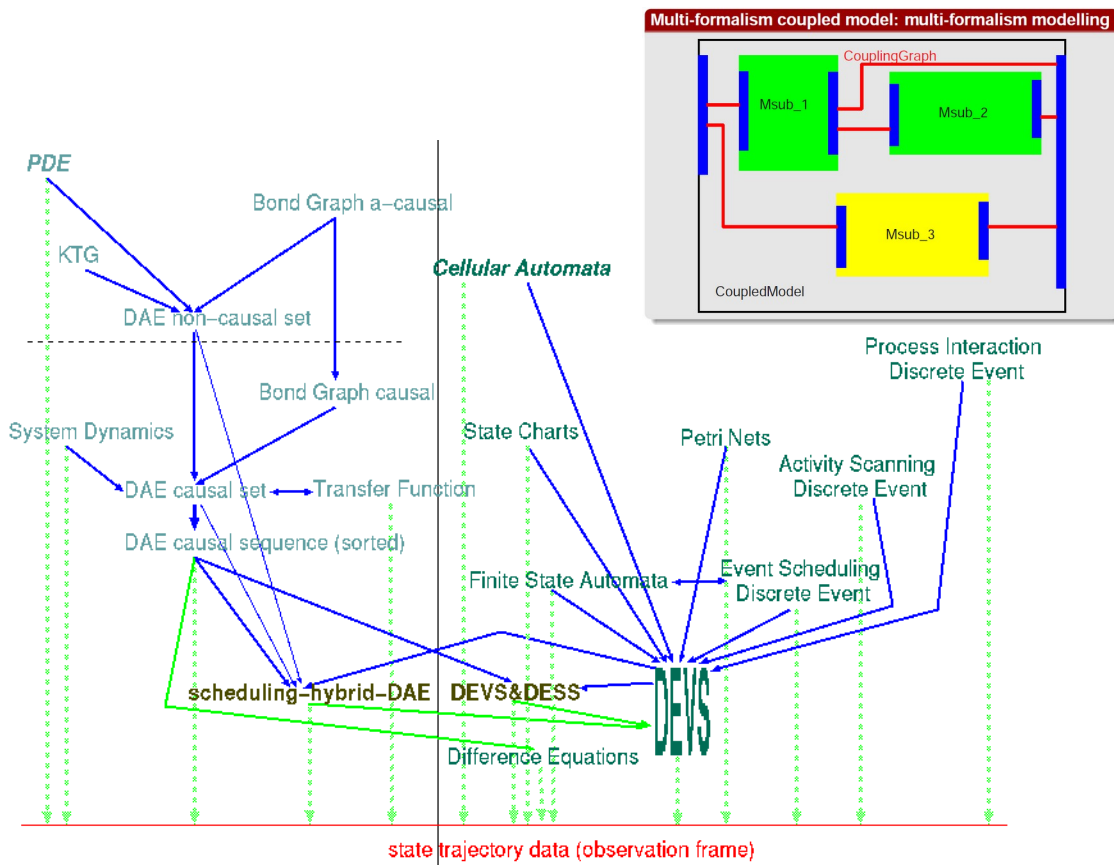
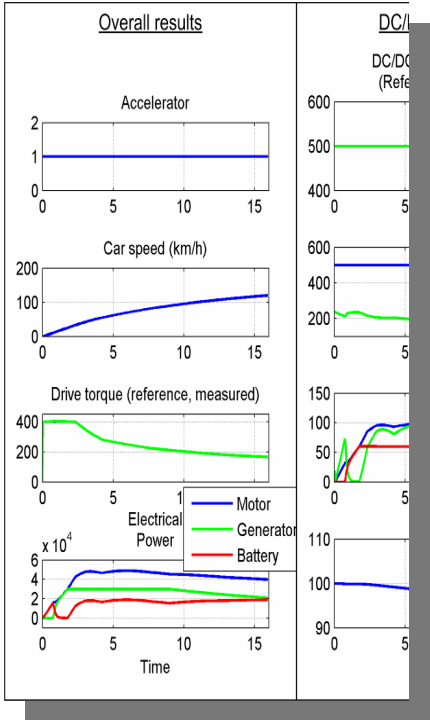
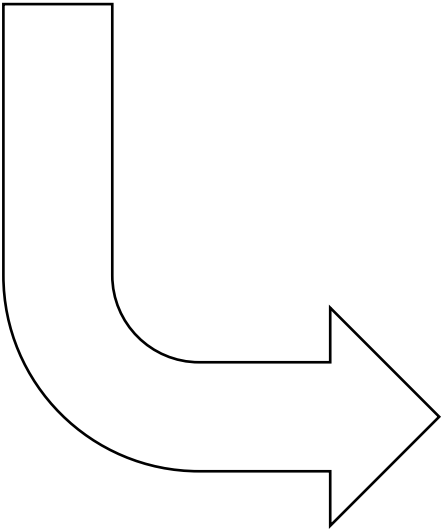
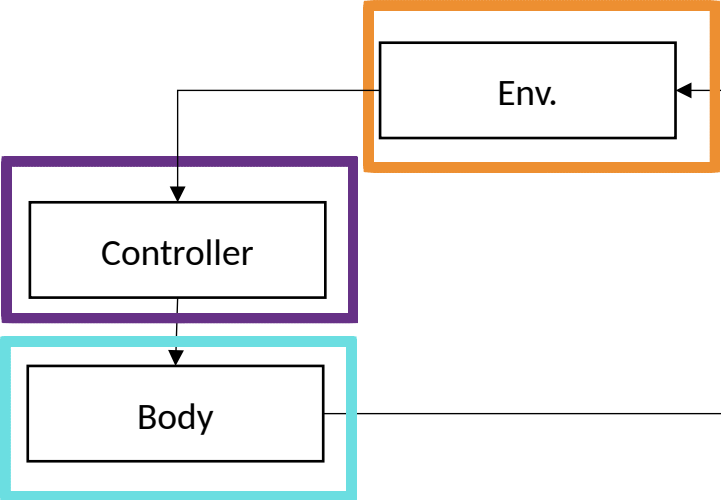




An introduction to **co-simulation** and to the **FMI model exchange and co-simulation standard** using OMEdit, SSP, and hand-coding

Rakshit Mittal, Cláudio Gomes, and **Hans Vangheluwe**

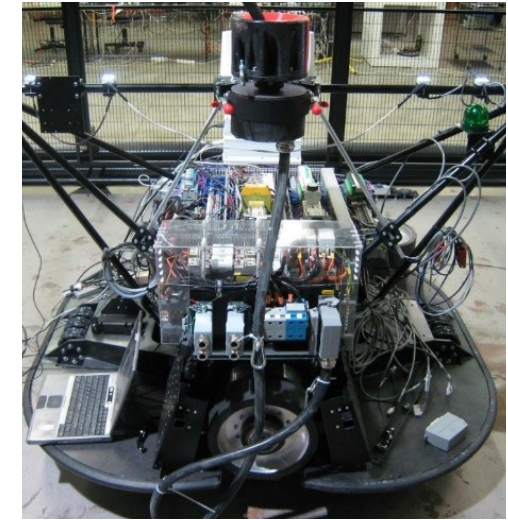
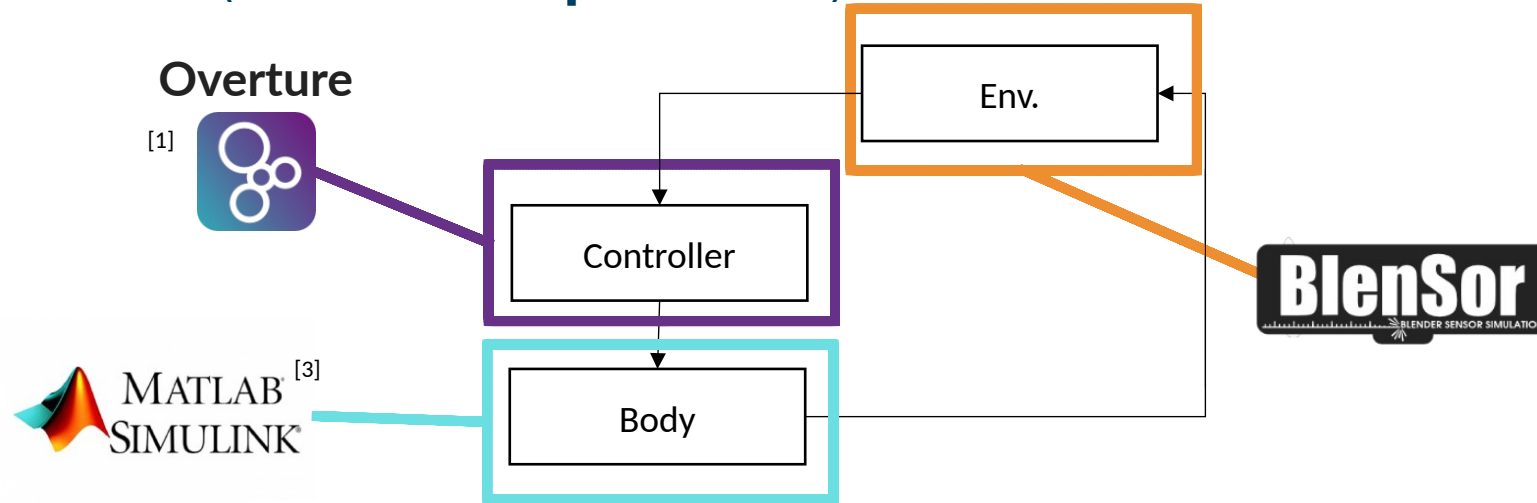
problem: **full-system analysis**



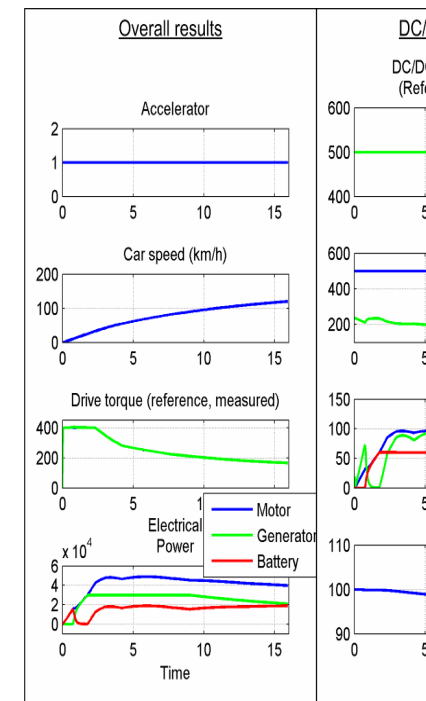
badminton playing robot

problem: **full-system analysis**

(also when **IP protected**)

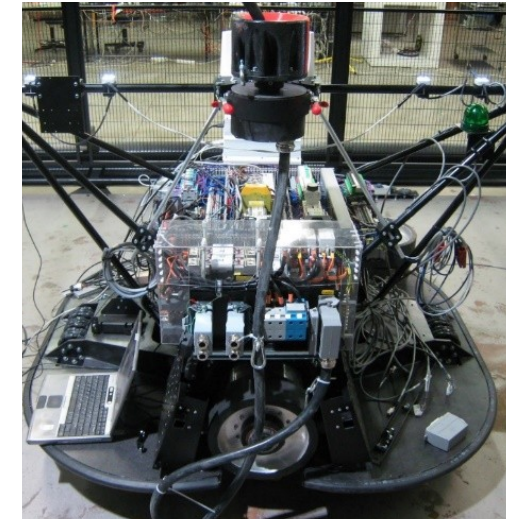
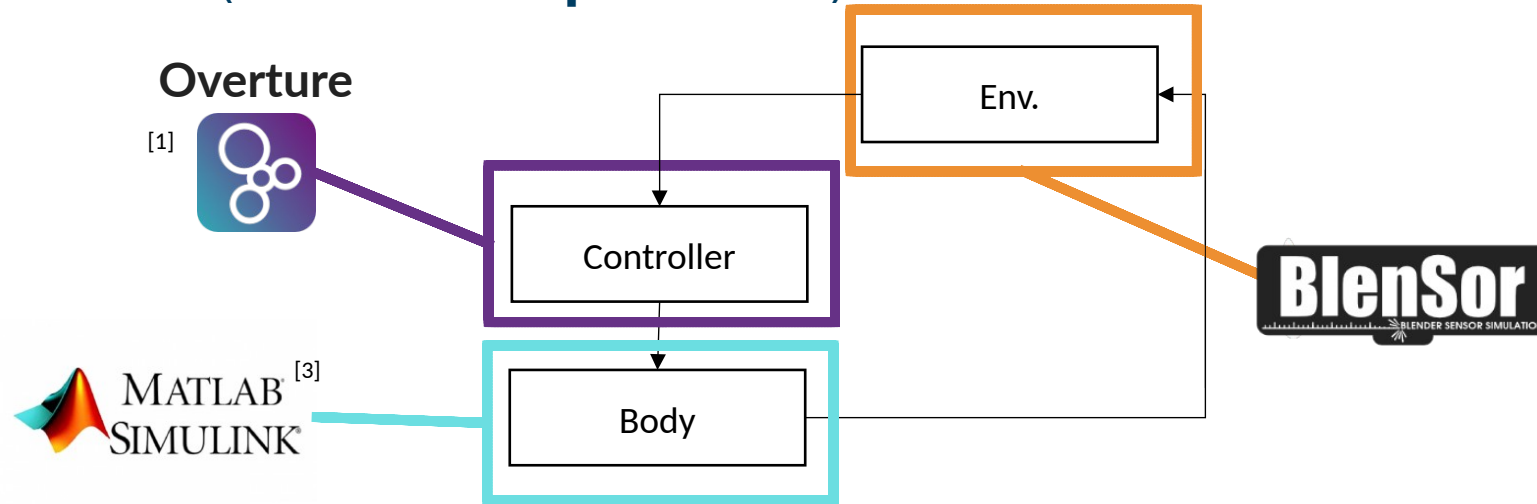


badminton playing robot



problem: **full-system analysis**

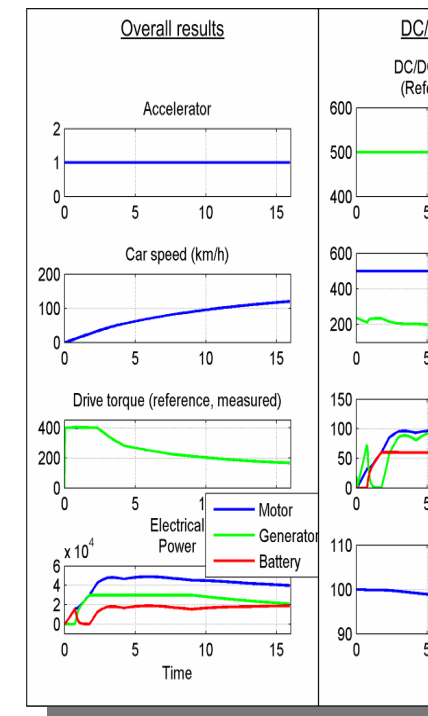
(also when **IP protected**)



badminton playing robot

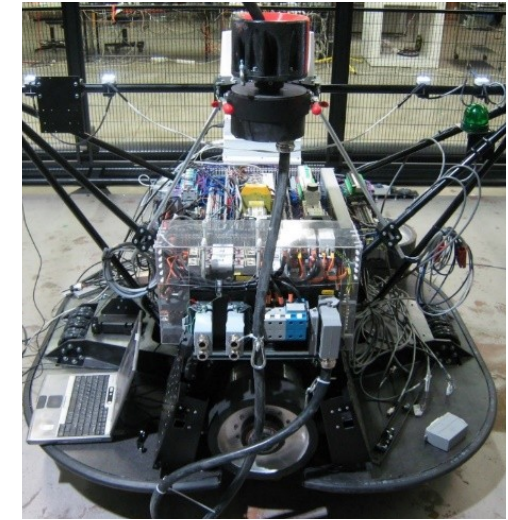
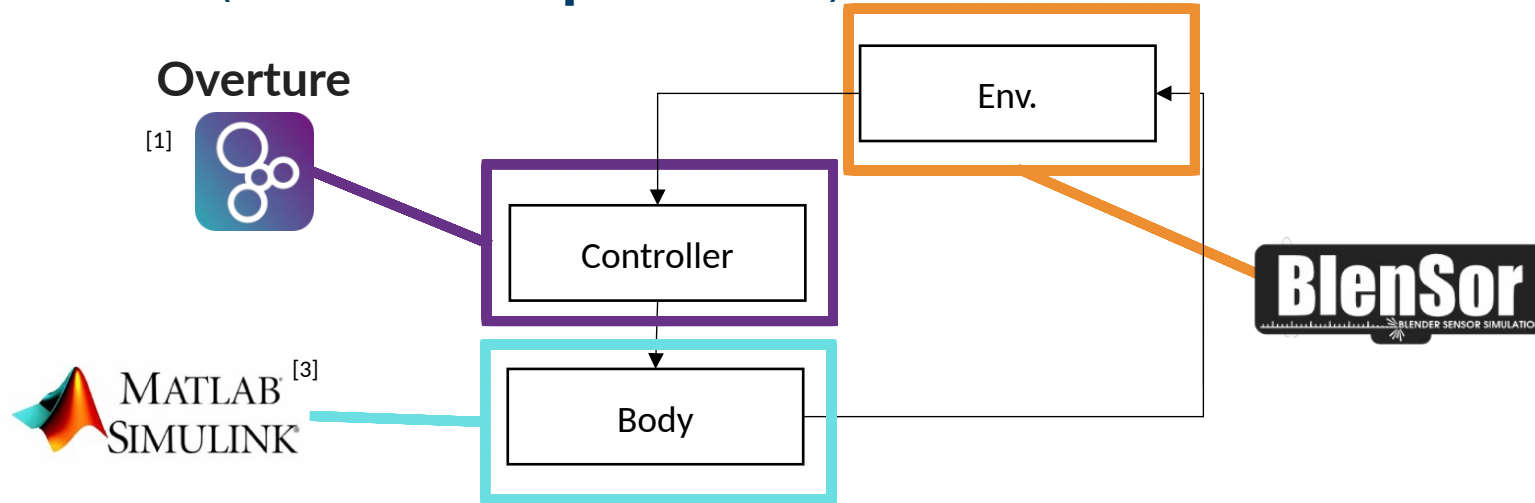
solution: combine
sub-system **simulators**

aka **co-simulation**



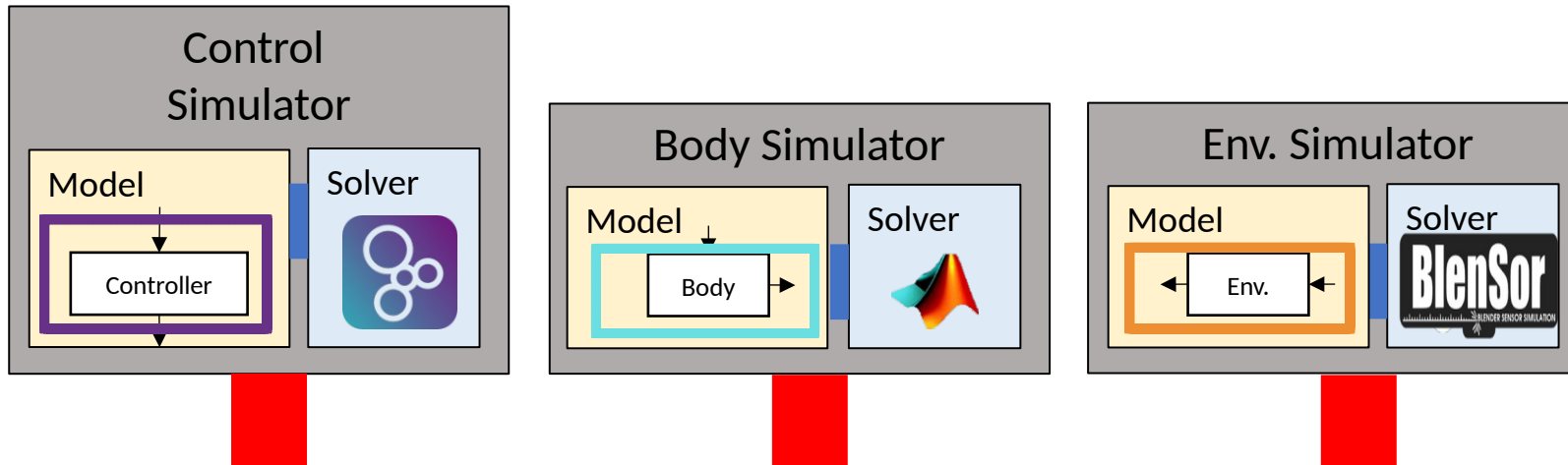
problem: **full-system analysis**

(also when **IP protected**)

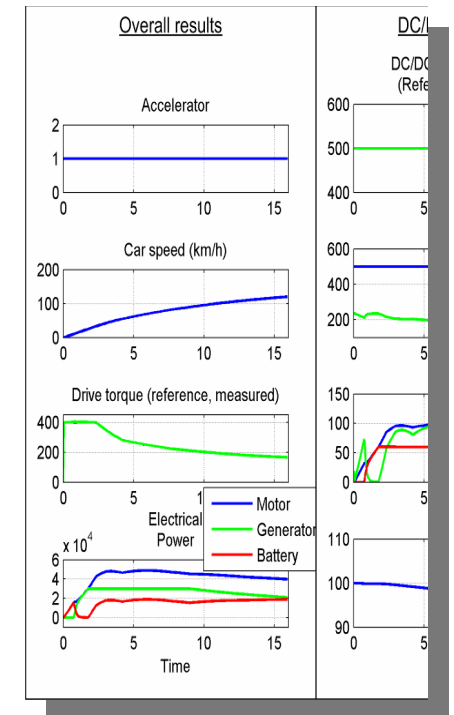
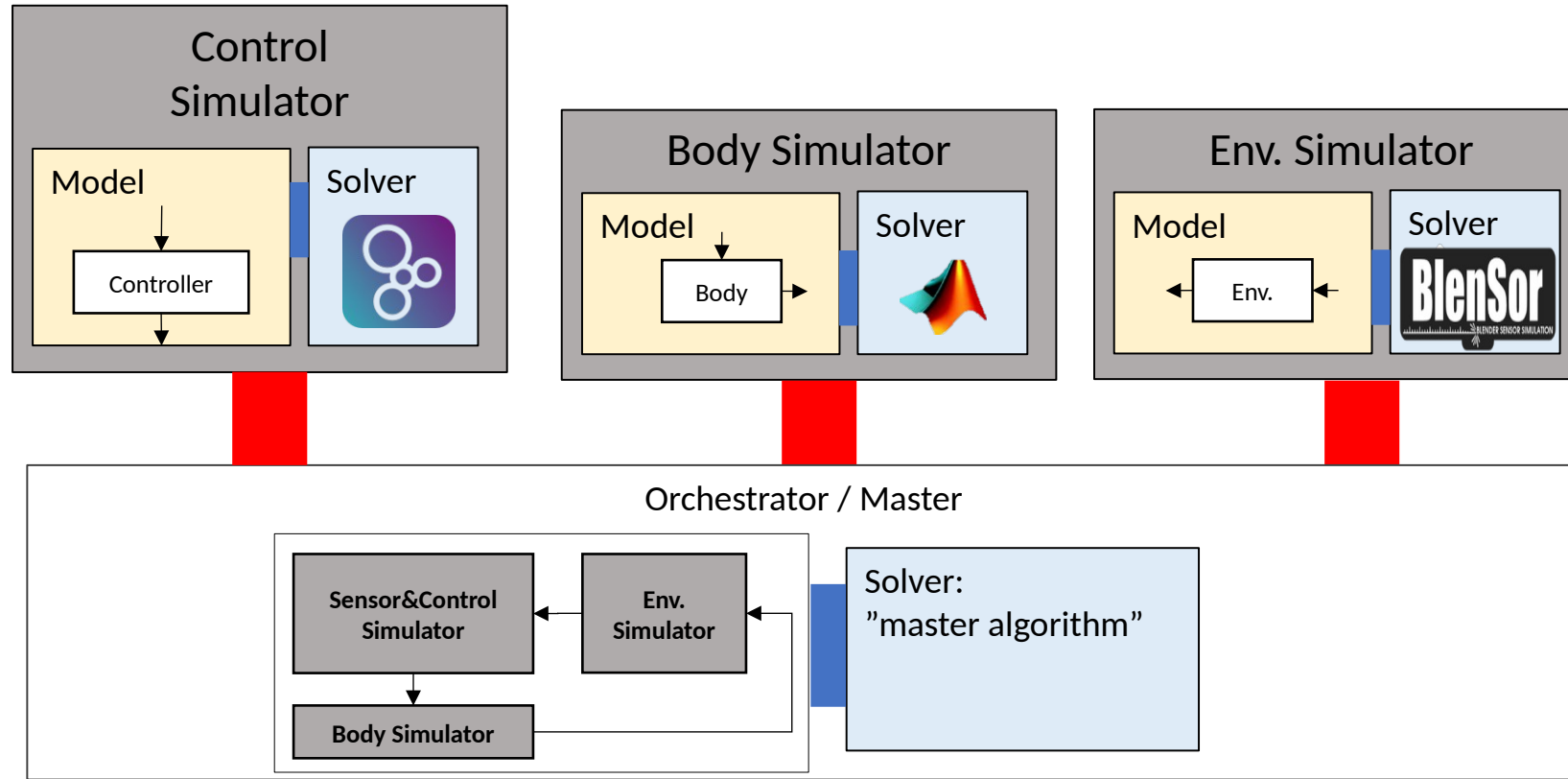


badminton playing robot

solution: **co-simulation**



co-simulation: how? (when IP protected “black box” simulators)



“master” algorithm: use a simulator/solver for Causal Block Diagrams (CBDs)

for CT-CBD: $t_{\text{approx}} = i \cdot \Delta t$ is “logical time”

$i = 0$

$t_{\text{approx}} = 0$

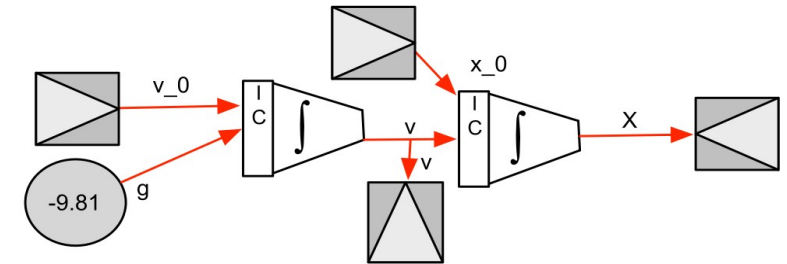
```
if (not end_condition(i, ...)):
    depGraph = buildDepGraph(CBD)
    schedule = loopDetectAndTopSort(depGraph)
    for gblock in schedule:
        gblock.compute() // is now one simulation step in block
else:
    exit()
```

$i = 1$

$t_{\text{approx}} += \Delta t$

```
while (not end_condition(i, ...)):
    depGraph = buildDepGraph(CBD)
    schedule = loopDetectAndTopSort(depGraph)
    for gblock in schedule:
        gblock.compute() // is now one simulation step in block
    i++
     $t_{\text{approx}} += \Delta t$ 
```

exit()



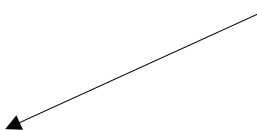
“master” algorithm: use a simulator/solver for Causal Block Diagrams (CBDs)

for CT-CBD: $t_{\text{approx}} = i \cdot \Delta t$ is “logical time”

$i = 0$

$t_{\text{approx}} = 0$

need “zero-delay feedthrough” information
(to find algebraic loops) (from FMI 2.0)



```
if (not end_condition(i, ...)):
```

```
    depGraph = buildDepGraph(CBD)
```

```
    schedule = loopDetectAndTopSort(depGraph)
```

```
    for gblock in schedule:
```

```
        gblock.compute() // is now one simulation step in block
```

```
else:
```

```
    exit()
```

$i = 1$

$t_{\text{approx}} += \Delta t$

```
while (not end_condition(i, ...)):
```

```
    depGraph = buildDepGraph(CBD)
```

```
    schedule = loopDetectAndTopSort(depGraph)
```

```
    for gblock in schedule:
```

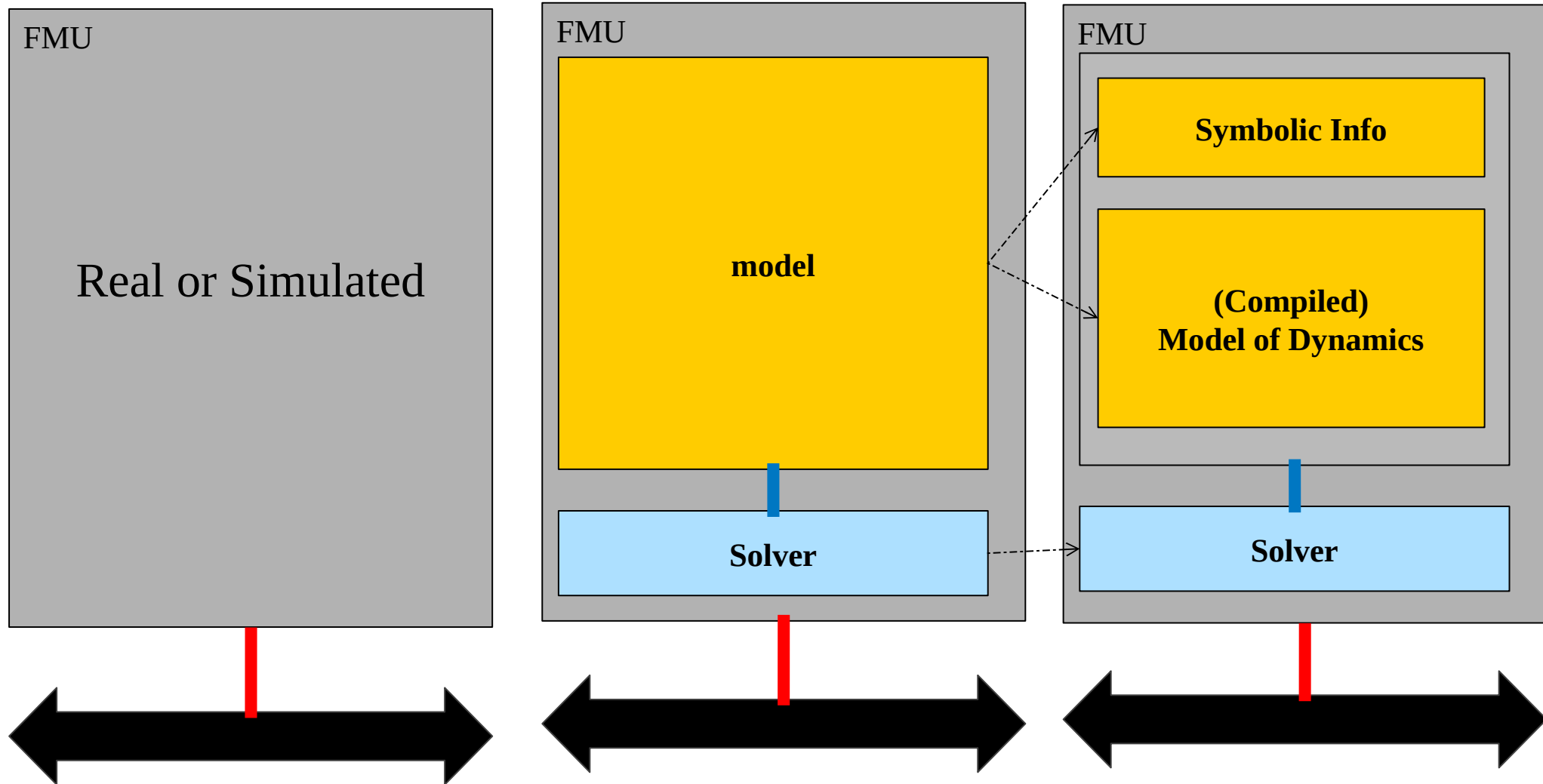
```
        gblock.compute() // is now one simulation step in block
```

```
    i++
```

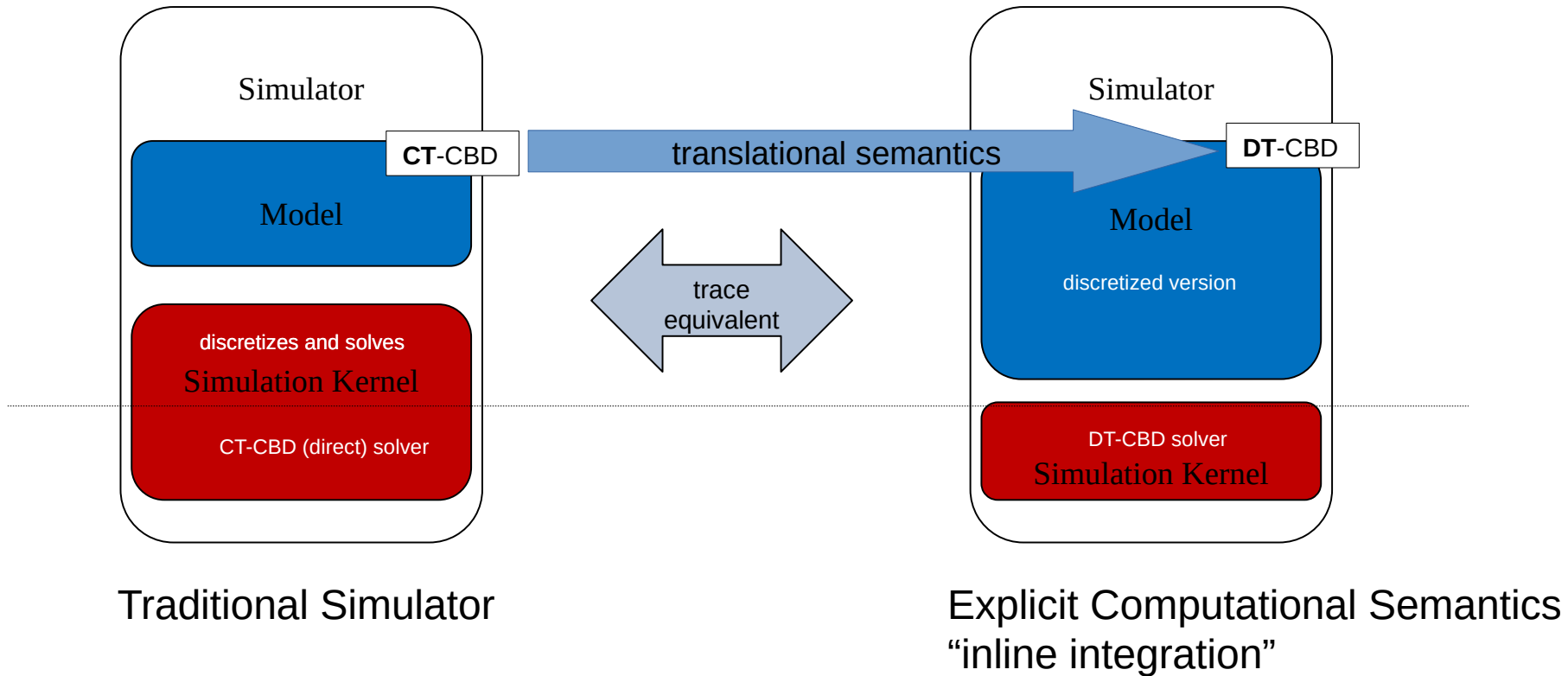
```
     $t_{\text{approx}} += \Delta t$ 
```

```
exit()
```

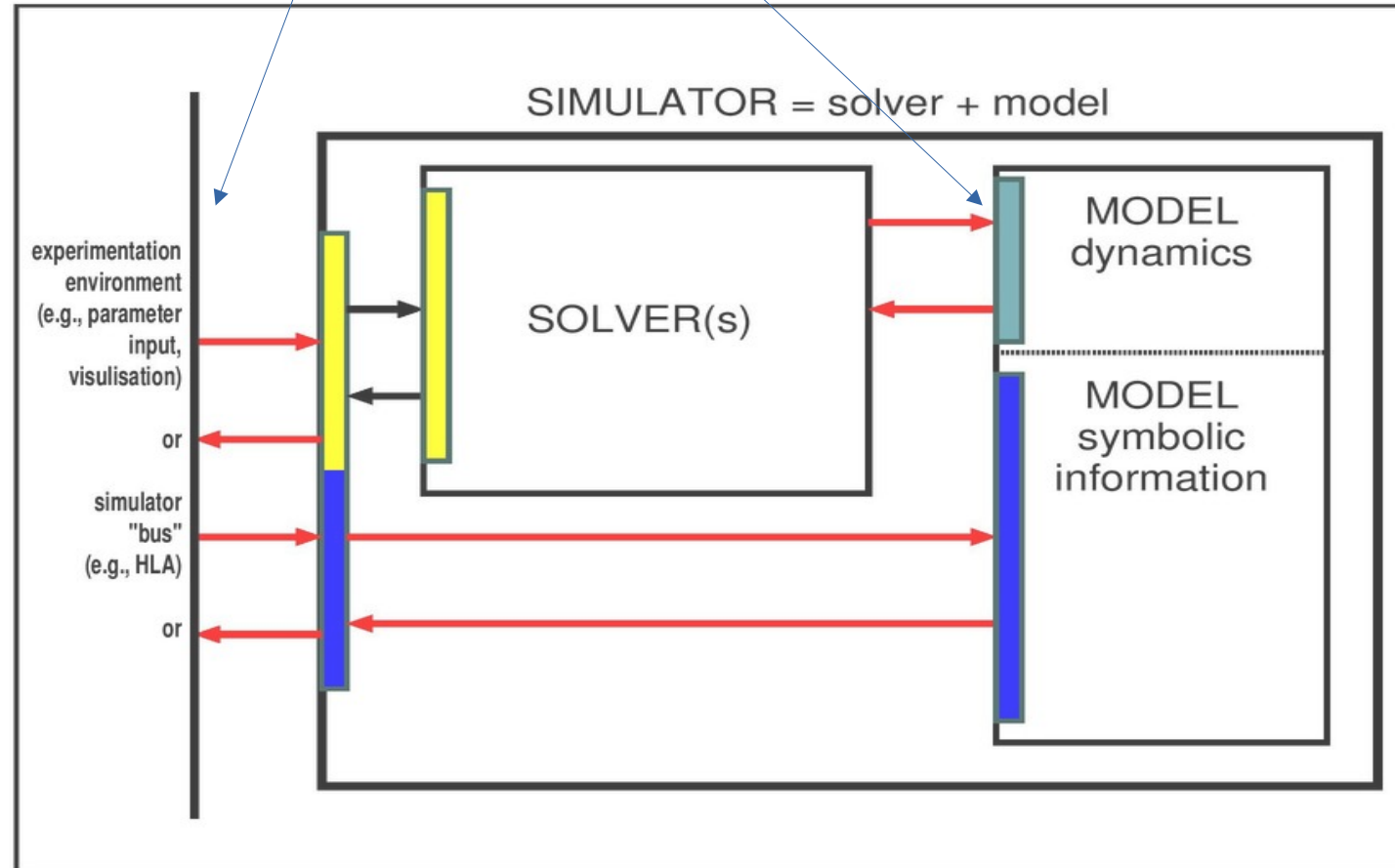
Deconstructing a Simulator: Functional Mockup Unit (FMU)



meaningful operational semantics (Models of Computation)



Interface: Simulator-Environment Interface: Model-Solver



DSblock

Martin Otter and Hilding Elmquist.
The DSblock interface for exchanging model components. Eurosime '95 Simulation Congress. pp. 505- 510. **1995**.

MSL-EXEC

Henk Vanhooren, Jurgen Meirlaen, Youri Amerlinck, Filip Claeys, Hans Vangheluwe, and Peter A. Vanrolleghem.
WEST: Modelling biological wastewater treatment. Journal of Hydroinformatics , 5(1):27--50, 2003.



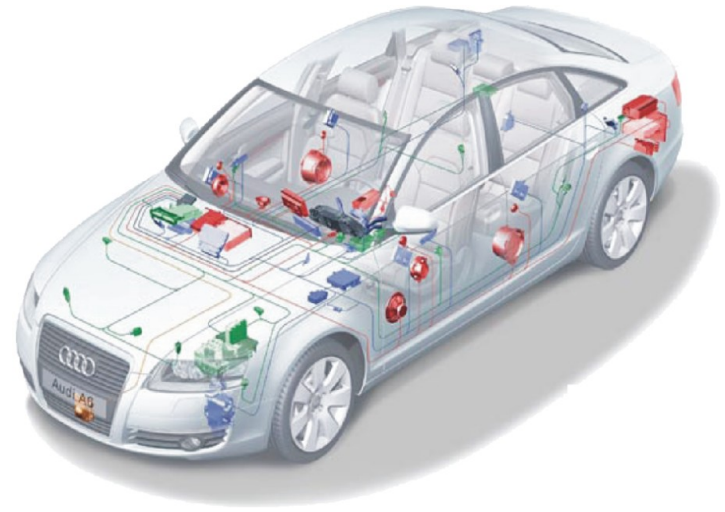
The leading standard to exchange dynamic simulation models

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 files, binaries and C code**, distributed as a **ZIP file**. It is supported by **180+ tools** and maintained as a Modelica Association Project.

[▶ Why FMI](#)[📄 Complete Package 3.0.1](#)[📄 Specification 3.0.1](#)[📄 Implementers' Guide](#)

Functional Mock-up Interface (FMI)

- **XML** + Binary Representation for Models
 - Standard
 - Modelling Tool Independent
 - +/- Black box ... (zero-delay info)



Symbolic information: FMU - XML

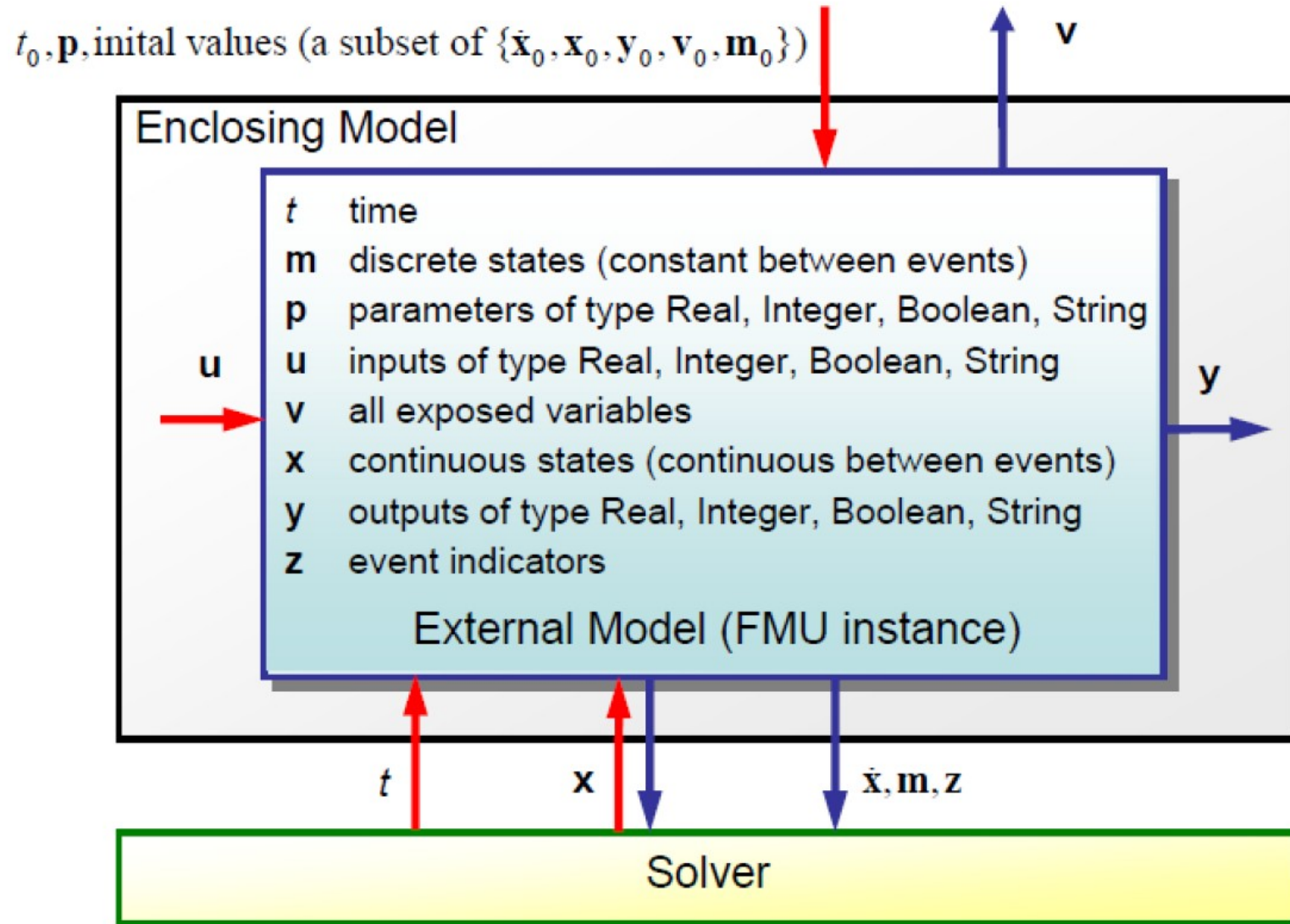
```
<?xml version="1.0" encoding="iso-8859-1"?>
<fmiModelDescription fmiVersion="1.0"
  modelName="BackEulerExmpl"
  modelIdentifier="BackEulerExmpl"
  guid="{7c4e810f-3da3-4a00-8276-176fa3c9f000}"
  numberOfContinuousStates="1"
  numberOfEventIndicators="0">
  <ModelVariables>
    <ScalarVariable
      name="Delay" valueReference="0">
      <Real start="0.0" fixed="true" />
    </ScalarVariable>
    <ScalarVariable
      name="stepSize" valueReference="1">
      <Real start="0.1" fixed="true" />
    </ScalarVariable>
    <ScalarVariable
      name="Product" valueReference="2">
      <Real start="0.0" fixed="true" />
    </ScalarVariable>
    <ScalarVariable
      name="Negator" valueReference="3">
      <Real start="0.0" fixed="true" />
    </ScalarVariable>
    <ScalarVariable
      name="Adder" valueReference="4">
      <Real start="0.0" fixed="true" />
    </ScalarVariable>
  </ModelVariables>
</fmiModelDescription>
```

```
fmi2Status fmi2DoStep(fmi2Component fc , fmi2Real currentCommPoint, fmi2Real commStepSize, fmi2Boolean
noPrevFMUState)
{
    FMUInstance* fi = (FMUInstance *)fc;
    fmi2Status simStatus = fmi2OK;
    printf("%s in fmi2DoStep()\n",fi->instanceName);
    fi->currentTime = currentCommPoint + commStepSize;
    printf("Motor_in: %f\n", fi->r[_motor_in]);
    printf("slave CBD_PART2 now at time: %f\n", fi->currentTime);

    fi->r[_position] = fi->r[_position] + fi->r[_velocity] * commStepSize;
    fi->r[_velocity] = fi->r[_velocity] + fi->r[_acceleration_after_friction] * commStepSize;
    fi->r[_friction] = fi->r[_velocity] * 5.81;
    fi->r[_motor_acceleration] = fi->r[_motor_in] * 40;
    fi->r[_acceleration_after_friction] = fi->r[_motor_acceleration] - fi->r[_friction];

    return simStatus;
}

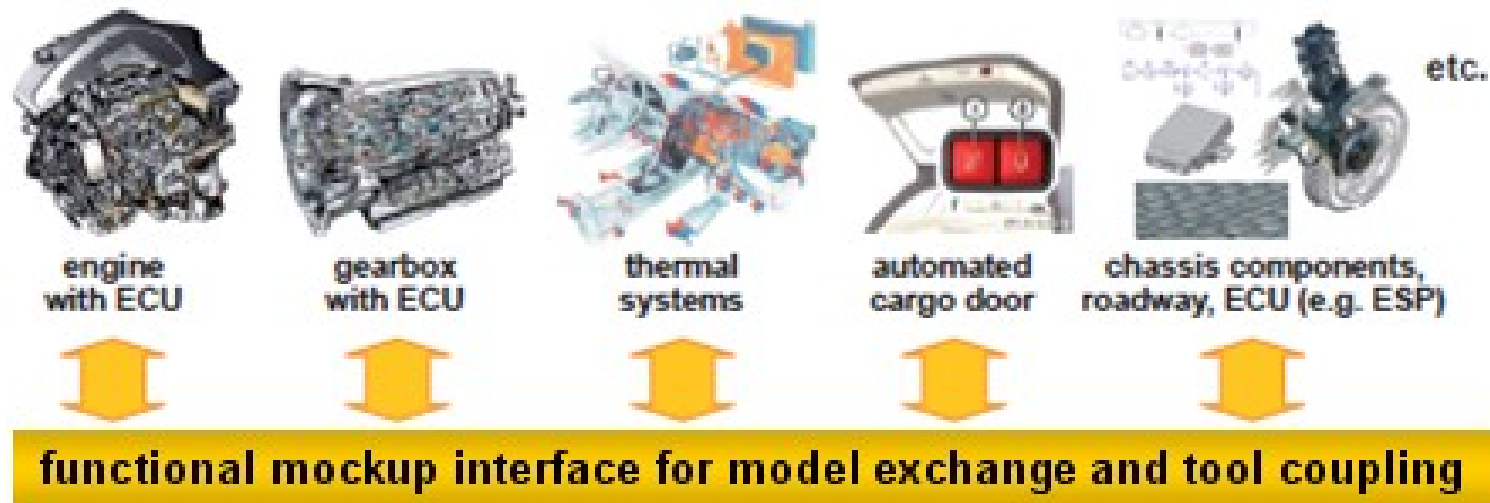
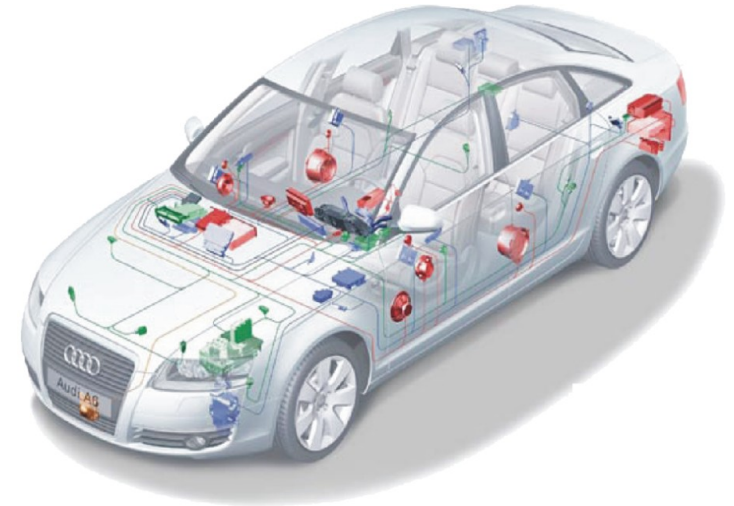
fmi2Status fmi2GetReal(fmi2Component fc, const fmi2ValueReference vr[], size_t nvr, fmi2Real value[])
{
    FMUInstance* comp = (FMUInstance *)fc;
    int i;
    for (i = 0; i < nvr; i++)
    {
        value[i] = comp->r[(vr[i])];
    }
    return fmi2OK;
}
```



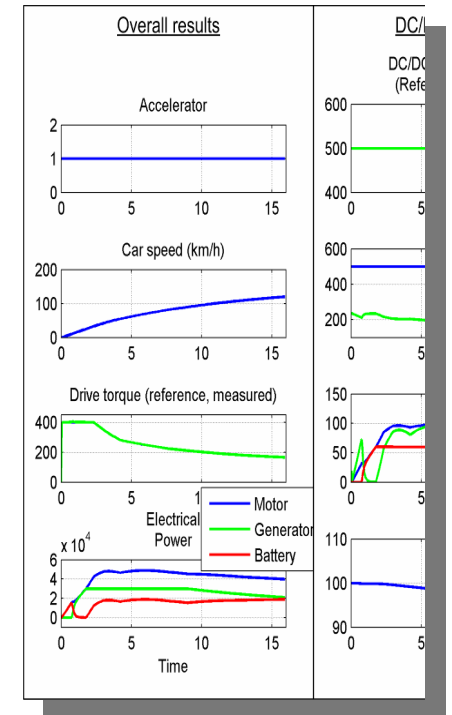
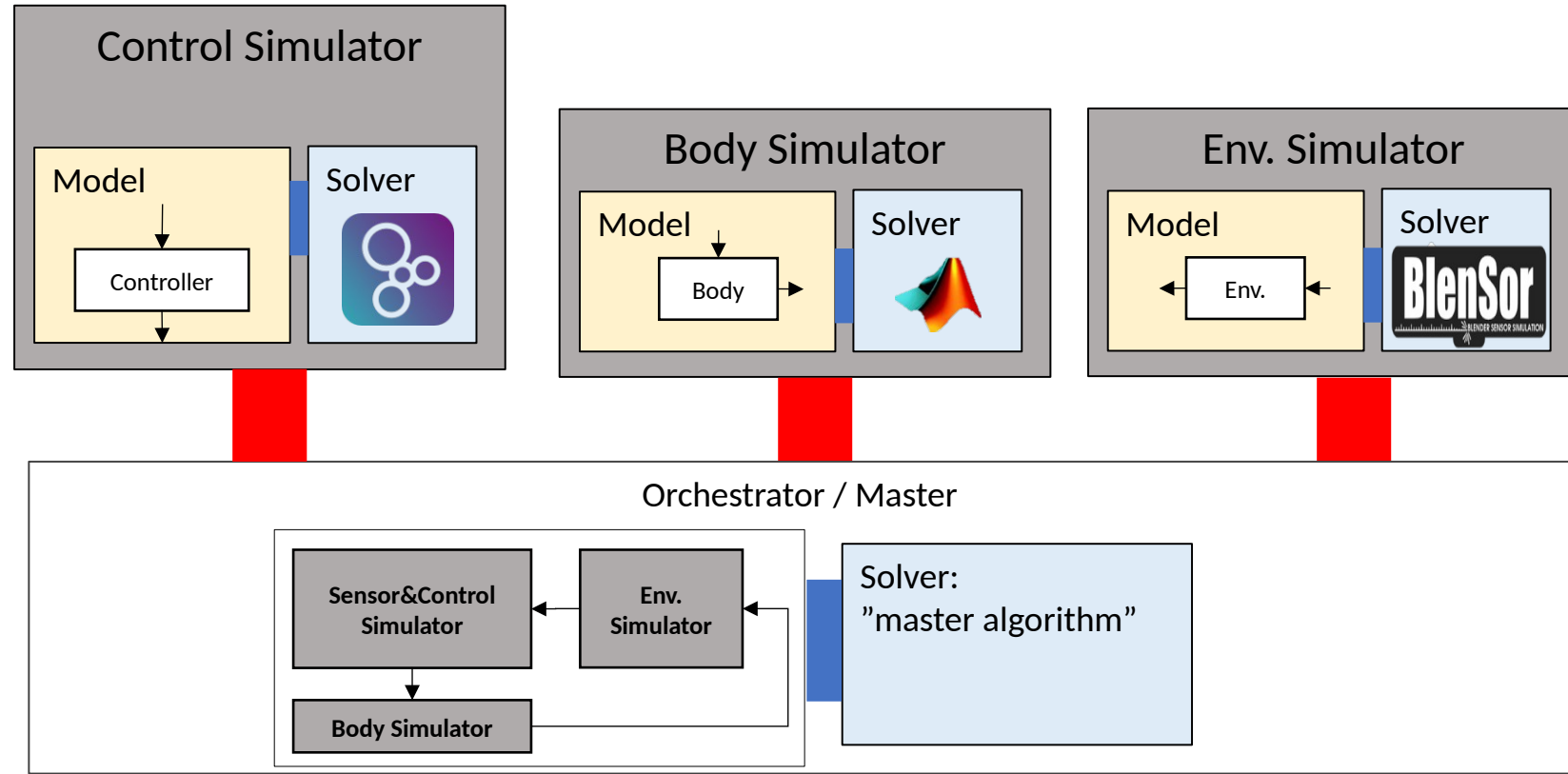
Function	FMI 2.0 for Co-Simulation									
	start, end	instantiated	Initialization Mode	stepComplete	stepInProgress	stepFailed	stepCanceled	terminated	error	fatal
fmi2GetTypesPlatform	x	x	x	x	x	x	x	x	x	
fmi2GetVersion	x	x	x	x	x	x	x	x	x	
fmi2SetDebugLogging		x	x	x	x	x	x	x	x	
fmi2Instantiate	x									
fmi2FreeInstance		x	x	x		x	x	x	x	
fmi2SetupExperiment		x								
fmi2EnterInitializationMode		x								
fmi2ExitInitializationMode			x							
fmi2Terminate				x		x				
fmi2Reset		x	x	x		x	x	x	x	
fmi2GetReal			2	x		8	7	x	7	
fmi2GetInteger			2	x		8	7	x	7	
fmi2GetBoolean			2	x		8	7	x	7	
fmi2GetString			2	x		8	7	x	7	
fmi2SetReal		1	3	6						
fmi2SetInteger		1	3	6						
fmi2SetBoolean		1	3	6						
fmi2SetString		1	3	6						
fmi2GetFMUstate		x	x	x		8	7	x	7	
fmi2SetFMUstate		x	x	x		x	x	x	x	
fmi2FreeFMUstate		x	x	x		x	x	x	x	
fmi2SerializedFMUstateSize		x	x	x		x	x	x	x	
fmi2SerializeFMUstate		x	x	x		x	x	x	x	
fmi2DeSerializeFMUstate		x	x	x		x	x	x	x	
fmi2GetDirectionalDerivative			x	x		8	7	x	7	
fmi2SetRealInputDerivatives		x	x	x						
fmi2GetRealOutputDerivatives				x		8	x	x	7	
fmi2DoStep				x						
fmi2CancelStep					x					
fmi2GetStatus				x	x	x		x		
fmi2GetRealStatus				x	x	x		x		
fmi2GetIntegerStatus				x	x	x		x		
fmi2GetBooleanStatus				x	x	x		x		
fmi2GetStringStatus				x	x	x		x		

Functional Mock-up Interface (FMI)

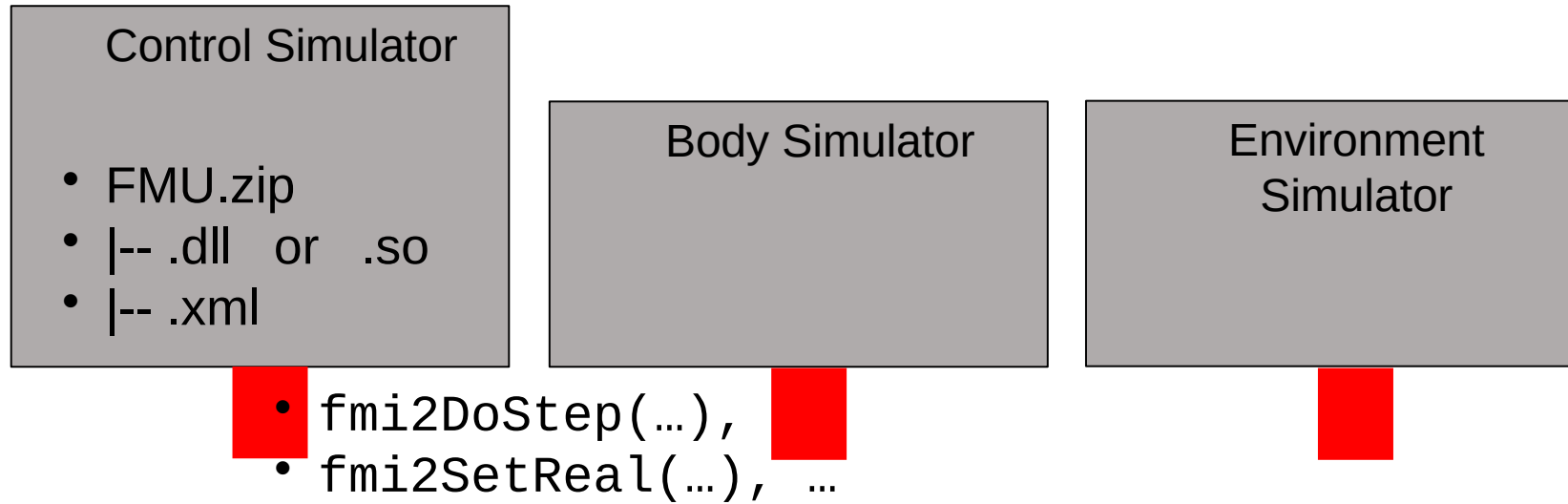
- **XML** + Binary Representation for Models
 - Standard
 - Modelling Tool Independent
 - +/- Black box ... (zero-delay info)
- Composed FMUs still need “orchestration”



co-simulation: how? (when IP protected “black box” simulators)

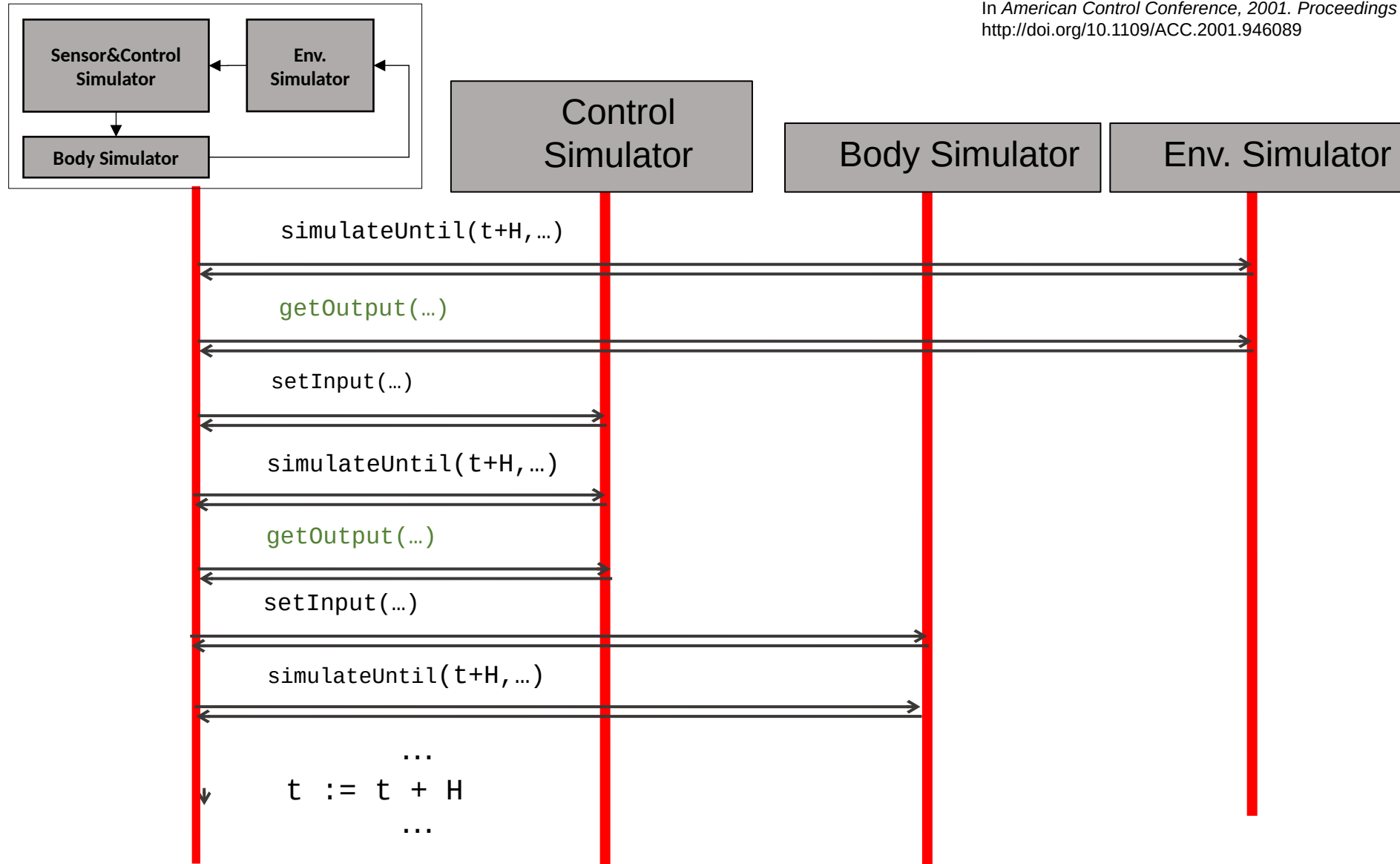


co-simulation: how? (when IP protected “black box” simulators)

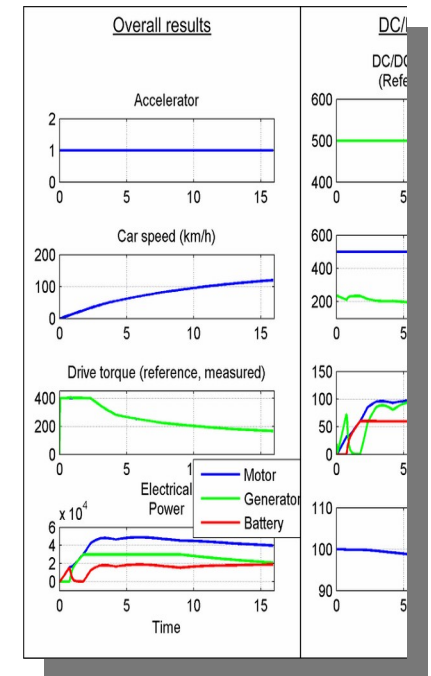


Co-simulation: how?

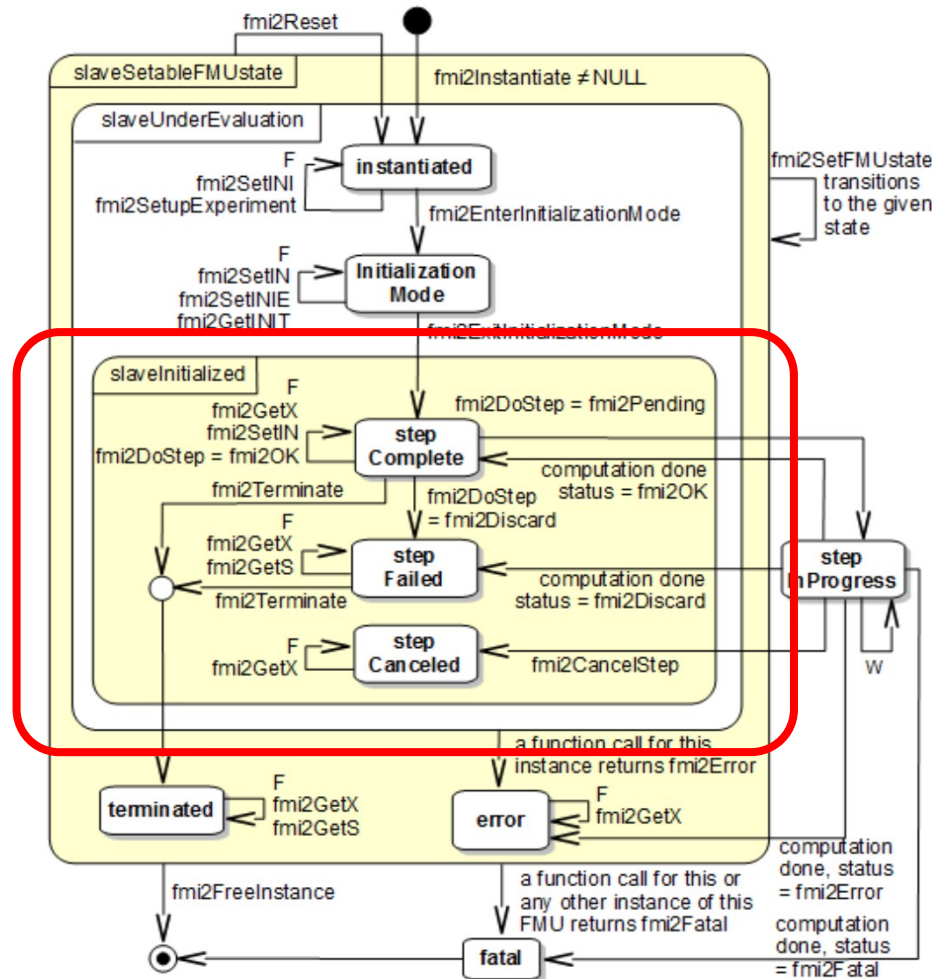
Gu, B., & Asada, H. H. (2001). Co-simulation of algebraically coupled dynamic subsystems. In *American Control Conference, 2001. Proceedings of the 2001* (Vol. 3, pp. 2273–2278 vol.3). <http://doi.org/10.1109/ACC.2001.946089>



H and h are other common notations for Δt



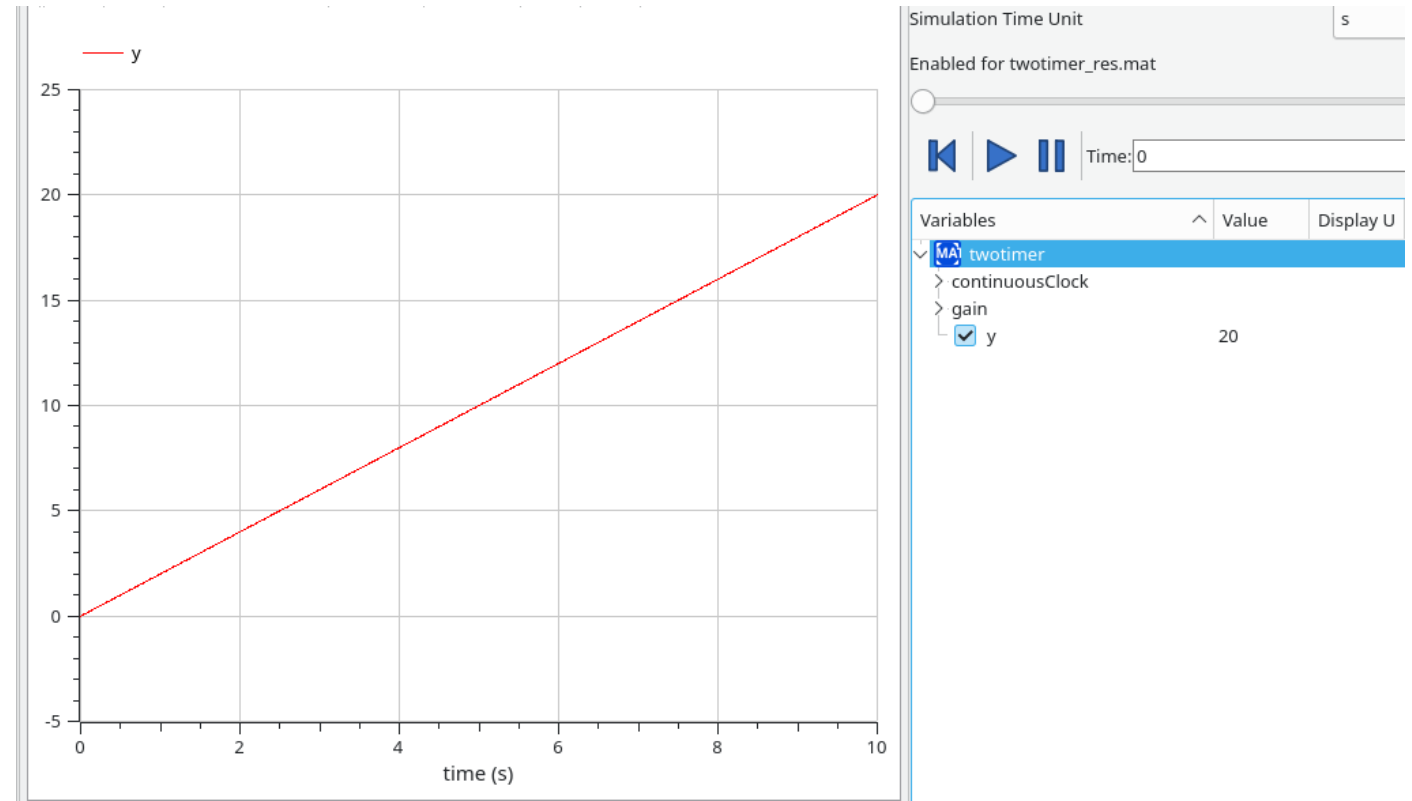
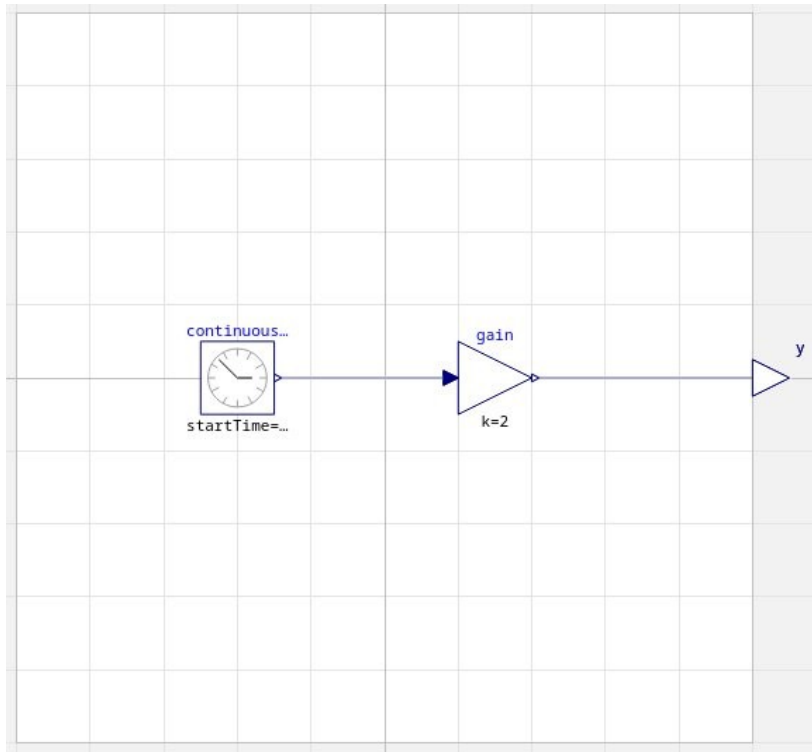
calling sequence Master-Slave (modelled as state machine)



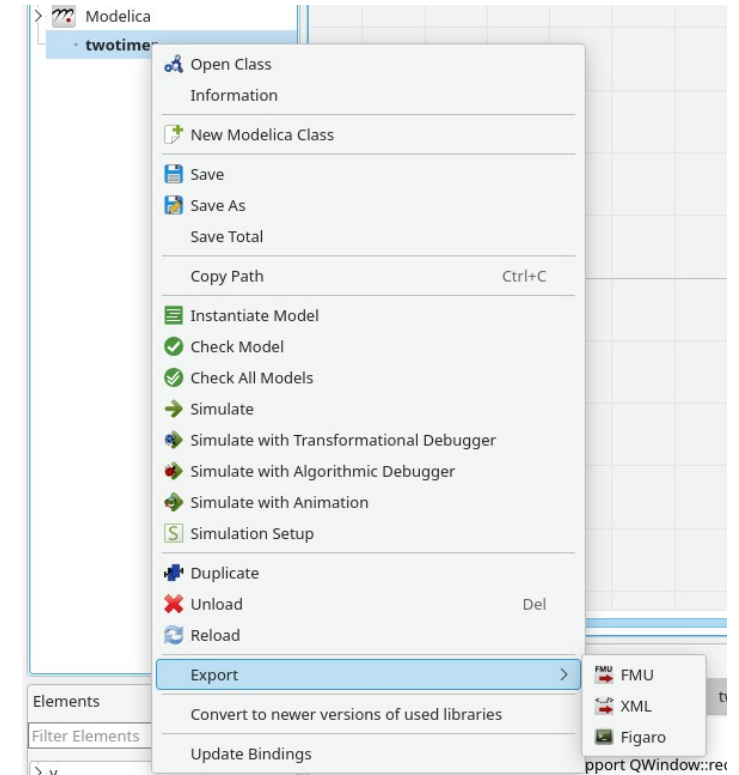
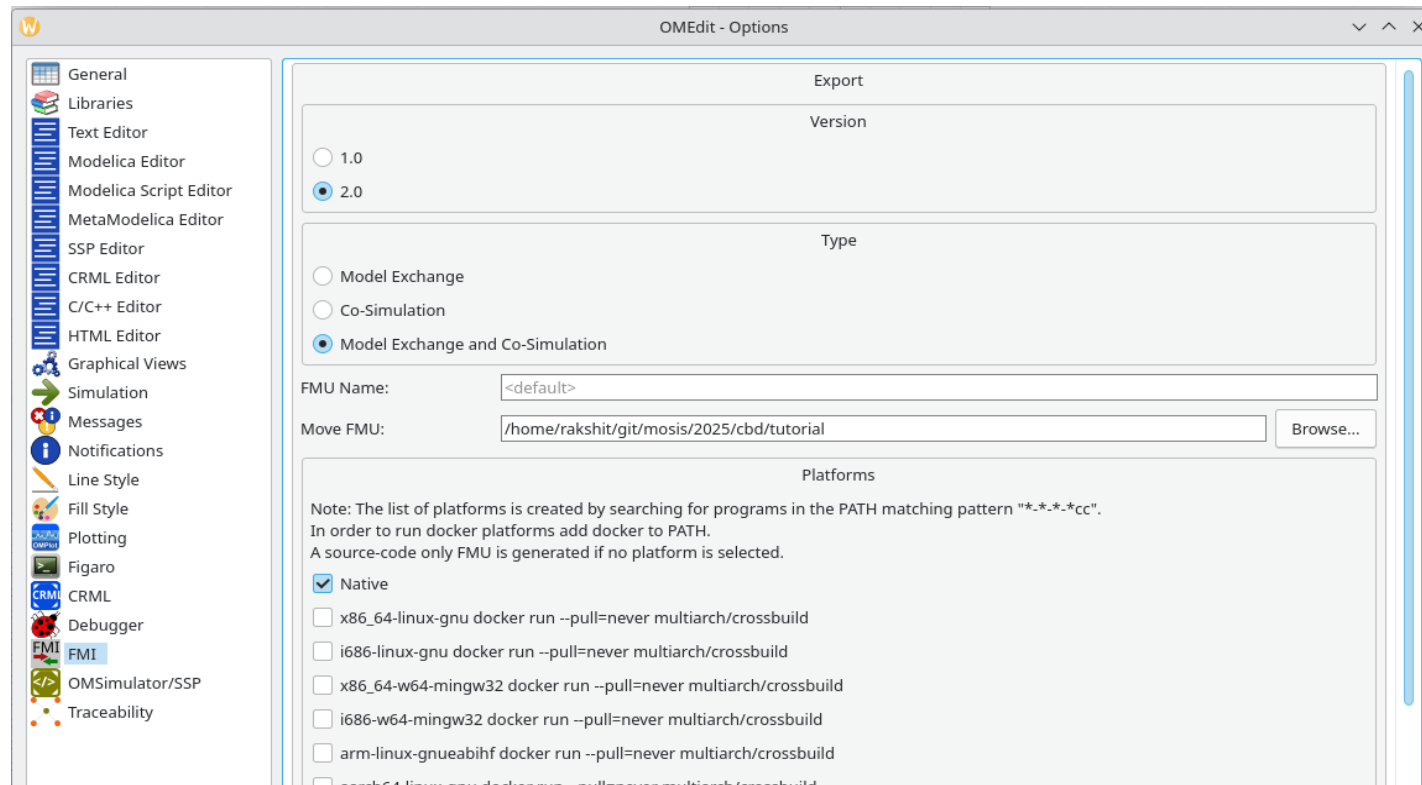
Simulation using a Functional Mockup Unit (FMU)

Simple example in OMEdit:

A block that outputs $2 \times \text{time}$



Exporting an FMU from OpenModelica



```
adding: sources/external_solvers/dgemm.c (deflated 75%)
adding: sources/external_solvers/s_copy.c (deflated 52%)
adding: sources/external_solvers/dgemv.c (deflated 71%)
adding: sources/external_solvers/clapack.h (deflated 91%)
adding: sources/external_solvers/dgetri.c (deflated 65%)
adding: sources/twotimer_06inz.c (deflated 69%)
adding: sources/twotimer_18spd.c (deflated 48%)
adding: modelDescription.xml (deflated 86%)
adding: binaries/ (stored 0%)
adding: binaries/linux64/ (stored 0%)
adding: binaries/linux64/twotimer.so (deflated 61%)
```

The FMU is generated at: /home/rakshit/git/mosis/2025/cbd/tutorial/twotimer.fmu

twotimer.fmu — Ark							
Extract Add Files Find Preview Open in External Application Remove from Archive							
Name	Original Size	Compressed Size	Mode	CRC checksum	Method	Date	
binaries	596.5 KiB		drwxr-xr-x		Store	10/12/25 4:46 PM	
resources	0 B		drwx-----		Store	10/12/25 4:46 PM	
sources	3.1 MiB		drwx-----		Store	10/12/25 4:46 PM	
modelDescription.xml	13.7 KiB	1.9 KiB	-rw-r--r--	D816F8D9	Deflate	10/12/25 4:46 PM	

twotimer.fmu

adding: sources/external_solvers/dgemm.c (deflated 75%)
 adding: sources/external_solvers/s_copy.c (deflated 52%)
 adding: sources/external_solvers/dgemv.c (deflated 71%)
 adding: sources/external_solvers/clapack.h (deflated 91%)
 adding: sources/external_solvers/dgetri.c (deflated 65%)
 adding: sources/twotimer_06inz.c (deflated 69%)
 adding: sources/twotimer_18spd.c (deflated 48%)
 adding: modelDescription.xml (deflated 86%)
 adding: binaries/ (stored 0%)
 adding: binaries/linux64/ (stored 0%)
 adding: binaries/linux64/twotimer.so (deflated 61%)

The FMU is generated at: </home/rakshit/git/mosis/2025/cbd/tutorial/twotimer.fmu>

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <fmiModelDescription
3   fmiVersion="2.0"
4   modelName="twotimer"
5   guid="{c4eff564-7c09-40aa-b7e6-7718b4007b6f}"
6   description=""
7   version=""
8   generationTool="OpenModelica Compiler OpenModelica 1.25.4"
9   generationDateAndTime="2025-10-12T16:46:49Z"
10  variableNamingConvention="structured"
11  numberOfEventIndicators="1">

```

Source files for:

Model exchange

Co-simulation

```
12 <ModelExchange
13   modelIdentifier="twotimer"
14   needsExecutionTool="false"
15   completedIntegratorStepNotNeeded="false"
16   canBeInstantiatedOnlyOncePerProcess="false"
17   canNotUseMemoryManagementFunctions="false"
18   canGetAndSetFMUstate="false"
19   canSerializeFMUstate="false"
20   providesDirectionalDerivative="false">
21   <SourceFiles>
22     <File name="twotimer.c" />
23     <File name="twotimer_functions.c" />
24     <File name="twotimer_records.c" />
25     <File name="twotimer_01exo.c" />
26     <File name="twotimer_02nls.c" />
27     <File name="twotimer_03lsy.c" />
28     <File name="twotimer_04set.c" />
29     <File name="twotimer_05evt.c" />
30     <File name="twotimer_06inz.c" />
31     <File name="twotimer_07dly.c" />
32     <File name="twotimer_08bnd.c" />
33     <File name="twotimer_09alg.c" />
34     <File name="twotimer_10asr.c" />
35     <File name="twotimer_11mix.c" />
36     <File name="twotimer_12jac.c" />
37     <File name="twotimer_13opt.c" />
38     <File name="twotimer_14lnz.c" />
39     <File name="twotimer_15syn.c" />
40     <File name="twotimer_16dae.c" />
41     <File name="twotimer_17inl.c" />
42     <File name="twotimer_18spd.c" />
43     <File name="twotimer_init_fmu.c" />
44     <File name="twotimer_FMU.c" />
45     <File name="./gc/memory_pool.c" />
46     <File name="./gc/omc_gc.c" />
47     <File name="./util/base_array.c" />
48     <File name="./util/boolean_array.c" />
49     <File name="./util/context.c" />
50     <File name="./util/division.c" />
51     <File name="./util/doubleEndedList.c" />
52     <File name="./util/generic_array.c" />
53     <File name="./util/index_spec.c" />
54     <File name="./util/integer_array.c" />
55     <File name="./util/list.c" />
56     <File name="./util/modelica_string_lit.c" />
```

```
128 <File name="ModelicaExternalC/ModelicaIO.c" />
129 <File name="ModelicaExternalC/ModelicaStandardC/ModelicaStandardC.c" />
130 <File name="ModelicaExternalC/sprintf.c" />
131 </SourceFiles>
132 </ModelExchange>
133 <CoSimulation
134   modelIdentifier="twotimer"
135   needsExecutionTool="false"
136   canHandleVariableCommunicationStepSize="true"
137   canInterpolateInputs="true"
138   maxOutputDerivativeOrder="1"
139   canRunAsynchronously="false"
140   canBeInstantiatedOnlyOncePerProcess="false"
141   canNotUseMemoryManagementFunctions="false"
142   canGetAndSetFMUstate="false"
143   canSerializeFMUstate="false"
144   providesDirectionalDerivative="false">
145   <SourceFiles>
146     <File name="twotimer.c" />
147     <File name="twotimer_functions.c" />
148     <File name="twotimer_records.c" />
149     <File name="twotimer_01exo.c" />
150     <File name="twotimer_02nls.c" />
151     <File name="twotimer_03lsy.c" />
152     <File name="twotimer_04set.c" />
153     <File name="twotimer_05evt.c" />
154     <File name="twotimer_06inz.c" />
155     <File name="twotimer_07dly.c" />
156     <File name="twotimer_08bnd.c" />
157     <File name="twotimer_09alg.c" />
158     <File name="twotimer_10asr.c" />
159     <File name="twotimer_11mix.c" />
160     <File name="twotimer_12jac.c" />
161     <File name="twotimer_13opt.c" />
162     <File name="twotimer_14lnz.c" />
163     <File name="twotimer_15syn.c" />
164     <File name="twotimer_16dae.c" />
165     <File name="twotimer_17inl.c" />
166     <File name="twotimer_18spd.c" />
167     <File name="twotimer_init_fmu.c" />
168     <File name="twotimer_FMU.c" />
169     <File name="./gc/memory_pool.c" />
170     <File name="./gc/omc_gc.c" />
```

```
adding: sources/external_solvers/dgemm.c (deflated 75%)
adding: sources/external_solvers/s_copy.c (deflated 52%)
adding: sources/external_solvers/dgemv.c (deflated 71%)
adding: sources/external_solvers/clapack.h (deflated 91%)
adding: sources/external_solvers/dgetri.c (deflated 65%)
adding: sources/twotimer_06inz.c (deflated 69%)
adding: sources/twotimer_18spd.c (deflated 48%)
```

```
adding: modelDescription.xml (deflated 86%)
adding: binaries/ (stored 0%)
adding: binaries/linux64/ (stored 0%)
adding: binaries/linux64/twotimer.so (deflated 61%)
```

The FMU is generated at: /home/rakshit/git/mosis/2025/cbd/tutorial/twotimer.fmu

```

257 <UnitDefinitions>
258   <Unit name="1">
259     <BaseUnit />
260   </Unit>
261   <Unit name="s">
262     <BaseUnit s="1" />
263   </Unit>
264 </UnitDefinitions>

```

Unit definitions

```

265 <LogCategories>
266   <Category name="logEvents" />
267   <Category name="logSingularLinearSystems" />
268   <Category name="logNonlinearSystems" />
269   <Category name="logDynamicStateSelection" />
270   <Category name="logStatusWarning" />
271   <Category name="logStatusDiscard" />
272   <Category name="logStatusError" />
273   <Category name="logStatusFatal" />
274   <Category name="logStatusPending" />
275   <Category name="logAll" />
276   <Category name="logFmi2Call" />
277 </LogCategories>

```

Log levels

Default experiment parameters

```

278 <DefaultExperiment startTime="0.0" stopTime="1.0" tolerance="1e-6" stepSize="0.002"/>

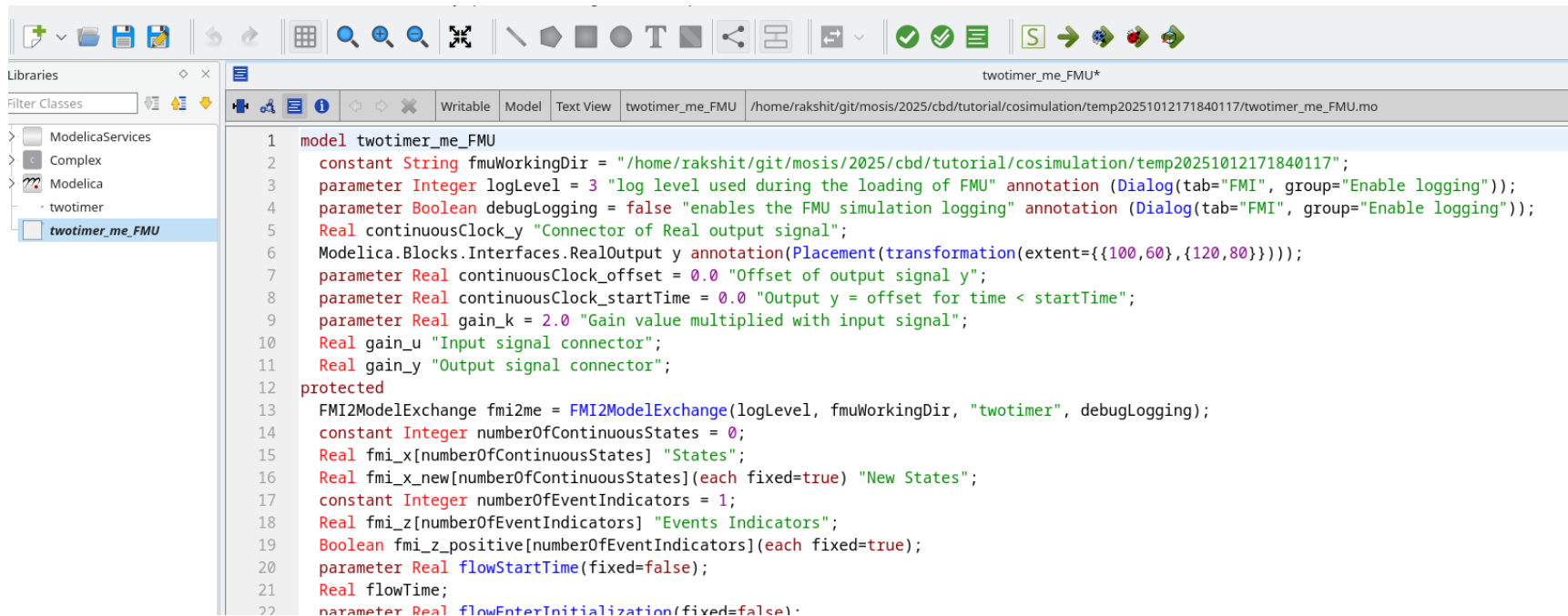
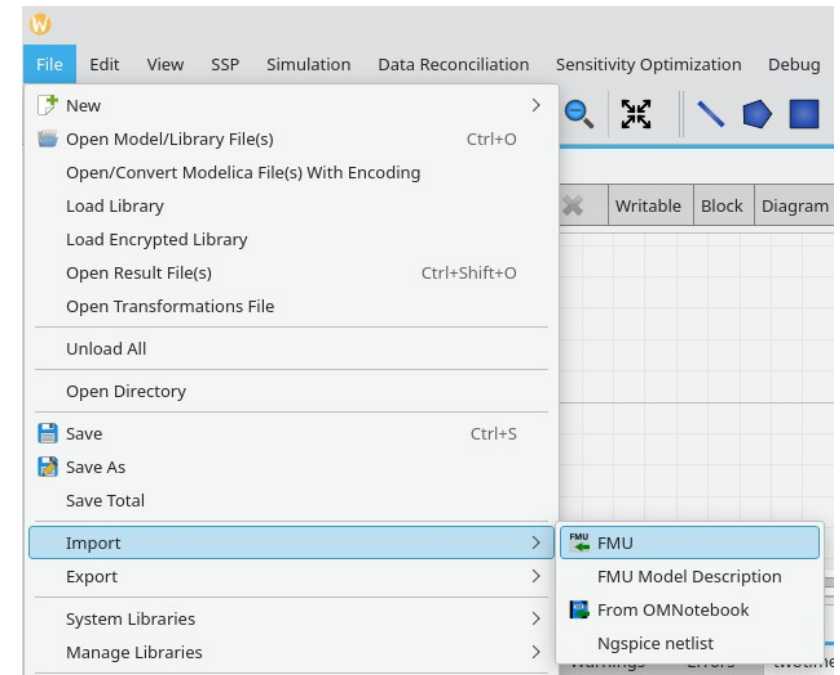
```

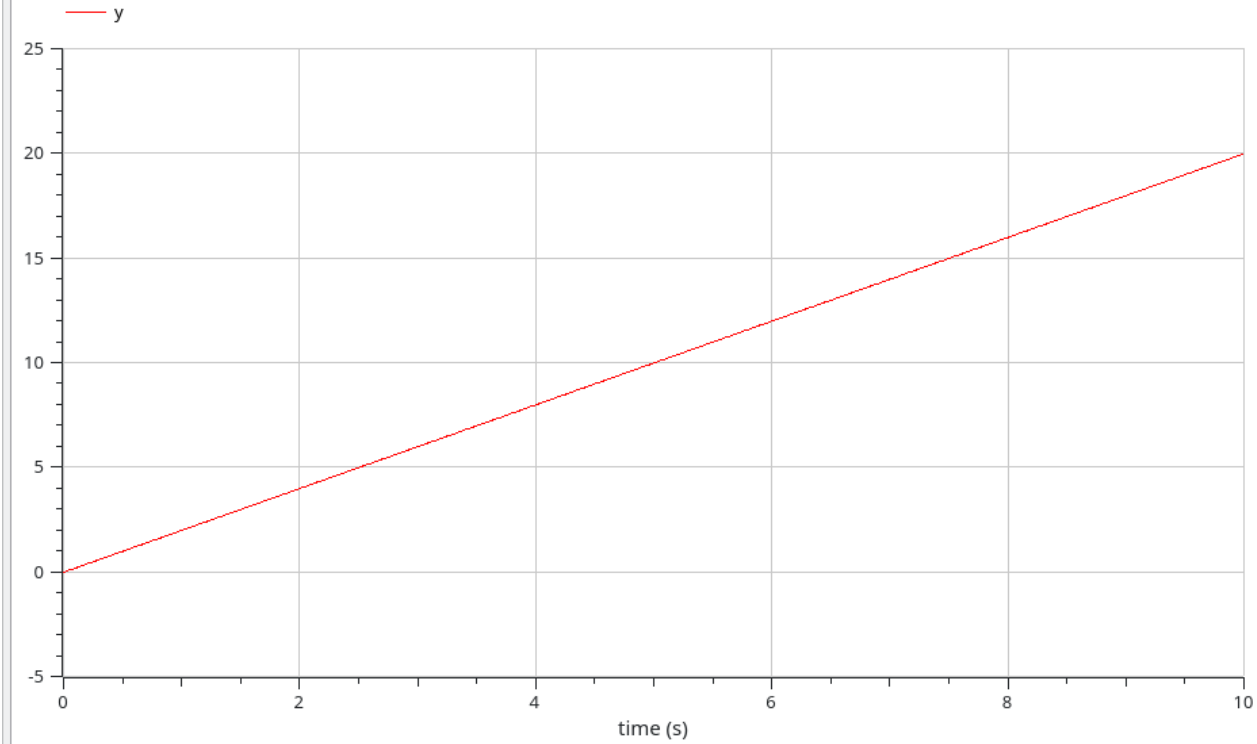
Model variables and structure

```
<ModelVariables>
<!-- Index of variable = "1" -->
<ScalarVariable
  name="continuousClock.y"
  valueReference="0"
  description="Connector of Real output signal"
  >
  <Real/>
</ScalarVariable>
<!-- Index of variable = "2" -->
<ScalarVariable
  name="y"
  valueReference="1"
  causality="output"
  >
  <Real/>
</ScalarVariable>
<!-- Index of variable = "3" -->
<ScalarVariable
  name="continuousClock.offset"
  valueReference="2"
  description="Offset of output signal y"
  variability="fixed"
  causality="parameter"
  >
  <Real start="0.0"/>
</ScalarVariable>
<!-- Index of variable = "4" -->
<ScalarVariable
  name="continuousClock.startTime"
  valueReference="3"
  description="Output y = offset for time &lt; startTime"
  variability="fixed"
  causality="parameter"
  >
  <Real start="0.0" unit="s"/>
</ScalarVariable>
<!-- Index of variable = "5" -->
<ScalarVariable
  name="gain.k"
  valueReference="4"
  description="Gain value multiplied with input signal"
  variability="fixed"
  causality="parameter"
  >
  <Real start="2.0" unit="1"/>
</ScalarVariable>
<!-- Index of variable = "6" -->
```

```
</ModelVariables>
<ModelStructure>
  <Outputs>
    <Unknown index="2" dependencies="" dependenciesKind="" />
  </Outputs>
  <InitialUnknowns>
    <Unknown index="2" dependencies="3 4 5" dependenciesKind="dependent dependent dependent" />
  </InitialUnknowns>
</ModelStructure>
</fmiModelDescription>
```

Importing an FMU in OpenModelica





Simulation time unit: s

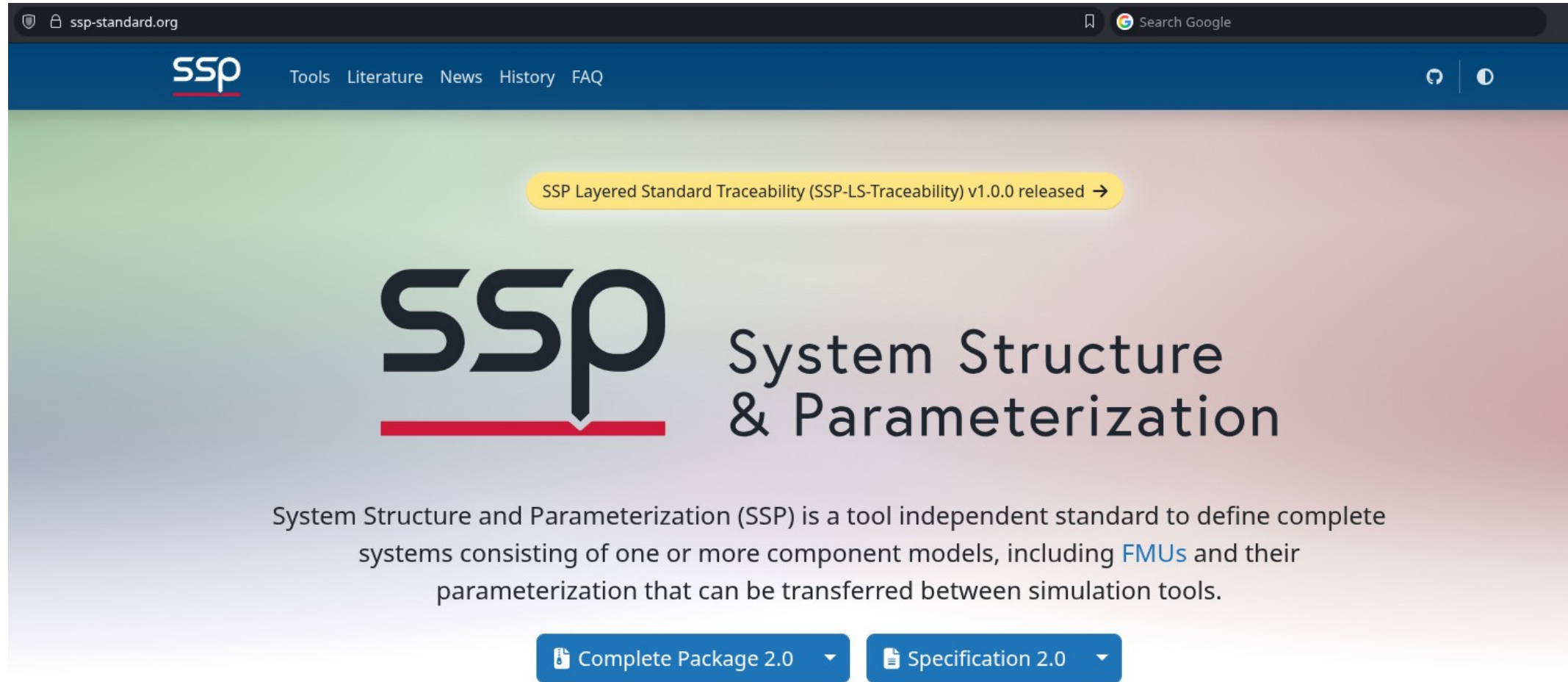
Enabled for twotimer_me_FMU_res.mat

Time: 0 Speed: 1

Variables	Value	Display U	Description
MA twotimer_me_FMU			
☐ continuousClock_offset	0.0		Offset of output signal y
☐ continuousC...k_startTime	0.0		Output y = offset for time < startTime
☐ continuousClock_y	10		Connector of Real output signal
☐ debugLogging	false		enables the FMU simulation logging
☐ gain_k	2.0		Gain value multiplied with input signal
☐ gain_u	10		Input signal connector
☐ gain_y	20		Output signal connector
☐ logLevel	3		log level used during the loading of FMU
☒ y	20		

What if we have **multiple** connected FMUs?

Co-simulate!



The screenshot shows the homepage of the ssp-standard.org website. The browser's address bar displays 'ssp-standard.org'. The website's header is dark blue with the 'ssp' logo on the left and navigation links for 'Tools', 'Literature', 'News', 'History', and 'FAQ' in the center. On the right side of the header are icons for a search and a user profile. A yellow banner below the header announces 'SSP Layered Standard Traceability (SSP-LS-Traceability) v1.0.0 released' with a right-pointing arrow. The main content area features a large 'ssp' logo with a red underline, followed by the text 'System Structure & Parameterization'. Below this, a paragraph states: 'System Structure and Parameterization (SSP) is a tool independent standard to define complete systems consisting of one or more component models, including FMUs and their parameterization that can be transferred between simulation tools.' At the bottom of the main content area are two blue buttons: 'Complete Package 2.0' and 'Specification 2.0', each with a document icon and a dropdown arrow.

ssp-standard.org

ssp

Tools Literature News History FAQ

SSP Layered Standard Traceability (SSP-LS-Traceability) v1.0.0 released →

ssp System Structure & Parameterization

System Structure and Parameterization (SSP) is a tool independent standard to define complete systems consisting of one or more component models, including FMUs and their parameterization that can be transferred between simulation tools.

Complete Package 2.0 Specification 2.0

<https://ssp-standard.org>

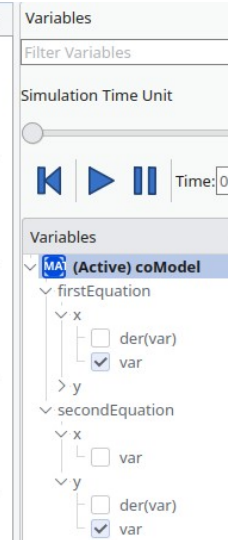
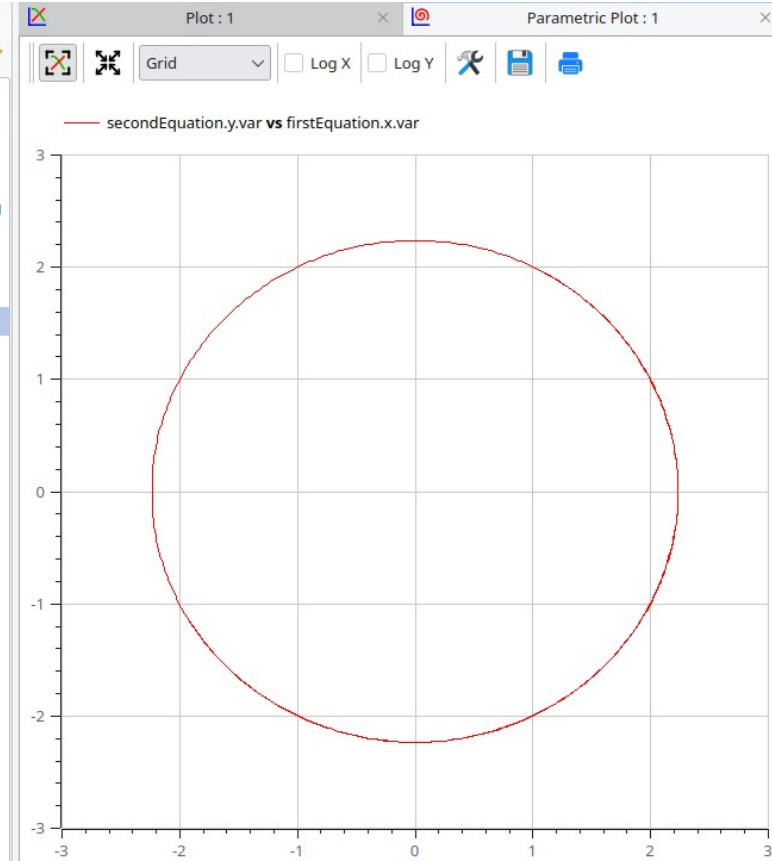
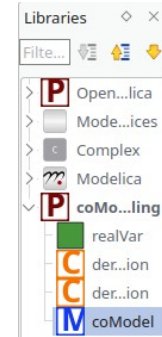
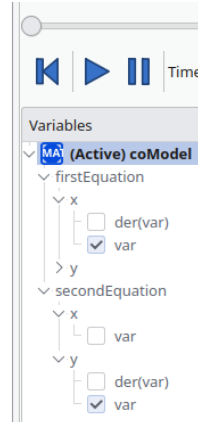
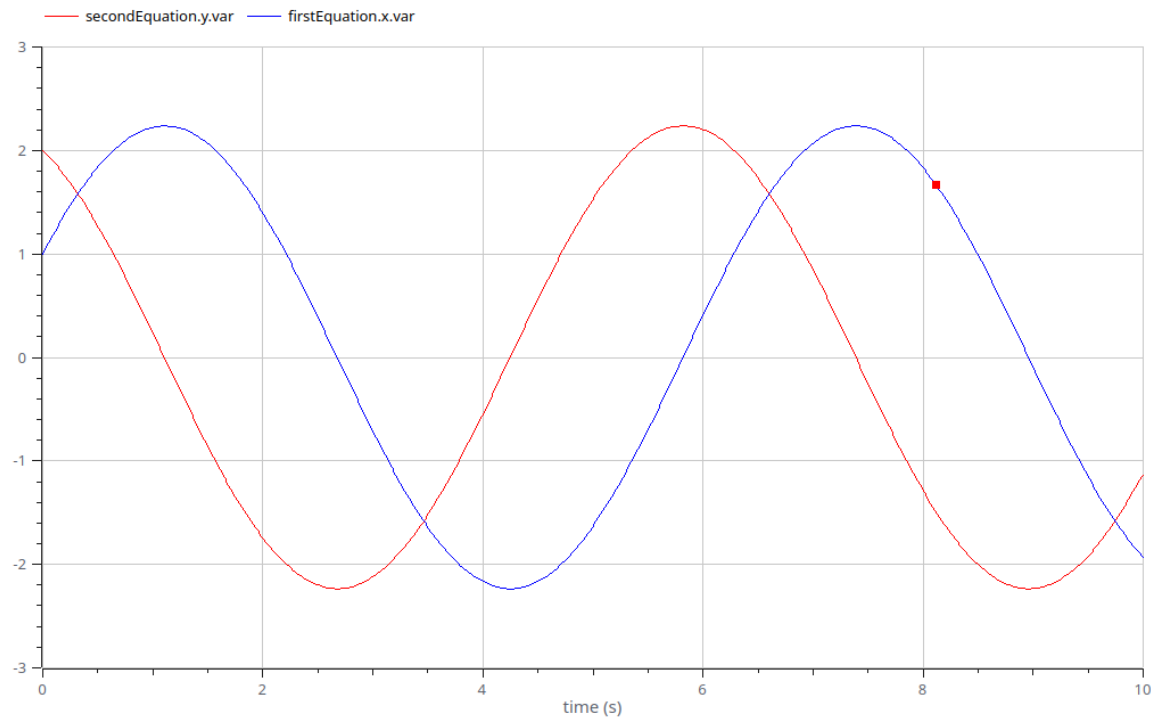
Simple example in OMEdit: circle test, (flattened, optimized and) **simulated**

```
1 package coModelling
2
3 connector realVar
4   Real var;
5 end realVar;
6
7 class derEquation
8   realVar x;
9   realVar y;
10 equation
11   der(x.var) = y.var;
12 end derEquation;
13
14 class derMinEquation
15   realVar x;
16   realVar y;
17 equation
18   der(y.var) = -x.var;
19 end derMinEquation;
20
21 model coModel
22   derEquation firstEquation;
23   derMinEquation secondEquation;
24 initial equation
25   firstEquation.x.var = 1;
26   secondEquation.y.var = 2;
27 equation
28   connect(firstEquation.x, secondEquation.x);
29   connect(firstEquation.y, secondEquation.y);
30 end coModel;
31
32 end coModelling;
```

1. model each equation (in acausal form)
2. specify the connection topology in Modelica
2. flatten and optimize the model with the Modelica compiler
3. compile and simulate

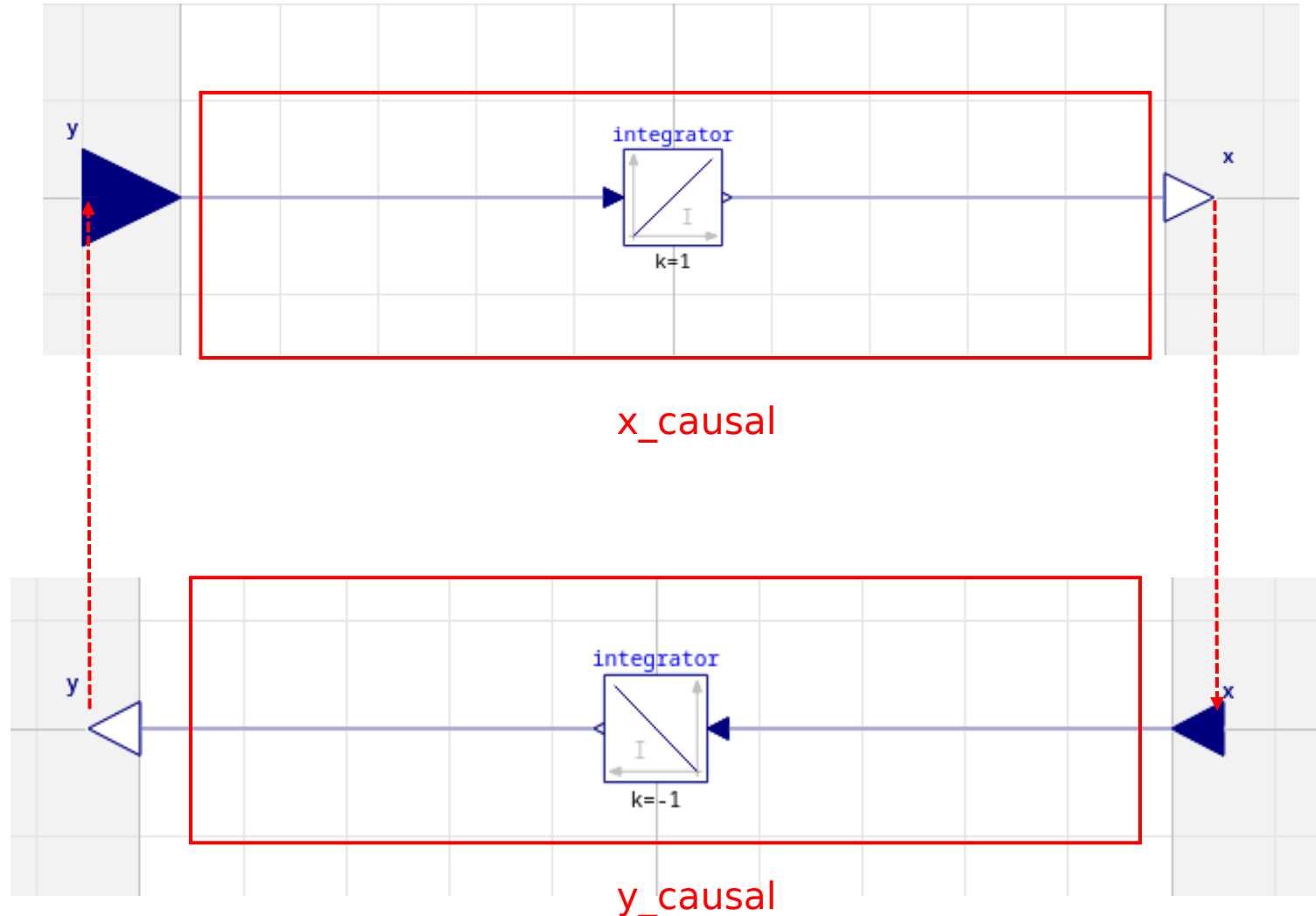
```
1 class coModelling.coModel
2   Real firstEquation.x.var;
3   Real firstEquation.y.var;
4   Real secondEquation.x.var;
5   Real secondEquation.y.var;
6 initial equation
7   firstEquation.x.var = 1.0;
8   secondEquation.y.var = 2.0;
9 equation
10   firstEquation.x.var = secondEquation.x.var;
11   firstEquation.y.var = secondEquation.y.var;
12   der(firstEquation.x.var) = firstEquation.y.var;
13   der(secondEquation.y.var) = -secondEquation.x.var;
14 end coModelling.coModel;
```

simulate as a single model (co-modelling, “white box”)



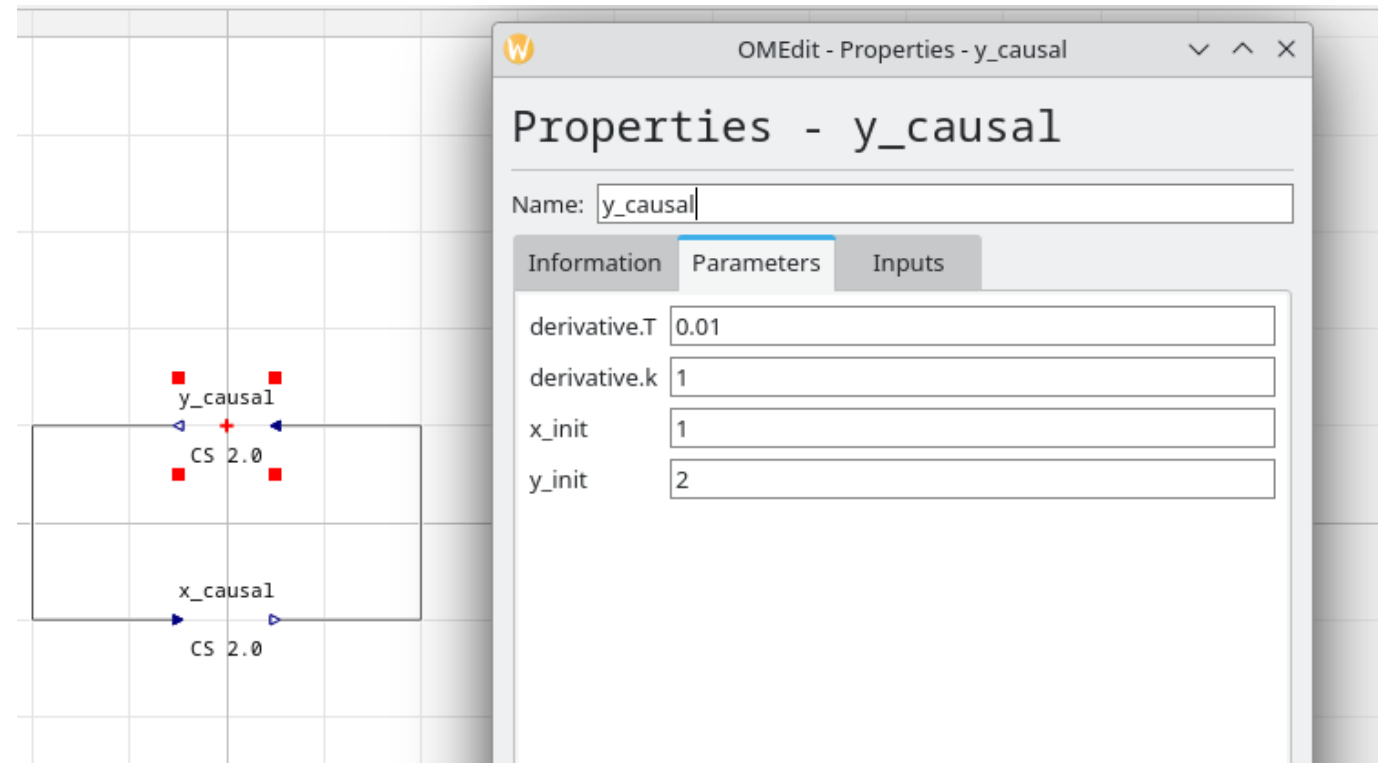
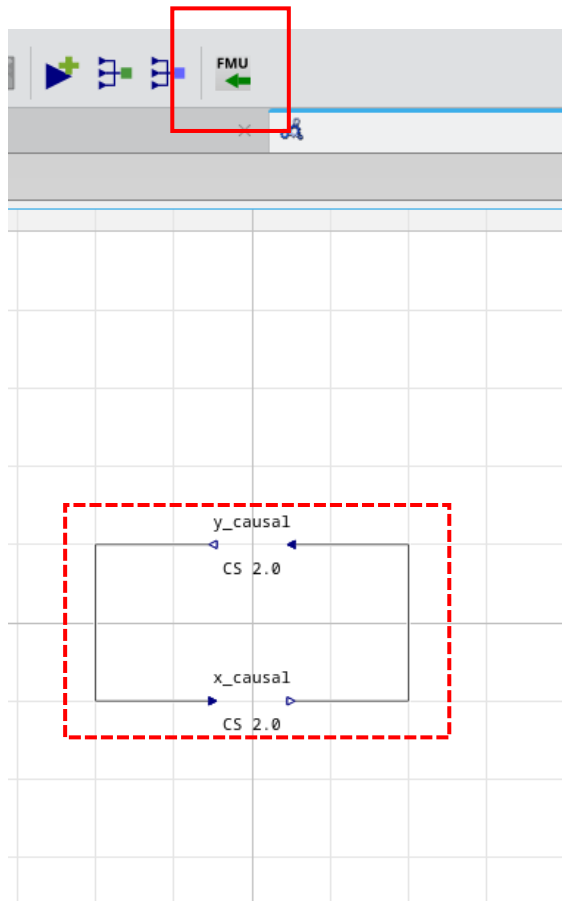
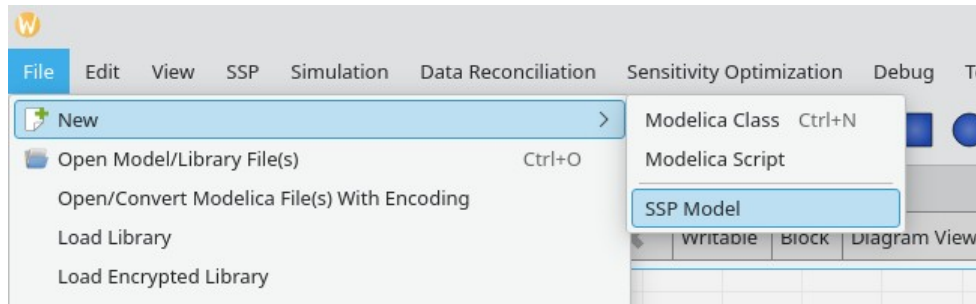
Simple example in OMEdit: circle test, (FMI/SSP) **co-simulated**

1. model each equation (in causal* form)
2. export each equation to an FMU
2. specify the connection topology in SSP
3. co-simulate using OMSimulator



*note: causal (block with input/output, but with equation section) $\neq$ ordered statements (algorithm section in a function block)

Simple example in OMEdit: circle test, (FMI/SSP) **co-simulated**



Simple example in OMEdit: circle test, (FMI/SSP) **co-simulated**

OMSimulator <https://github.com/OpenModelica/OMSimulator>

license OSMC-PL  28%

The OpenModelica FMI & SSP-based co-simulation environment.

Branch Overview

- **master** : Development branch for active, unstable updates.
- **maintenance/v2.1** : Stable branch for version 2.1 with patch updates (latest release: **v2.1.3**).
- **maintenance/v2.0** : Stable branch for version 2.0 with patch updates (latest release: **v2.0.1**).

Overview

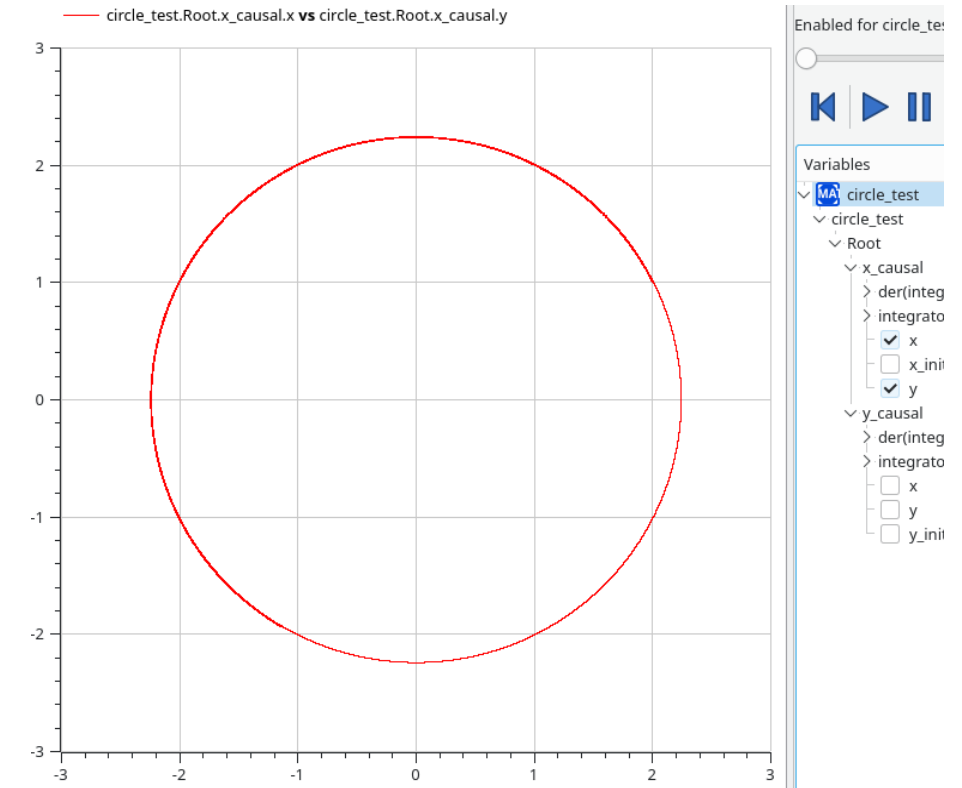
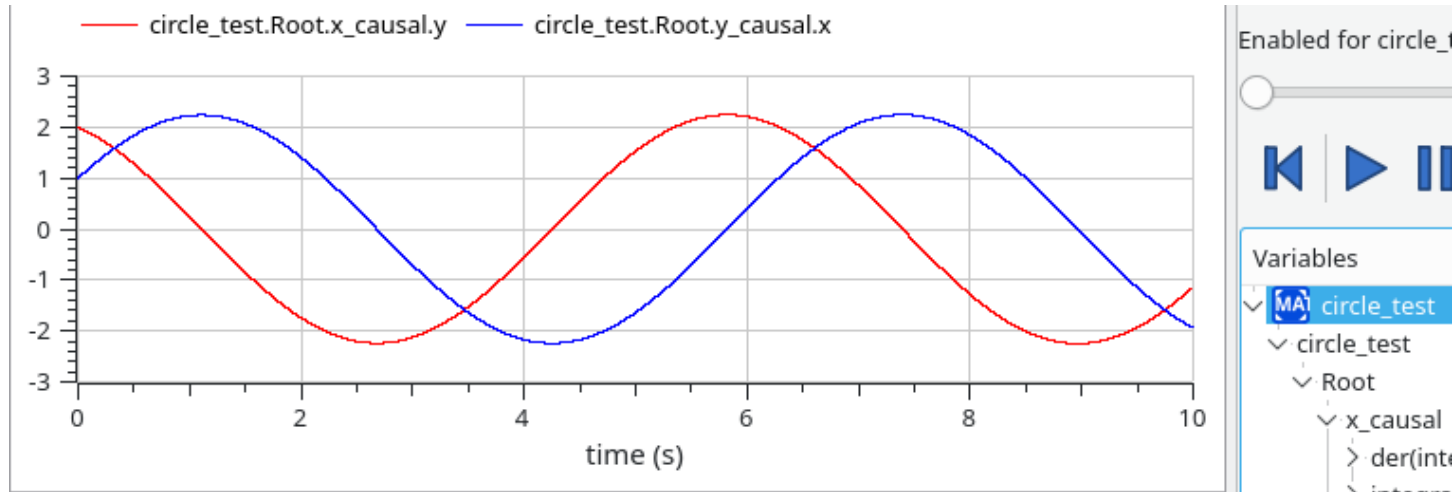
OMSimulator can be used as:

- **Stand-alone application**
- **C library** for integration into custom applications
- **Python package** (`pip install OMSimulator`)
- **OMSimulatorGui** -- a lightweight, standalone PySide6 GUI for editing and simulating SSP models, built directly on the Python package with no `omc` dependency. See [its own README](#) or the [User's Guide](#).

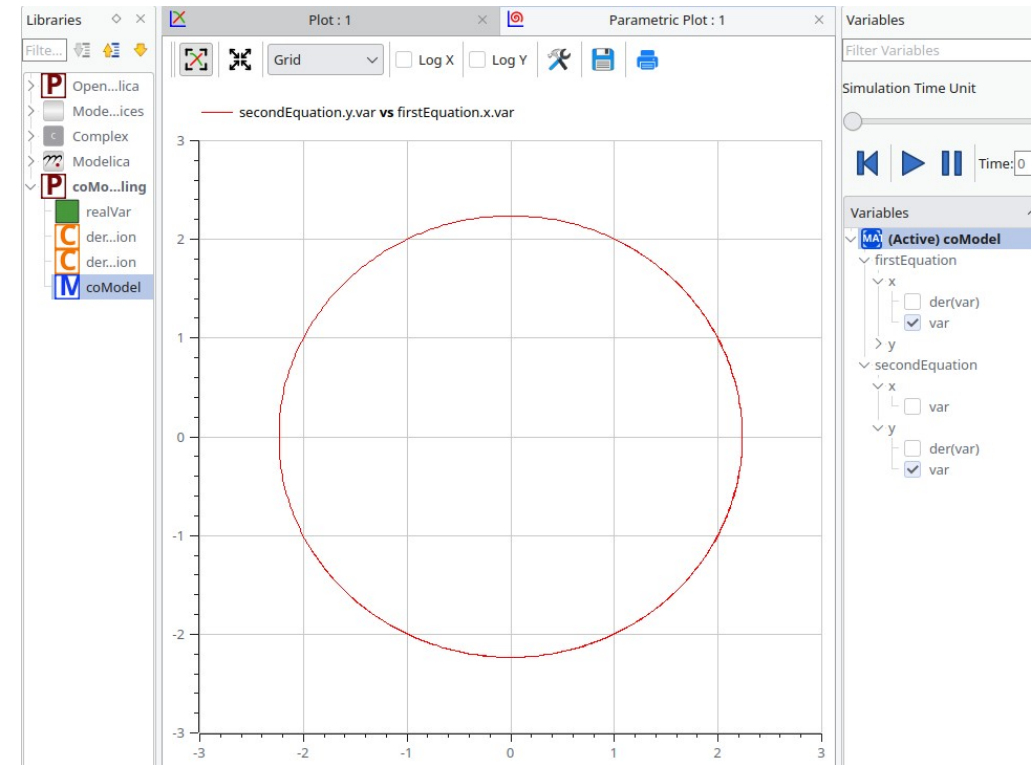
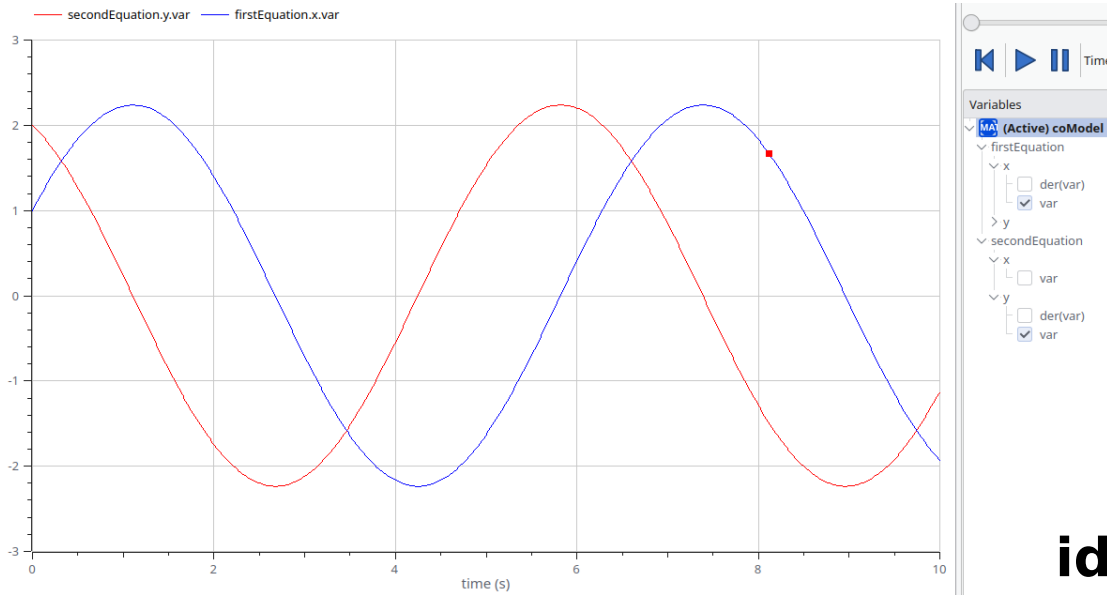
OMSimulator is also included with the OpenModelica installer, which includes OMEdit, a graphical editor.

```
(omsimulator-venv) rakshit@debian-x13:~/git/mosis/2025/cbd/tutorial/cosimulation$ OMSimulator --stopTime=10 --stepSize=0.01 --tolerance=1e-6 circle_test.ssp
warning: invalid "SystemStructureDescription" detected in file "circle_test.ssp" at line: 83 column: 23, element 'Connectors' is not allowed for content model
ings?,Annotations?)'
warning: "SystemStructureDescription" does not conform to the SSP standard schema
info: Result file: circle_test_res.mat (bufferSize=10)
info: Alg. loop (size 2/4)
      circle_test.Root.y_causal
      circle_test.Root.x_causal
info: 2 warnings
info: 0 errors
(omsimulator-venv) rakshit@debian-x13:~/git/mosis/2025/cbd/tutorial/cosimulation$
```

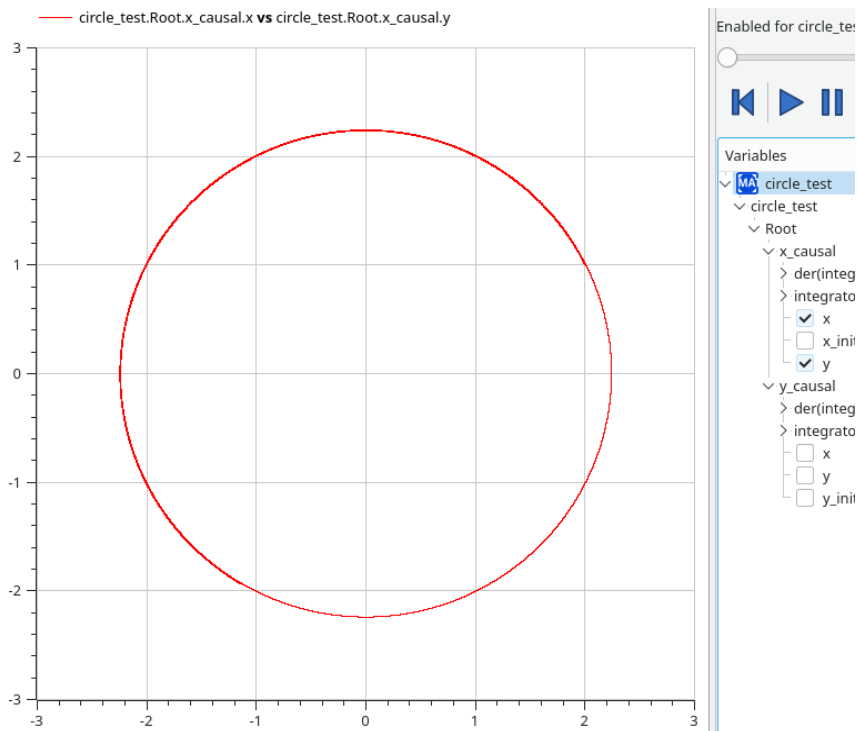
orchestrate multiple simulators (**co-simulation**, “black box”)



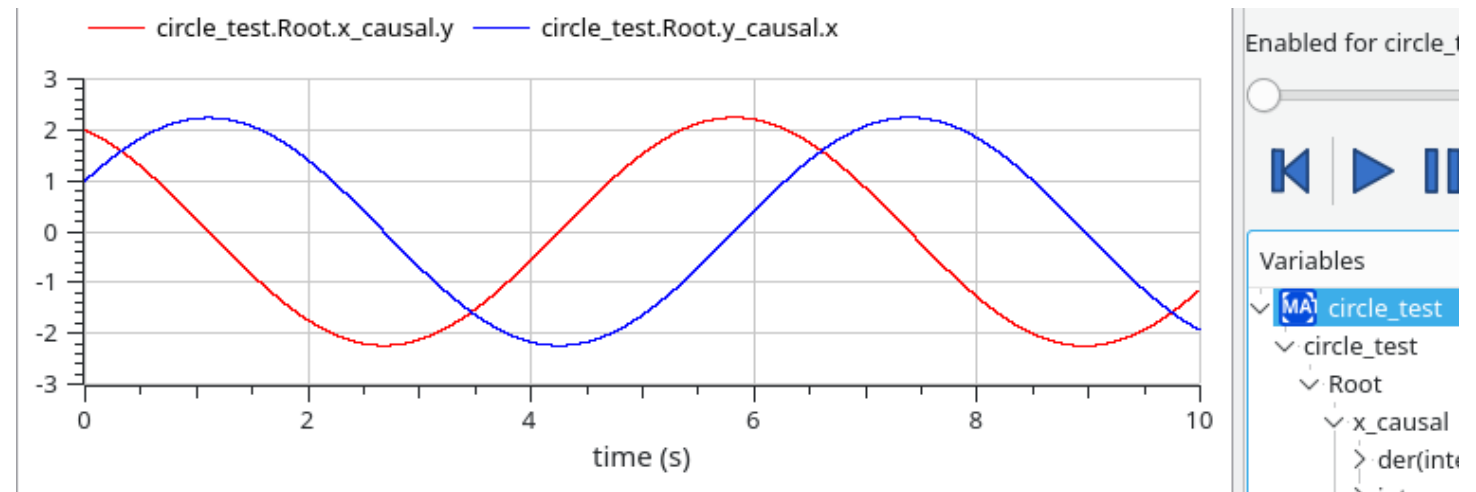
simulate as a single model (**co-modelling**, “white box”)



identical
traces



orchestrate multiple simulators (**co-simulation**, “black box”)



Tradeoff between co-modelling and co-simulation

time vs. accuracy: for the same run-time, co-simulation is less accurate; for the same accuracy, co-simulation is slower.

```
(omsimulator-venv) rakshit@debian-x13:~/git/mosis/2025/cbd/tutorial/cosimulation$ time OMSimulator --stopTime=10 --stepSize=0.01 --tolerance=1e-6 circle_test.ssp
warning: invalid "SystemStructureDescription" detected in file "circle_test.ssp" at line: 83 column: 23, element 'Connectors' is not allowed for content model '(Connections?,Annotations?)'
warning: "SystemStructureDescription" does not conform to the SSP standard schema
info:   Result file: circle_test_res.mat (bufferSize=10)
info:   Alg. loop (size 2/4)
        circle_test.Root.y_causal
        circle_test.Root.x_causal
info:   2 warnings
info:   0 errors

real    0m0.403s
user    0m0.328s
sys     0m0.072s
```

REAL TIME here means “wall clock” or “elapsed” time,
not a guaranteed 1-to-1 correspondence between simulated and wall clock time.

For the circle test:

n = 10 runs
step size = 0.01 s
stop time = 10 s

for the **same tolerance** = 10^{-6}
co-simulation is roughly **10x slower**

REAL TIME:

simulate():

Mean: 0.0153s
StdDev: 0.0013s
Min: 0.0140s
Max: 0.0180s

cosimulate():

Mean: 0.1802s
StdDev: 0.0120s
Min: 0.1620s
Max: 0.2020s

USER TIME:

simulate():

Mean: 0.0101s
StdDev: 0.0014s
Min: 0.0080s
Max: 0.0120s

cosimulate():

Mean: 0.1334s
StdDev: 0.0072s
Min: 0.1220s
Max: 0.1470s

SYS TIME:

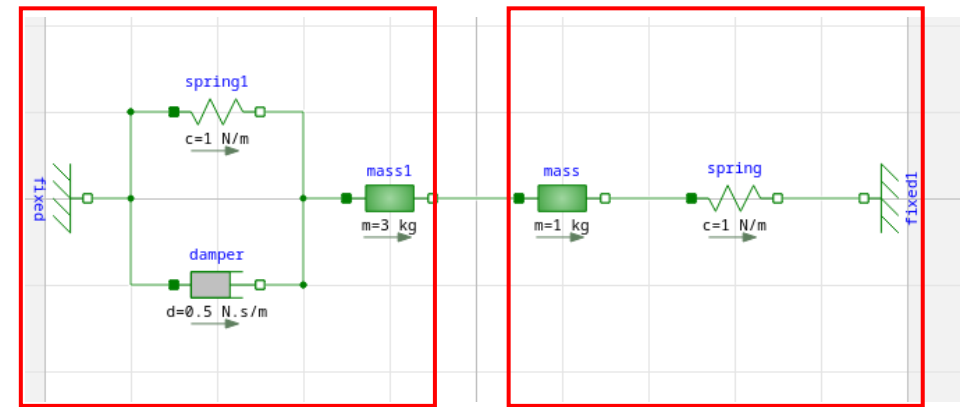
simulate():

Mean: 0.0060s
StdDev: 0.0017s
Min: 0.0040s
Max: 0.0090s

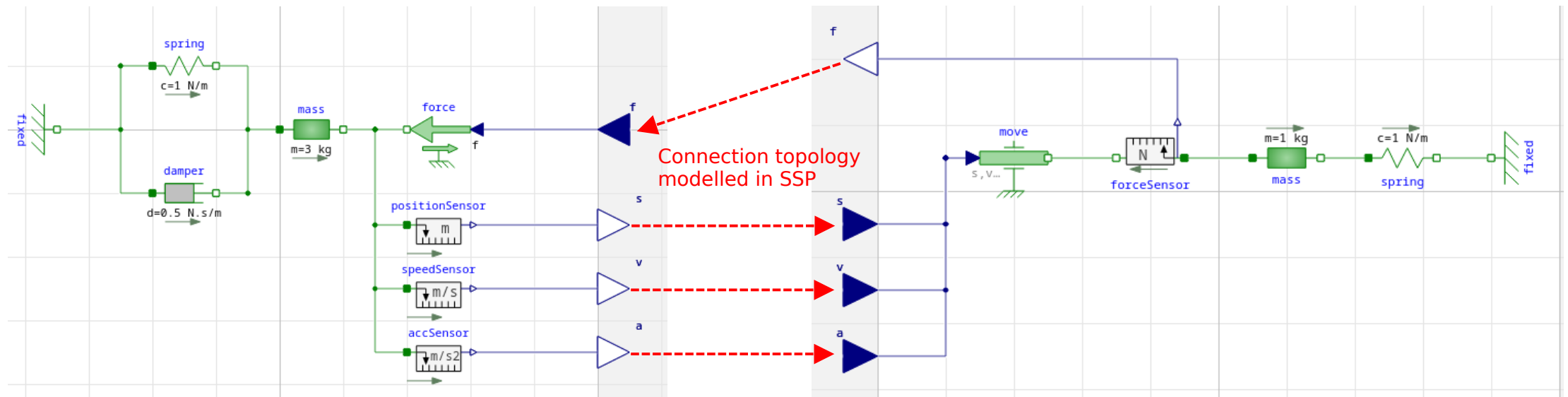
cosimulate():

Mean: 0.0466s
StdDev: 0.0100s
Min: 0.0240s
Max: 0.0560s

Another example with a physical connection:
Coupled mass-spring-damper systems



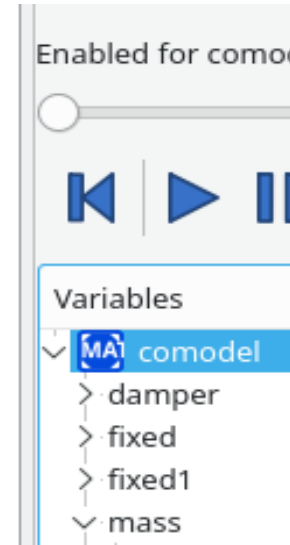
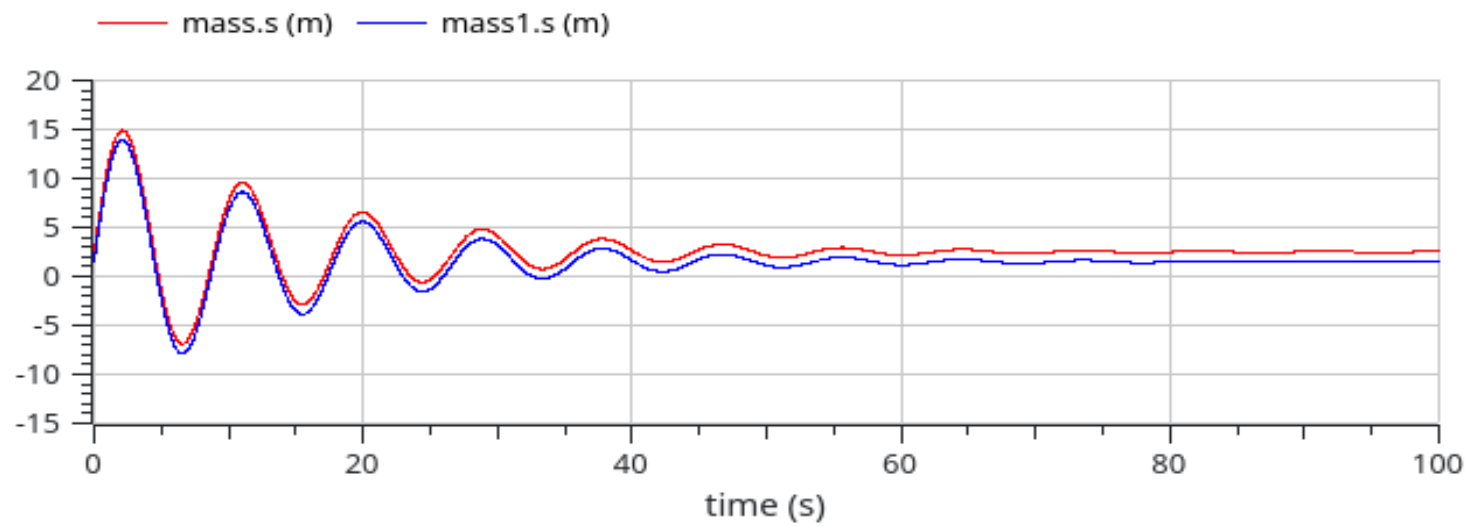
a-causal system_1 a-causal system_2



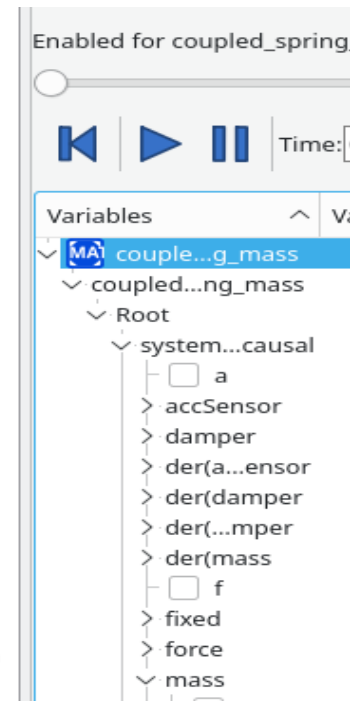
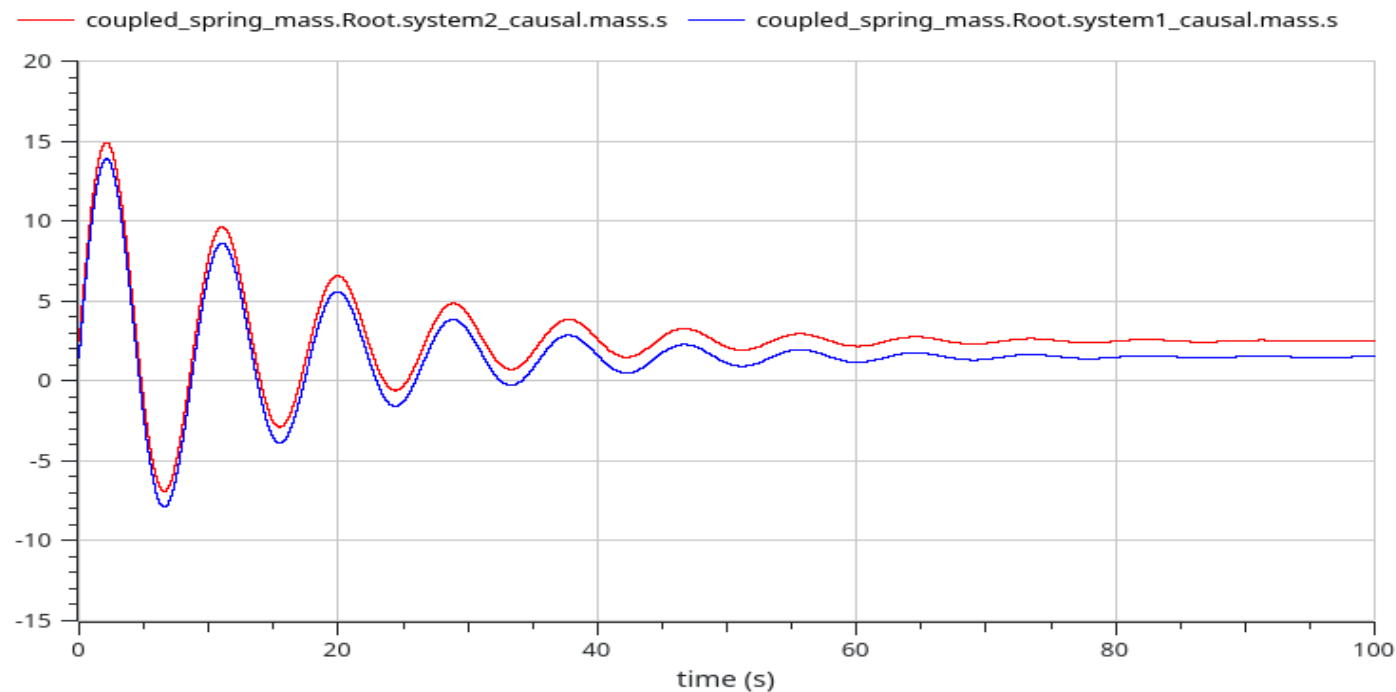
causal system_1 (exported as FMU)

causal system_2 (exported as FMU)

simulate as a single model (**co-modelling**, “white box”)



orchestrate multiple simulators (**co-simulation**, “black box”)



Tradeoff between co-modelling and co-simulation

time vs. accuracy: for the same run-time, co-simulation is less accurate; for the same accuracy, co-simulation is slower.

For the coupled spring mass :

n =10 runs

step size = 0.01 s

stop time = 10 s

for the same tolerance = 10^{-6}

co-simulation is roughly 100x slower

REAL TIME:

simulate():

Mean: 0.0350s

StdDev: 0.0064s

Min: 0.0280s

Max: 0.0450s

cosimulate():

Mean: 3.7715s

StdDev: 0.5358s

Min: 3.0850s

Max: 5.1550s

USER TIME:

simulate():

Mean: 0.0251s

StdDev: 0.0054s

Min: 0.0170s

Max: 0.0370s

cosimulate():

Mean: 3.2279s

StdDev: 0.4412s

Min: 2.6740s

Max: 4.3610s

SYS TIME:

simulate():

Mean: 0.0091s

StdDev: 0.0048s

Min: 0.0040s

Max: 0.0180s

cosimulate():

Mean: 0.5056s

StdDev: 0.0821s

Min: 0.3810s

Max: 0.6800s

Hand-craft an FMU (instead of generating from OMEdit)

... so you can build your own FMU exporter/generator



Functional Mock-up Interface

The Functional Mock-up Interface (FMI) is a free standard that defines an interface to exchange dynamic models using a combination of XML files, binaries and C code. It's supported by [280+ tools](#) and maintained as a [Modelica Association Project](#).

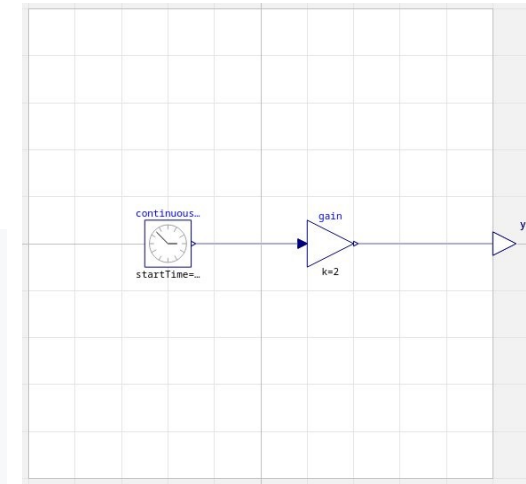
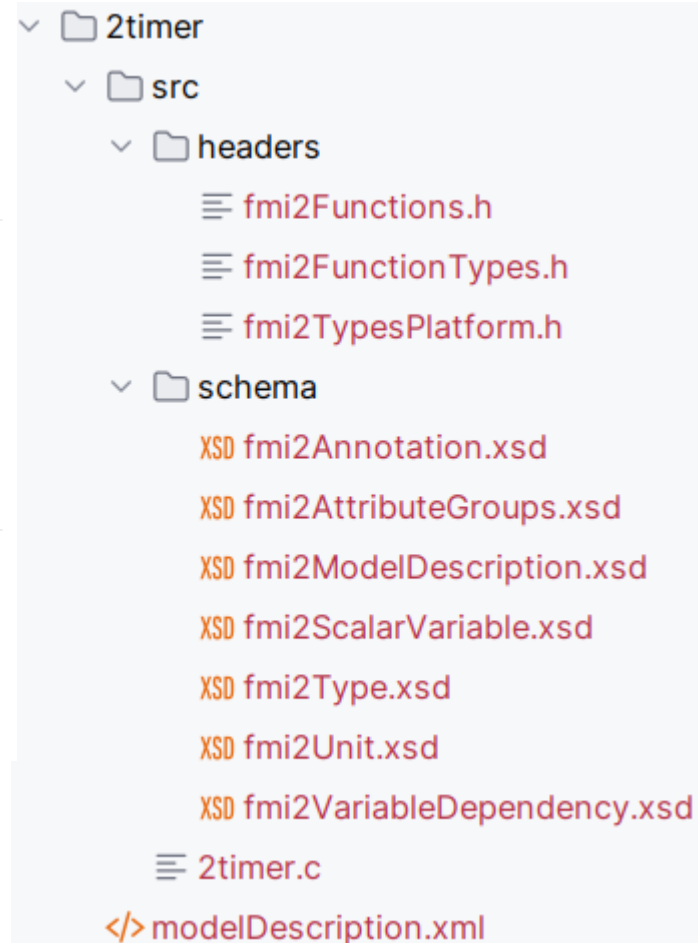
Releases are available on the [FMI website](#) and the latest development version of the specification is available [here](#).

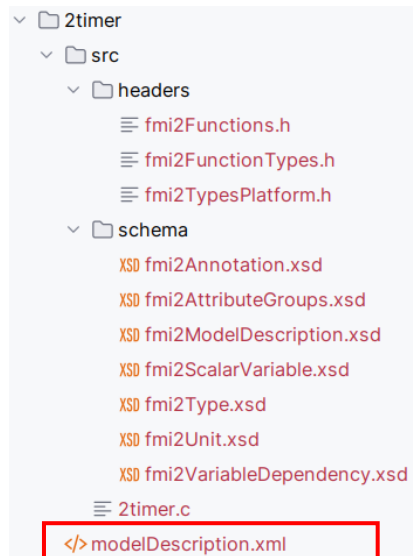
Repository Structure

- [docs](#) — sources of the specification document
- [headers](#) — header files of the C-API
- [schema](#) — XSD schema for the modelDescription.xml
- [CONTRIBUTING.md](#) — info on contributing to the project and building the specification document
- [README.md](#) — this document

<https://github.com/modelica/fmi-standard>

Going back to the simple example of the block that outputs (time x 2)





```
<?xml version="1.0" encoding="UTF-8"?>

<fmiModelDescription fmiVersion="2.0"
                    modelName="2timer"
                    guid="1234567890"
                    description="Timer FMU that outputs 2 times the current time"
                    generationTool="manual">

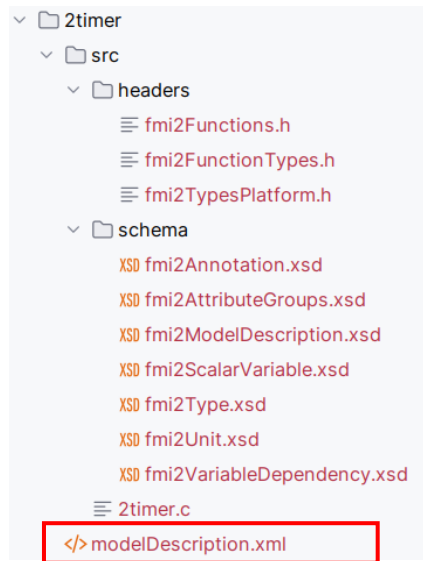
  <CoSimulation modelIdentifier="Timer"
                canHandleVariableCommunicationStepSize="true">
    <SourceFiles>
      <File name="src/2timer.c"/>
    </SourceFiles>
  </CoSimulation>

  <UnitDefinitions>
    <Unit name="s">
      <BaseUnit s="1"/>
    </Unit>
  </UnitDefinitions>

  <DefaultExperiment startTime="0" stopTime="10" stepSize="0.1" />

  <ModelVariables>
    <ScalarVariable name="output"
                    valueReference="0"
                    initial="calculated"
                    causality="output"
                    variability="continuous">
      <Real unit="s"/>
    </ScalarVariable>
  </ModelVariables>

</fmiModelDescription>
```



```
<?xml version="1.0" encoding="UTF-8"?>

<fmiModelDescription fmiVersion="2.0"
    modelName="2timer"
    guid="1234567890"
    description="Timer FMU that outputs 2 times the current time"
    generationTool="manual">

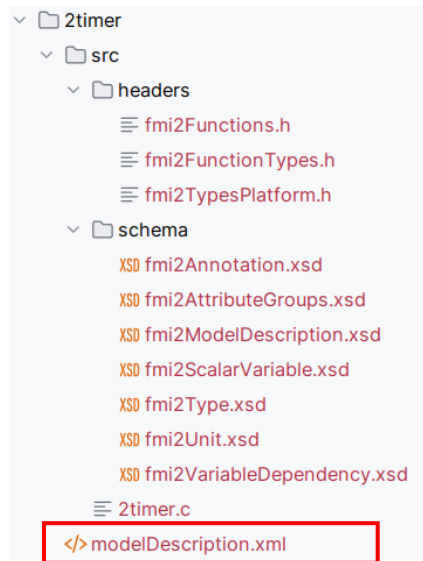
    <CoSimulation modelIdentifier="Timer"
        canHandleVariableCommunicationStepSize="true">
        <SourceFiles>
            <File name="src/2timer.c"/>
        </SourceFiles>
    </CoSimulation>

    <UnitDefinitions>
        <Unit name="s">
            <BaseUnit s="1"/>
        </Unit>
    </UnitDefinitions>

    <DefaultExperiment startTime="0" stopTime="10" stepSize="0.1" />

    <ModelVariables>
        <ScalarVariable name="output"
            valueReference="0"
            initial="calculated"
            causality="output"
            variability="continuous">
            <Real unit="s"/>
        </ScalarVariable>
    </ModelVariables>

</fmiModelDescription>
```



```
<?xml version="1.0" encoding="UTF-8"?>

<fmiModelDescription fmiVersion="2.0"
    modelName="2timer"
    guid="1234567890"
    description="Timer FMU that outputs 2 times the current time"
    generationTool="manual">

    <CoSimulation modelIdentifier="Timer"
        canHandleVariableCommunicationStepSize="true">

        <SourceFiles>
            <File name="src/2timer.c"/>
        </SourceFiles>
    </CoSimulation>

    <UnitDefinitions>
        <Unit name="s">
            <BaseUnit s="1"/>
        </Unit>
    </UnitDefinitions>

    <DefaultExperiment startTime="0" stopTime="10" stepSize="0.1" />

    <ModelVariables>
        <ScalarVariable name="output"
            valueReference="0"
            initial="calculated"
            causality="output"
            variability="continuous">
            <Real unit="s"/>
        </ScalarVariable>
    </ModelVariables>
```

- 2timer
 - src
 - headers
 - fmi2Functions.h
 - fmi2FunctionTypes.h
 - fmi2TypesPlatform.h
 - schema
 - fmi2Annotation.xsd
 - fmi2AttributeGroups.xsd
 - fmi2ModelDescription.xsd
 - fmi2ScalarVariable.xsd
 - fmi2Type.xsd
 - fmi2Unit.xsd
 - fmi2VariableDependency.xsd
 - 2timer.c
 - modelDescription.xml

```
<?xml version="1.0" encoding="UTF-8"?>

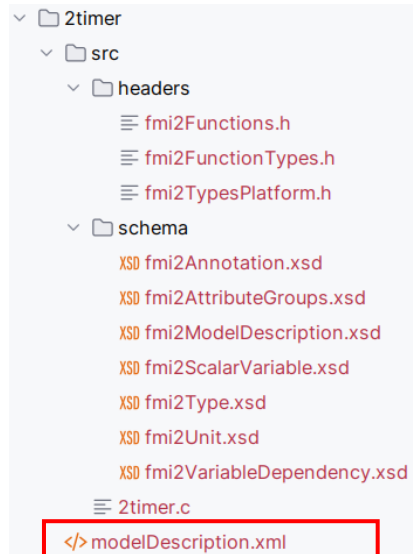
<fmiModelDescription fmiVersion="2.0"
    modelName="2timer"
    guid="1234567890"
    description="Timer FMU that outputs 2 times the current time"
    generationTool="manual">

    <CoSimulation modelIdentifier="Timer"
        canHandleVariableCommunicationStepSize="true">
        <SourceFiles>
            <File name="src/2timer.c"/>
        </SourceFiles>
    </CoSimulation>

    <UnitDefinitions>
        <Unit name="s">
            <BaseUnit s="1"/>
        </Unit>
    </UnitDefinitions>

    <DefaultExperiment startTime="0" stopTime="10" stepSize="0.1" />

    <ModelVariables>
        <ScalarVariable name="output"
            valueReference="0"
            initial="calculated"
            causality="output"
            variability="continuous">
            <Real unit="s"/>
        </ScalarVariable>
    </ModelVariables>
```



```
<?xml version="1.0" encoding="UTF-8"?>

<fmiModelDescription fmiVersion="2.0"
                    modelName="2timer"
                    guid="1234567890"
                    description="Timer FMU that outputs 2 times the current time"
                    generationTool="manual">

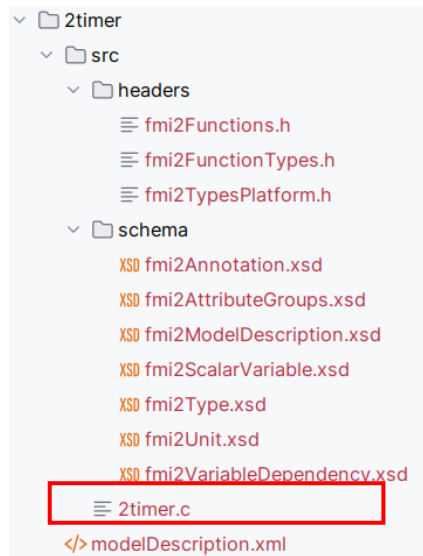
  <CoSimulation modelIdentifier="Timer"
                canHandleVariableCommunicationStepSize="true">
    <SourceFiles>
      <File name="src/2timer.c"/>
    </SourceFiles>
  </CoSimulation>

  <UnitDefinitions>
    <Unit name="s">
      <BaseUnit s="1"/>
    </Unit>
  </UnitDefinitions>

  <DefaultExperiment startTime="0" stopTime="10" stepSize="0.1" />

  <ModelVariables>
    <ScalarVariable name="output"
                    valueReference="0"
                    initial="calculated"
                    causality="output"
                    variability="continuous">
      <Real unit="s"/>
    </ScalarVariable>
  </ModelVariables>

</fmiModelDescription>
```



```
2timer.c x
1  #include "headers/fmi2Functions.h"
2  #include <stdlib.h>
3  #include <string.h>
4
5  typedef struct {
6      fmi2Real time;
7      fmi2Real output;
8  } TimerState;
9
```

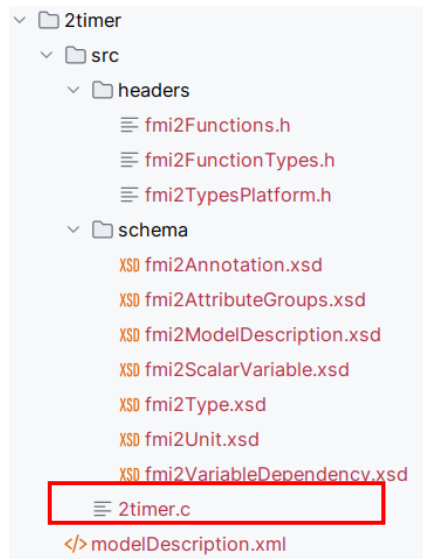
The state of the timer

```
fmi2Component fmi2Instantiate(fmi2String instanceName,
                             fmi2Type fmuType,
                             fmi2String fmuGUID,
                             fmi2String fmuResourceLocation,
                             const fmi2CallbackFunctions* functions,
                             fmi2Boolean visible,
                             fmi2Boolean loggingOn) {

    TimerState* model = (TimerState*)malloc( size: sizeof(TimerState));
    if (!model) return NULL;

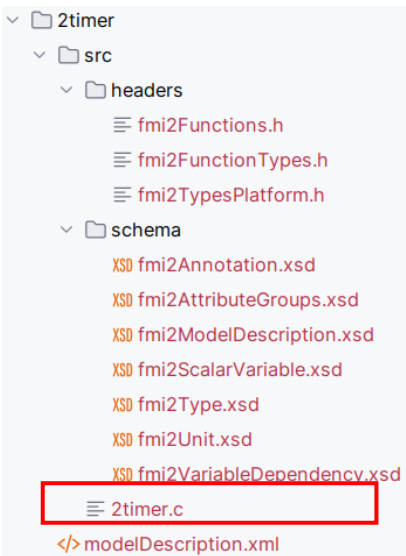
    return (fmi2Component)model;
}
```

Instantiate the state struct
(allocate memory)



```
fmi2Status fmi2EnterInitializationMode(fmi2Component c) {  
    TimerState* model = (TimerState*)c;  
  
    model->time = 0.0;  
    model->output = 0.0;  
  
    return fmi2OK;  
}  
  
fmi2Status fmi2ExitInitializationMode(fmi2Component c) {  
    return fmi2OK;  
}
```

Initialize



```
fmi2Status fmi2SetupExperiment(fmi2Component c,
                              fmi2Boolean toleranceDefined,
                              fmi2Real tolerance,
                              fmi2Real startTime,
                              fmi2Boolean stopTimeDefined,
                              fmi2Real stopTime) {

    TimerState* model = (TimerState*)c;

    model->time = startTime;
    model->output = 2.0 * startTime;

    return fmi2OK;
}
```

Setup experiment

```
fmi2Status fmi2DoStep(fmi2Component c,
                     fmi2Real currentCommunicationPoint,
                     fmi2Real communicationStepSize,
                     fmi2Boolean noSetFMUStatePriorToCurrentPoint) {

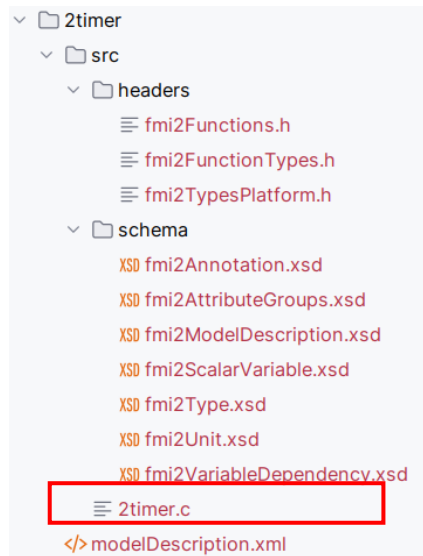
    TimerState* model = (TimerState*)c;

    model->time = currentCommunicationPoint + communicationStepSize;

    model->output = 2.0 * model->time;

    return fmi2OK;
}
```

Do (communication) step



```
fmi2Status fmi2GetReal(fmi2Component c,  
                      const fmi2ValueReference vr[],  
                      size_t nvr,  
                      fmi2Real value[]) {
```

```
    TimerState* model = (TimerState*)c;
```

```
    for (size_t i = 0; i < nvr; i++) {  
        if (vr[i] == 0) {  
            value[i] = model->output;  
        } else {  
            return fmi2Error;  
        }  
    }
```

```
    return fmi2OK;
```

```
}
```

```
fmi2Status fmi2SetReal(fmi2Component c,  
                      const fmi2ValueReference vr[],  
                      size_t nvr,  
                      const fmi2Real value[]) {
```

```
    return fmi2Error;
```

```
}
```

```
void fmi2FreeInstance(fmi2Component c) {  
    if (c) {  
        free(ptr: c);  
    }  
}
```

Free up memory

Get and set Reals (nothing to set!)

Build Binaries for a specific platform (e.g., Linux 64 bit ELF shared objects (.so))

```
(omsimulator-venv) rakshit@debian-x13:~/git/mosis/2025/cbd/tutorial/fmi/2timer$ mkdir -p binaries/linux64/
(omsimulator-venv) rakshit@debian-x13:~/git/mosis/2025/cbd/tutorial/fmi/2timer$ gcc -shared -fPIC -I src/ -I src/headers/ src/2timer.c -o binaries/linux64/Timer.so -lm
(omsimulator-venv) rakshit@debian-x13:~/git/mosis/2025/cbd/tutorial/fmi/2timer$ zip -r ../2timer.fmu modelDescription.xml src/ binaries/
updating: modelDescription.xml (deflated 60%)
  adding: src/ (stored 0%)
  adding: src/headers/ (stored 0%)
  adding: src/headers/fmi2FunctionTypes.h (deflated 76%)
  adding: src/headers/fmi2TypesPlatform.h (deflated 62%)
  adding: src/headers/fmi2Functions.h (deflated 74%)
  adding: src/schema/ (stored 0%)
  adding: src/schema/fmi2VariableDependency.xsd (deflated 62%)
  adding: src/schema/fmi2ModelDescription.xsd (deflated 78%)
  adding: src/schema/fmi2Type.xsd (deflated 63%)
  adding: src/schema/fmi2Unit.xsd (deflated 71%)
  adding: src/schema/fmi2AttributeGroups.xsd (deflated 64%)
  adding: src/schema/fmi2Annotation.xsd (deflated 54%)
  adding: src/schema/fmi2ScalarVariable.xsd (deflated 76%)
  adding: src/2timer.c (deflated 82%)
  adding: binaries/ (stored 0%)
  adding: binaries/linux64/ (stored 0%)
  adding: binaries/linux64/Timer.so (deflated 79%)
```

What if we want to hide Intellectual Property (IP)?

Even though the Modelica model is not included in an FMU, the C code may reveal too much information (i.e., can be reverse engineered).

1. Remove source files sections from modelDescription.xml

```
</> modelDescription.xml ×
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <fmiModelDescription fmiVersion="2.0"
4      modelName="2timer"
5      guid="1234567890"
6      description="Timer FMU that outputs 2 times the current time"
7      generationTool="manual">
8
9      <ModelExchange modelIdentifier="Timer">
10         </ModelExchange>
11
12      <CoSimulation modelIdentifier="Timer"
13          canHandleVariableCommunicationStepSize="true">
14         </CoSimulation>
15
16      <UnitDefinitions>
17         <Unit name="s"> <BaseUnit s="1"/></Unit>
18      </UnitDefinitions>
19
20      <DefaultExperiment startTime="0" stopTime="10" stepSize="0.1" />
21
22      <ModelVariables>
23         <ScalarVariable name="y"
24             valueReference="0"
25             initial="calculated"
26             causality="output"
27             variability="continuous">
28             <Real start="0.0"/>
29         </ScalarVariable>
30     </ModelVariables>
31
32     <ModelStructure>
33         <Outputs>
34             <Unknown index="1"/>
35         </Outputs>
36     </ModelStructure>
37 </fmiModelDescription>
```

What if we want to hide Intellectual Property (IP)?

1. Remove source files sections from modelDescription.xml
2. Only compress modelDescription.xml and binaries folder
Do not include the source files ...

```
rakshit@debian-x13:~/git/mosis/2025/cbd/tutorial/fmi/2timer$ zip -r ../2timer.fmu modelDescription.xml binaries/
updating: modelDescription.xml (deflated 59%)
updating: binaries/ (stored 0%)
updating: binaries/linux64/ (stored 0%)
updating: binaries/linux64/Timer.so (deflated 77%)
rakshit@debian-x13:~/git/mosis/2025/cbd/tutorial/fmi/2timer$
```

What if we want to hide Intellectual Property (IP)?

When generating an FMU from OMEdit:

un-check "Include Source Code" in Tools > Options > FMI (before exporting FMU)

