The Functional Mock-up Interface Beginners' Tutorial

Presenters:
Cinzia Bernardeschi, University of Pisa
Christian Bertsch, Bosch Research
Claudio Gomes, Aarhus University
Maurizio Palmieri, University of Pisa
Torsten Sommer, Dassault Systèmes

Online Materials:

https://github.com/modelica/fmi-beginners-tutorial-2023/tree/main

Agenda

Part 1: Introduction to FMI (40Min)

- Motivation / History
- How does FMI work?
- Tool support
- New features in FMI 3.0

Part 2: Working with (single) FMUs (45 min)

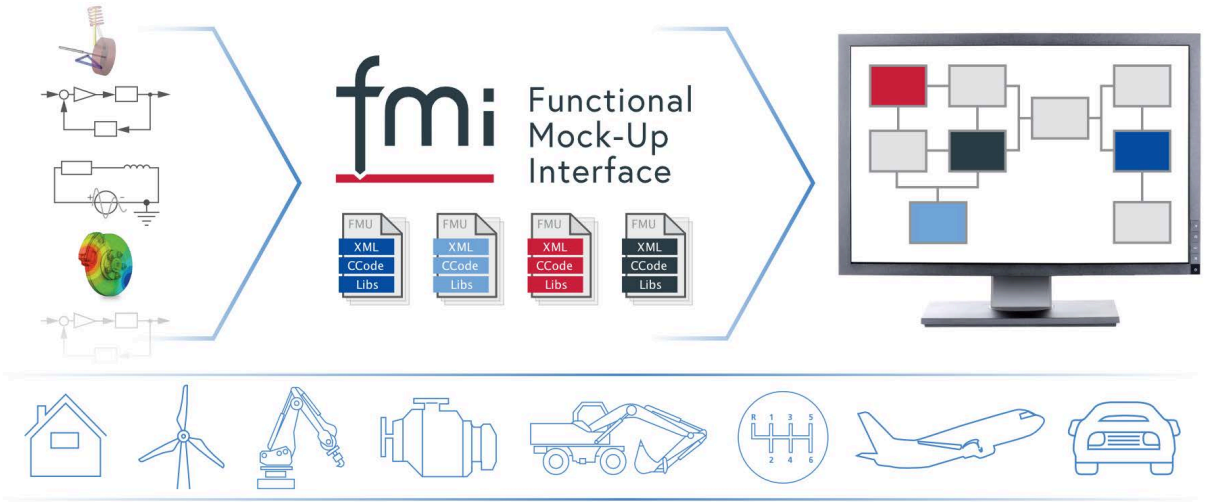
Part 3: Interacting with multiple FMUs (45 min)

Part 4: Closing Session (10 min)

Q&A

Introduction to FMI

Presenter:
Christian Bertsch, Bosch Research



Online Materials:

https://github.com/modelica/fmi-beginners-tutorial-2023/tree/main

Motivation, Timeline

Motivation

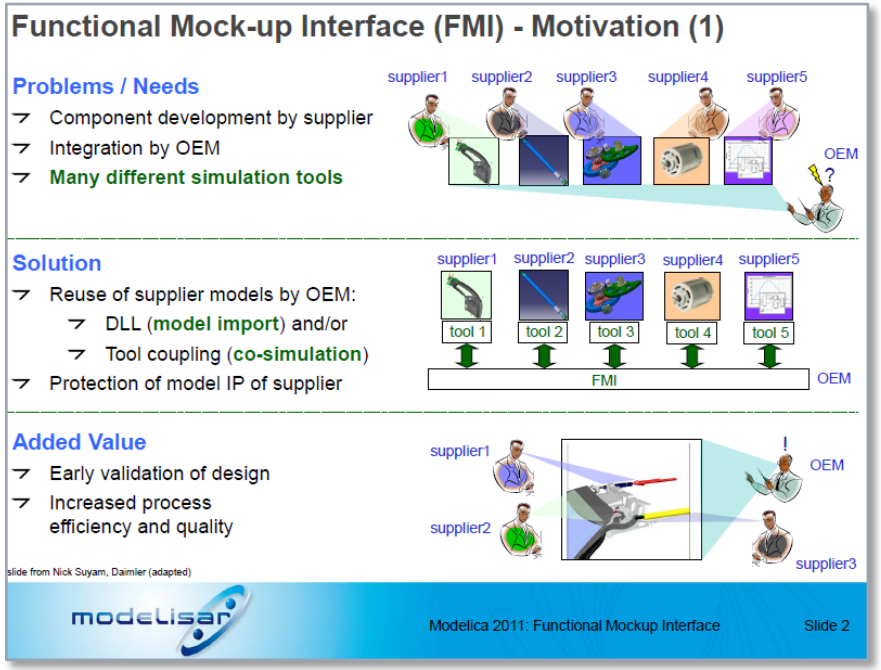
- Define **a tool-independent**, **free-to-use** standard for exchange and co-simulation of models between different simulation tools
- **Provide models in a containerized form that** allow for the **deployment to different targets**
- **Decouple Know-How**
  - between producers and users of FMUs
  - between different specialized engineers and software programmers
- Massive **Re-use** of modelling investment
- **IP Protection** possible vial black-box model exchange

Timeline : Modelisar Project → Modelica Association Project FMI

- FMI 1.0 (and most part of FMI 2.0) was developed in the publicly funded project Modelisar

- 2008-2011: MODELISAR project
- 2011: Release of FMI 1.0
- 2012: Foundation of MAP FMI in MA

  Members: see FMI Webpage
- 2013: Release of FMI 2.0 → *focus of this tutorial*
- 2021: Release of FMI 3.0



Versioning of FMI

FMI uses semantic versioning:

Major releases

- not backwards or forward compatible changes

e.g. **„FMI 2.0.4“**

Minor releases

- Can introduce new features
- Backwards compatible
  „FMI 3.0 FMUs will be valid FMI 3.1 FMUs“

Bugfix/Maintenance releases

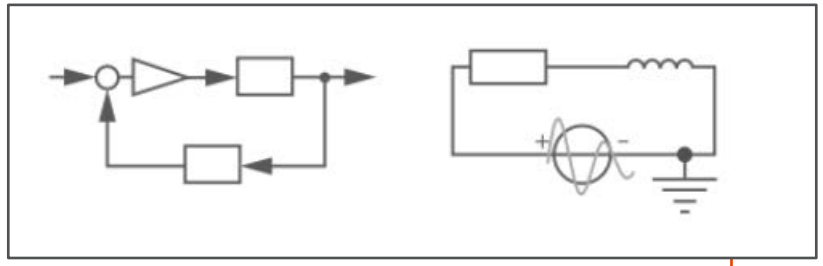
- No new features, only bugfixes and clarifications possible
- Backwards and forward compatibility

FMI: Technical Fundamentals

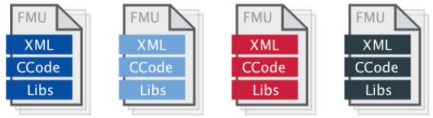
“How does FMI work?”

Exporting vs. Importing tool

Exporting tools



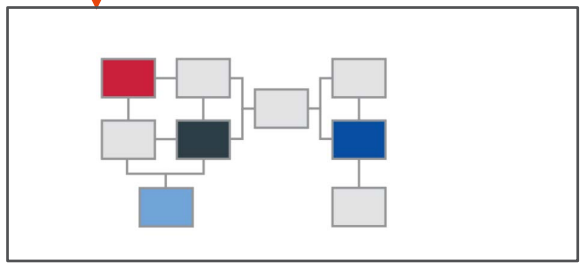
1. develop model
2. export as **Functional Mockup Unit (FMU)** accordingg to the **FMI Standard**



FMU

3. import FMU

4. (co-) **simulate model**

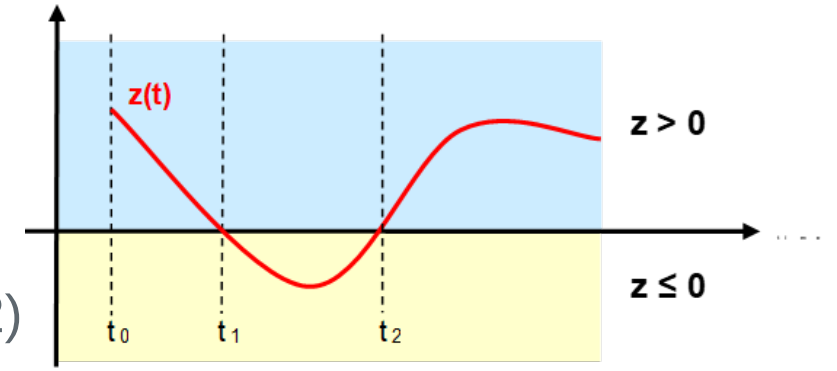


Importing tool

Models in Scope

Original Scope of FMI:

- Ordinary differential equations (**ODEs**) with **events**
  - You need a **numerical solver** for their solution
- Distinction between continuous and discrete variables
- There is a notion of **time** (or more general „independent variabl2)



Extended scope:

- Purely algebraic equations
- Complex discrete behavior with clocks and model partitions (FMI 3.0)
- A very broad scope of use cases such as
  - virtual electronic control units (ECUs),
  - AI models
  - ...

Co-Simulation (CS)

Co-Simulation: (from the FMI 3.0.1 glossary)

- **Coupling of several *simulation programs*** in order to compute the global behavior of a system that consists of several subsystems.
- Subsystems are coupled in the sense that the **behavior of each subsystem depends on the behavior of the remaining subsystems**, so that the co-simulation must be **computed in a step-by-step fashion**.
- Each simulation program is responsible for computing the behavior of a subsystem, using the outputs produced by the other simulation programs.

- There can be an **additional error** introduced by the co-simulation, that has to be limited to an acceptable size by using a **suitable co-simulation algorithms and communication pattern** (e.g. step size)

The FMI Standard

The FMI Standard defines the “Functional Mock-up Interface” FMI, an interface between a model and a simulation environment.

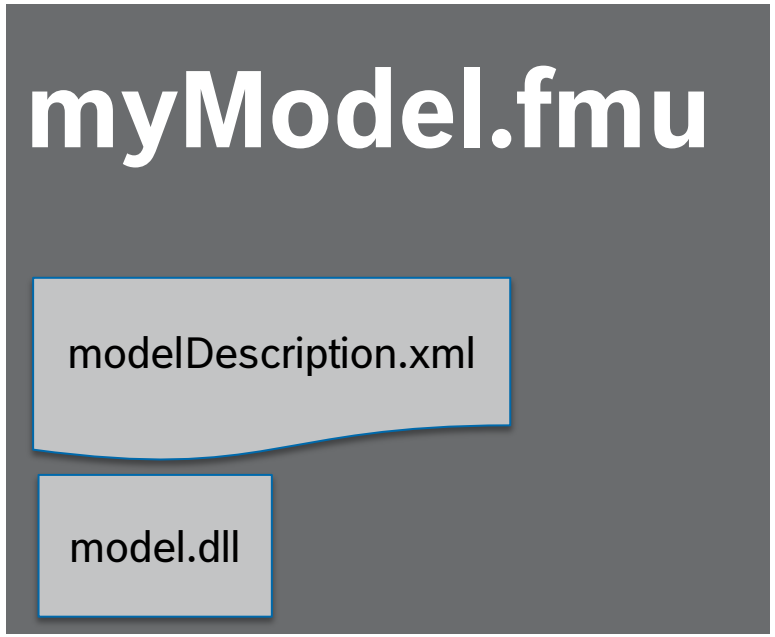
It consists of the definition of

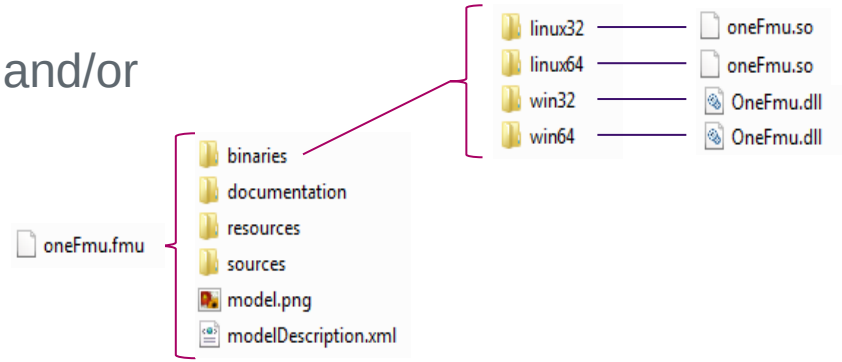
- a **C-API** (application programming interface)
- an **interface description** (modelDescription.xml) according to a defined schema
- the **definition of an exchangeable unit**, an **“FMU”** (Functional Mock-up Unit), technically a zip file

The FMI Standard only defines the interface of a single FMU,
not the co-simulation algorithm or solver for multiple FMUs!

FMU – „Functional Mock-up Unit“

- A model container, that can be distributed
- Technically it is realized a zip File, with ending „.fmu“
- Content:
  - **modelDescripotion.xml:**
    meta-data with information on the model variables, interface, capabilities and to a limited extent model structure
  - **Model representation**
    - **Binaries f**or one or multiple platforms and/or (“black box”), and/or
    - **Source code** (e.g C-Code)
  - Optionally: Resources, documentation, Icons, port definitions
  - /extra information (defined in layered standards)

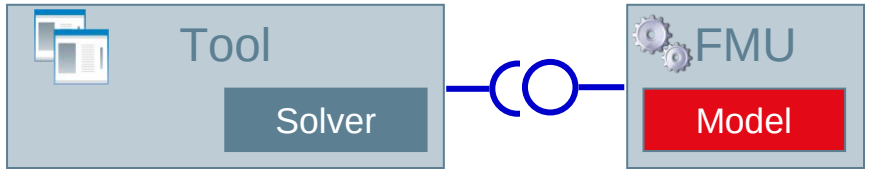




FMU for Model Exchange vs. Co-Simulation (FMI 2.0)

Model Exchange (ME)

- Importing tool provides the solver.



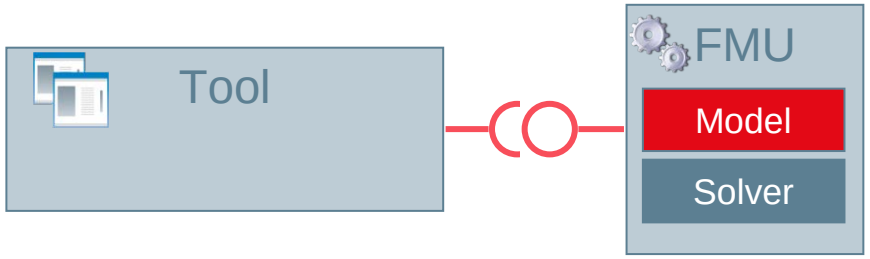
Properties:

- Very tight integration of model in simulation tool
- Complex interface between importing tool and model
- Importing tool must provide a suitable solver for the model

- Used e.g. for inclusion of Modelica support in non-Modelica tools (using the solver of the importing tool)

Co-Simulation (CS)

- Exporting tool provides the solver.



Properties:

- Tight coupling of model and a suitable solver
- Simpler interface between importing tool and model
- Freedom in the selection of a co-simulation algorithm and communication timestep to reach a stable and accurate solution

- Used in most industrial applications

Co-Simulation (CS) Interface

- Communication timestep can be different from internal steps (e.g. Variable step solver)
- Calling sequence CS
  - Set inputs: **`fmi2SetXXX(...)`**
  - Trigger calculation until next communication point: **`fmi2doStep(...)`**
  - Get outputs: **`fmi2GetXXX(...)`**

For an implementation in C, see e.g. fmusim in the Reference FMUs:
https://github.com/modelica/Reference-FMUs/blob/main/fmusim/fmusim_fmi2_cs.c

You can view the calling sequence `with fmusim --log-fmi-calls`

- The co-simulation algorithm is not part of the FMI standard. It is responsible for:
  - advancing the overall simulation time,
  - exchange input and output data,
  - triggering of input clocks, and handling events.
- Internally the FMU can have different timesteps (e.g. a variable step solver)

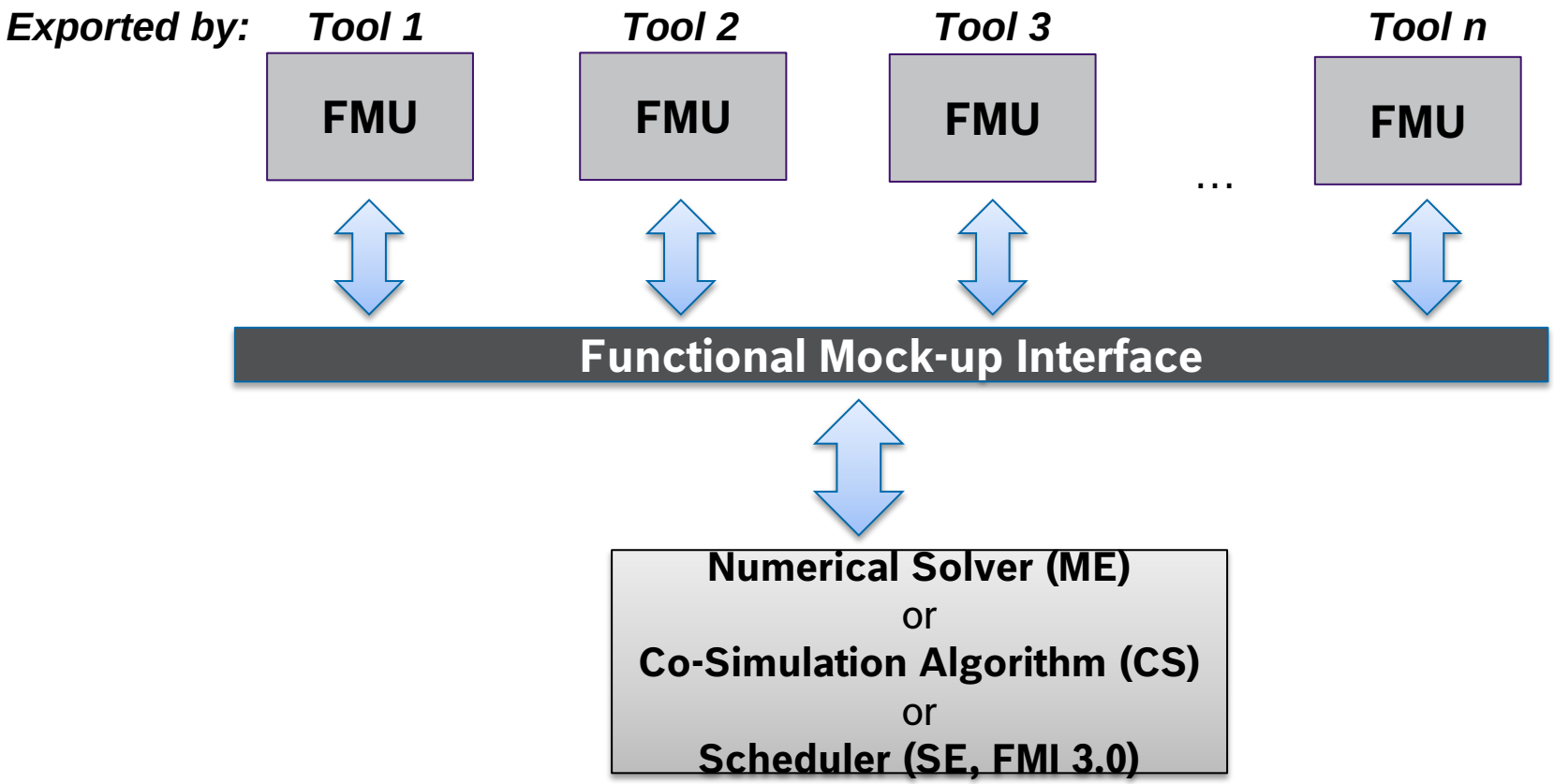
Model Exchange (ME) Interface

Main idea: the ME interface exposes the **right hand side** (RHS)
of a **hybrid ODE** to the external solver.

- The **importer controls the data exchange** and the synchronization between FMUs.

- **The solver algorithm itself is not part of the FMI standard**. It is responsible for:
  - advancing the overall simulation time,
  - exchange input and output data,
  - computation of continuous state variables by time integration,
  - triggering of input clocks, and
  - handling events.

Simulation of multiple FMUs



Not defined by FMI Standard, but by the importing tool

The modelDescription.xml

It contains all static information about the FMU

- Definition of **supported interface kinds** (ME, CS, SE)
- **Model variables**
  - Inputs, outputs, parameters
- Certain information on the **model structure**

```
<?xml version="1.0" encoding="UTF-8"?>
<fmiModelDescription
  fmiVersion="2.0"
  modelName="BouncingBall"
  description="This model calculates the trajectory, over time,
  generationTool="Reference FMUs (development build)"
  guid="{1AE5E10D-9521-4DE3-80B9-D0EAAA7D5AF1}"
  numberOfEventIndicators="1">

  <ModelExchange
    modelIdentifier="BouncingBall"
    canNotUseMemoryManagementFunctions="true"
    canGetAndSetFMUstate="true"
    canSerializeFMUstate="true">
    <SourceFiles>
      <File name="all.c"/>
    </SourceFiles>
  </ModelExchange>
```

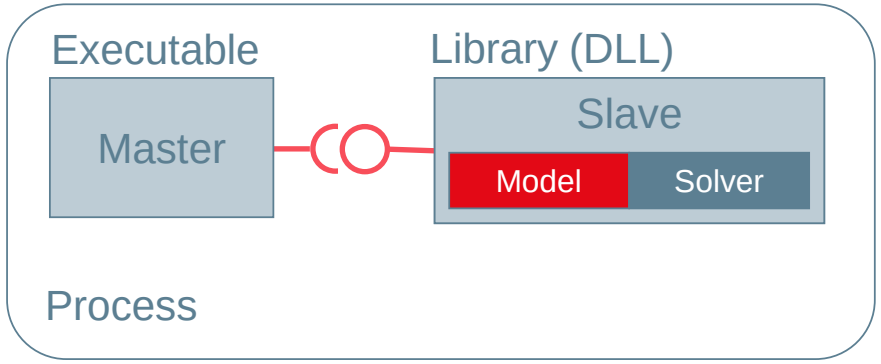
- Example: https://github.com/modelica/Reference-FMUs/blob/main/BouncingBall/FMI2.xml

- Attributes and capability flags, such as
  - “needsExecutionTool”
  - …

FMI for Co-simulation Tool Wrapper Variant

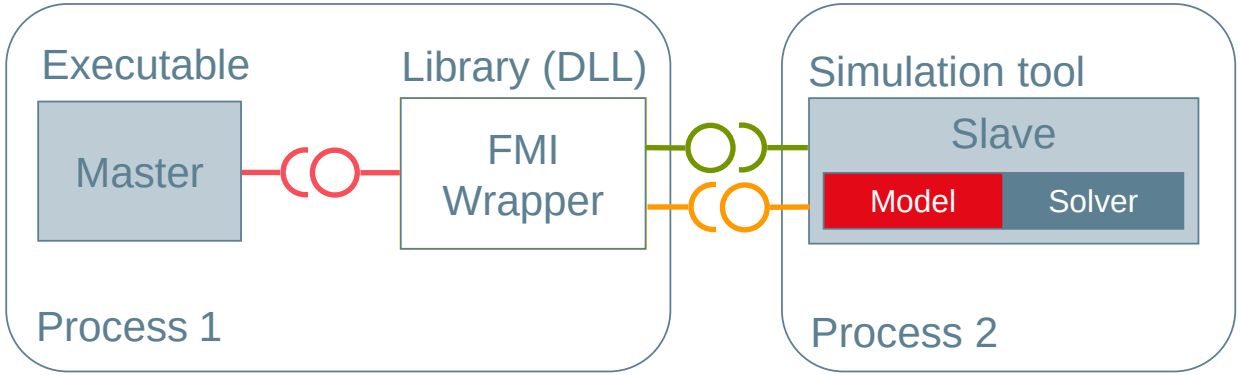
Co-simulation stand-alone FMU

- self-contained FMU



Co-simulation FMI Tool Wrapper

- FMU dependent on an existing tool installation



Attribute: “needsExecutionTool”

FMI Tool Support

https://fmi-standard.org/tools/: FMI supported by >= 180 Tools

- Exporting tools
  - Modeling and simulation tools
  - Model-based software creation
  - Wrapping of algorithms, virtual ECUs
  - …
- Importing tools
  - Modeling and simulation tools
  - Co-simulation tools and integration platforms
  - Implementations for programming and scripting languages
  - ….

Compatibility Information

- Replaces the cross-check that helped to improve he maturity of FMI 1.0 and 2.0 supporting tools.

- Information on on how FMI export and import have been tested provided by the tool vendors and linked in the tools list

- Tools providing compatibility information are marked with a golden star and listed on top of the tools list https://fmi-standard.org/tools/:

Licenses and licensing mechanism

- The **FMI standard deals with the technical part** of model-exchange and co-simulation only.

- There might be other **legal or technical restrictions**
  - Exporting tools might use a **licensing mechanism** such as FlexNet
  - **Other legal obligations**, e.g., w.r.t. to the distribution of FMUs might be imposed by the exporting tools.

- **License information** can be stored in the documentation folder

New features in FMI 3.0

Motivation for FMI 3.0

180+ tools support FMI now: many users now, many new use-case requests:

- Virtual Electronic Control Units (vECUs):
  - FMI 2.0 works well for physics simulations: better support for vECUs is needed
- Advanced Co-Simulation
  - Co-Simulation is the more popular interface type: improved co-simulation methods are needed to improve performance and accuracy
- Multi-FMU simulations are getting more common
  - Events are necessary in complex control systems
  - Events must be synchronized across FMUs
- New ML and AI applications
  - More derivatives computations is required

vECU

Co-Sim

Events

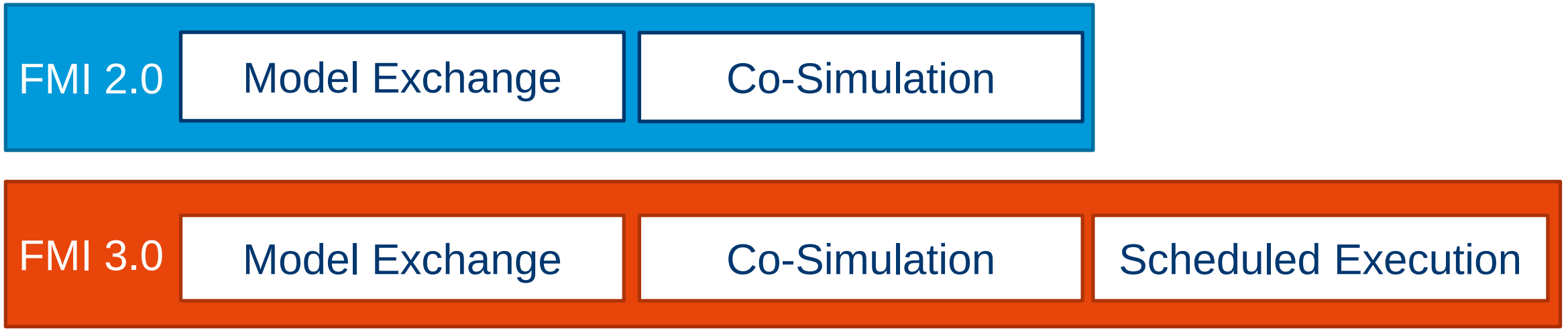
AI

API Efficiencies

FMI 3.0: New Interface Type – Scheduled Execution

vECU

Events



Scheduled Execution allows coupling several FMUs with one, external scheduler (OS)

FMI 3.0: Main Improvements

- Event mode for Co-Simulation
- Intermediate variable update
- Clocks
- New variable types
- Array variables
- Terminals and icon
- FMI for Scheduled Execution (SE)
- Preparation for layered Standards

Performance

Accuracy

New Application

Resources

- FMI Webpage
  - FMI tools list
  - FMU validation
  - Publications

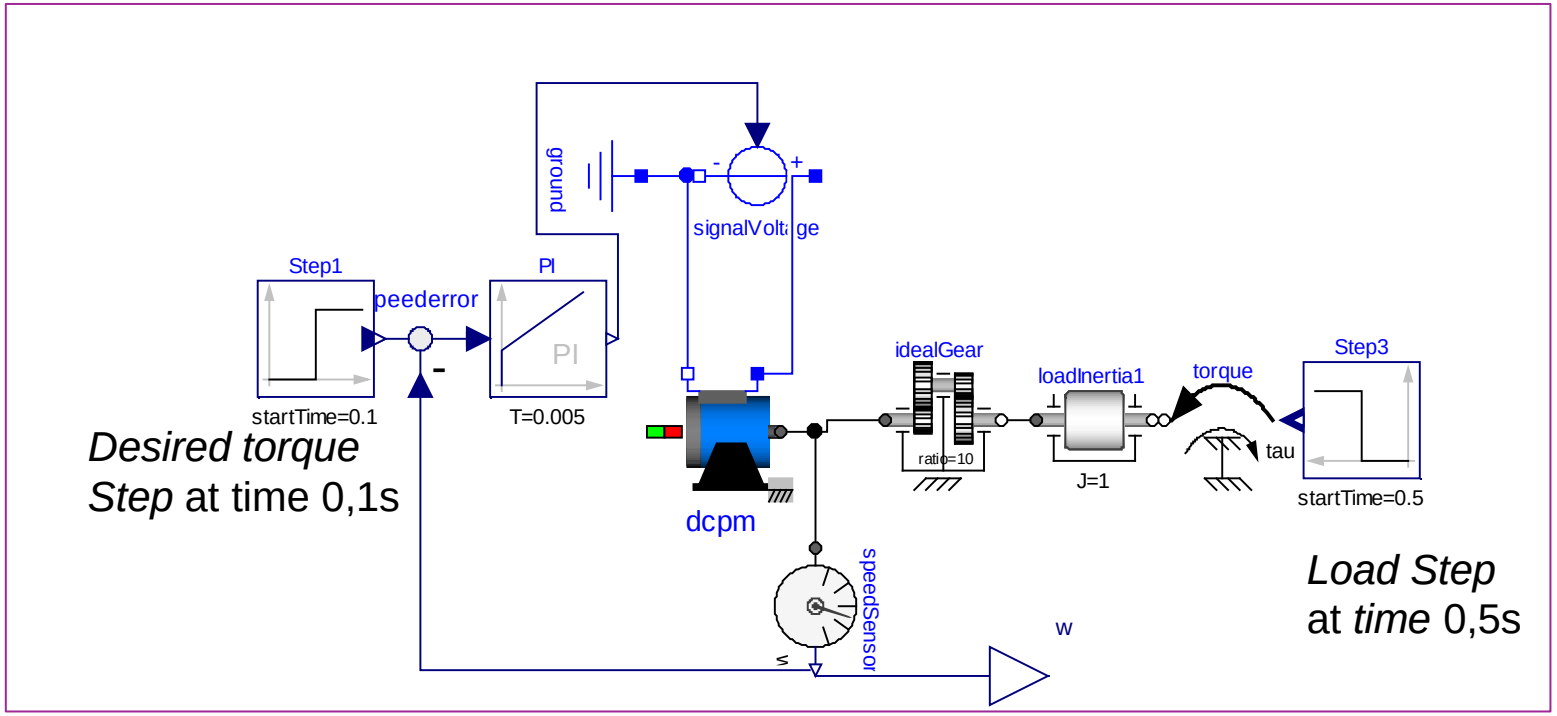
- Reference FMUs
  - A set of hand-coded FMUs for development, testing and debugging of FMI.

- In case of questions we recommend to use StackOverflow with tag „fmi“
- You are welcome to join the (unofficial) FMI LinkedIn Group

Example model for hands-on part

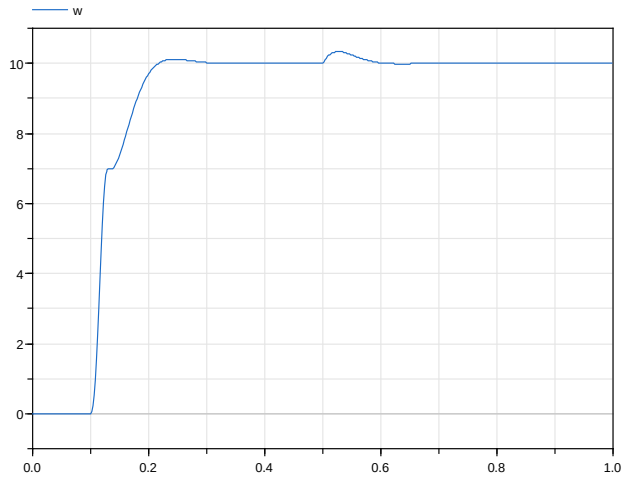
Example model: Controlled Motor Drive

Motor Drive with PI control modelled in Modelica



Desired torque
Step at time 0,1s

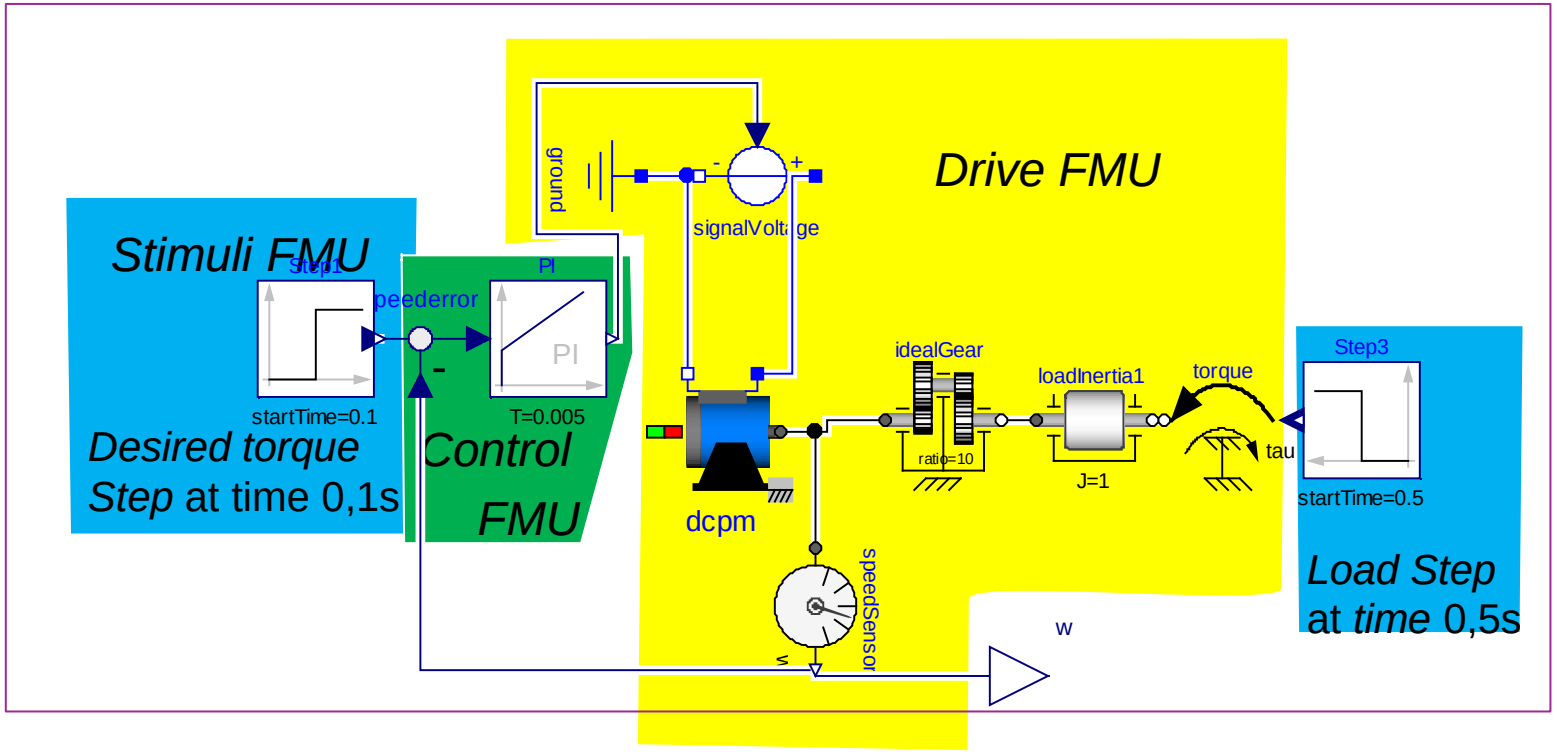
Load Step
at *time* 0,5s



Reference solution
Angular speed

Example model: Controlled Motor Drive

Model exported as one overall FMU or split as three FMUs.



Overall FMU