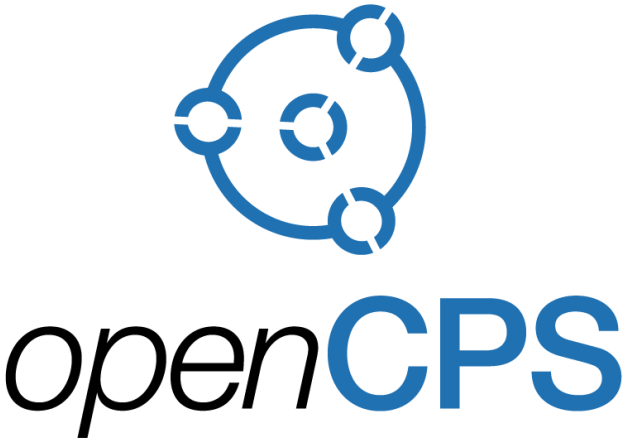
D2.3	Test cases for MST with evaluation
Access[1]:	**PU**
Type[2]:	**Report**
Version:	**1.0**
Due Dates[3]:	**M36**



Open Cyber-Physical System Model-Driven Certified Development

Executive summary[4]:

This document contains test cases for the Master Simulation Tool and FMU export tool requirements. The test cases verify the ability of the Master Simulation Tool to execute the simulation of such models correctly. The evaluation of the test cases is attached to each of the tests. The test results provided are produced on a 64-bit Windows 8 operating system, with MST version v2.0.0-dev-860-gf06327c-mingw

1 Access classification as per definitions in PCA; PU = Public, CO = Confidential. Access classification per deliverable stated in FPP.
2 Deliverable type according to FPP, note that all non-report deliverables must be accompanied by a deliverable report.
3 Due month(s) according to FPP.
4 It is mandatory to provide an executive summary for each deliverable.

Deliverable Contributors:

	Name	Organisation	Primary role in project	Main Author(s)[5]
Deliverable Leader[6]	Sébastien REVOL	CEA	T2.3 Leader	
Contributing Author(s)[7]	Zoltán GERA	ELTE-Soft	Contributor	X
	Boldizsár NÉMETH	ELTE-Soft	Contributor	X
	András NAGY	ELTE-Soft	Contributor	
	Máté SZOKOLAI	ELTE-Soft	Contributor	
Internal Reviewer(s)[8]	Sébastien REVOL	CEA	T2.3 Leader	

Document History:

Version	Date	Reason for Change	Status[9]
1.0	20/11/2018	First Version	Final

5 Indicate Main Author(s) with an "X" in this column.
6 Deliverable leader according to FPP, role definition in PCA.
7 Person(s) from contributing partners for the deliverable, expected contributing partners stated in FPP.
8 Typically person(s) with appropriate expertise to assess deliverable structure and quality.
9 Status = "Draft", "In Review", "Released".

CONTENTS

ABBREVIATIONS

List of abbreviations/acronyms used in document:

Abbreviation	**Definition**
FMI	Functional Mock-up Interface
FMU	Functional Mock-up Unit
MST	Master Simulation Tool

1.1.1 Connectivity – One model

Id	test-connectivity-one-model
Tested requirement	Req1

1.1.1.1 Description

Given an OpenModelica model that has a single output variable
and this model exported as an FMU
when the model is simulated
then the value of the output parameter will increase linearly.

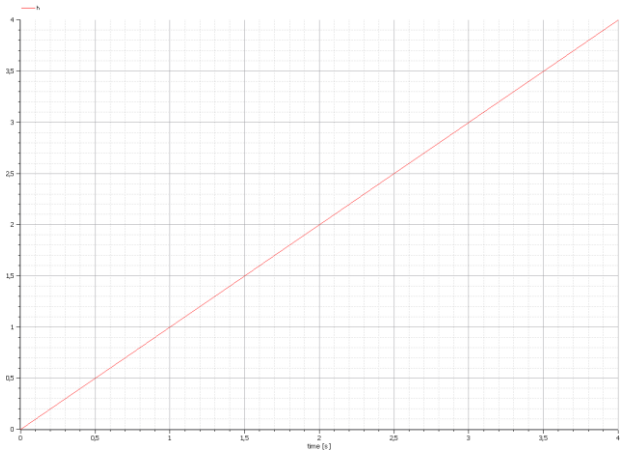
1.1.1.2 Input

The test input is an FMU model, exported from OpenModelica, that has one output variable:

```
model OneModel
  output Real h;
initial equation
  h = 0;
equation
  der(h) = 1;
end OneModel;
```

1.1.1.3 Output

When running the simulation, the master simulation tool should display a constantly increasing output.



1.1.1.4 Rationale

The model validates that the master simulation tool correctly evaluates a model in itself.

1.1.1.5 Results

Building up the connections between models in two different FMUs is not successful (the problem is probably related to variable names in the models). However when exporting the models using the connector, the resulting FMU yields the expected results.

1.1.2 Connectivity – One direction

Id	test-connectivity-one-direction
Tested requirement	Req1

1.1.2.1 Description

Given two Modelica models that are in a one-direction relationship
and these models are exported as an FMU
when the models are simulated
then the value of the output parameter will increase quadratically.

1.1.2.2 Input

The test input is two FMU models exported from OpenModelica. One model has only an output variable while the other has input and output variables. The output of the first model should be connected to the input of the second model.

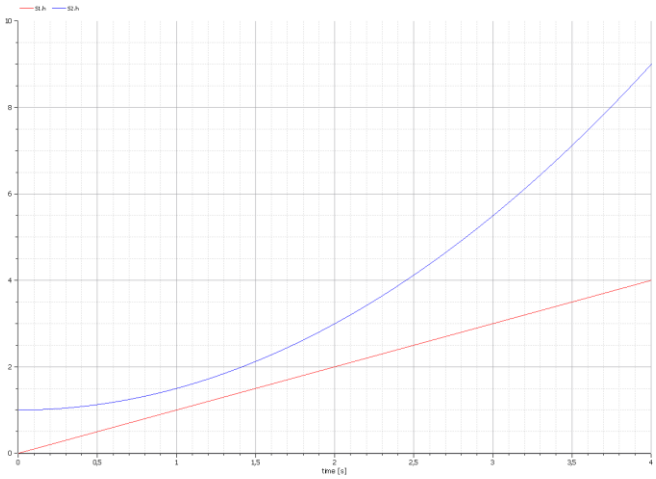
```
package OneDirection
  model Source
    output Real h;
  initial equation
    h = 0;
  equation
    der(h) = 1;
  end Source;

  model Target
    input Real d;
    output Real h;
  initial equation
    h = 1;
  equation
    der(h) = d;
  end Target;

  model OneDir
    Source S1;
    Target S2;
  equation
    connect(S1.h, S2.d);
  end OneDir;
end OneDirection;
```

1.1.2.3 Output

When running the simulation, one of the model outputs will be linearly increasing, the other will be quadratically increasing.



1.1.2.4 Rationale

The master simulation tool should allow the loaded models to be connected by their input and output variables. It should allow the connection to be made if the type of the variables are the same and one is input while the other is an output variable. With the connections made, the values should be passed from one model to the other at each step of the simulation.

1.1.2.5 Results

Building up the connections between models in two different FMUs is not successful (the problem is probably related to variable names in the models). However when exporting the models using the connector, the resulting FMU yields the expected results.

1.1.3 Connectivity – Two directions

Id	test-connectivity-two-direction
Tested requirement	Req1

1.1.3.1 Description

Given two Open Modelica models that are in a two-direction relationship
and these models are exported as an FMU
when the models are simulated
then the value of the output parameters will increase together.

1.1.3.2 Input

The test input is two FMU models exported from OpenModelica. Both models have one input and one output variable. The output of the first model should be connected to the input of the second model and vice versa.

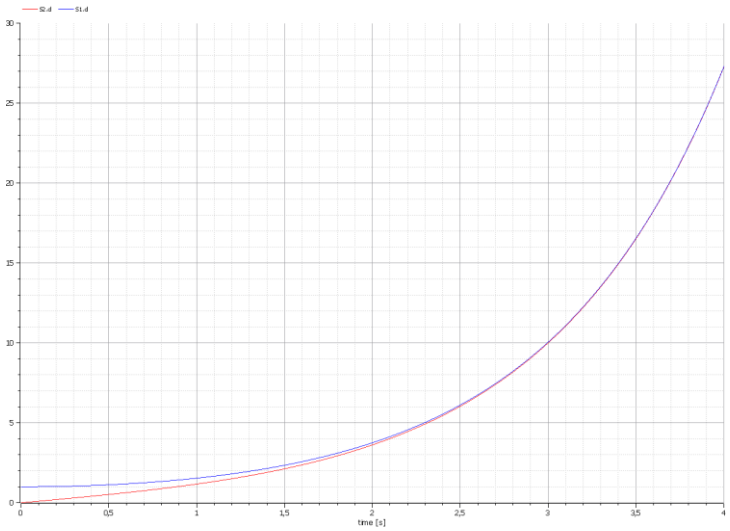
```
package TwoWayConnection
  model A
    output Real h;
    input Real d;
  initial equation
    h = 0;
  equation
    der(h) = d;
  end A;

  model B
    output Real h;
    input Real d;
  initial equation
    h = 1;
  equation
    der(h) = d;
  end B;

  model Connect
    A S1;
    B S2;
  equation
    connect(S1.h, S2.d);
    connect(S2.h, S1.d);
  end Connect;
end TwoWayConnection;
```

1.1.3.3 Output

When running the simulation, both models output variables will be non-linearly increasing together.



1.1.3.4 Rationale

The master simulation tool should allow the loaded models to be connected by their input and output variables. It should allow the connection to be made if the type of the variables are the same and one is input while the other is an output variable. With the connections made, the values should be passed from one model to the other at each step of the simulation. The master simulation tool should have no problem when the connections form a loop.

1.1.3.5 Results

Building up the connections between models in two different FMUs is not successful (the problem is probably related to variable names in the models). However when exporting the models using the connector, the resulting FMU yields the expected results.

1.1.4 FMI compliance – Best-case behavior

Id	test-fmi-complience-best-case
Tested requirement	Req24

1.1.4.1 Description

Given an FMU defined using a C++
when the models are simulated
then the model will run without notifying errors.

1.1.4.2 Input

An FMU model that has no input or output variables, but checks that the master behaves according to the FMI standard.

```
fmi2Status fmi2DoStep(
    fmi2Component c,
    fmi2Real currentCommunicationPoint,
    fmi2Real communicationStepSize,
    fmi2Boolean noSetFMUStatePriorToCurrentPoint) {
  return fmi2OK;
}
```

Output

The simulation should be successful, without any messages written to the standard error, or the corresponding log channel of the master simulation tool.

1.1.4.3 Rationale

The FMU validates the master simulation tool by checking that it only calls the functions declared by the FMI standard in states where they are valid.

1.1.4.4 Results

The simulation runs without any error messages.

1.1.5 FMI compliance – callback usage

Id	test-fmi-complience-callbacks
Tested requirement	Req24

1.1.5.1 Description

Given an FMU defined using a C++
when the models are simulated
then the model will run without notifying errors.

1.1.5.2 Input

An FMU model that has one output variable and checks that the master behaves according to the FMI standard.
The simulation should be successful, without errors. The master simulation tool writes log messages at each simulation step. The log messages should be: ”Just info”, ” Just a warning”, ” An error message”, ” A fatal message”, in the info, warning, error and fatal channels respectively. It should also log ” h (variable name here) #1 (hashmark here)” at each step.

```
fmi2Status fmi2DoStep(
    fmi2Component c,
    fmi2Real currentCommunicationPoint,
    fmi2Real communicationStepSize,
    fmi2Boolean noSetFMUStatePriorToCurrentPoint) {
  fmi2CallbackFunctions* cf
    = (fmi2CallbackFunctions*) c;
  // test logging
  cf->logger(cf->componentEnvironment, "Cat1", fmi2OK, "Cat1", "Just
info");
  cf->logger(cf->componentEnvironment, "Cat2", fmi2Warning, "Cat2",
"Just a warning");
  cf->logger(cf->componentEnvironment, "Cat1", fmi2Error, "Cat1", "An
error message");
  cf->logger(cf->componentEnvironment, "Cat2", fmi2Fatal, "Cat2", "A
fatal message");


  cf->logger(cf->componentEnvironment, "Cat1", fmi2OK, "Cat1", "#r1#
(variable name here) ##1 (hashmark here)");

  // test allocate and deallocate
  fmi2Byte* allocated = static_cast<fmi2Byte*>(cf->allocateMemory(8,
8));
  for (int i = 0; i < 64; ++i) {
    allocated[i] = i;
  }
  cf->freeMemory(allocated);

  return fmi2OK;
}
```

1.1.5.3 Rationale

The FMU validates the master simulation tool by checking that the callback functions passed to the FMU are valid. The allocation, deallocation and logging features are tested.

1.1.5.4 Results

The messages that should be displayed after the simulation are not found. The problem may be a technical issue with retrieving the logs or an issue with the logging system of MST.

1.1.6 FMI compliance – null value when initialized

Id	test-fmi-compliance-init-null
Tested requirement	Req24

1.1.6.1 Description

Given an FMU defined using a C++
when the models are simulated
then the model will handle the failure gracefully.

1.1.6.2 Input

An FMU model thats initialization returns NULL.
Running the simulation should display an error to the user that the initialization of the FMU was not successful.

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

1.1.6.3 Rationale

The master simulation tool should handle when the initialization of a model returns NULL and inform the user.

1.1.6.4 Results

During the testing, the expected error did not happen, and the simulation is actually executed. This could mean that the MST does not adhere to the FMI standard in this instance.

1.1.7 Integration test – Moon Landing

Id	test-integration-moon-landing
Tested requirement	Integration test

1.1.7.1 Description

Given an FMU defined using txtUML and an FMU defined using OpenModelica
when the models are simulated
then the model will behave correctly, as specified below.

1.1.7.2 Input

A txtUML model that defines the control logic of the lander. And an OpenModelica model that defines the environment in which the control logic runs. They are connected by multiple variables that represent the distance of the lander from the surface, its speed, the remaining fuel and the acceleration.

```
model World
output Real h(start = 1000), v(start = 0), m(start = 1000);
input Real u;
parameter Real g = 1.63, k = 500;
equation
   der(h) = v;
   der(v) = k * u / m - g;
   der(m) = -u;
end World;
```

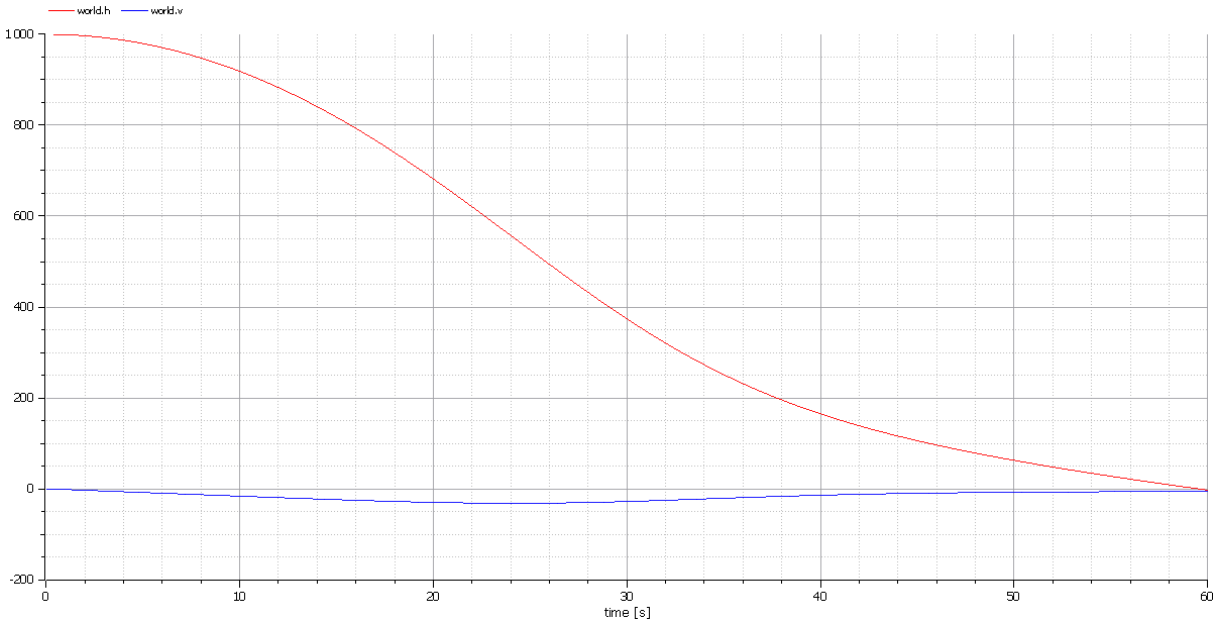
```
    public class Controlled extends State {

     @Override
     public void entry() {
          World world =
assoc(LanderWorld.world.class).selectAny();
          // better because of rounding in generated code
          Action.send(new
ControlSignal(world.v*world.h*world.h/500000 - (-
world.v*world.h/1000) - world.v/2), world);
     }
    }

    @Trigger(ControlCycleSignal.class)
    @From(Controlled.class)
    @To(Controlled.class)
    public class StillControlled extends Transition {
     @Override
     public boolean guard() {
          ControlCycleSignal signal
               = getTrigger(ControlCycleSignal.class);
          return signal.h >= 1;
     }
    }
```

1.1.7.3 Output

The expected result of the simulation is that the lander unit lands on the surface in a controlled way, with velocity less than 10m/s. The lander unit should be able to land around 60s of the simulation, but at most 100s. After the landing the simulated variables are irrelevant.



1.1.7.4 Rationale

This scenario is a case study for a simulation thats control logic is defined by an UML model, while the behavior of the environment is controlled by mathematical equations.

1.1.7.5 Results

Exporting and loading the models were successful. Simulating parts of the models have shown correct results. However, integrated testing yielded erroneous results.