

1

2 **Wrapper for Functional Mock-up Interface in TLM-based**  
3 **Asynchronous Co-Simulation**

4 Robert Braun, SICS East Swedish ICT

5 2017-04-12

|                  |          |        |       |
|------------------|----------|--------|-------|
| OPENCPS DOCUMENT |          |        | Pages |
|                  |          |        | 1/11  |
| Reference number | Revision | Author |       |
|                  |          |        |       |

## 6 1. Introduction

7 Functional Mock-up Interface (FMI) is a tool-independent standard for connecting simulation tools  
 8 [2]. One tool can export a model as a Functional Mockup Unit (FMU), a ZIP package with the file ex-  
 9 tension FMU. This file is in turn loaded by the master simulation tool, which can connect and simulate  
 10 the model. An FMU file contains a model description XML file called `modelDescription.xml`,  
 11 binaries for different platforms and other optional content. It is important that the FMU contains a  
 12 binary file for the platform where the master simulation tool is executed.

13 There are two versions of the FMI standard: FMI for Co-Simulation and FMI for Model Exchange.  
 14 The main difference is that FMUs for Co-Simulation contain their own built-in solvers, and only  
 15 exchange data at predefined communication points. FMUs for Model Exchange require a solver in  
 16 the master simulation tool.

17 The TLM framework is able to simulate aggregated systems of connected sub-models using asyn-  
 18 chronous TLM communication. Currently, only 3D multi-body mechanical sub-models are supported.  
 19 Other physical domains, 1D connections and causal signal communication is intended to be imple-  
 20 mented in the future. Including FMUs in the TLM framework requires a wrapper. `FMIWrapper` is  
 21 a generic wrapper for connecting functional mockup units (FMUs) to the TLM framework. It uses  
 22 the FMI Library from Modelon [1] to load an FMU, and the `TLMPlugin` for socket communication  
 23 with the framework, see fig. 1.

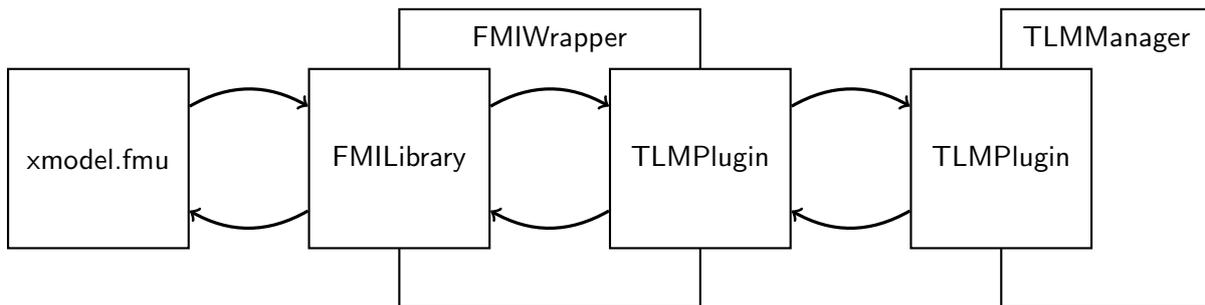


Figure 1: `FMIWrapper` uses `FMILibrary` to import FMUs and `TLMPlugin` for socket communication.

24 Both FMI for co-simulation and FMI for model exchange are supported. Model exchange requires  
 25 a solver in the wrapper executable. For this reason, two solvers from the Sundials package are  
 26 included.

|                  |          |        |       |
|------------------|----------|--------|-------|
| OPENCPS DOCUMENT |          |        | Pages |
|                  |          |        | 1/11  |
| Reference number | Revision | Author |       |
|                  |          |        |       |

## 27 2. Setting up a simulation

28 The FMI wrapper is started by the StartTLMFMIWrapper startup script. This script generates the  
 29 `tlm.config` file and calls the FMIWrapper executable. The executable takes the following argu-  
 30 ments:

31  
 32 FMIWrapper <path> <fmufile> <solver> <debug>

33  
 34 Example:

35  
 36 FMIWrapper C:\temp\folder mymodel.fmu solver=CVODE -d

37  
 38 The last two arguments are optional. Available solvers are Euler, RungeKutta, CVODE and  
 39 IDA. Section 4 contains more details about the different solvers. These can currently only be changed  
 40 by modifying the startup script, i.e. not from the graphical interface.

41 An FMU keep track of its variables by integer numbers called `value references`. However,  
 42 it does not provide any information about the mapping between its variables and the TLM interface.  
 43 Hence, this information must be provided by the user. A configuration file called `fmi.config` is  
 44 used for this purpose, see listing 1.

Listing 1: A configuration file maps value references to TLM variables

```

45 substeps, 100
46 name, tlm1
47 position, 6, 7, 8
48 orientation, 136, 139, 142, 137, 140, 143, 138, 141, 144
49 speed, 9, 10, 11
50 ang_speed, 145, 146, 147
51 force, 133, 134, 135, 167, 168, 169
52 name, tlm2
53 position, 12, 13, 14
54 orientation, 145, 146, 147, 148, 149, 150, 151, 152, 153
55 speed, 15, 16, 17
56 ang_speed, 154, 155, 156
57 force, 136, 137, 138, 170, 171, 172
    
```

|                  |          |        |       |
|------------------|----------|--------|-------|
| OPENCPS DOCUMENT |          |        | Pages |
|                  |          |        | 2/11  |
| Reference number | Revision | Author |       |
|                  |          |        |       |

Listing 2: Value references for variables are obtained from `modelDescription.xml`

```

<ScalarVariable
  name="fMITLMInterface3D1.t[1]"
  valueReference="167"
  variability="continuous"
  causality="local"
>
  <Real/>
</ScalarVariable>

```

58 As can be seen, data is stored in a comma-separated format. The first line specifies the number of  
 59 substeps used for FMI for Co-Simulation (see section 3). After this comes the port information. Each  
 60 port is specified by name, position, orientation, speed, angular speed and force, according to the TLM  
 61 interfaces in the TLM framework. The numbers after each keyword are the value references. These  
 62 can be obtained by analyzing the `modelDescription.xml` file. Listing 2 shows an example. The  
 63 variable first torque component has the value reference 167. Hence, this number should be inserted  
 64 as number four on the "force" line in `fmi.config`.

|                  |          |        |       |
|------------------|----------|--------|-------|
| OPENCPS DOCUMENT |          |        | Pages |
|                  |          |        | 3/11  |
| Reference number | Revision | Author |       |
|                  |          |        |       |

### 65 3. FMI for Co-Simulation

66 With FMI for Co-Simulation the solver is embedded within the FMU. Variables can only be exchanged  
 67 at predefined communication points. Hence, it is not possible for the solver to obtain interpolated  
 68 force variables during internal iteration steps. Keeping the force constant during the entire commu-  
 69 nication interval may, however, have a negative effect on numerical stability. For this reason it is  
 70 possible to divide each communication interval into a fixed number of *sub steps*, as defined in the  
 71 `fmi.config`. In this way the forces can at least be updated in the FMU at more fine-grained  
 72 intervals. Pseudo-code for the simulation loop in the wrapper is shown in listing 3. The real code  
 73 also contains error handling, but it has been excluded here to enhance readability.

Listing 3: Pseudo code for the simulation loop with FMI for co-simulation

```

74 while (tcur < tend) {
75     double hsub = hmax/nSubSteps;
76     for(size_t i=0; i<nSubSteps; ++i) {
77         x = fmu.get_real(x_vr,3);
78         T = fmu.get_real(T_vr,9);
79         v = fmu.get_real(v_vr,3);
80         w = fmu.get_real(w_vr,3);
81         f = TLMPugin.GetForce(tcur,x,T,v,w);
82         fmu.set_real(f_vr[j],6,f);
83
84         TLMPugin.SetMotion(tcur,x,T,v,w);
85
86         fmu.do_step(tcur,hsub);
87         tcur+=hsub;
88
89         x = fmu.get_real(x_vr,3);
90         T = fmu.get_real(T_vr,9);
91         v = fmu.get_real(v_vr,3);
92         w = fmu.get_real(w_vr,3);
93     }
94 }
    
```

95 Note that it is necessary to read the motion variables from the FMU before obtaining the force  
 96 from the TLMPugin. At the end of each major step it is also necessary to call `GetForce()` before  
 97 calling `SetMotion()`. The reason for this is that `SetMotion()` requires updated input variables  
 98 which are retrieved by `GetForce()`.

|                  |          |        |       |
|------------------|----------|--------|-------|
| OPENCPS DOCUMENT |          |        | Pages |
|                  |          |        | 4/11  |
| Reference number | Revision | Author |       |
|                  |          |        |       |

99 **4. FMI for model exchange**

100 With model exchange, the wrapper must provide a solver for the FMU. Three solvers are available:  
 101 explicit Euler, 4th order explicit Runge-Kutta, and the CVODE and IDA solvers from Sundials [3].  
 102 Listing 4 shows pseudo code for one major step (i.e. one communication interval) with the IDA  
 103 solver. Note that the solver is used with one step mode. This means that it takes one step at a  
 104 time, until its internal time exceeds the next communication interval. The CVODE solver requires a  
 105 callback function for obtaining derivatives of state variables (i.e. "right-hand side"). The IDA solver  
 106 requires a similar callback for obtaining the residuals.

Listing 4: Pseudo code for the simulation loop with FMI for model exchange

```

107 double position[3], orientation[9], speed[3], ang_speed[3], force[6];
108
109 x = fmu.get_real(x_vr, 3);
110 T = fmu.get_real(T_vr, 9);
111 v = fmu.get_real(v_vr, 3);
112 w = fmu.get_real(w_vr, 3);
113 f = TLMPugin.GetForce(tcur, x, T, v, w);
114 fmu.set_real(f_vr[j], 6, f);
115
116 y = fmu.get_continuous_states();
117 dy = fmu.get_derivatives();
118
119 tcur += h;
120
121 while(tc < tcur){
122     IDASolve(mem, tcur, &tc, y, dy, IDA_ONE_STEP);
123 }
124
125 fmu.set_continuous_states(y);
    
```

|                  |          |        |       |
|------------------|----------|--------|-------|
| OPENCPS DOCUMENT |          |        | Pages |
|                  |          |        | 5/11  |
| Reference number | Revision | Author |       |
|                  |          |        |       |

126 **5. Example: Double Pendulum with Dymola and OpenModelica**

127 This section will explain how to build a meta model of a double pendulum using FMUs exported from Dymola and OpenModelica. Figure 2 shows a sketch of the model. The resulting meta model  
 128 is found in the /MetaModels/FmuFmuPendulum folder. One FMU for Co-simulation and one  
 129 for Model Exchange will be used. Both models use a custom Modelica library called FMILIB for the  
 130 TLM interfaces. This library is shipped together with the FMI wrapper.  
 131

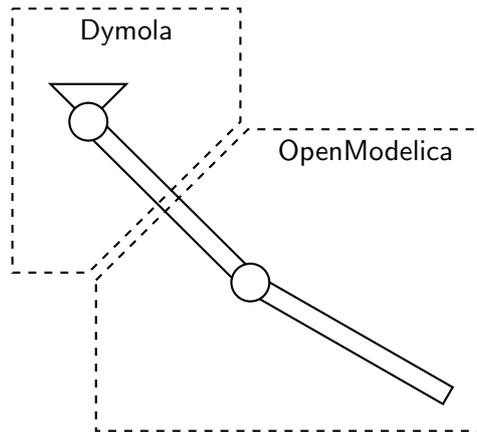


Figure 2: A double pendulum is simulated using FMUs exported from Dymola and OpenModelica

132 **5.1. Preparing Dymola FMU of first part pendulum**

133 The Dymola model consists of half the first pendulum and the fixed attachment to the inertial system, see figure fig. 3. It is normally advisable to decouple a model at its weakest point, which in this case  
 134 would be at the joint between the two pendulum arms. In this case the model is decoupled in the  
 135 middle of the first pendulum only to demonstrate the possibilities. Force and torque variables must  
 136 be specified as input variables at the top level of the model, in order to get the correct number of  
 137 equations and variables. Hence, we need to add the equations shown in listing 5.  
 138

Listing 5: Input variables must be specified on top level in the Modelica models

```

139 [...]
140 //Define FMI interface model
141 FMITLM.FMITLM_Interface_3D.FMITLMInterface3D fMITLMInterface3D;
142
143 //Define input force and input torque
144 input Real f[3] (start = zeros(3));
145 input Real t[3] (start = zeros(3));
146 equation
147 //Assign force and torque in interface model with input variables
148 fMITLMInterface3D.f = f;
149 fMITLMInterface3D3D.t = t;
150 [...]
    
```

|                  |          |        |       |
|------------------|----------|--------|-------|
| OPENCPS DOCUMENT |          |        | Pages |
|                  |          |        | 6/11  |
| Reference number | Revision | Author |       |
|                  |          |        |       |

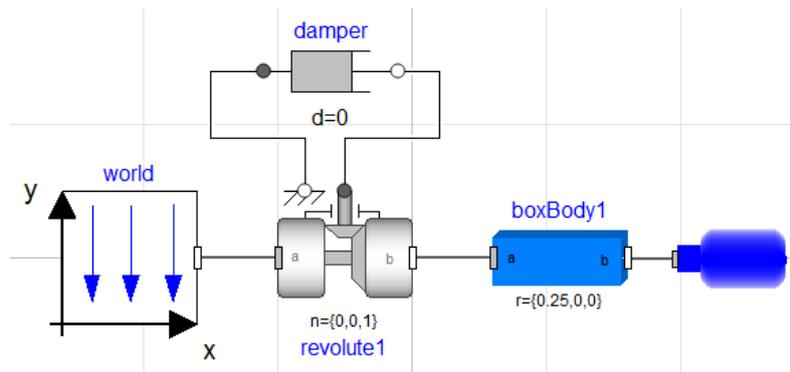


Figure 3: The first half of the first beam is modelled in Dymola

151 Now it is time to export the model so it can be used with the framework.

152 **1. Transform the model to an FMU for co-simulation.**

153 Use a suitable solver, for example CVODE.

154 **2. Put the .fmu file in a new subfolder.**

155 **3. Create an empty text file called fmi.config in the same folder.**

156 **4. Specify number of substeps on the first line.**

157 For example: "substeps,10"

158 **5. Open modelDescription.xml.**

159 By opening the .fmu as a zip package.

160 **6. Locate the TLM variables and write the value references in fmi.config.**

161 The desired variables are position, orientation, velocity, angular velocity, force and torque:

162 fMITLMInterface3D.r[x], fMITLMInterface3D.A[x,y],  
 163 fMITLMInterface3D.v[x], fMITLMInterface3D.w[x],  
 164 fMITLMInterface3D.f[x], fMITLMInterface3D.t[x]

165 **7. Enter the value reference for each variable in fmi.config.**

166 See section section 2 for more information.

167 **5.2. Preparing OpenModelica FMU of second part of pendulum**

168 The OpenModelica model as shown in fig. 4 consists of the second half of the first beam and the  
 169 second beam. The export process is exactly the same as for Dymola, except that the FMU exported  
 170 should be for model exchange.

|                  |          |        |       |
|------------------|----------|--------|-------|
| OPENCPS DOCUMENT |          |        | Pages |
|                  |          |        | 7/11  |
| Reference number | Revision | Author |       |
|                  |          |        |       |

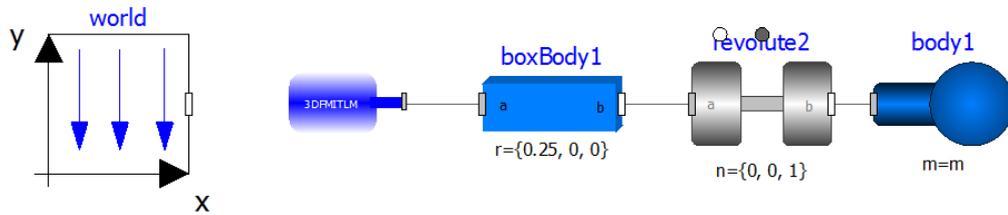


Figure 4: The second half of the first beam and the second beam are modelled in OpenModelica

171 **5.3. Building co-simulation system model**

172 When both FMUs have been generated together with one `fmi.config` file each, it is time to build  
 173 the meta model.

- 174 1. **Create a new meta model.**  
 175 Either by hand or by using the graphical user interface.
- 176 2. **Add the FMUs as sub-models.**  
 177 Use `StartTLMFmiWrapper` as start command.
- 178 3. **Fetch TLM interfaces.**
- 179 4. **Add a connection between the two models.**  
 180 Default TLM parameters should work fine.
- 181 5. **Make sure interfaces are aligned.**
- 182 6. **Choose solver for model exchange.**  
 183 By editing `StartTLMFmiWrapper.bat`.  
 184 Use for example `solver=CVODE` or `solver=IDA`.

185 When done, the meta model XML file should look similar to listing 6. In the graphical interface,  
 186 the meta model should look like fig. 5. The model is now ready to be simulated!

|                  |          |        |       |
|------------------|----------|--------|-------|
| OPENCPS DOCUMENT |          |        | Pages |
|                  |          |        | 8/11  |
| Reference number | Revision | Author |       |
|                  |          |        |       |

Listing 6: XML description of the complete meta model

```

<?xml version='1.0' encoding='ISO-8859-1'?>
<Model Name="doublePendulum">
  <SubModels>
    <SubModel Name="doublePendulum1"
      ModelFile="doublePendulum1.fmu"
      StartCommand="StartTLMFmiWrapper"
      Position="0,0,0"
      Angle321="0,0,0">
      <InterfacePoint Name="t1m"
        Position="0.25,0,0"
        Angle321="0,0,0"/>
    </SubModel>
    <SubModel Name="doublePendulum2"
      ModelFile="doublePendulum2.fmu"
      StartCommand="StartTLMFmiWrapper"
      Position="0.25,0,0"
      Angle321="0,0,0">
      <InterfacePoint Name="t2m"
        Position="0,0,0"
        Angle321="0,0,0"/>
    </SubModel>
  </SubModels>
  <Connections>
    <Connection From="doublePendulum1.t1m"
      To="doublePendulum2.t2m"
      Delay="1e-4"
      Zf="10000"
      Zfr="100"
      alpha="0.2">
    </Connection>
  </Connections>
  <SimulationParams ManagerPort="11113" StopTime="3" StartTime="0"/>
</Model>

```

|                  |          |        |       |
|------------------|----------|--------|-------|
| OPENCPS DOCUMENT |          |        | Pages |
|                  |          |        | 9/11  |
| Reference number | Revision | Author |       |
|                  |          |        |       |

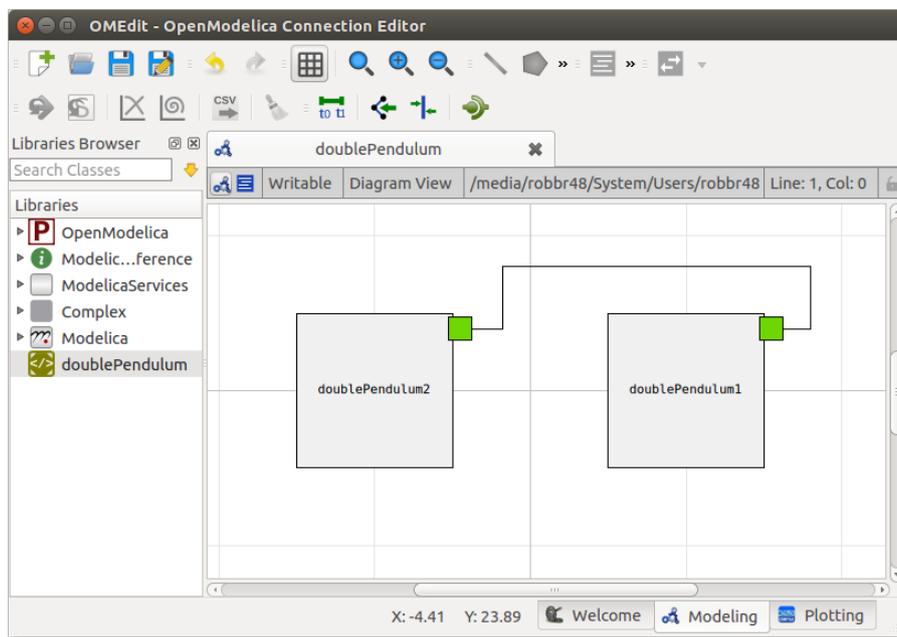


Figure 5: Graphical view of the complete meta model

|                  |          |        |       |
|------------------|----------|--------|-------|
| OPENCPS DOCUMENT |          |        | Pages |
|                  |          |        | 10/11 |
| Reference number | Revision | Author |       |
|                  |          |        |       |

# Bibliography

- 187
- 188 [1] Modelon AB. Fmi library: part of jmodelica.org, 2014.
- 189 [2] T. Blochwitz, M. Otter, M. Arnold, C. Bausch, C. Clauß, H. Elmqvist, A. Junghanns, J. Mauss,  
190 M. Monteiro, T. Neidhold, D. Neumerkel, H. Olsson, J.-V. Peetz, and S. Wolf. The Functional  
191 Mockup Interface for tool independent exchange of simulation models. In *8th International  
192 Modelica Conference 2011*, Como, Italy, September 2009.
- 193 [3] Alan C Hindmarsh, Peter N Brown, Keith E Grant, Steven L Lee, Radu Serban, Dan E Shumaker,  
194 and Carol S Woodward. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers.  
195 *ACM Transactions on Mathematical Software (TOMS)*, 31(3):363–396, 2005.

|                  |          |        |       |
|------------------|----------|--------|-------|
| OPENCPS DOCUMENT |          |        | Pages |
|                  |          |        | 11/11 |
| Reference number | Revision | Author |       |
|                  |          |        |       |