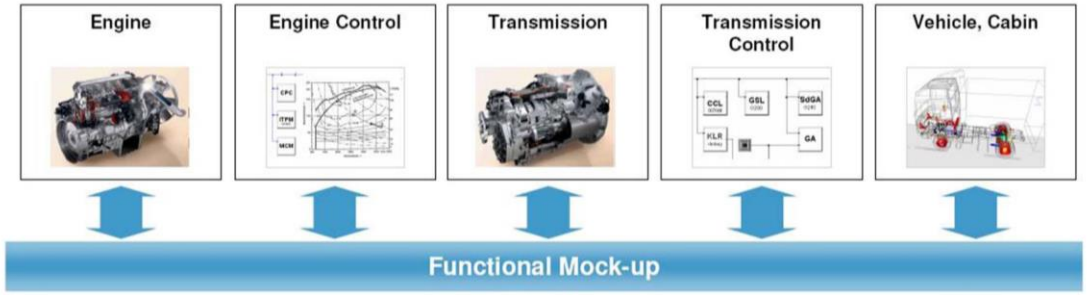
The FMI++ MATLAB Toolbox

A toolbox for MATLAB for importing and exporting FMUs

Edmund Widl
AIT – Austrian Institute of Technology
Center for Energy
Research Field Integrated Energy Systems

FMI – Functional Mock-up Interface (1/2)

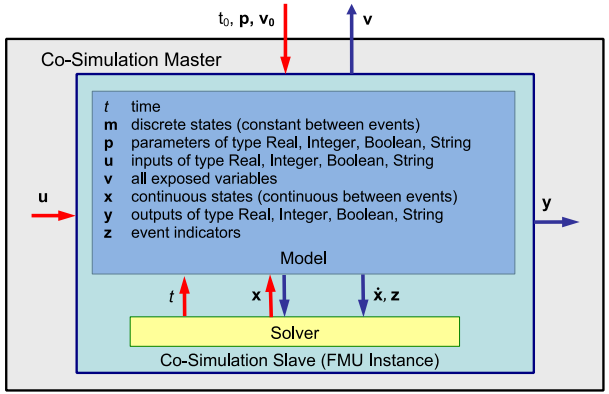
- FMI has been developed to **encapsulate** and **link** models and simulators
  - developed within MODELISAR project
  - driven by a community from *industry and academia*
  - *standardized encapsulation* of *models* and *tools*
  - first version published in 2010, second version published in 2014
  - initially supported by 35 tools, currently *supported by more than 100 tools*
  - see: https://www.fmi-standard.org/



Cosimulation of the behavioral models and the embedded controller software

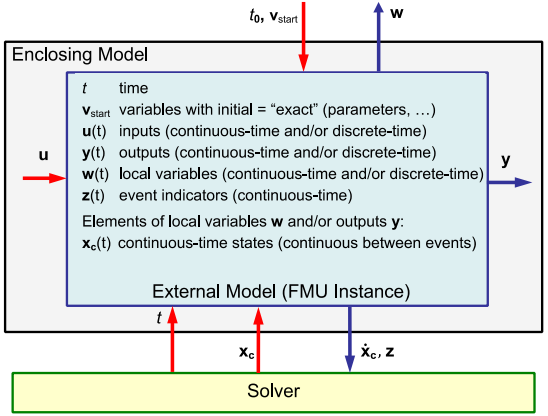
FMI – Functional Mock-up Interface (2/2)

Co-Simulation (CS)



- stand-alone black-box simulation components
- data exchange restricted to discrete communication points
- between two communication points system model is solved by internal solver
- may call another tool at run-time (tool coupling)

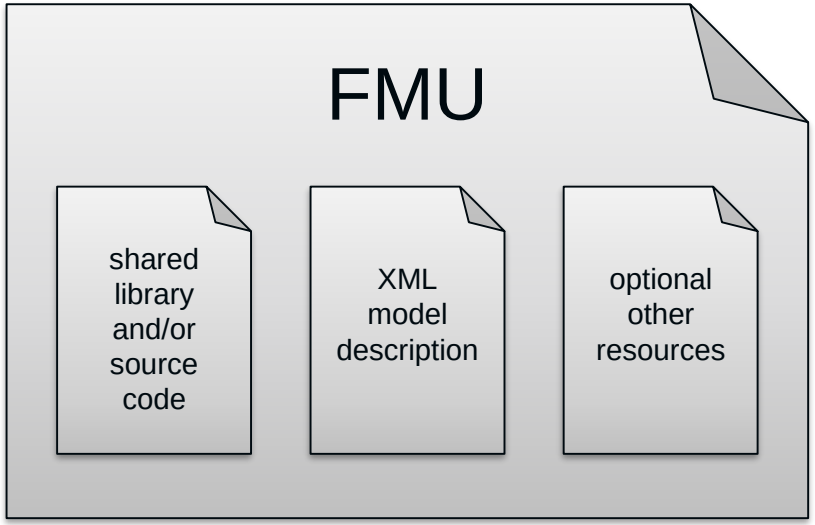
Model Exchange (ME)



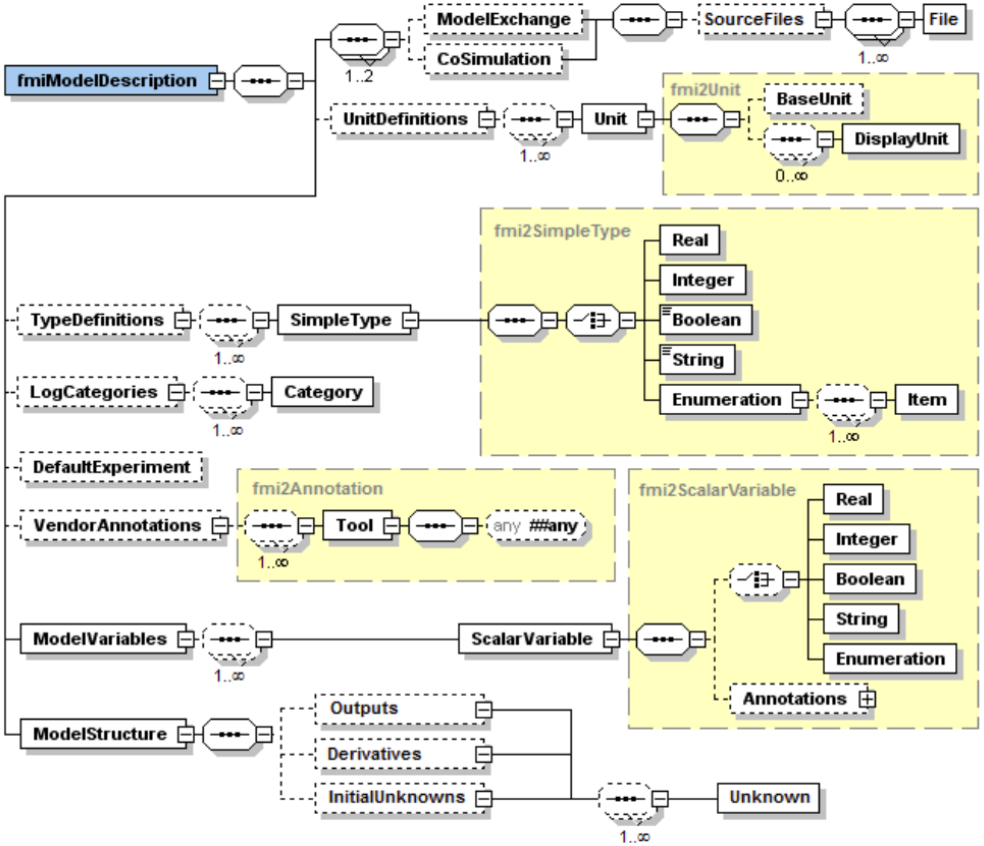
- standardized access to model equations
- models described by differential, algebraic and discrete equations
- time-events, state-events and step-events
- solved with integrators provided by embedding environment.

Functional Mock-up Unit (FMU)

- FMU ≡ **simulation component** compliant with FMI specification
- **ZIP file** that contains:
  - *shared library* and/or source code
  - XML-based *model description*
  - optional other resources (icon, etc.)
- shared library (or source code) implements **FMI API**
- all static information related to an FMU is stored in an XML text file according to the **FMI Description Schema**



FMI Description Schema

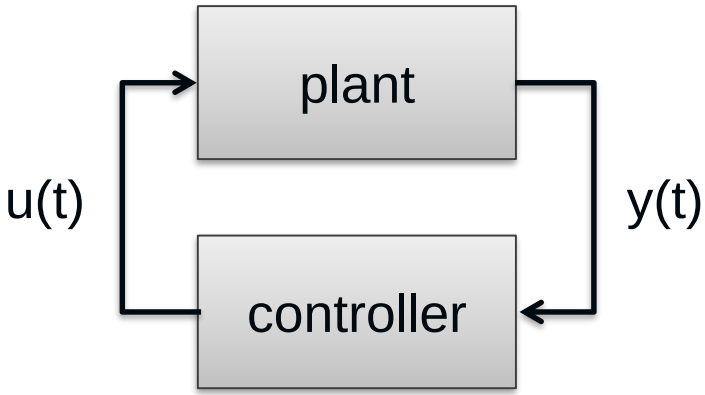


The FMI++ MATLAB toolbox (for Windows)

- An easy-to-use MATLAB toolbox for handling FMUs
  - based on the FMI++ Library (uses SWIG, SUNDIALS and BOOST)
  - open-source, available at http://matlab-fmu.sourceforge.net
- Features:
  - *import* FMUs for Model Exchange (FMI 1.0 and FMI v2.0) *into MATLAB scripts*
  - *import* FMUs for Co-Simulation (FMI 1.0) *into MATLAB scripts*
  - *export MATLAB code* as FMUs for Co-Simulation (FMI 1.0)
- Note: toolbox does not require Simulink or Simulink Coder, but *works with plain MATLAB code*
  - eases the use of procedural MATLAB code (matrix manipulation, optimization, etc.)
  - complements the Modelon Toolbox or Dassault Systemes FMU Kit

FMI++ MATLAB Toolbox

Example application: Rapid prototyping of controls



Example application: Rapid prototyping of controls

Option 1:

MATLAB script

- implements controller
- executes main simulation loop (master)

DYMOLA model

- implements plant model
- imported as FMU for ME

u(t)

plant

controller

y(t)

Option 2:

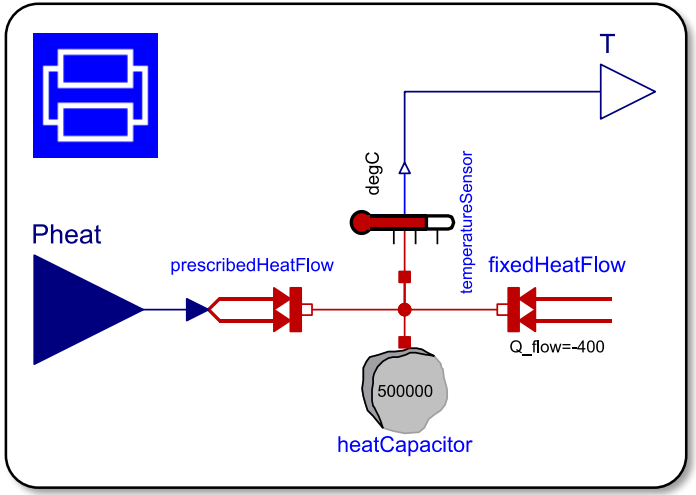
DYMOLA model

- implements plant model
- executes main simulation loop (master)

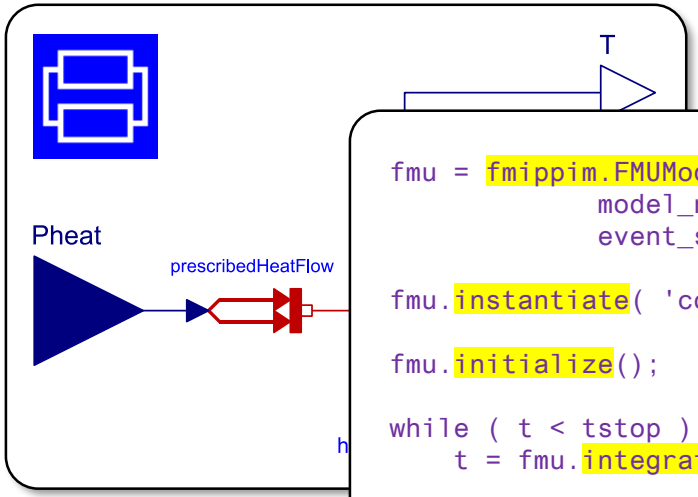
MATLAB script

- implements controller
- imported as FMU for CS

Option 1: Import FMUs in MATLAB scripts

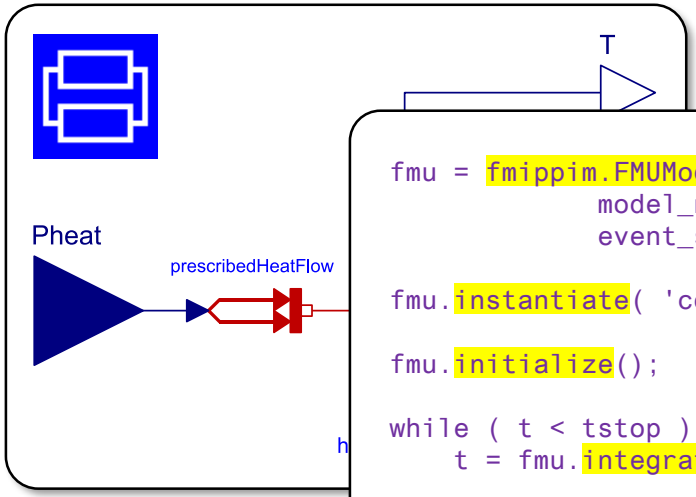


Option 1: Import FMUs in MATLAB scripts

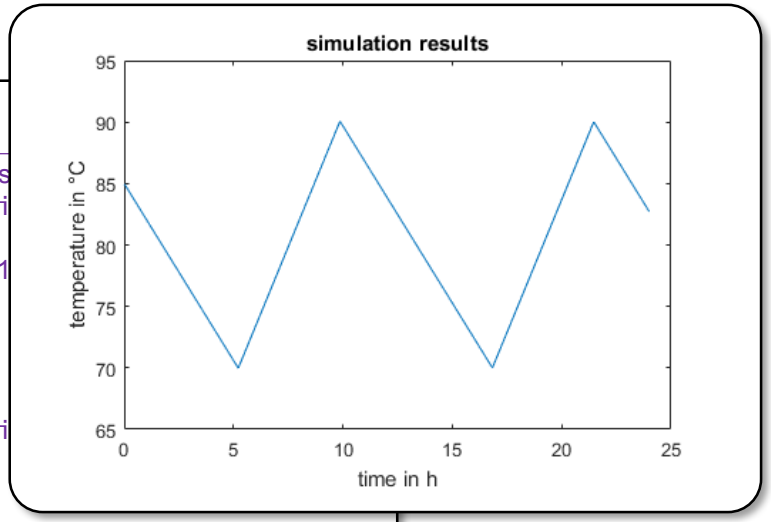



```
fmu = fmippim.FMUModelExchangeV2( uri_to_extracted_fmu, …
            model_name, logging_on, stop_before_event, …
            event_search_precision, integrator_type );

fmu.instantiate( 'controlled_radiator1' );

fmu.initialize();

while ( t < tstop )
    t = fmu.integrate( t + stepsize, integrator_stepsize );

    T = fmu.getRealValue( 'T' );

    if ( T >= Thigh ) fmu.setRealValue( 'Pheat', 0.0 );
    elseif ( T <= Tlow ) fmu.setRealValue( 'Pheat', 1e3 );
    end;

end;
```

Option 1: Import FMUs in MATLAB scripts




```
fmu = fmippim.FMUModelExchangeV2( uri_
            model_name, logging_on, s
            event_search_precision, i

fmu.instantiate( 'controlled_radiator1

fmu.initialize();

while ( t < tstop )
    t = fmu.integrate( t + stepsize, i

    T = fmu.getRealValue( 'T' );

    if ( T >= Thigh ) fmu.setRealValue( 'Pheat', 0.0 );
    elseif ( T <= Tlow ) fmu.setRealValue( 'Pheat', 1e3 );
    end;

end;
```

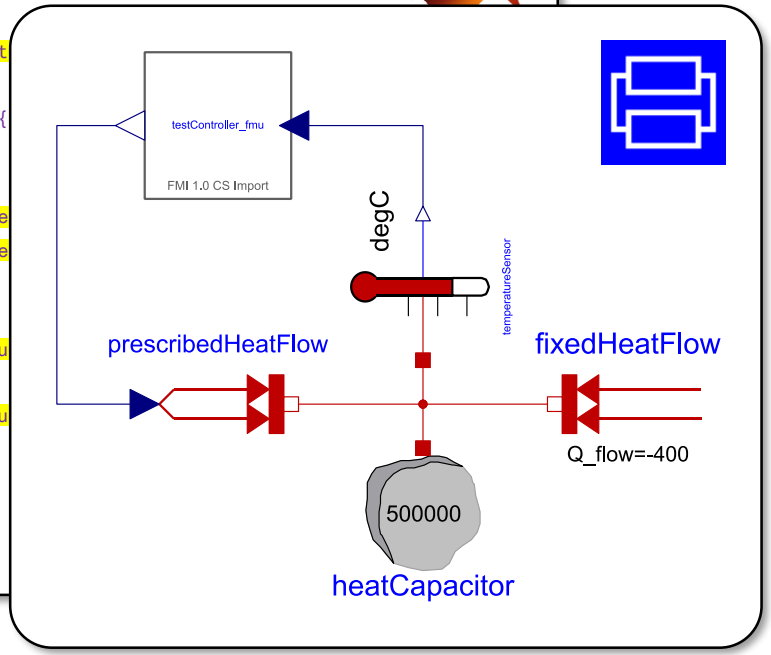


Option 2: Export MATLAB script as FMU

```
classdef SimpleController < fmipputils.FMIAdapter

   methods

      function init( obj, currentCommunicationPoint )
         obj.defineRealInputs( { 'T' } );
         obj.defineRealOutputs( { 'Pheat' } );
      end

      function doStep( obj, currentCommunicationPoint, communicationStepSize )
         realInputValues = obj.getRealInputValues();
         T = realInputValues(1);
         if ( T >= 90 )
            obj.setRealOutputValues( 0 );
         elseif ( T <= obj.Tlow_ )
            obj.setRealOutputValues( 1e3 );
         end
      end
   end
end
```

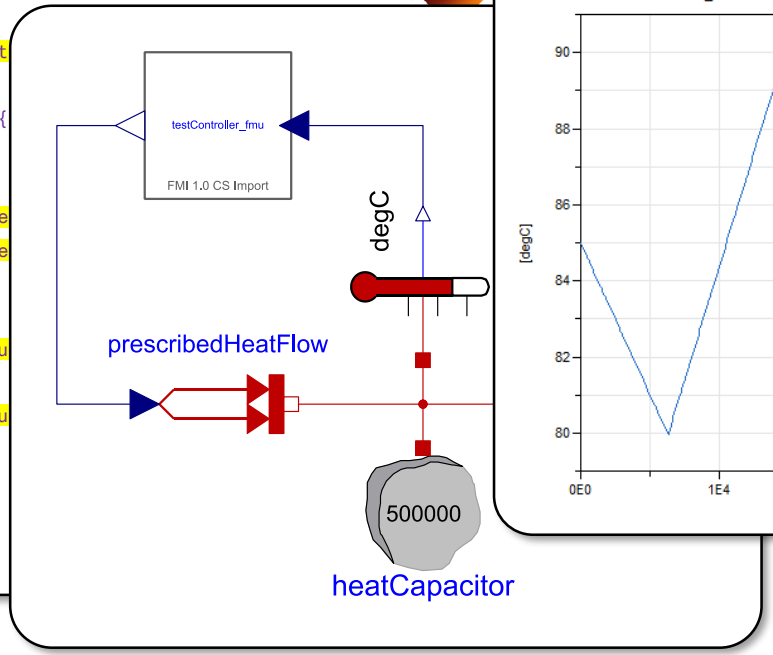
Option 2: Export MATLAB script as FMU

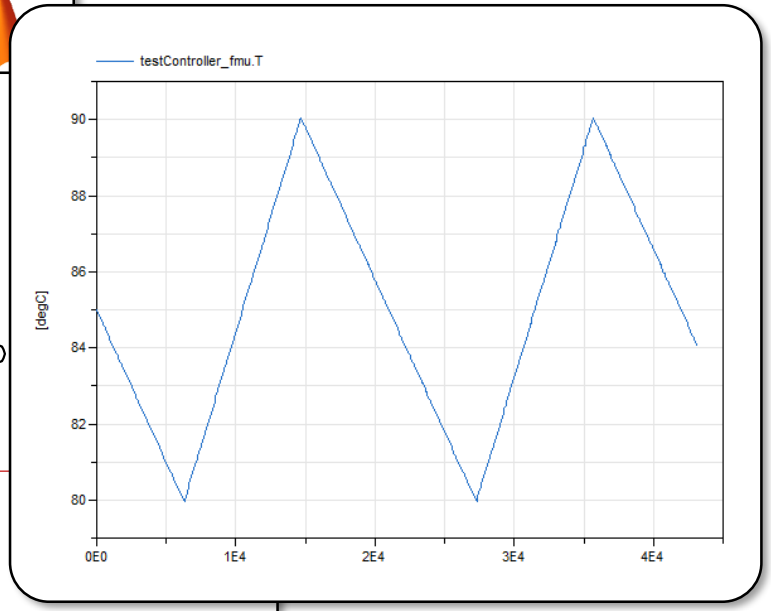
```
classdef SimpleController < fmipputils.FMIAdapter

   methods

      function init( obj, current
         obj.defineRealInputs( {
         obj.defineRealOutputs( {
      end

      function doStep( obj, curre
         realInputValues = obj.ge
         T = realInputValues(1);
         if ( T >= 90 )
            obj.setRealOutputValu
         elseif ( T <= obj.Tlow_
            obj.setRealOutputValu
         end
      end
   end
end
```



Option 2: Export MATLAB script as FMU

```
classdef SimpleController < fmipputils.FMIAdapter

   methods

      function init( obj, current
         obj.defineRealInputs( {
         obj.defineRealOutputs( {
      end

      function doStep( obj, curre
         realInputValues = obj.ge
         T = realInputValues(1);
         if ( T >= 90 )
            obj.setRealOutputValu
         elseif ( T <= obj.Tlow_
            obj.setRealOutputValu
         end
      end
   end
end
```





Debugging MATLAB code before FMU export

- Implemented MATLAB code can be *tested* and *debugged* before exporting it as an FMU for Co-Simulation

- This can be done using the *dedicated methods* of class `FMIAdapter`:
  - `debugSetRealInputValues(…)`
  - `debugGetRealOutputValues(…)`
  - etc.

```
import SimpleController
test = SimpleController();
test.init( 0. );
test.debugSetRealInputValues( [ 95 ] );
test.doStep( 0., 0. );
output = test.debugGetRealOutputValues();
assert( 0 == output(1) );
```

Links

Download the FMI++ MATLAB Toolbox (for Windows) at

http://matlab-fmu.sourceforge.net

FMI++ MATLAB Toolbox

- Related links:
  - Official FMI homepage: https://www.fmi-standard.org/
  - FMI++ Library: http://fmipp.sourceforge.net
  - FMI++ Python Interface (for Windows): https://pypi.python.org/pypi/fmipp
  - FMI++ PowerFactory Export Utility: http://powerfactory-fmu.sourceforge.net
  - FMI++ TRNSYS Export Utility: http://trnsys-fmu.sourceforge.net

AUSTRIAN INSTITUTE OF TECHNOLOGY
TOMORROW TODAY

THANK YOU FOR YOUR ATTENTION!