FMI TUTORIAL

Hubertus Tummescheit

INCOSE/NAFEMS SMSWG

INCOSE IW

January 26th, 2015
Torrance / CA
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 formalisms into coherent co-simulation
  ▪ Physical models, 1D-3D
  ▪ Controls

Modelica / 1-D Systems Simulation

Block Diagrams
Controls

CFD

FEA

FMI-based System Simulation

2015-01-28
USE CASE III: 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

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

Vehicle Simulation
High-fidelity, configurable
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
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

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)

**FMU=Model w/ Standard Interface**
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

Requirements ➔ Formalized requirements ➔ Executable model of requirements (e.g. FMU)

Physical plant ➔ Model of plant ➔ Deployable model of plant (FMU)

Software spec ➔ Software model or prototype ➔ Deployable model of software (FMU)

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

- FMU deployed (native tool) to support multiple applications

Mechanical

Electrical

Systems

Thermal
“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”
MULTIDOMAIN 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!!

Mechanical

Systems

Thermal

Electrical

Control

Fluid Power

FMUs
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
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 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('p1',3.1)
# Simulate
res = vdp.simulate(t_final=10)
# Get the results
x1 = res['x1']
t = res['time']
# Plot
plt.figure()
plt.plot(t,x1)
WHAT IS NEW IN FMI 2.0?

• Unification of the two flavors FMI-CS and FMI-ME
• Clarifications of the specification
• Improvement of the cross-checking protocol
• Interface improvements that allow for much higher quality coupling of simulations
  ▪ Dependency information for inputs/outputs/states
  ▪ Derivatives & analytic Jacobians
  ▪ Combinations of discrete/continuous systems
  ▪ Initialization
FMI ADVANTAGE

- Same model – different applications
ALL CONNECTED!