Functional Mock-up Interface

Thierry S. Nouidui and Michael Wetter
Simulation Research Group

July 22, 2015

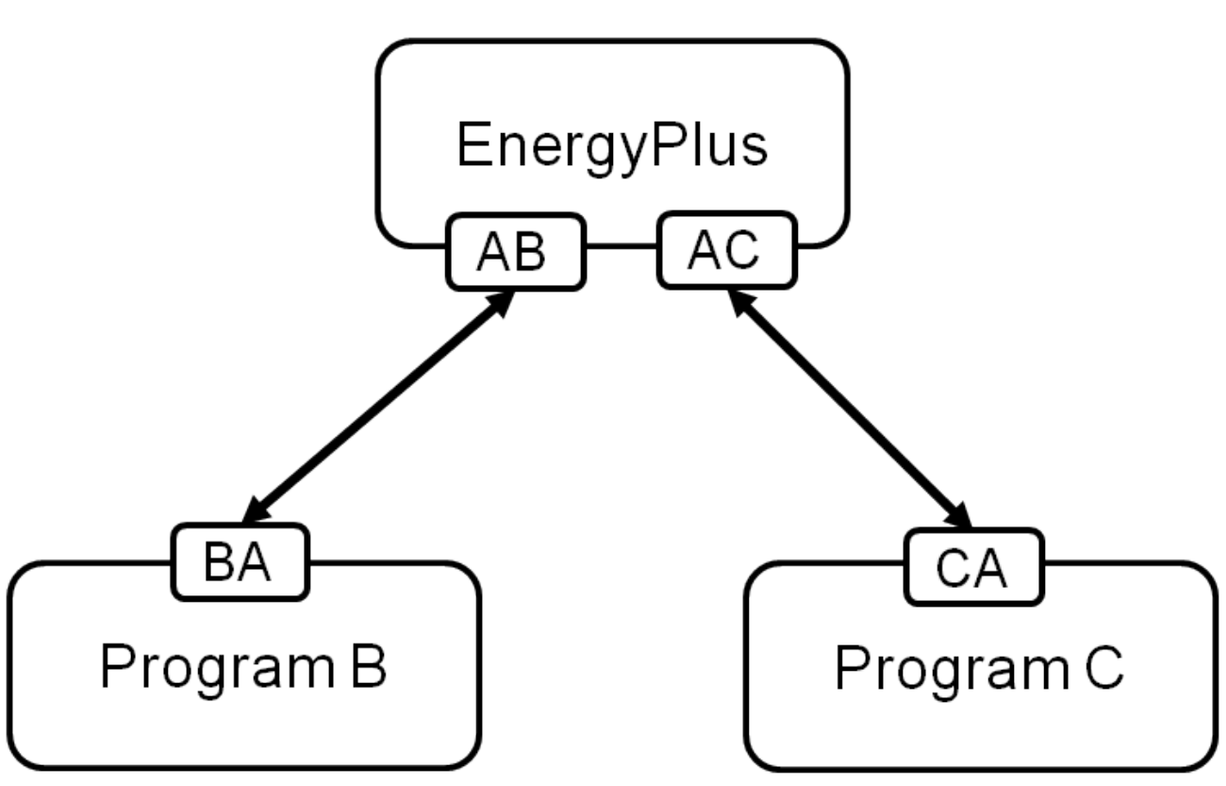
Lawrence Berkeley National Laboratory

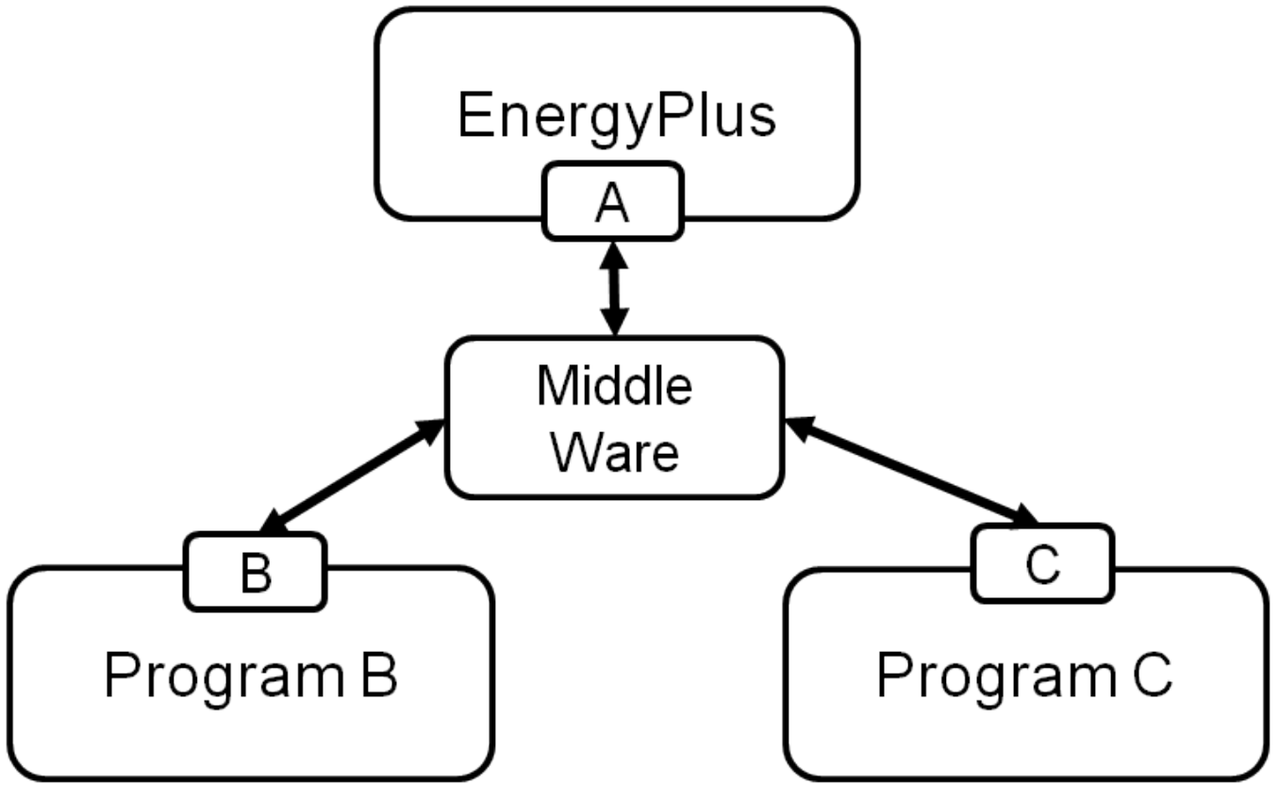
Overview

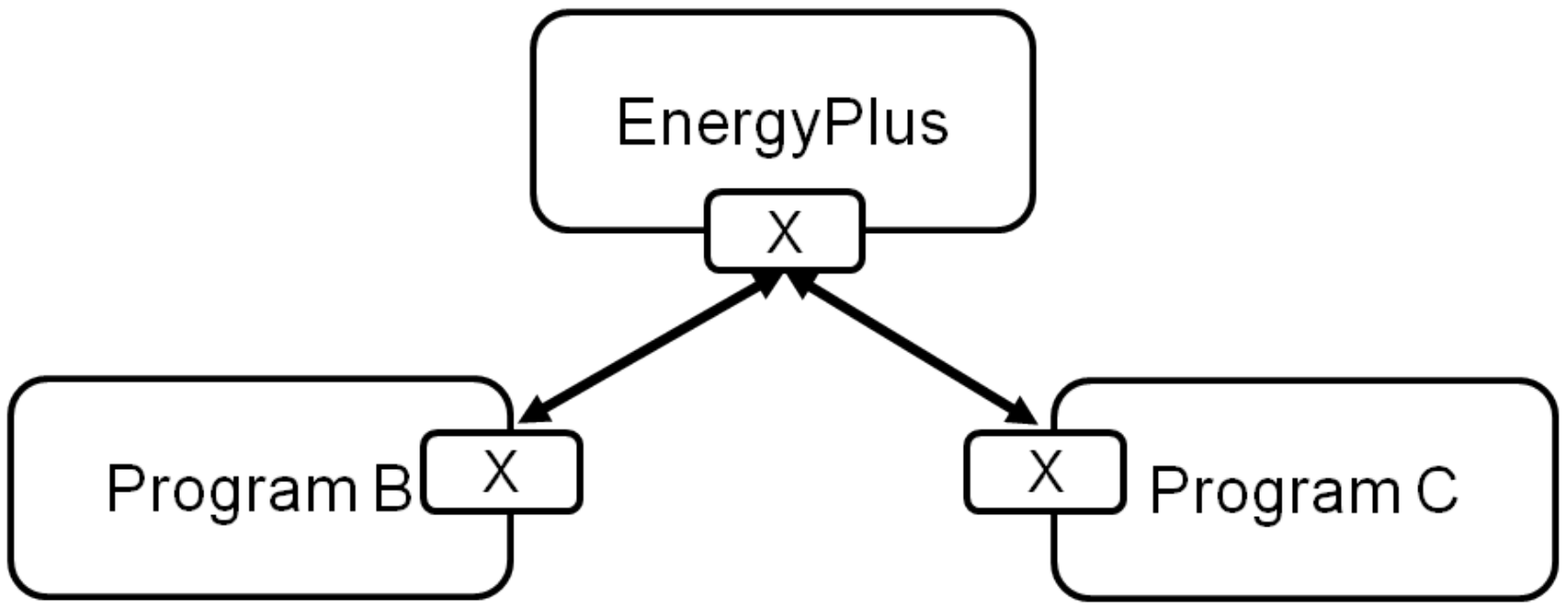
The purpose is to

1. understand the Functional Mock-up Interface (FMI) and Functional Mock-up Unit (FMU)
2. learn how to create and simulate a Functional Mock-up Unit

Use Case - Interoperability between Simulation Programs







X = Functional Mock-up Interface

Functional Mock-up Interface (FMI)

The FMI Standard has been developed to encapsulate and link models and simulators

Developed within MODELISAR.

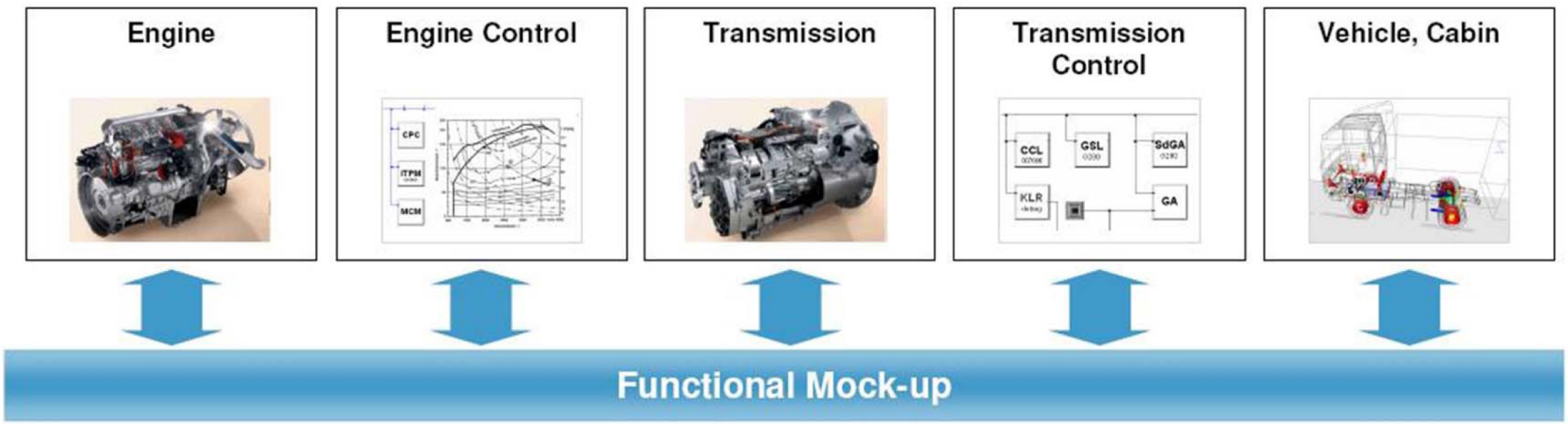
Initially a 28 million € ITEA2 project with 29 partners.

Standardizes API and encapsulation of models and simulators.

Scales from embedded systems to high performance computers.

First version published in 2010. Second version published in 2014.

Initially supported by 35 tools, now supported by 72 tools.



Cosimulation of the behavioral models and the embedded controller software

The FMI Standard has been developed to encapsulate and link models and simulators

FMI standardizes

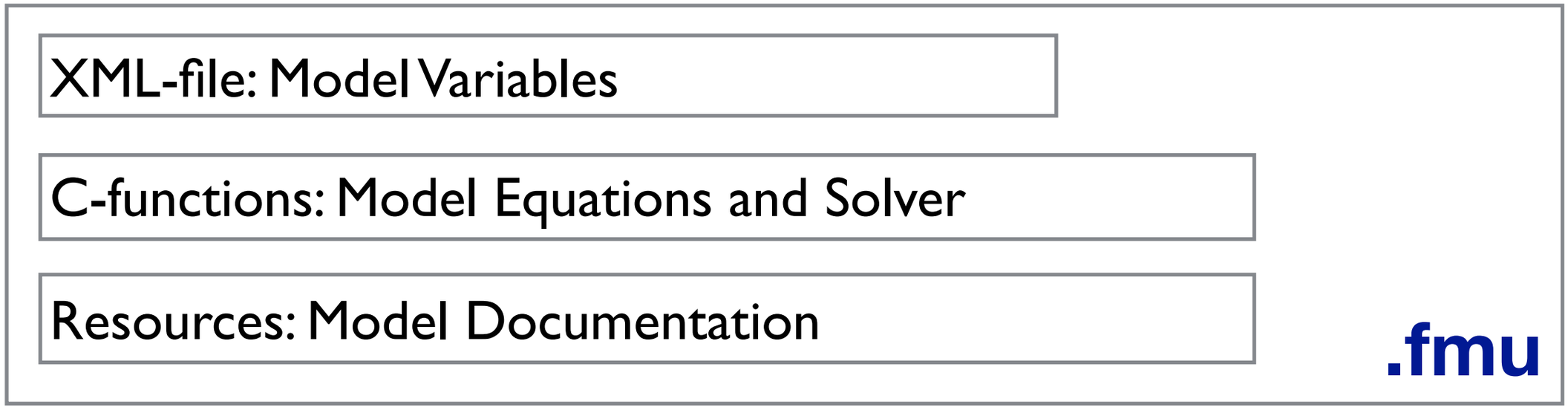
a) a set of C-functions to be implemented by a model/simulator

b) a XML-model description file to be provided by a model/simulator

c) the distribution file format to be used by a model/simulator

A model/simulator which implements FMI is called a Functional Mock-up Unit (FMU)

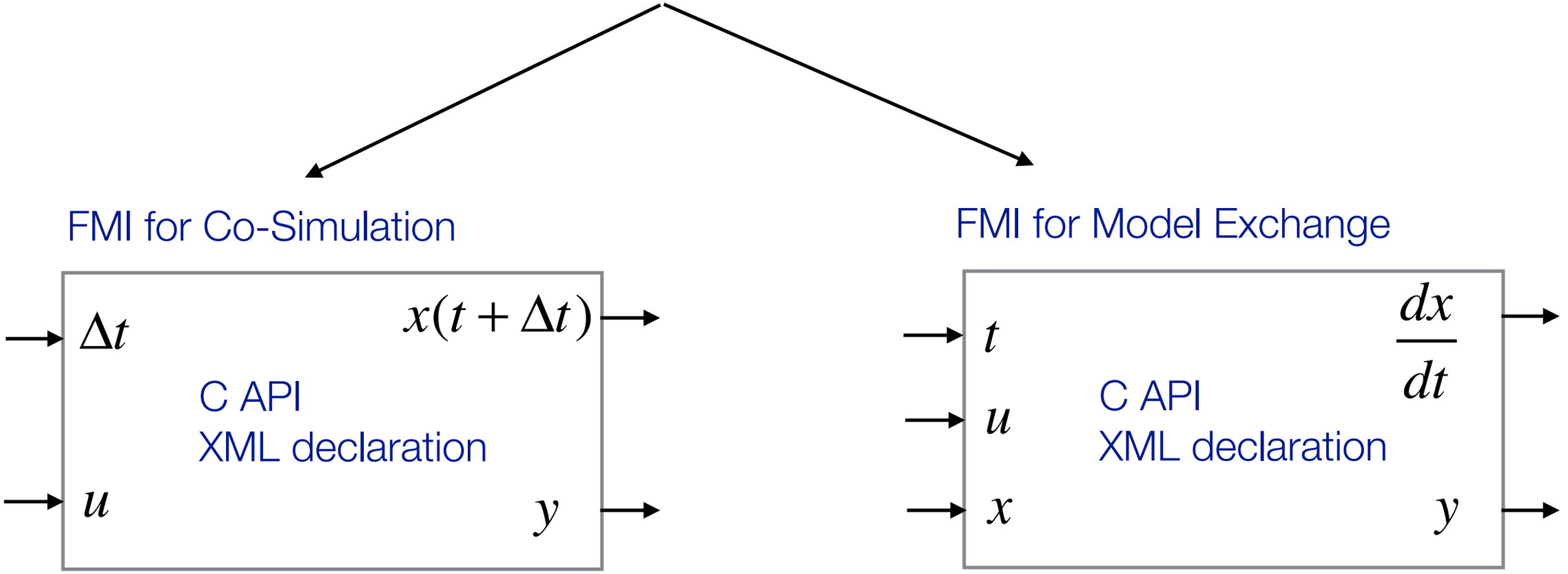
An FMU is a .zip file with the extension .fmu



FMI for Co-Simulation and for Model Exchange

$$\frac{dx}{dt} = u - x,\ x(0) = 1.0$$

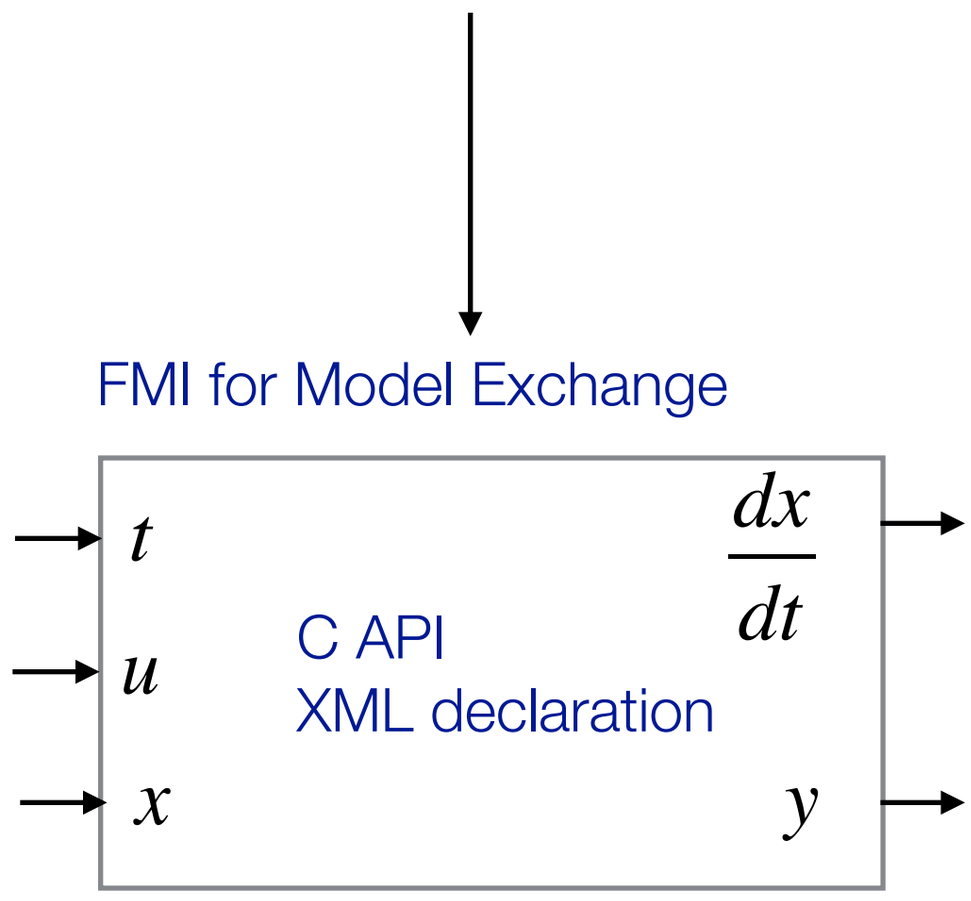
$$y = -x$$



From a model to an FMU

$$\frac{dx}{dt} = u - x,\ x(0) = 1.0$$

$$y = -x$$



XML*- file contains

a) name of variables
b) value reference of variables
c) causality of variables
d) variable dependencies

C API* contains functions to

a) initialize the model
b) set continuous states
c) set inputs
d) get derivatives
e) get outputs
f) terminate the model

*XML and C API contains additional information which are not listed for simplicity.

Master Algorithm for FMU Simulation

```
m = fmi2Instantiate("m", ...)  // "m" is the instance name
Tstart = 0     // start time
Tend   = 10    // stop time
dt     = 0.01  // fixed step size of 10 ms

// set the start time
Tnext = Tend
time  = Tstart
fmi2SetTime(m, time)

// set all variable start values and
// set the input values at time = Tstart
fmi2SetReal/Integer/Boolean/String(m, ...)

// initialize
fmi2SetupExperiment(m,fmi2False,0.0, Tstart, fmi2True,Tend)
fmi2EnterInitializationMode(m)
fmi2ExitInitializationMode(m)

// retrieve initial state x and
fmi2GetContinuousStates(m, x, nx)

// retrieve solution at t=Tstart, for example for outputs
fmi2GetReal/Integer/Boolean/String(m, ...)

while time < Tend loop
  // compute derivatives
  fmi2GetDerivatives(m, der_x, nx)

  // advance time
  h    = min(dt, Tnext-time)
  time = time + h
  fmi2SetTime(m, time)

  // set inputs at t = time
  fmi2SetReal/Integer/Boolean/String(m, ...)

  // set states at t = time and perform one step
  x = x + h*der_x  // forward Euler method
  fmi2SetContinuousStates(m, x, nx)

  if terminateSimulation then goto TERMINATE_MODEL

// terminate simulation and retrieve final values
TERMINATE_MODEL:
fmi2Terminate(m)
```

Exercise on Functional Mock-up Interface (FMI)

Creating an FMU from Dymola

a) Start Dymola and switch to the modeling tab

b) Implement following first order model

$$\frac{dx}{dt} = u - x,\ x(0) = 1.0$$

$$y = -x$$

The Modelica code for the model is

```
model MyFirstFMU
  "This model simulates the exponential decay curve."
  Real x(start = 1.0);
  Modelica.Blocks.Interfaces.RealInput u;
  Modelica.Blocks.Interfaces.RealOutput y;
equation
    der(x) = u-x;
    y = -x;
  annotation (uses(Modelica(version="3.2.1")));
end MyFirstFMU;
```

This defines an input of the FMU

This defines an output of the FMU

c) Export the model as an FMU for Model Exchange 2.0

d) Unzip the FMU and look at the model description file

Handwrite the same FMU

a) Change to the `sources` folder of `myFirstFMU (myFirstFMU/src/sources/)`

b) On Windows OS, edit `build_fmu.bat` and adjust the path to the C-compiler (line 77)

c) Open `myFirstFMU.c`

d) Look at the implementation of following functions:

- `fmi2Instantiate()`

- `fmi2SetContinuousStates()`

- `fmi2SetReal()`

- `fmi2GetDerivatives()`

- `fmi2GetReal()`

e) Open a MS-DOS/shell command prompt

f) Change to the `sources` folder of `myFirstFMU`

g) Run `build_fmu.bat myFirstFMU` (Windows) or `make` (Linux) to create an FMU

This will create `myFirstFMU.fmu` which is two levels up from the `sources` folder

Run FMUs in Ptolemy II

a) Start Ptolemy II

b) Open `MyFirstFMU.xml`

c) From the File/Menu select

```
Import FMU as a Ptolemy Actor for QSS Integration
```

d) Browse to the folder where the handwritten FMU is

e) Select the FMU and import it

f) Do the same with the Dymola generated FMU

g) Connect the output of the `SingleEvent` to the inputs of both FMUs

h) Observe the integrated continuous state variables `x` on the same plot

Summary

The purpose was to

1. understand the Functional Mock-up Interface (FMI) and Functional Mock-up Unit (FMU)
2. learn how to create and simulate a Functional Mock-up Unit

Questions