FMI TUTORIAL

Hubertus Tummescheit / John Batteh
Modelon Inc. & INCOSE/NAFEMS SMSWG

INCOSE IW
January 25th, 2015
Torrance / CA
AGENDA

• FMI Overview
• FMI News
• Tutorial Overview
• Hands-on Exercises
  ▪ Presentation of Model (Modelica / Dymola)
  ▪ Hands on: import of FMUs into Excel
• Demo of FMI import in Dymola
• Demo/hands on exercise: FMI import into Python
• Demo/hands on exercise: FMI import into Simulink
• Demo: FMI export from Simulink
DOWNLOAD & INSTALLATION

• Material for hands-on exercises available on USB-sticks and public download

• Public Link: https://app.sugarsync.com/iris/wf/D1068299_77975033_6553767

• Installation instructions under folder Software/READMEFIRST.pdf

• Licenses for FMI Add-in for Excel and FMI Add-in for Matlab/Simulink under Licenses

• All Licenses valid until February 14th

• Open Source option: PyFMI, requires Python 2.7 installation
DOWNLOAD MATERIAL

Installation prerequisites and instructions

Tutorial instructions

License for Excel- and Matlab FMI tools
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.
1. WHAT IS IT ALL ABOUT?

Solution

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

• Combine different modeling formalisms into coherent simulation
  ▪ Physical models, 1D-3D
  ▪ Controls

FMI-based System Simulation
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 over 60 tools (see www.fmi-standard.org for most up to date list)
• Active development as Modelica Association project
• FMI 2.0 just released and brings additional functionality to FMI standard

Problems / Needs

⇒ Component development by supplier
⇒ Integration by OEM
⇒ Many different simulation tools
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


From the official FMI presentation (adapted)
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)
XML schema (.xsd) defined by the FMI specification

From the official FMI presentation (adapted)
FMI XML Schema

- **Information** not needed during execution is stored in one xml-file:
  - Complex data structures give still simple interface.
  - Reduced overhead in terms of memory.

From the official FMI presentation (adapted)
C-interface

• Two C-header files:
  - **Platform dependent definitions** (basic types):

```c
/* Platform (combination of machine, compiler, operating system) */
#define fmiModelTypesPlatform "standard32"

/* Type definitions of variables passed as arguments */
typedef void*    fmiComponent;
typedef unsigned int fmiValueReference;
typedef double fmiReal    ;
typedef int fmiInteger    ;
typedef char fmiBoolean    ;
typedef const char* fmiString    ;

/* Values for fmiBoolean */
#define fmiTrue  1
#define fmiFalse 0

/* Undefined value for fmiValueReference (largest unsigned int value) */
#define fmiUndefinedValueReference (fmiValueReference)(-1)
```

• **C-functions**:  
  - 18 core functions  
  - 6 utility functions  
  - no macros  
  - C-function name: `<ModelIdentifier>_<name>`, e.g. `Drive_fmiSetTime`
C-interface

• Instantiation:

```c
fmiComponent fmiInstantiateXXX(fmiString instanceName, ...)
```

• `fmiComponent` is a parameter of the other interface functions
  - Opaque `void*` for the importing tool
  - Used by FMU to hold any necessary information.

• Functions for initialization, termination, destruction

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

• **Set and Get** functions for each type:

```c
fmiStatus fmiSetReal (fmiComponent c, 
    const fmiValueReference vr[], size_t nvr, 
    const fmiReal value[])
```

```c
fmiStatus fmiSetInteger(fmiComponent c, 
    const fmiValueReference vr[], size_t nvr, 
    const fmiInteger value[])
```

• Identification by `valueReference`, defined in the XML description file for each variable

From the official FMI presentation (adapted)
FMI for Model Exchange

- Import and export of input/output blocks (FMU – Functional Mock-up Unit)
- Described by
  - differential-, algebraic-, discrete equations,
  - with time-, state, and step-events
- FMU can be large (e.g. 100000 variables)
- FMU can be used in an embedded system (small overhead)
- FMUs can be connected together

From the official FMI presentation (adapted)
Signals of a Model Exchange FMU

For example: 10 input/output signals (u/y) for connection and 100000 internal variables (v) for plotting.

From the official FMI presentation (adapted)
Co-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

• Motivation:
  ▪ Simulation of heterogeneous systems
  ▪ Partitioning and parallelization of large systems
  ▪ Multi-rate integration
  ▪ Software-in-the-loop simulation
  ▪ Hardware-in-the-loop simulation

From the official FMI presentation (adapted)
FMI for Co-Simulation

• Master/slave architecture
• Considers different capabilities of simulation tools
• Support of simple and sophisticated coupling algorithms:
  ▪ Iterative and straightforward algorithms
  ▪ Constant and variable communication step size
• Allows (higher order) interpolation of continuous inputs
• Support of local and distributed co-simulation scenarios

• FMI for Co-Simulation does not define:
  ▪ Co-simulation algorithms
  ▪ Communication technology for distributed scenarios

From the official FMI presentation (adapted)
FMI for Co-Simulation

• Signals of an FMU for Co-Simulation

\[ t_0, p, \text{initial values (a subset of } \{x_0, x_0, y_0, v_0, m_0\}) \]

- Inputs, outputs, and parameters, status information
- Derivatives of inputs, outputs w.r.t. time can be set/retrieved for supporting of higher order approximation

From the official FMI presentation (adapted)
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

Separate the model authoring tool from the model execution tool!
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
"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"
MULTIDOMAN COLLABORATION

• Engineers in different domains work in one formalism/tool
  - Share models, distributed collaboration, work in tool of choice, reduced life cycle costs, protect intellectual property carefully!!
REUSABILITY

- Reusable models in standard Modelica language
  - Off-the-shelf model libraries and components, focus on core knowledge, innovative systems from standard component models

Free open source
Commercial off-the-shelf
Consulting services
Partners / suppliers / customers / academia
In-house
REUSABILITY

- Reusable models in standard Modelica language as FMUs
  - Compiled models generated internally, from suppliers, from partners, etc.
  - Protect IP as required
  - Couple carefully!!
DEVELOPMENT TO DEPLOYMENT

Functional Mockup Interface (FMI)

FMU Export via Model Export

PyFMI

Custom GUI

Excel

FMU Simulator

SIMULATION OF PHYSICAL MODELS IN PYTHON

with PyFMI and Assimulo

CODE EXAMPLE

# Imports
from pyfmi import FMUModel
import matplotlib.pyplot as plt

# Load model
vdp = FMUModel('MyFMU.fmu')

# Set a parameter
vdp.set('p',3.1)

# Simulate
res = vdp.simulate(final_time=10)

# Get the results
x1 = res['x1']
t = res['time']

# Plot
plt.figure()
plt.plot(t,x1)
USE CASE: VIRTUALIZATION FOR CONTROLS

Virtualization: Objectives

- Running accurate closed-loop simulation of the complete system – On a PC
- All engineers equipped with a virtual vehicle
- System integration and feed-back within minutes

FMI

Virtual ECUs
Simulation of the ECUs, working as in the vehicle

Vehicle Simulation
High-fidelity, configurable
Silver allows:
- Easy sharing of information/results
- Use of simulation technology by non-specialists, without the simulation tool
- Protection of IP
- System behaviour on the laptop of every engineer (concurrent engineer.)
- Extremely fast change-validation-change cycles (few minutes!)
- Engineers immediately experience their changes in a system context

Silver runs a virtual prototype:
- On standard Windows PC
- Connecting virtual control and virtual plant using SiL-technology
- Using compiled behavioural models from many different tools without sources
- Allowing efficient and intuitive communication definition
- Allowing simple interaction with the virtual prototype: User “drives” system

FMU export: any FMI compatible tool
FMU import: Silver
FMI ADVANTAGE

• Same model – different applications
ALL CONNECTED!
PART II

Tutorial Overview
TUTORIAL OVERVIEW

- Goal: demonstrate FMI-based workflows in several FMI compliant tools with hands-on exercises
  - Modelica model (vehicle thermal management)
  - FMU creation
  - FMU import, simulation, and post-processing
- FMI is a standard but we need tools to work with it
- Several tools are provided to support tutorial, both open source and commercial (evaluation licenses)
- Choose exercises based on interest and tools (note some have tool pre-requisites, i.e. MATLAB/Simulink)
- FULL DISCLOSURE: tutorial based on tools in use at or developed by Modelon (full list of FMI-compliant tools at [www.fmi-standard.org](http://www.fmi-standard.org))
TUTORIAL USE CASE

Thermal Management

Multi-Engineering Modeling and Simulation

Python

FMI TOOLBOX for MATLAB®

FMI ADD-IN FOR EXCEL®

2015-01-25
GETTING STARTED

• USB sticks passed around with all tutorial files (instructions, software, licenses, sample files, FMUs)

• Public link: https://app.sugarsync.com/iris/wf/D1068299_77975033_6553767

• Open FMIWorkshop_Incose.pdf with full tutorial instructions

• Feel free to try your own FMI tools with workshop

• Ask questions if you need help or have problems
LICENSING LOGISTICS

• Dymola
  ▪ Full Dymola license required for FMI import and export

• MATLAB/Simulink
  ▪ FMI Toolbox (evaluation license provided) + MATLAB/Simulink required for FMU import into Simulink
  ▪ FMU export from Simulink also requires Simulink Coder
  ▪ Sample FMUs are 32 bit and require MATLAB/Simulink 32 bit

• Excel
  ▪ FMI Add-in for Excel (evaluation license provided)
  ▪ Requires 32 bit Microsoft Office

• Evaluation licenses expire on February 14th, 2015 but all FMUs included with tools execute with demo licenses per Users’ Guide (contact Modelon for more information)
THANK YOU FOR YOUR ATTENTION

john.batteh@modelon.com, (734) 274-5933