**Functional Mock-up Unit Export of EnergyPlus**

**Original: 2012-Nov-30**

**Updated: 12/10/2012 (Better definition of Key Value for EMS variables)**

**Updated 01/10/2013 (changed in key field names)**

**Thierry Stephane Nouidui (LBNL), Michael Wetter (LBNL), David Lorenzetti (LBNL)**

**Justification for Feature**:

EnergyPlus 7.1 was extended to support the import of simulation programs that are packaged as Functional Mock-up Unit (FMU) for co-simulation.

FMUs are formally specified in the Functional Mock-up Interface (FMI) standard, an open standard designed to enable links between disparate simulation programs. The standard is available from [http://www.functional-mockup-interface.org/](http://www.functional-mockup-interface.org/.%20). A FMU comes in form of a zip file, which may contain models, model descriptions, source code, and executable programs for multiple platforms.

The current NFP aims to export and expose EnergyPlus itself as a FMU for co-simulation. This will allow other software tools to import and run EnergyPlus as part of a larger simulation. To do so, the outside software needs to implement the FMI standard and be able to import FMUs for co-simulation. This new feature will significantly increase and facilitate the co-simulation of EnergyPlus with different simulation programs.

**Conference Call Conclusions**:

**Other Conference Call Topics (not in scope of current proposal):**

* N/A

**Overview:**

The FMU export of EnergyPlus allows EnergyPlus to be accessed from other simulation environments, as an FMU. An FMU is a software component that implements the FMI standard. The FMI standard consists of two main parts.

* The first part is FMI for model exchange. This part of the standard specifies how a modeling environment can generate C-code of a dynamic system model for use by other modeling and simulation environments. The exported model is independent of the target simulator (Modelisar, 2010a).
* The second part is FMI for co-simulation, an interface standard for coupling two or more simulation programs in a co-simulation environment. The data exchange between sub-systems is restricted to discrete communication points in time. In the time between two communication points, the sub-systems are solved independently from each other by their individual solvers. A master algorithm controls the data exchange between sub-systems, and synchronizes all slave simulation programs (slaves). All information about the slaves, which is relevant for the communication in the co-simulation environment, is provided in a slave specific XML-file (Modelisar, 2010b).

Here is important to note that the FMI for Co-Simulation is designed both for coupling with subsystem models, which have been exported by their simulators together with its solvers as runnable code (), and for coupling of simulation tools (tool coupling ()).

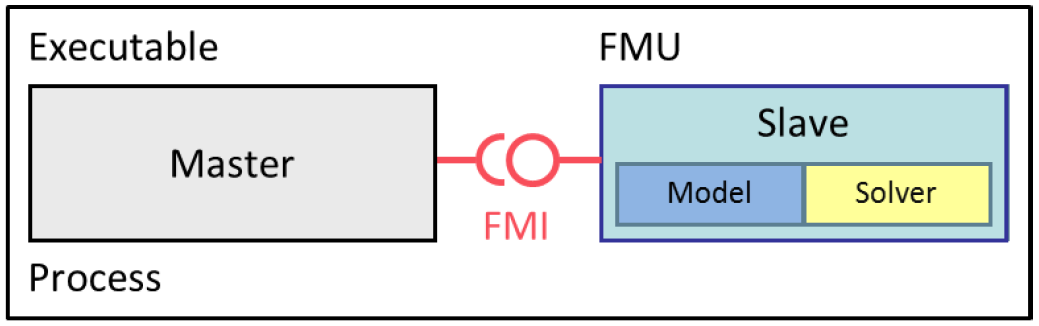


Figure 1: Co-simulation with generated code on a single computer (Modelisar, 2012c)

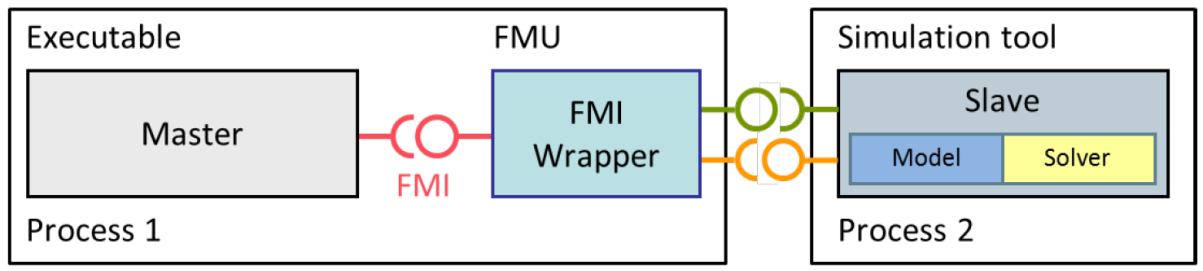


Figure 2: Co-simulation with tool coupling on a single computer (Modelisar, 2012c)

In the tool coupling case the FMU implementation wraps the FMI function calls to API calls which are provided by the simulation tool. Additionally to the FMU the simulation tool is needed to run a co-simulation.

In the current implementation, EnergyPlus is exported and exposed to simulation programs as an FMU with FMI 1.0 for co-simulation API for tool coupling.

**Approach:**

To export EnergyPlus as an FMU, we developed a parser which exports EnergyPlus as an FMU. To link EnergyPlus to simulation programs which can import FMUs, the EnergyPlus module *ExternalInterface* has been modified so that data can be exchanged during run-time between EnergyPlus and the simulation programs. In the current implementation, the data exchange is done at each zone time step. The data exchange is synchronized by the master simulation program. Minor additions have been made to DataRuntimeLanguage.f90, EMSManager.f90, and ScheduleManager.f90. These modifications were made to define and make the objects available in the different modules.

**Architecture of the FMU Export of EnergyPlus**

shows the architecture of the connection between EnergyPlus and a simulation program that can import FMU for co-simulation. EnergyPlus exposes its input and output variables to the simulation program through a new set of EnergyPlus objects (see the IO Ref section, below). The simulation program can then be linked to EnergyPlus for co-simulation. There are currently more than 20 tools that can import FMUs for co-simulation (See [https://www.fmi-standard.org/tools](https://www.fmi-standard.org/tools%20) ).

The **ExternalInterface:FunctionalMockupUnitExport:From:Variable** object is used to map the outputs of the FMU to the EnergyPlus objects **Output:Variable** and **EnergyManagementSystem:OutputVariable**.

The **ExternalInterface:FunctionalMockupUnitExport:To:Schedule**, **ExternalInterface:FunctionalMockupUnitExport:To:Actuator**, and **ExternalInterface:FunctionalMockupUnitExport:To:Variable** are used to map the inputs of the FMU to EnergyPlus schedule and EMS actuators and variables.

The ExternalInterface:FunctionalMockupUnitExport:To:Schedule can be used to overwrite schedules, and the other two objects can be used in place of Energy Management System (EMS) actuators and EMS variables. The objects have similar functionality as the objects Schedule:Compact, EnergyManagementSystem:Actuator and EnergyManagementSystem:GlobalVariable, except that their numerical value is obtained from the external interface at the beginning of each zone time step, and will remain constant during this zone time step.

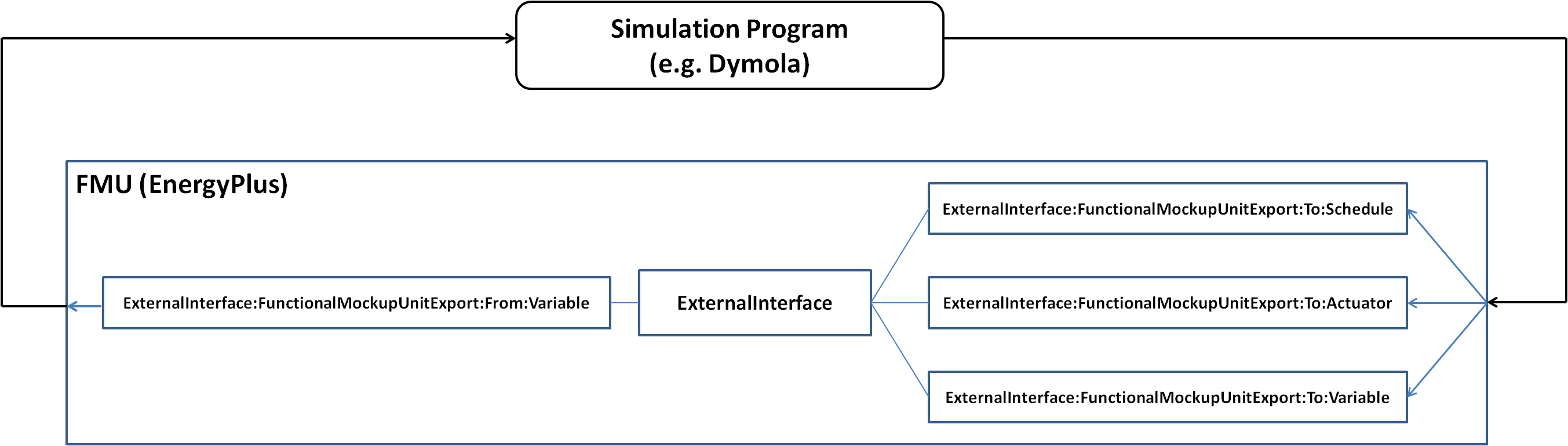
****

Figure 3: Architecture of the FMU Export of EnergyPlus

**Workflow of the FMU Export of EnergyPlus**

To export EnergyPlus as an FMU, the user first creates an IDF file which declares the input and output variables to be exposed to the co-simulation master. These declarations are made via the following new EnergyPlus objects:

* ExternalInterface:FunctionalMockupUnitExport:From:Variable,
* ExternalInterface:FunctionalMockupUnitExport:To:Schedule,
* ExternalInterface:FunctionalMockupUnitExport:To:Actuator,
* ExternalInterface:FunctionalMockupUnitExport:To:Variable.

The user then runs the EnergyPlusToFMU tool, to create the FMU.

**EnergyPlusToFMU**

The **EnergyPlusToFMU** is a batch/shell script. It parses an IDF input-file, and generates a Functional Mockup Unit (FMU) for co-simulation. The system command that is used to process the input file is

**EnergyPlusToFMU** xxx.idf, where xxx.idf is a valid EnergyPlus idf file.

The steps involved inEnergyPlusToFMUare

* checking whether the requirements specified for the input file are met,
* writing a model description file (**modelDescription.xml**)file, which contains the definition of all exposed variables in the FMU and other static information

required to run the FMU in a co-simulation environment,

* writinga**variables.cfg** filewhich lists the variables to be exchanged,
* compiling the binaries for the target platform,
* compressing binaries, input file, and any resources into a zip file which will have the same name as the input file.

This zip file, which will have the name xxx.fmu, will be the file that can be imported into a co-simulation master.

The batch/shell script will not be distributed with EnergyPlus, but rather be available from <http://SimulationResearch.lbl.gov> because its use does not require an EnergyPlus distribution.

**Testing/Validation/Data Source(s):**

N/A

**IO Ref (draft):**

**Group – ExternalInterface**

### **ExternalInterface**

The external interface that has been implemented in EnergyPlus for linking EnergyPlus with the BCVTB and FMU has been modified. It has a new entry **FunctionalMockupUnitExport**. If this entry exists in the EnergyPlus input-file, the **FunctionalMockupUnitExport** objects listed below will be activated.

#### Field: Name

This field contains the name of the external interface. The valid entries are **PtolemyServer**, **FunctionalMockupUnitImport**, and **FunctionalMockupUnitExport**.

ExternalInterface,

A1 ; \field Name of external interface

\required-field  
 \type choice

\key PtolemyServer

\key FunctionalMockupUnitImport \key FunctionalMockupUnitExport

\note Currently, the only valid entries are PtolemyServer,

\note FunctionalMockupUnitImport, and FunctionalMockupUnitExport.

**IDD Object (New):**

### **ExternalInterface:FunctionalMockupUnitExport:From:Variable**

This object exposes the output variables of EnergyPlus (**Output:Variable** or **EnergyManagementSystem:OutputVariable**) to the outside simulation program.

#### Field: Output:Variable Index Key Name

This field contains a Key Value for an EnergyPlus output variable. For an **EnergyManagementSystem:OutputVariable**, the Output:Variable Index Key Name need to be set to “**EMS**”.

#### Field: Output:Variable Name

This field contains the Variable Name as defined in the Input Output Reference. For an **EnergyManagementSystem:OutputVariable**, the Output:Variable Name is the name of the **EnergyManagementSystem:OutputVariable**’s object.

#### Field: FMU Variable Name

This field contains the name of the variable in the model description file of the FMU that will be mapped to the corresponding variable in EnergyPlus.

ExternalInterface:FunctionalMockupUnitExport:From:Variable,

\memo This object declares an FMU input variable

\min-fields 3

A1 , \field Output:Variable Index Key Name

\required-field

\type alpha

A2 , \field Output:Variable Name

\required-field

\type alpha

A3 ; \field FMU Variable Name

\required-field

\type alpha

\retaincase

### **ExternalInterface:FunctionalMockupUnitExport:To:Schedule**

This object is similar to Schedule:Compact. However, during the time stepping, its value is set to the value received from the external interface. During the warm-up period and the system sizing, its value is set to the value specified by the field “initial value.”

#### Field: Schedule Name

This field contains a unique (within all DaySchedules) designation for this schedule in EnergyPlus. It is referenced by WeekSchedules to define the appropriate schedule values.

#### Field: Schedule Type Limits Name

This field contains a reference to the Schedule Type Limits object. If found in a list of Schedule Type Limits (see above), then the restrictions from the referenced object will be used to validate the current field values.

#### Field: FMU Variable Name

This field contains the name of the variable in the model description file of the FMU that will be mapped to the schedule in EnergyPlus.

#### Field: Initial Value

This field contains the schedule value that is used during the warm-up period and during the system sizing.

ExternalInterface:FunctionalMockupUnitExport:To:Schedule,

\memo This objects contains only one value, which is used during the first

\memo call of EnergyPlus

\min-fields 4

A1 , \field Schedule Name

\required-field

\type alpha

\reference ExternalInterfaceScheduleNames

A2 , \field Schedule Type Limits Names

\type object-list

\object-list ScheduleTypeLimitsNames

A3 , \field FMU Variable Name

\required-field

\type alpha

\retaincase

N1 ; \field Initial Value

\type real

\required-field

\note Used during warm-up and system sizing.

### **ExternalInterface:FunctionalMockupUnitExport:To:Actuator**

This object maps a value received from the external interface to an actuator of the Energy Management System. The object is similar to EnergyManagementSystem:Actuator. However, during the time stepping, its value is set to the value received from the external interface. During the warm-up period and the system sizing, its value is set to the value specified by the field “initial value.”

#### Field: Name

This field contains a unique name for the actuator. No spaces are allowed in the object name. This name will be a global read-only variable in Erl programs and cannot duplicate any other global scope Erl variable.

#### Field: Actuated Component Unique Name

This field defines a unique name for the specific entity that is to be controlled. The names for each individual component are listed in the EDD output file when Verbose mode is used – see the input object **Output:EnergyManagementSystem** for more on the EDD file. These will often be user-defined names of input objects or system nodes, but some actuators are automatically setup by the program and will not be completely user-defined.

#### Field: Actuated Component Type

The field defines the type of the entity that is to be controlled by the actuator. The component types available vary with the specifics of individual models. The types of components that can be used as actuators in a specific model are listed in the EDD output file – see the input object **Output:EnergyManagementSystem** for more on the EDD file. Components can be object types defined elsewhere in the IDD but there are other types of entities such as nodes and system-level actuators that do not directly correspond to IDF objects.

#### Field: Actuated Component Control Type

This field defines the type of control to be done on the specific entity being controlled. The control types available are listed in the EDD output. Specific components may have more than one type of control available, such as flow rate or temperature, and this field is used to distinguish between them.

#### Field: FMU Variable Name

This field contains the name of the variable in the model description file of the FMU that will be mapped to the actuator in EnergyPlus.

#### Field: Initial Value

This field contains the initial value. If a value is specified, then this value is used during the warm-up period and the system sizing. If no value is specified, then the actuated component will only be updated once the time stepping starts, i.e., after the warm-up and the system-sizing.

ExternalInterface:FunctionalMockupUnitExport:To:Actuator,

\memo Hardware portion of EMS used to set up actuators in the model

\memo that are dynamically updated from the FMU.

\min-fields 6

A1 , \field Name

\required-field

\type alpha

\note This name becomes a read-only variable for use in Erl programs

\note no spaces allowed in name

A2 , \field Actuated Component Unique Name

\required-field

\type alpha

A3 , \field Actuated Component Type

\required-field

\type alpha

A4 , \field Actuated Component Control Type

\required-field

\type alpha

A5 , \field FMU Variable Name

\required-field

\type alpha

\retaincase

N1 ; \field Initial Value

\type real

\required-field

\note Used during warm-up and system sizing.

### **ExternalInterface:FunctionalMockupUnitExport:To:Variable**

This input object is similar to EnergyManagementSystem:GlobalVariable. However, during the time stepping, its value is set to the value received from the external interface. During the warm-up period and the system sizing, its value is set to the value specified by the field “initial value.” This object can be used to move data into Erl subroutines.

#### Field: Name

This field becomes the global Erl variable name that can be referenced in the EnergyPlus Runtime Language. No spaces are allowed in the object name. The name must be unique across all global scope variables including those declared as sensor and actuators and the built-in variables.

#### Field: FMU Variable Name

This field contains the name of the variable in the model description file of the FMU that will be mapped to the corresponding variable in EnergyPlus.

#### Field: Initial Value

This field contains the initial value that is used during the warm-up period and during the system sizing.

ExternalInterface:FunctionalMockupUnitExport:To:Variable,

\memo Declares Erl variable as having global scope

\memo No spaces allowed in names used for Erl variables

\min-fields 3

A1 , \field Name

\required-field

\type alpha

\note This name becomes a variable for use in Erl programs

\note no spaces allowed in name

A2 , \field FMU Variable Name

\required-field

\type alpha

\retaincase

N1 ; \field Initial Value

\type real

\required-field

\note Used during warm-up and system sizing.

Proposed Report Variables: None

Proposed additions to Meters: None

**EngRef (draft):**

This engineering reference describes the algorithm for exchanging data between EnergyPlus, packaged as an FMU, and an external program.

Suppose we have a system with two simulation programs. Let simulation program 1 be EnergyPlus, the slave simulation program, which is packaged as an FMU for co-simulation; and let simulation program 2 be the master simulation program which supports the import of FMU for co-simulation. Suppose each program solves an initial-value ordinary differential equation that is coupled to the differential equations of the other program.

Let  denote the number of time steps and let  denote the time steps. We will use the subscripts *1* and *2* to denote the state variable and the function that computes the next state variable of the simulator *1* and *2*, respectively.

Then program *1* computes, for the sequence

*x1(k+1) = f1(x1(k), x2(k))*

and, similarly, program *2* computes the sequence

*x2(k+1) = f2(x2(k), x1(k))*

with initial conditions *x1(0) = x1,0* and *x2(0) = x2,0*.

To advance from time *k* to *k+1*, each program uses its own integration algorithm. At the end of the time step, program *1* sends its new state *x1(k+1)* to program 2, and receives the state *x2(k+1)* from program 2. The same procedure is done with the program *2*. Program 2, which is the master simulation program, imports the FMU, and manages the data-exchange between the two programs.

In comparison to numerical methods of differential equations, this scheme resembles an explicit Euler integration, which is an integration algorithm that solves an ordinary differential equation with specified initial values,

*dx/dt = h(x),*

*x(0) = x0,*

on the time interval *t ∈ [0, 1]*, the following sequence:

|  |  |
| --- | --- |
| **Step 0:** | Initialize counter *k=0* and number of steps . |
|  | Set initial state *x(k) = x0* and set time step *Δt = 1/N*. |
| **Step 1:** | Compute new state *x(k+1) = x(k) + h(x(k)) Δt*. |
|  | Replace *k* by *k+1.* |
| **Step 2:** | If *k=N* stop, else go to Step 1. |

However, this scheme does not require each simulation tool to use explicit Euler for its internal time-stepping; the analogy to explicit Euler applies only to the data exchange between programs. In the situation where the differential equation is solved using co-simulation, the above algorithm becomes

|  |  |
| --- | --- |
| **Step 0:** | Initialize counter *k=0* and number of steps . |
|  | Set initial state *x1(k) = x1,0* and *x2(k) = x2,0*. Set the time step *Δt = 1/N*. |
| **Step 1:** | Compute new states  *x1(k+1) = x1(k) + f1(x1(k), x2(k)) Δt*, and  *x2(k+1) = x2(k) + f2(x2(k), x1(k)) Δt*. |
|  | Replace *k* by *k+1*. |
| **Step 2:** | If *k=N* stop, else go to Step 1. |

# References

Modelisar. 2010a. “Functional Mock-up Interface for Model Exchange.” [https://svn.modelica.org/fmi/branches/public/specifications/FMI\_for\_ModelExchange\_v1.0.pdf](https://svn.modelica.org/fmi/branches/public/specifications/FMI_for_ModelExchange_v1.0.pdf%20) [Last *accessed: 09/30/2012]*.

Modelisar. 2010b. “Functional Mock-up Interface for Co-Simulation.” [https://svn.modelica.org/fmi/branches/public/specifications/FMI\_for\_CoSimulation\_v1.0.pdf](https://svn.modelica.org/fmi/branches/public/specifications/FMI_for_CoSimulation_v1.0.pdf%20) *[Last accessed: 09/30/2012]*.

Modelisar. 2010c. “FMI for Model Exchange and Co-Simulation, version 2.0 Beta 4.” [https://svn.modelica.org/fmi/branches/public/specifications/FMI\_for\_ModelExchange\_and\_CoSimulation\_v2.0\_Beta4.pdf](https://svn.modelica.org/fmi/branches/public/specifications/FMI_for_ModelExchange_and_CoSimulation_v2.0_Beta4.pdf%20) *[Last accessed: 09/30/2012]*.