HYBRID CO-SIMULATION OF FMUs
USING DEV&DESS IN MECSYCO

Benjamin CAMUS, Vincent CHEVRIER, Julien VAUBOURG, Laurent CIARLETTA, Christine BOURJOT

Virginie GALTIER

Mathieu CAUJOLLE
FMI 2.0: PRESENTATION

FMI, a standard for Model Exchange & Co-Simulation

FMU for « Model Exchange »

FMU for « Co-Simulation »

Exchange of source models

Exchange of executable models

$t_0, \mathbf{p}$: initial values (a subset of $v(t_0)$)

$v$

Models are managed by time-stepped solvers

Solution preferred by EDF
Based on a communication points execution strategy

**Basic computation functions to:**
- perform an integration to a given time-step (*fmi2DoStep*)
- set inputs’ values (*fmi2SetReal/Integer/Boolean/String*)
- get outputs’ values (*fmi2GetReal/Integer/Boolean/String*)

**Optional functions** enabling to export/import the model state
- *fmi2GetFMUState / fmi2SetFMUState*

⇒ **Essential to enable roll-back**, i.e. go a single integration step back

Many more functions available in the FMI API…
Hybrid co-simulation: FMU must interact with event-based model
- We must integrate input events into an FMU,
- and manage the generation of state-event from an FMU

Exact detection of occurrence times is difficult
- **State-changes are not tagged**: No persistent variable to inform if (when) an event occurred during a simulation step
- **Bias when event (states or input) occurs between two communication points**
  - state events are localized at the upper communication point
  - new inputs are integrated at the upper communication point
DEV&DESS:

- offers a sound framework for formalizing how an equation based component interacts with the discrete-world.
- embedded by Zeigler in DEVS which is a universal formalism for describing discrete-event models

If it is wrapped into DEV&DESS, an FMU can interact with discrete-event models.

Issues:

- the original DEVS version of DEV&DESS relies on a quantized integrator approach
- Not compliant with the time-stepped framework of FMI for co-simulation

We need to adapt the original DEVS version of DEV&DESS
Continuous component = the FMU to integrate

Discrete-event component = the discrete behavior of the FMU – i.e. :
- How the FMU state changes when events occurs?
- What are the output event produced by the model?

Event-detection component – i.e. :
- Is there a state-event according to the FMU state?
GETTING THE TIME OF THE NEXT INTERNAL EVENT

The Time of the Next Internal Event is the minimum between:

- the date of the next internal event scheduled in the discrete event component
- the date of the next communication point of the FMU
- the date of the next state-event

\texttt{getNextInternalEventTime()} must not change the state of the model

Thus, we:

- Perform a time exploration with the FMU to see if a state-event occurs before its next communication point
- Use its rollback capability to restore the state of the FMU before the exploration, if required
Bisectional search for state-event localization

**INPUT:** \( \Delta T \in \mathbb{R}^+, m \in \mathbb{N}_0^+ \)
- \( \delta t \leftarrow 0 \)
- \( \Delta t \leftarrow \Delta T \)

for 1 to m do
  - solver.rollback()
  - \( \Delta t \leftarrow \Delta t/2 \)
  - solver.doStep(\( \delta t + \Delta t \))
  - if \( \neg \text{detFunction.stateEventOccurrence()} \) then
    - \( \delta t \leftarrow \delta t + \Delta t \)
  - end if
end for

If event detected, rollback & \( D_{t_{i+1}} = \max(\min \text{ step}, D_t/2) \)
**PROCESSING EVENTS**

*processExternalEvent* reports the occurrence of an external input event at $t$

1. The FMU is rolled back to its previous state
2. A new simulation step to $t$ is computed to reach the point where the event occurs (no new event will occur in the meanwhile)

*processInternalEvent* works similarly

- But, the FMU state after the exploration is reused if the internal event time is the same
- This limits the computational burden of the exploration method
Co-simulation middleware
Using a DEVS wrapping strategy
  1 wrapper = 1 model/simulator

Based on Agent & Artifact (A&A) concepts :
  - Autonomous agents
  - Passive artifacts proposing services to Agents

Advantages for co-simulation :
  - Multi-representation integration (e.g. multi-level modeling)
  - Multi-formalism integration (thanks to DEVS)
  - Simulators interoperability (e.g. NS-3, OMNeT++, NetLogo, ad-hoc)
  - Modular view
  - Parallel & decentralized execution
  - Distributable architecture
    thanks to the multi-agent paradigm
STUDIED USE CASE: A BARREL FACTORY

A simple use case to illustrate the generic aspect of our methodology:

- applicable outside from the context of Smart Grids

A fully implemented and functional wrapper for integrating FMU 2.0 components into a hybrid co-simulation was developed.
STUDIED USE CASE: MODELING

Simple control systems generating time-events

Equation-based systems

The tank model

The barrel model
Outputs of the continuous and discrete components

Taking simultaneously account of continuous and discrete dynamics within the FMU components.
## RESULTS

Accuracy of the state event detection

<table>
<thead>
<tr>
<th>Step Size</th>
<th>$t_{barrel1}$</th>
<th>$t_{barrel2}$</th>
<th>$t_{barrel3}$</th>
<th>$t_{barrel4}$</th>
<th>$t_{barrel5}$</th>
<th>$t_{barrel6}$</th>
<th>$t_{barrel7}$</th>
<th>$t_{barrel8}$</th>
<th>$t_{barrel9}$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Series 1</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.001</td>
<td>1.016</td>
<td>1.613</td>
<td>1.910</td>
<td>2.715</td>
<td>5.773</td>
<td>7.342</td>
<td>8.535</td>
<td>10.412</td>
<td>15.139</td>
</tr>
<tr>
<td>0.01</td>
<td>1.016</td>
<td>1.613</td>
<td>1.910</td>
<td>2.715</td>
<td>5.773</td>
<td>7.342</td>
<td>8.535</td>
<td>10.412</td>
<td>15.139</td>
</tr>
<tr>
<td>0.1</td>
<td>1.016</td>
<td>1.613</td>
<td>1.910</td>
<td>2.715</td>
<td>5.773</td>
<td>7.342</td>
<td>8.535</td>
<td>10.412</td>
<td>15.139</td>
</tr>
<tr>
<td><strong>Series 2</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.001</td>
<td>1.017</td>
<td>1.615</td>
<td>1.910</td>
<td>2.716</td>
<td>5.774</td>
<td>7.344</td>
<td>8.538</td>
<td>10.419</td>
<td>15.167</td>
</tr>
<tr>
<td>0.01</td>
<td>1.020</td>
<td>1.620</td>
<td>1.900</td>
<td>2.710</td>
<td>5.770</td>
<td>7.340</td>
<td>8.540</td>
<td>10.430</td>
<td>15.230</td>
</tr>
<tr>
<td>0.1</td>
<td>1.100</td>
<td>1.800</td>
<td>1.800</td>
<td>2.600</td>
<td>5.700</td>
<td>7.300</td>
<td>8.500</td>
<td>10.400</td>
<td>15.300</td>
</tr>
</tbody>
</table>

**Series 1**: proposed implementation of the DEV&DESS model artifact  
**Series 2**: classic FMU co-simulation strategy with constant time step  

Bisectional search = same accuracy, independently of the step size
A fully functional and generic wrapper for integrating FMU 2.0 components into a hybrid co-simulation

- Respects the DEV&DESS semantics and the FMI 2.0 operational constraints
- Rigorous interactions of FMU and event-based models within DEVS framework

Further improvements of the event location strategy:
- Illinois algorithm or a combination of other existing algorithms are considered.

A solution applied to large Smart Grids and Smart Spaces
- Large use cases involving several domains are being considered, among which: electric systems, telecom systems, thermal systems and information systems