Fontys UNIVERSITY OF APPLIED SCIENCES

> FOR SOCIETY

CO-SIMULATION

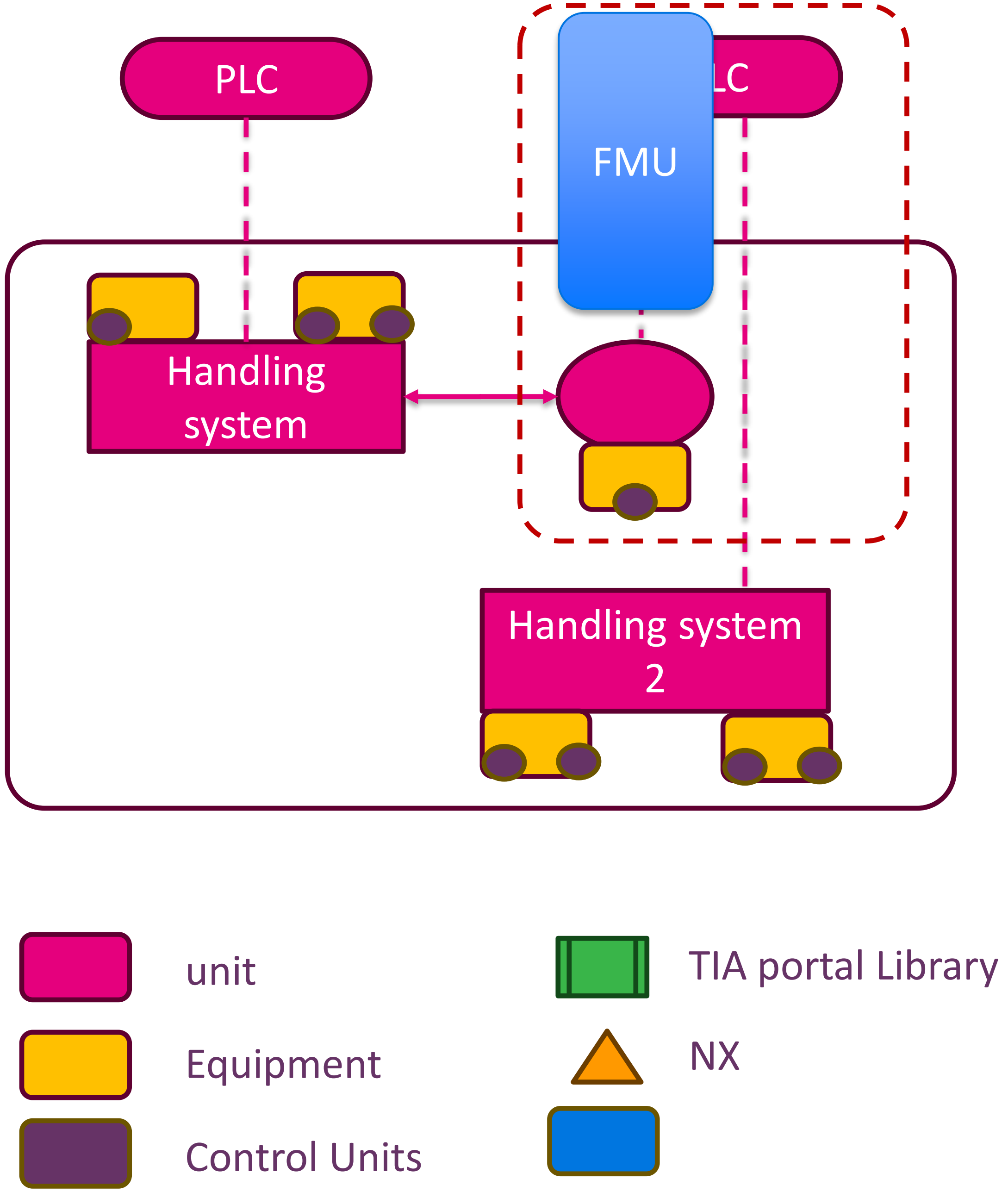
Introduction

- System
- Multiple Units (ISA88)

- Robot Control
  - Tia Portal Library
  - NX

- What if behaviour is programmed in another environment?

How can and flexible architecture for dynamic simulation be created with FMU?



Fontys UNIVERSITY OF APPLIED SCIENCES

Introduction

What is FMI? FMI stands for **Functional Mock-up Interface**. It is a **standard** for exchanging dynamic models between different simulation tools.

What is an FMU? An FMU is the **zip file container** that holds a model compliant with the FMI standard (code, XML, resources). It's a "black box" model you can share.

•*Key Takeaway:* FMI/FMU allows **tool independence** and **model reuse**.

Fontys
UNIVERSITY OF
APPLIED SCIENCES

Co Simulation

Functional Mock-up Interface (FMI)

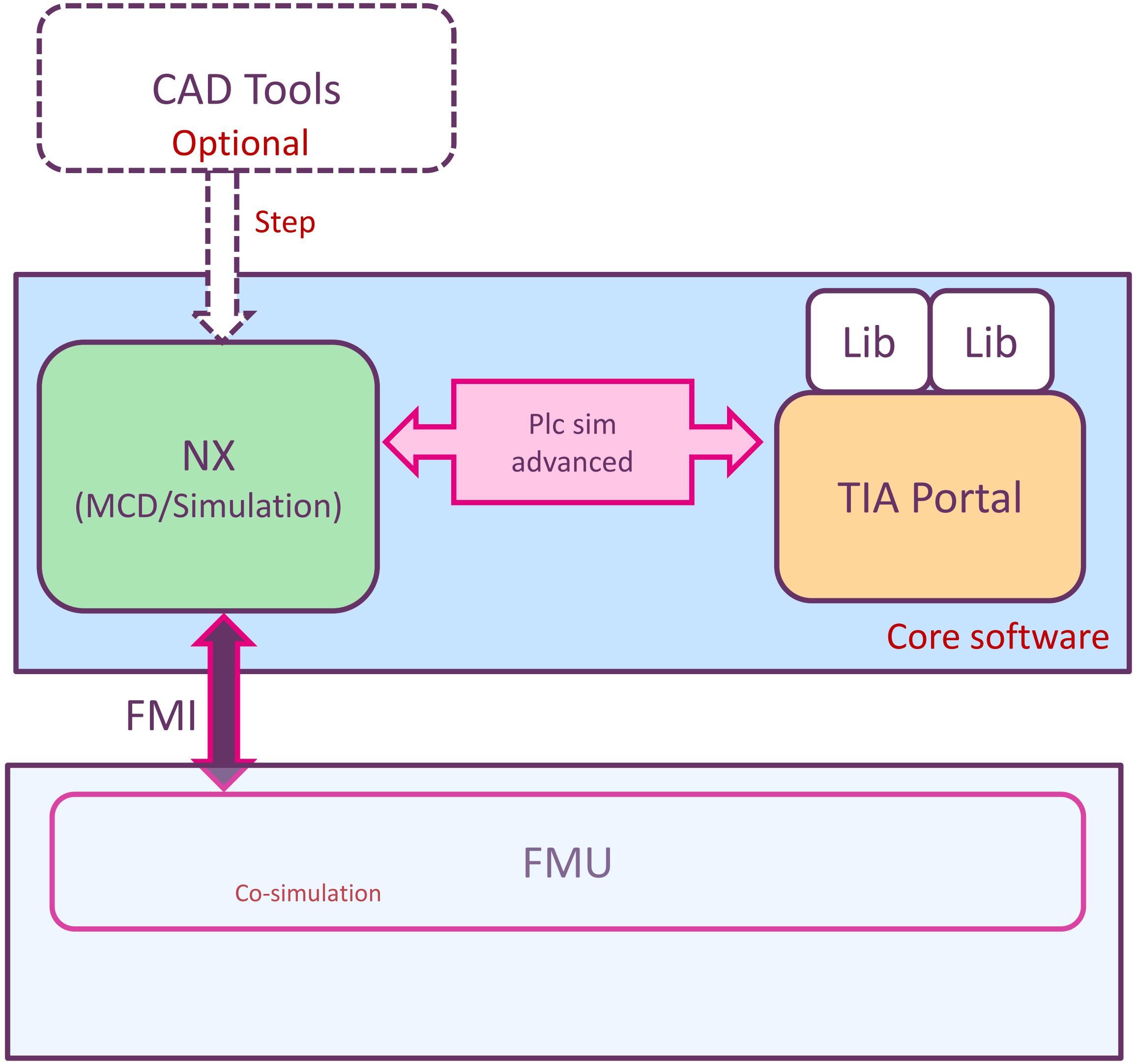
FMI is a standardized interface enabling model exchange and co-simulation across diverse simulation tools.

Functional Mock-up Unit (FMU)

FMU encapsulates models, metadata, and binaries into a single file for easy sharing and reuse.

Model Exchange and Co-Simulation

FMI supports both external solver model exchange and FMU-included solver co-simulation for flexibility.



Why FMU?

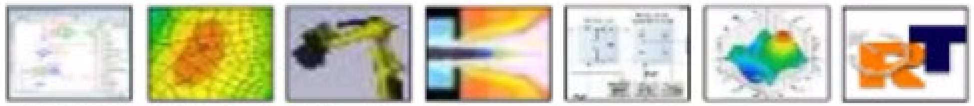
- Industrial automated systems are **complex** (mechanical, electrical, control software).
- Need for **efficient integration** and **early-stage validation** across different engineering domains (e.g., Mechanical CAD, Electrical Systems, Control Code).
- Traditional simulation is often fragmented.
- Open Source standard for exchange dynamic simulation models

Fontys
UNIVERSITY OF
APPLIED SCIENCES

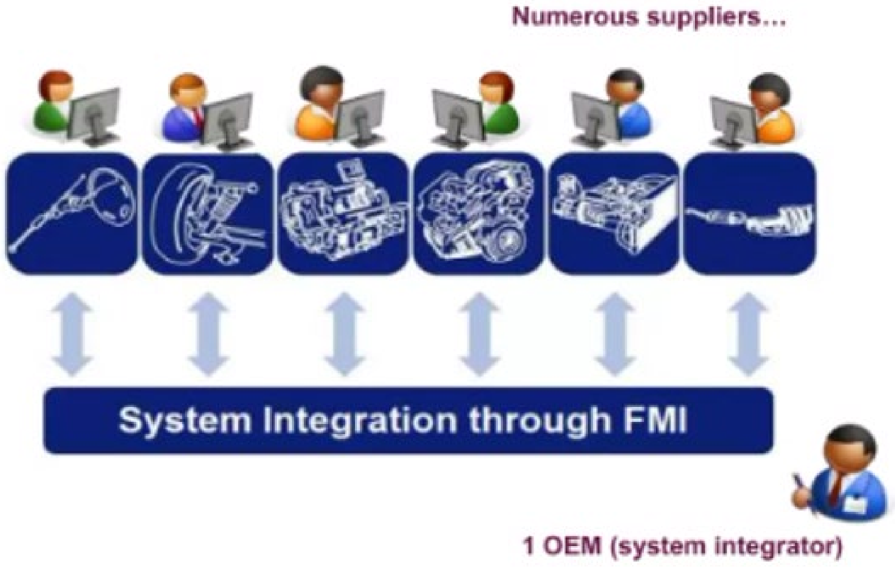
FMU Use Cases and Motivations

Use cases to address:

- Collbarions MBSE
- Between OEM’s and supplies
- Between departments of the same company
- Including different domains and complex levels



- \- 1D, 3D FEA/MBS, CFD, Controls, RSM, Real-Time…



fmi FUNCTIONAL MOCK-UP INTERFACE

- Tool neutral software
- Open format, with public available specifications: https:///www.fmi-standard.org.downloads

Fontys UNIVERSITY OF APPLIED SCIENCES

FONTYS UNIVERSITY OF APPLIED SCIENCES > FOR SOCIETY

FMU/FMI

The functional mock-up interface

The functional Mock-up Interface is a free standard that defines a container and an interface

- To exchange dynamic simulation models
- Using a combination of XML, binaries and C code, distributed as a ZIP file
- Current release FMI 3.0
- 200+ tools and libraries support FMI

Introduction

The *Functional Mock-Up Interface* (**FMI**) is a **tool-independent standard for making submodels binary compatible** with each other.

- It is completely open and free to use and is supported by a large and growing number of tools, for example by Dymola, JModelica.org, SIMPACK, SimulationX, and Simulink.
- Note that FMI only specifies how the (co-)simulation software interacts with the models;
- it is **not in itself a simulation software**, nor does it specify or restrict any other parts of the architecture of such a software.
- FMI does not specify how the sub-simulators are time synchronized, nor in what format, data are transported between them.

Fontys UNIVERSITY OF APPLIED SCIENCES

Co-simulation - Open Simulation Platform

Background info

- A model which implements FMI is called a ***Functional Mock-Up Unit*** (**FMU**).
  - In effect, an FMU is an *archive file (ZIP format*) consisting of model code for one or more platforms (C or binary),
  - a description of the interface data (XML format), and optional documentation and metadata.
  - The FMI standard specifies the **APIs** that must be implemented by the model code.
  - Note that an FMU can also represent interfaces to hardware such as sensors, actuators, or devices for human input.

Fontys UNIVERSITY OF APPLIED SCIENCES

Functional Mock-up Interface Specification

ModelDescription.XML

```
modelDescription.xml                 // description of FMU (required file)
documentation                        // directory containing the documentation (optional)
   index.html                        // entry point of the documentation
   diagram.png                       // descriptive diagram view of the model (optional)
   diagram.svg                       // if provided, diagram.png is also required (optional)
   externalDependencies.{txt|html}   // documentation of external resources required to load
                                     // or simulate the FMU
   <other documentation files>
   licenses                          // directory for licenses (optional)
      license.{txt|html|spdx}        // entry point for license information
      <license files>                // for example BSD licenses (optional)
   staticLinking.{txt|html}          // entry point for static link information (optional)
terminalsAndIcons                    // FMU and terminal icons (optional)
   terminalsAndIcons.xml             // description of terminals and icons (optional)
   icon.png                          // image file of icon without terminals (optional)
   icon.svg                          // if existing the icon.png is required (optional)
                                     // all terminal and fmu icons referenced in the
                                     // graphical representation
sources                              // directory containing the C sources (optional)
   buildDescription.xml
binaries                             // directory containing the binaries (optional)
   x86_64-windows                    // binaries for Windows on Intel 64-bit
      <modelIdentifier>.dll          // shared library of the FMI implementation
      <other files>                  // other platform dependend files
                                     // needed at linking or loading time
   x86-linux                         // binaries for Linux on Intel 32-bit
      <modelIdentifier>.so           // shared library of the FMI implementation
   aarch32-linux                     // binaries for Linux on ARM 32-bit
      <modelIdentifier>.so           // shared library of the FMI implementation
   x86_64-darwin                     // binaries for macOS
      <modelIdentifier>.dylib        // shared library of the FMI implementation
resources                            // resources used by the FMU (optional)
extra                                // Additional (meta-)data of the FMU (optional)
```

Fontys
UNIVERSITY OF
APPLIED SCIENCES

Key features of FMI 3.0

- Advanced Co-Simulation: Enables high-quality, robust co-simulation of complex models, making it suitable for more demanding applications.
- Virtual Electronic Control Units (vECUs): Supports the development and testing of embedded software by converting FMUs into full-fledged vECUs.
- Layered Standards: Allows artifacts from other standards to be included in the FMI container, which improves interoperability.
- Next Generation Digital Twins: Enhances support for system-level digital twins, which can run in the cloud or at the edge.
- Artificial Intelligence Applications: Facilitates the use of machine learning and other AI techniques for calibrating model parameters.

Fontys
UNIVERSITY OF
APPLIED SCIENCES

Interface type FMI

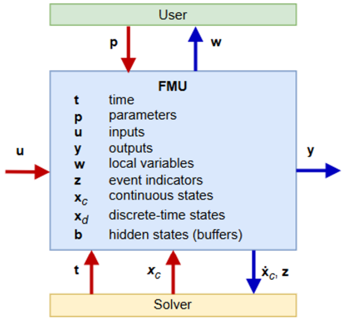
FMI 3.0 defines three interface types:

Co-Simulation (CS): Enables the integration of simulation units that can exchange data at runtime.

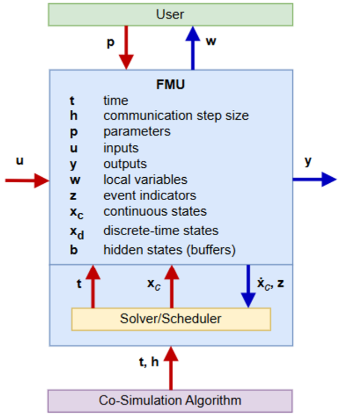
Model Exchange (ME): Supports the exchange of models that can be integrated into different simulation environments.

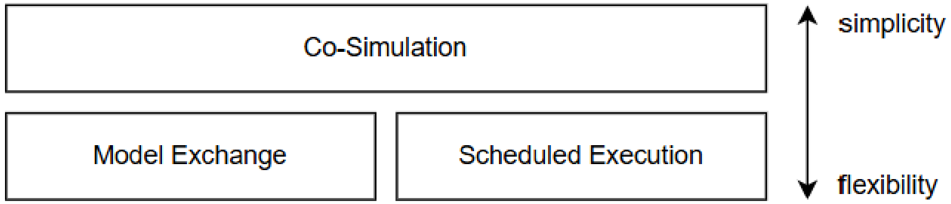
Scheduled Execution (SE): Facilitates real-time simulations by enabling precise control over the computational time of submodels.

FMI For model Exchange (ME)



FMI for Co-simulation





FMI co-simulation

- *FMI for co-simulation* is based on a master/slave model of communication and control, where sub-simulators are *slaves* that are controlled by a *master algorithm* (the co-simulation algorithm).
- The sub-simulators do not have any information about each other, nor about the simulation environment, except for the values they receive for their input variables. Thus, they have no knowledge about or control over which other sub-simulators they are coupled to; the data are routed by the master algorithm.

Features interface type

Table 1. Non-normative overview of features per interface type.

Feature	Model Exchange	Co-Simulation	Scheduled Execution
Advancing Time	Call `fmi3SetTime`	Call `fmi3DoStep` and monitor argument `lastSuccessfulTime`	Call `fmi3ActivateModelPartition`
Solver Included	✖	Possibly	Possibly
Scheduler included	Possibly	Possibly	✖
Event Indicators	✔	✖	✖
Early Return	Includes similar or better mechanism	✔	✖
Intermediate Update or Clock Update	Includes similar or better mechanism	✔	Signal output Clock ticks: ✔ Inputs/Outputs: ✖
Clocks	✔	✔	✔
Direct Feedthrough	✔	In **Event Mode**: ✔ Else: ✖	✖

Fontys UNIVERSITY OF APPLIED SCIENCES

[Functional Mock-up Interface Specification]

FONTYS UNIVERSITY OF APPLIED SCIENCES

› FOR SOCIETY

THE POTENTIONAL OF FMI FOR THE DEVELOPMENT OF DIGITAL TWINS FOR LARGE MODULAR MULTI-DOMAIN SYSTEMS

Digital Twins

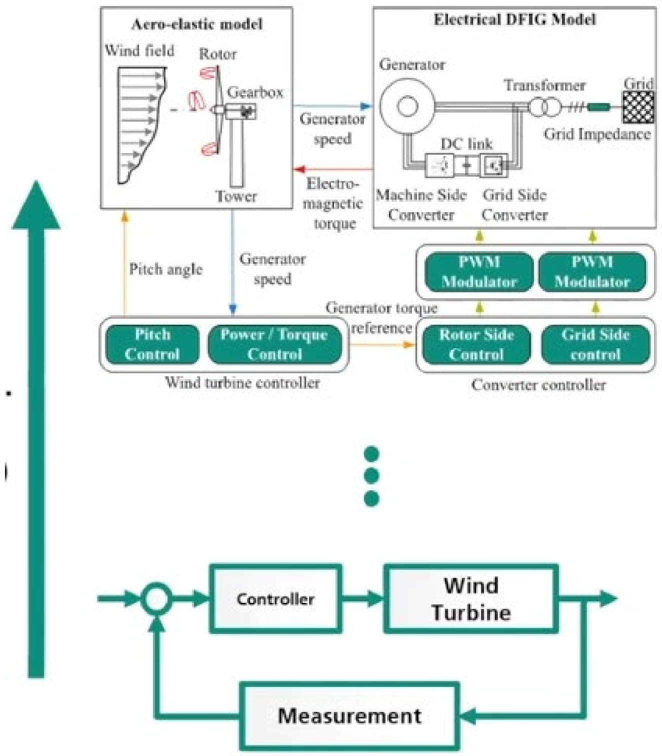
- **Digital Model**
  - Virtual representation of physical system
  - No interaction between system and model
- **Digital Shadow**
  - Model follows physical system
- **Digital Twin,**
  - Virtual representation of a physical object or process
  - Bidirectional exchange of data between physical and virtual system
  - User for process optimization, observation, prediction

Fontys
UNIVERSITY OF
APPLIED SCIENCES

Development of a Digital Twin

- Digital Twin requires simulation
- Representation of a complex system
- Specific simulation of singular problems
  - Generator models (lineair, saturation effects, ..), grid, etc
  - Wind turbine model (rigid, flexible, inflow conditions, …)
  - Controller (pitch, torque, current flow, …), Electrolyser
- Many different models (modularly exchangeable)
- Interfaces are often similar.

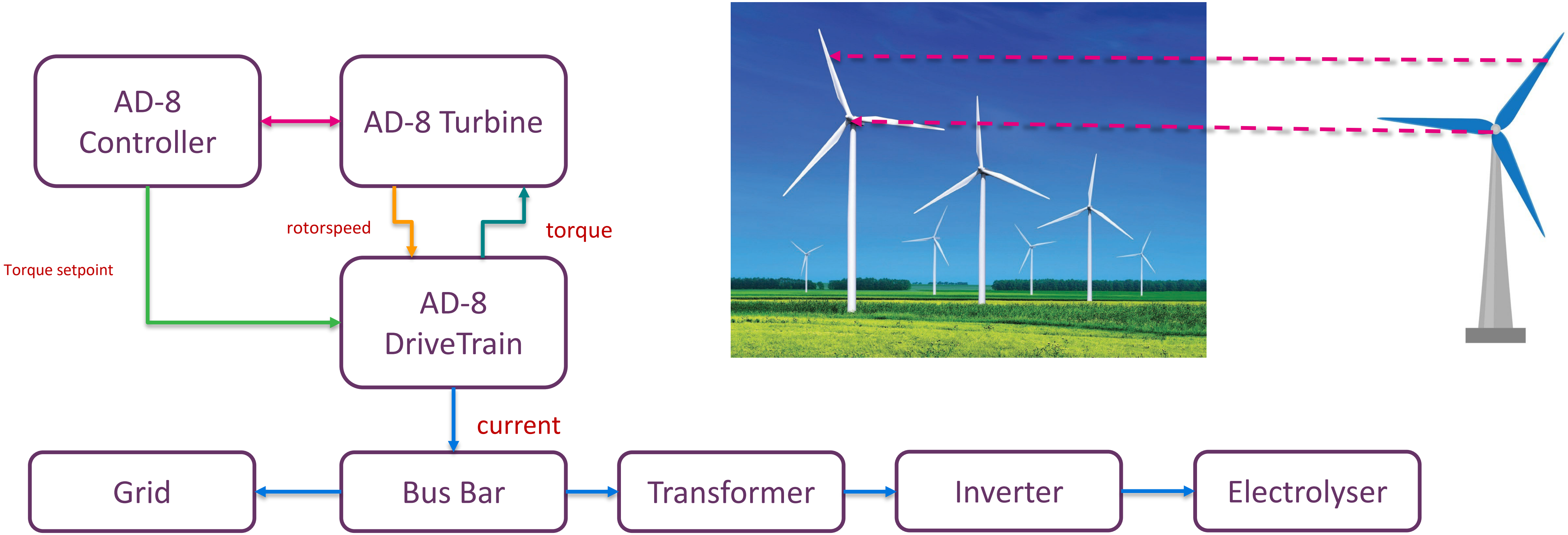
Example: Wind Energy Field



Fontys UNIVERSITY OF APPLIED SCIENCES

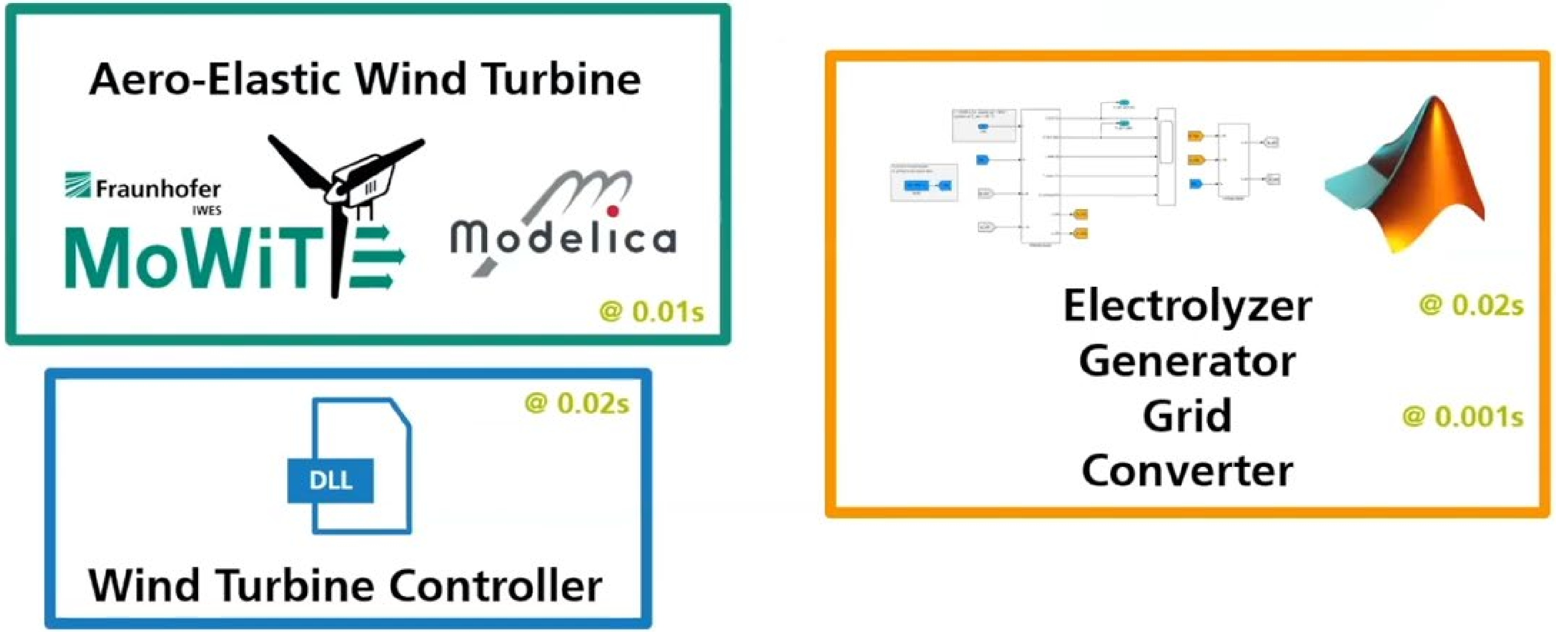
Fraunhofer IWES

Modular Multi-Domain System



How to combine many different models efficiently?

Simulation Model from varios Sources

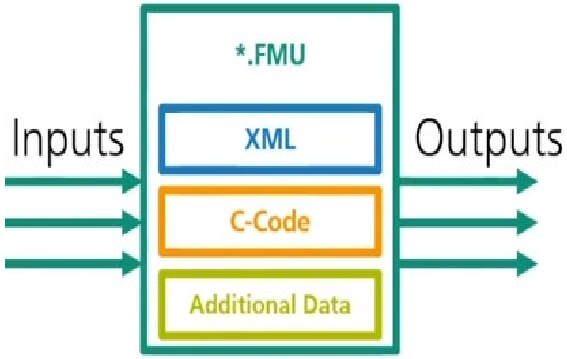


Fontys
UNIVERSITY OF
APPLIED SCIENCES

Co Simulation

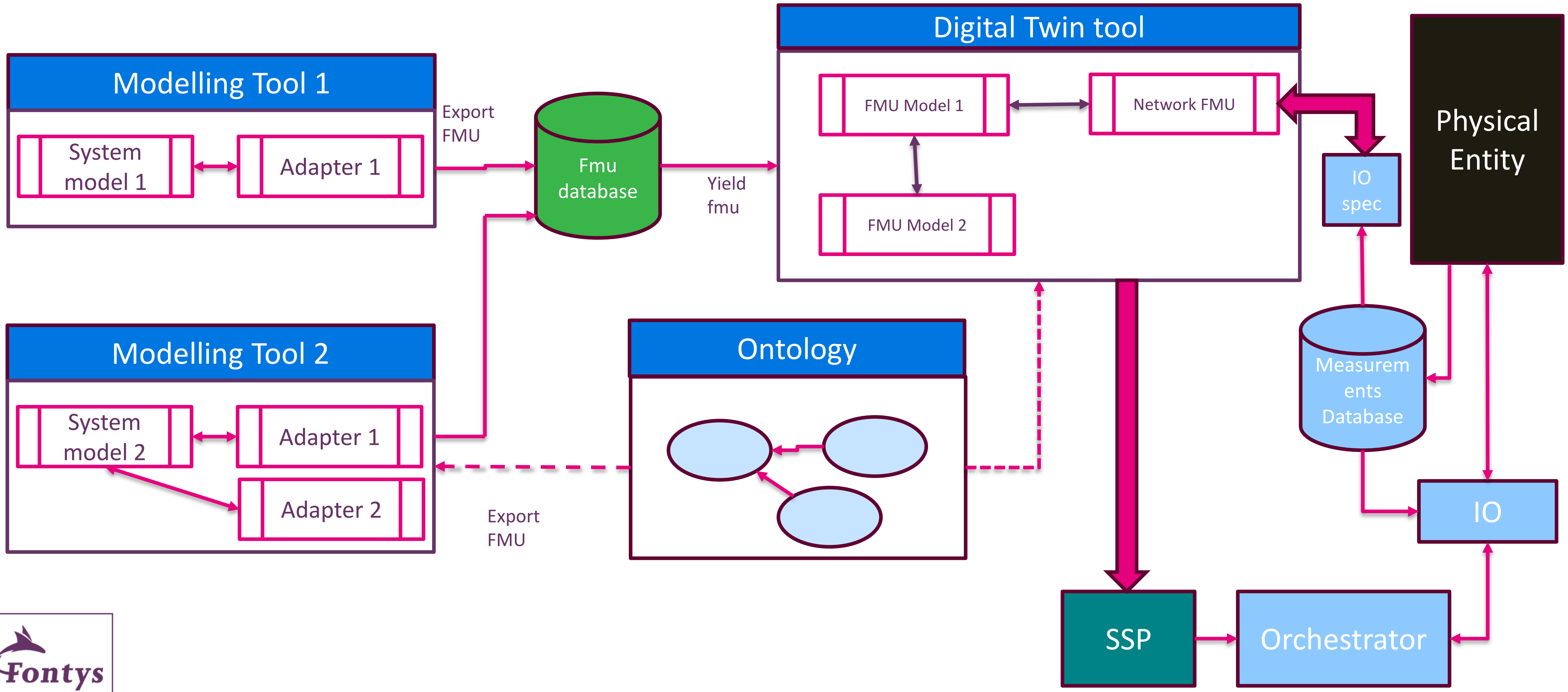
- Co-Simulation with FMU
- Problems:
  - Interconnection of complex models requires multi-dimensions
  - No addiational information e.g. unit
- Standardized interfaces
- How to include that already in modelling?

fmi Functional Mock-Up Interface



Fontys UNIVERSITY OF APPLIED SCIENCES

Building a Digital Twin Tool



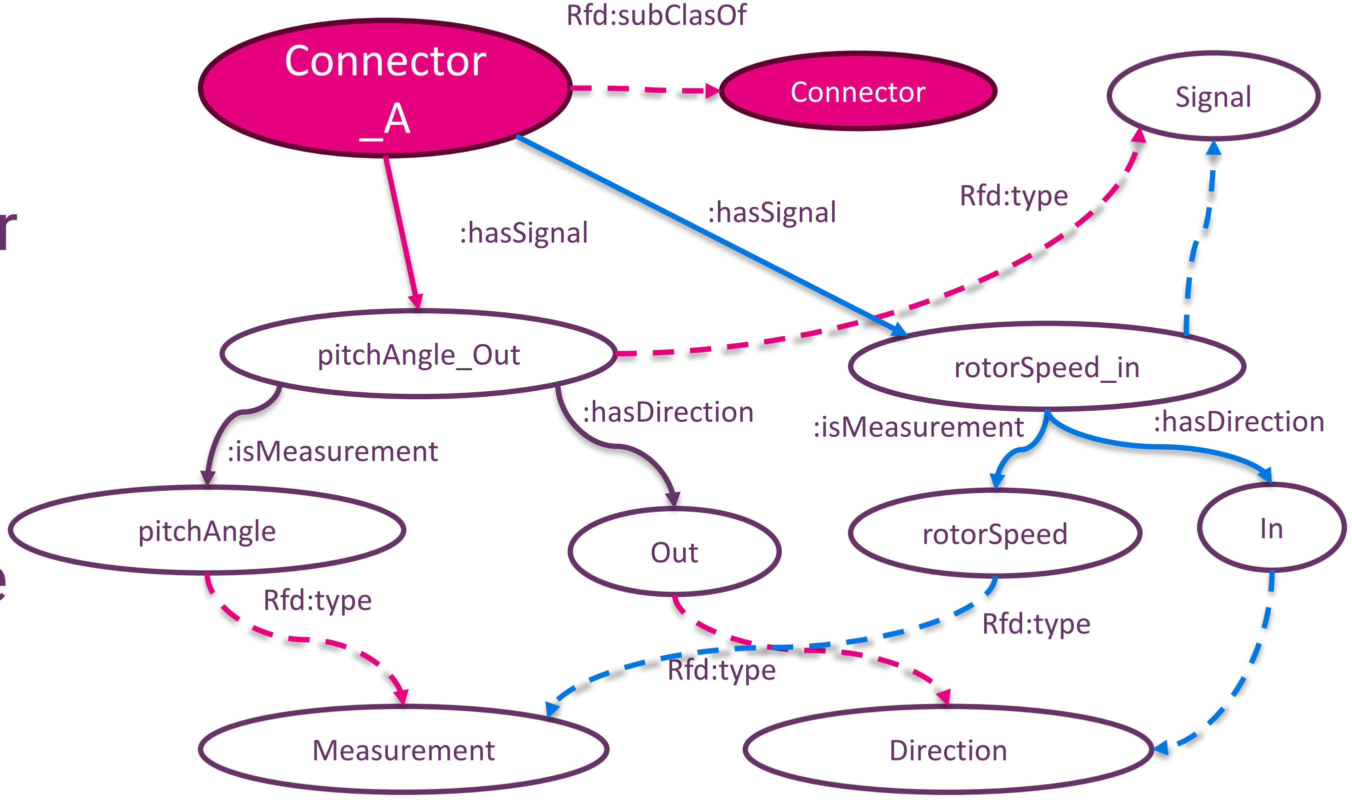
Building a Digital Twin Tool

- Define Modeling Tool
  - Create FMU Database
- Ontology (the heartbeat of the system)
  - Define adapters
  - Define how the simulation is setup
- Digital Twin Tool for
  - Derives the interaction between the FMU from the ontology
- Export SSP (System Structure and Parameterization)

Fontys
UNIVERSITY OF
APPLIED SCIENCES

Connect Ontology (Single Entity)

- Enables automated connection between models
  - Digital Twin Modelling Tool is improved
- Defines interfaces for models for the FMU export → helps the modelling engineer
- Connectors are <u>composed</u> of Signals
- Compatible Connectors have the same measurement but opposite directions



Fontys
UNIVERSITY OF APPLIED SCIENCES

More details

- More information about structure and setup:

[Functional Mock-up Interface Specification]

FOR SOCIETY

EXERCISES AND GOALS

FMU tutorial

- Since FMU’s are open-source, vendors have implemented methods of integrated FMU’s in their software programs

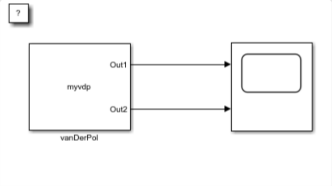
- Various tools of how FMU can be used
  - Matlab Simulink
  - Python
  - Openmodellica

- Explore tools to investigate the functionality of FMU
  - Matlab * (advanced tool for import and export FMU)
  - Python * (script based FMU / dedicated libraries)
  - Openmodellica * (lots of examples)

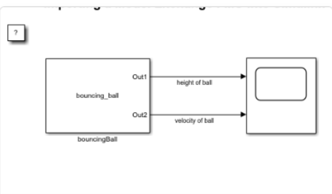
Fontys
UNIVERSITY OF
APPLIED SCIENCES

Matlab

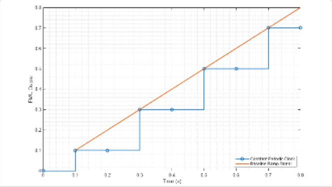
FMU Importing — Examples



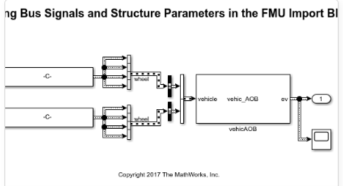
Import Co-Simulation FMU into Simulink
Use the FMU Import block to load an FMU file. The FMU file supports execution in co-simulation mode. Simulink® software supports...



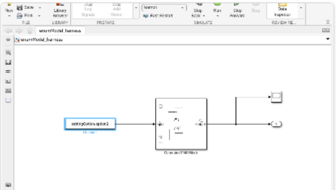
Importing a Model Exchange FMU into Simulink
Model showing how to use the FMU Import block to load an FMU file. The FMU file supports execution in Model Exchange mode.



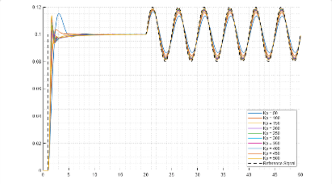
Import and Simulate FMU with Time-Based Clocks in Simulink
Demonstrates the use of a Functional Mockup Unit (FMU) containing time-based clocks in Simulink®. You can import standalone co-simulation...



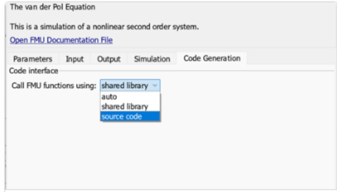
Simplify Interface for Structured Data with FMU Import Block
Use bus signals and structure parameters in an FMU block.



Simulate FMU with Enum Type Ports Using FMU Import Block
The example demonstrates how to use the FMU import block to simulate FMUs that have enumerated data type ports. The FMU import block...



Capture Simulation State, Fast Restart, and Step Through Model Containing FMU
Use the internal states of a standalone cosimulation FMU to enhance the simulation capability of a model with fixed-step solver FMU i... (Simulink Compiler)



Integrate FMI APIs in Generated Code Using FMU Source Code or Binary
Generate code for FMU from source code or binary

Fontys
UNIVERSITY OF APPLIED SCIENCES

Bouncingball

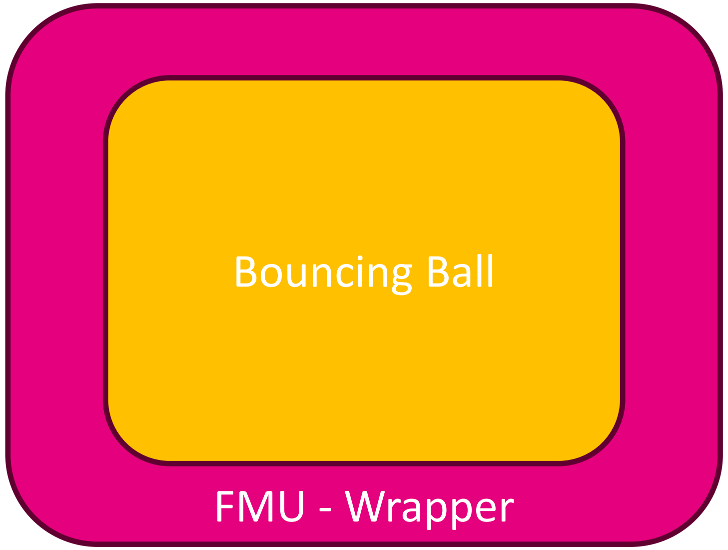
- BouncingBall Example
  - Frequently used as FMU example
  - Implementation of the model
  - Create interface for FMU
  - Test FMU

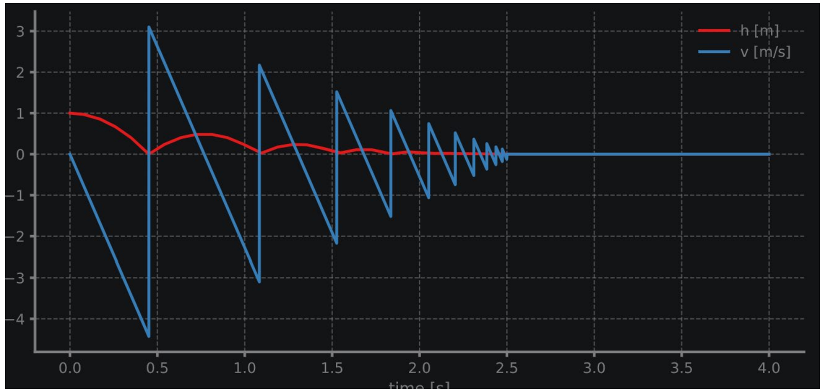
The bouncingBall implements the following equation:

der(h) = v;
der(v) = -g;
when h<0 then v := -e* v

with start values h=1, e=0.7, g = 9.81 and h: height [m], used as state

v: velocity of ball [m/s], used as state
der(h): velocity of ball [m/s]
der(v): acceleration of ball [m/s2]
g: acceleration of gravity [m/s2], a parameter
e: a dimensionless parameter

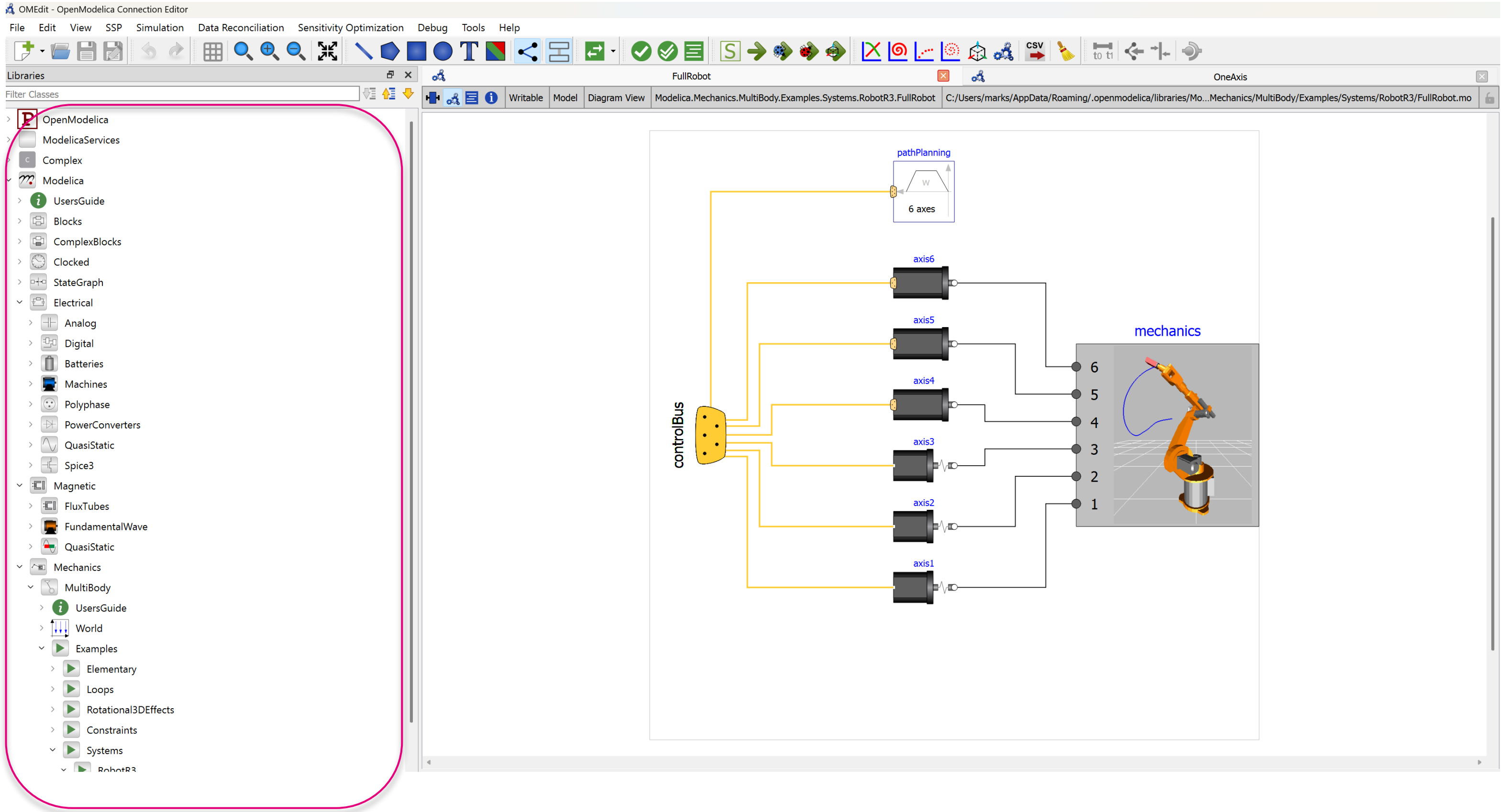




Fontys
UNIVERSITY OF APPLIED SCIENCES

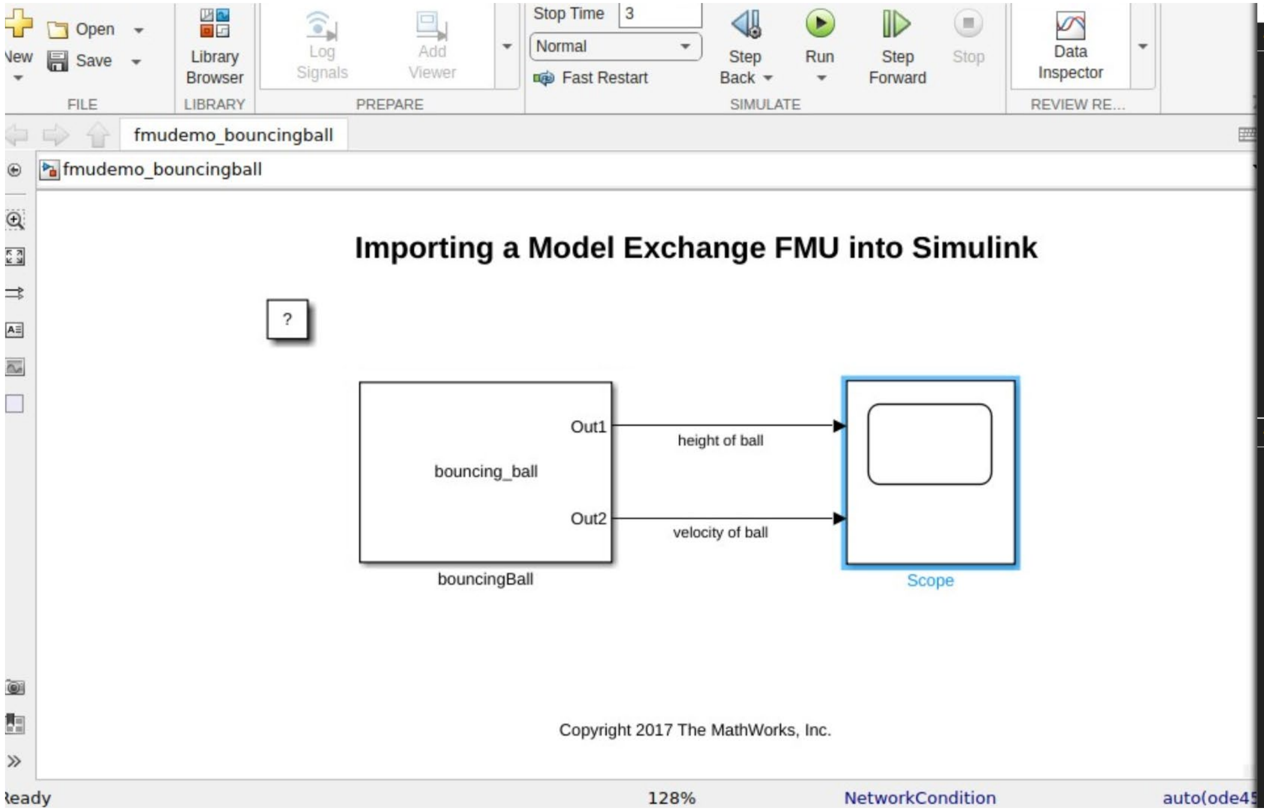
Openmodellica

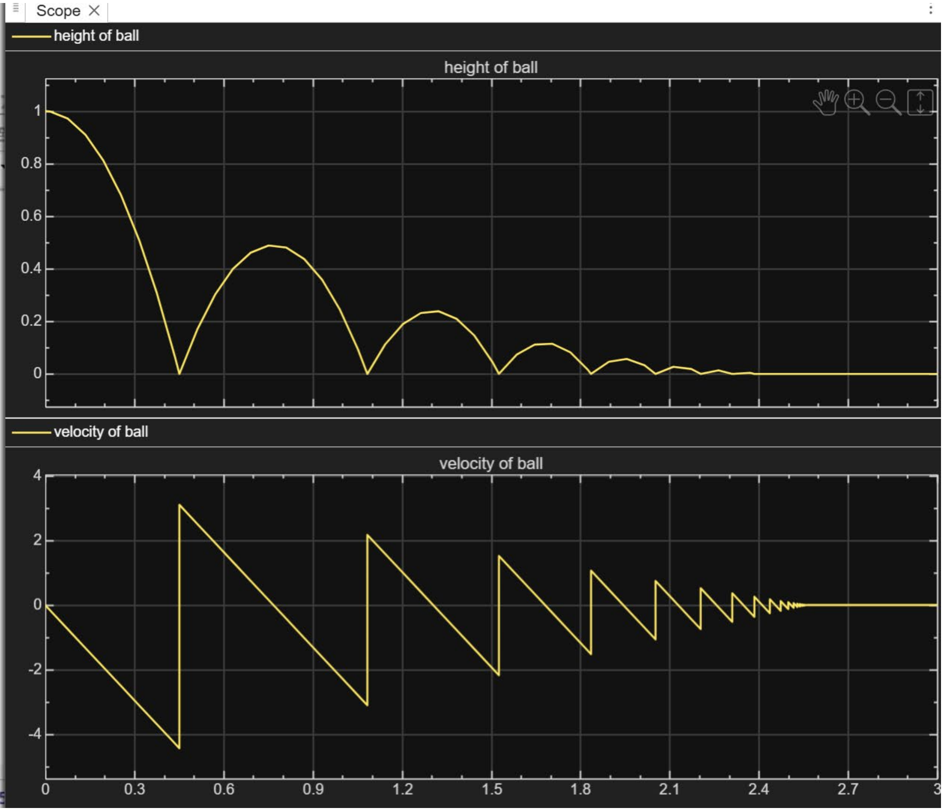
Examples of Models



Fontys UNIVERSITY OF APPLIED SCIENCES

Bouncing Ball Matlab





Goal

- Understanding how FMU can be used
- Implement an FMU with a tool you prefer
- Each tool has advantage disadvantage

- Final goal: development of individual FMU
  - FMU can be deployed and tested with another program that can reads FMU's.
  - Fmu correlation with use case

- Group:
  - Describe General the background of FMU's and how it can be used in mechatronic systems.
  - Technical background of complete overview standard
  - Make references to used sources
  - How could the standard be used for your system

- Individual:
  - can make a report about FMU (as component)
  - Goal and requirements
  - Applied model (type of simulation)
  - Implemented model
  - References model (based on research papers/ implemented from source / link to paper
  - Explanation of model
  - Baseline as an implementation of a model and a test. Complex models will be graded higher.
    - Complex model minimal linked to reference of scientific research paper
    - Copied /implemented model.

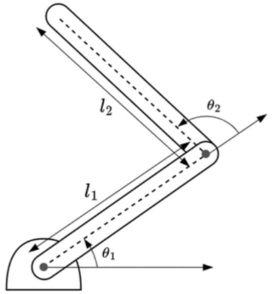


Figure 2: Two degress-of-freedom robot arm

$$\ddot{q} = \begin{bmatrix}\ddot{\theta}_1 \\ \ddot{\theta}_2\end{bmatrix} = \underbrace{B(q)^{-1}\left[-C(\dot{q}, q) - g(q)\right]}_{\text{dynamic}} + \underbrace{\begin{bmatrix} K_{P1}(\theta_{1r} - \theta_1) + K_{D1}\dot{\theta}_1 + K_{I1}e_1 \\ K_{P2}(\theta_{2r} - \theta_2) + K_{D2}\dot{\theta}_2 + K_{I2}e_2 \end{bmatrix}}_{\text{PID control}} \quad (1)$$

Fontys UNIVERSITY OF APPLIED SCIENCES

› FOR SOCIETY

FMU TUTORIALS

Tutorial overview

- Creating Functional Mock-up Unit (FMU) models using PythonFMU and component-model - DNV Technology Insights
- Creating Functional Mock-up Unit (FMU) models using C++ - DNV Technology Insights
- Tutorial — PyFMI 2.5 documentation

FMU matlab

- Importing a Model Exchange FMU into Simulink - MATLAB & Simulink
- Import Co-Simulation FMU into Simulink - MATLAB & Simulink
- Implement an FMU Block - MATLAB & Simulink

Tutorial python and

FMU PathSIM

- FMU Co-Simulation - PathSim Documentation
- FMU ME: Bouncing Ball - PathSim Documentation
- FMU ME: Van der Pol - PathSim Documentation