
- FAQ
- Download specifications
- List of tools and cross check results

Ambiguities in specification, feature requests etc.:
- Public error tracking system trac.fmi-standard.org
- Send e-mail to contact@fmi-standard.org
- Contact your FMI-tool vendor

Mailing lists:
- FMI-Info, used for public announcements, subscribe via contact page
- FMI-Design, for active developers, send e-mail to contact@fmi-standard.org
Conclusions

- FMI is a unique initiative for Model Exchange and Co-Simulation
- Tool independent
- Developed in close cooperation between leading European CAE tool vendors
- Proof of concept in industrial use cases during development
- Fast adoption by tool vendors (100+ tools with FMI support)
- Used in industry and research
- Open and free access to FMI specification and additional material
- Continued maintenance and development as Modelica Association Project