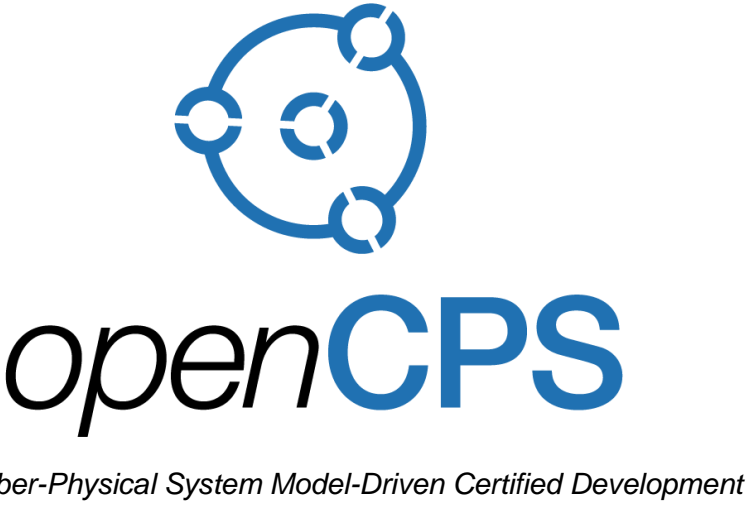


D2.1	FMI Master Simulation Tool Requirement Specification
Access ¹ :	PU
Type ² :	Report
Version:	1.1
Due Dates ³ :	M12
 <p><i>Open Cyber-Physical System Model-Driven Certified Development</i></p>	
Executive summary⁴:	
<p>One of the core areas of the OPENCPS project is the development of an open source FMI Master Simulation Tool (MST). This report provides a number of identified use cases from which functional and non-functional MST requirements have been derived. In the requirements definition phase, the ambition has been to obtain as a complete picture as possible regarding required MST functionality, ensuring industrial relevance in terms of simulation efficiency, robustness, and scalability. However, development of a MST compliant with all identified requirements is not in the scope of the OPENCPS project. Rather, the project is focused on the most essential requirements enabling realization of the industrial demonstrators in WP6. This requirement specification will guide and prioritize further development and implementation efforts in WP 2-5.</p>	

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	Hällqvist Robert	Saab	T2.1, T5.5, T6.3 Leader	X
Contributing Author(s) ⁷	Saadia DHOUIB	CEA List	WP2 Leader	X
	Francois DUBE	ESI	T2.1 Member	
	Francisco Gómez	KTH	T2.1 Member	
	Magnus Eek	Saab	Project Co.	
	Tommi Karhela	VTT	NC Finalnd	
	Sune Horkeby	Siemens TU	NC Sweden, T6.2 Leader	
	Bernhard Thiele	SICSEast	WP3 Leader	
	Martin Sjölund	LiU	WP4 Leader	
	Philippe Fiani	Sherpa	NC France, WP6 Leader	
	Akos Horvath	IncQuery Labs	NC Hungary	
	Gergely Dévai	ELTE-Soft	T2.1 Member	
	Robert Braun	LiU/SKF	T2.4, T5.6 Leader	
Internal Reviewer(s) ⁸				
	Sebastien Revol	CEA	T2.1 Member	
	Peter Fritzon	LiU	Scientific Co.	

Document History:

Version	Date	Reason for Change	Status ⁹
0.1	03/06/2016	First Draft Version	Draft
0.2	04/11/2016	Second Draft Version	In Review
1.0	14/11/2016	First Release	Released
1.1	24/11/2016	Second Release	Released

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.....	3
1 FUNCTIONAL REQUIREMENTS	4
1.1 Actors.....	4
1.2 Use Cases	5
1.3 Requirements	28
1.4 Requirements Diagram	36
2 NON FUNCTIONAL REQUIREMENTS	42
3 VERIFICATION.....	44
3.1 Test Cases	45
4 NON MST REQUIREMENTS	49
4.1 FMU export tool requirements.....	49
4.2 FMI standard extension requirements.....	50
5 REFERENCES	50
Appendix	51

ABBREVIATIONS

List of abbreviations/acronyms used in document:

Abbreviation	Definition
FMI	Functional Mock-up Interface
FMU	Functional Mock-up Unit
M&S	Modelling and Simulation
MST	Master Simulation Tool
UML	Unified Modeling Language
WP	Work Package
N/A	Not Applicable

1 FUNCTIONAL REQUIREMENTS

Functional requirements describe what the system should do to be useful within the stakeholders context. To specify these requirements for the Master Simulation Tool (MST), OPENCPS partners have provided use case diagrams describing the expected functionalities of the MST. Consequently all the functional requirements have been derived from the use cases and shown in SysML requirements diagrams as well as being described textually.

1.1 Actors

1.1.1 Hardware/Physical Systems Model Developer:

The model developer designs models of physical systems which can be exported as FMUs.

1.1.2 Systems Engineer:

The systems engineer utilizes developed models and simulators to develop and analyse sub-system behavior.

1.1.3 Software Model Developer:

The software model developer designs models of software systems, e.g. control systems, which can be exported as FMUs. These models cover continuous control systems and discrete event-based software.

1.1.4 Test Engineer:

The test engineer utilizes test rigs and models/simulators for system verification, including verification of software functionality.

1.1.5 Simulation Model Integrator:

The simulation model integrator integrates models of hardware and software into simulators.

1.1.6 Simulation Tool Developer:

The simulation tool developer is responsible for the maintenance of the simulator platform (the integrating environment).

1.1.7 Software Simulation Tool:

A software simulation tool is a tool that enable modeling and simulation of parts of cyber physical systems. Examples are: Matlab-Simulink, OpenModelica, Dymola, etc.

1.1.8 Domain Expert:

The domain expert supports the systems engineer in refinement of system behaviours depending on specific domain expertise (electromagnetism, fluid dynamics, mechanical dynamics, electronic, embedded software, ...)

1.2 Use Cases

To represent the functional requirements of the MST, the OPENCPS partners have chosen to adopt UML Use Case diagrams to derive functional and non-functional requirements on the MST. Use Cases are already in wide-spread practice for graphically representing functional requirements in common methodologies, such as Object-Oriented Software Engineering (OOSE). The key concepts specified in Use Case diagrams are Actors, Use Cases, and subjects. Each Use Case subject represents a system under consideration to which the Use Case applies. Users and any other systems that may interact with a subject are represented as Actors.

There are four types of relationships in a Use Case diagram: communication, generalization, include and extend.

- The «generalization» relationship occurs from actor to actor or from use case to use case. The semantics is the same used in other diagrams, such as the UML Class diagram: the child element inherits all behavior of its parent, and can add some more specific behavior.
- The «include» relationship provides a mechanism useful when a sequence of events is common to more than one use case. These sequences of events can be encapsulated as one use case and reused by other use cases. The execution of the base use case implies also in the execution of the included use cases.
- The «extend» relationship provides optional functionality, which extends the base use case at defined extension points under specified conditions. This relationship is useful when a use case is too complex, with many alternatives and optional sequences of interactions. The solution is to separate each alternative or option of the base use case into another use case, and relate them using the «extend» keyword. The base use case is independent of the extended ones, which may only be executed if the condition in the base use case that causes it to execute is set to true [1].

1.2.1 Master Simulation Tool Use Cases

This section contains the MST use cases identified by the project partners. Rather than concentrating the effort in obtaining a few but fully complete and formally correct use case definitions, the objective is to get as a complete picture as possible of the overall MST needs. That is, each use case is detailed to a level enabling derivation of relevant requirements.

1.2.1.1 Initialization of dynamic variables:

Description:

Define initial values for the dynamic states of the model. These values set the initial condition or initial points to start the simulation.

Pre-Condition: FMU simulation model with application specific parameter values.

Post-Condition: Guess values for dynamic states.

Main Scenario:

The actor defines initial values for the dynamic states of the component's model equations. These values set the initial conditions for starting the simulation.

(For example, initial conditions are specified within the Modelica language construct initial

equation).

Alternative, Exception, Extensions:

1. The actor loads initial conditions from file. It should be possible to load a file of initial conditions for each individual FMU or for the simulator configuration as a whole.
2. Store final values from a simulation for reuse as initial conditions for new simulations.

Associated actors: Systems Engineer,

Extended Use Case: FMU Unit Test Simulation Setup, Load Existing System Model,

1.2.1.2 Time-Domain Simulation:

Description:

Application of numerical integration routine for calculating the trajectories of time-dynamic variables in time.

The integrator method calculates the state of the simulation model at every time step, for all the equation unknowns defining the simulation model.

Pre-Condition:

FMU containing a complete model, with or without solver. The term complete models considers the map of a physical system with application specific parameter values, steady-state solution and initial conditions of dynamic equations. And the model of possible disturbances affecting a specific component or connection nodes within the simulation model. If the FMU contains a solver, the FMU has the solver configuration.

Post-Condition:

N/A

Main Scenario:

Application of mathematical integration method to solve all the states of the simulation model equations.

Alternatives, Exceptions, Extensions:

This use case is considered specific for the Power Systems field. It applies the same description but for power network models.

Associated actors: Simulation Model Integrator,

Extended Use Case: Execute Co-Simulation,

1.2.1.3 Steady-State Calculation:

Description:

Calculation of the values for the state variables of the simulation model when there is no event affecting the model. For example, in power network model, current, voltage, power and angle values, at each connection point of the network, represent the energy flow through the transmission lines.

Main Scenario:

1. Select solver algorithm
2. Solve the non-linear equation of the model

Pre-Condition:

N/A

Post-Condition: Simulation model is updated, with steady-state values indicated in the connecting nodes of the model

Associated actors: Simulation Model Integrator,

Extended Use Case: Execute Co-Simulation, Compose Simulation Models,

1.2.1.4 Initialization of Steady-State:

Description:

Define initial values for the interconnected variables within the simulation model. For power network models, these values state starting points to be used by a power flow solvers.

Pre-Condition:

Simulation model with values in all its components

Post-Condition:

Guess values for all interconnected component variables

Associated actors: Systems Engineer,

Extended Use Case: FMU Unit Test Simulation Setup, Load Existing System Model,

1.2.1.5 Compose Simulation Models:

Description:

Simulation Composition. Connecting inputs/outputs of the FMUs that compose the simulation model, and provide the parameter values for the components and connections between the components of the model.

Pre-Condition:

Existing FMU elementary models, or existing system architecture model (i.e : SysML Internal Block Diagram, ...)

Post-Condition:

System Model ready for Co-Simulation (defined blocks, blocks relationships, interfaces, configured co-simulation parameters).

Main Scenario:

1. The MST display a block and inputs/outputs ports corresponding to each imported model I/Os.
2. The System Architect composes relations between the models by connecting compatible ports together.
3. Save the the system model (composed simulation models) into an open cps standard for

system modeling.

Alternatives, Exceptions, Extensions:

N/A

Associated actors: Software Model Developer, Hardware/Physical Systems Model Developer, Systems Engineer,

Extended Use Case:

1.2.1.6 Import FMU:

Description:

Import FMU for co-simulation or model exchange.

Pre-Condition:

Available FMU from continuous model or a discrete-events model.

Post-Condition:

FMU for co-simulation or model exchange imported into the MST.

Main Scenario:

1, Import of one or more FMU/FMUs for co-simulation or model exchange into Master Simulation Tool.

Alternatives, Exceptions, Extensions:

N/A

Associated actors: Systems Engineer, Simulation Model Integrator, Software Model Developer, Hardware/Physical Systems Model Developer,

Extended Use Case:

1.2.1.7 Configure co-simulation parameters:

Description:

Configure co-simulation parameters.

Pre-Condition:

Post-Condition:

Specified co-simulation parameters.

Main Scenario:

1. Actor configures the co-simulation parameters before starting the co-simulation (start time, stop time, communication time step)

Alternatives, Exceptions, Extensions:

N/A

Associated actors: Systems Engineer, Simulation Model Integrator, Software Model Developer, Hardware/Physical Systems Model Developer,

Extended Use Case:

1.2.1.8 Schedule FMUs execution:

Description:

Schedule the execution of FMUs.

Pre-Condition:

Available FMU composition.

Defined co-simulation parameters .

Post-Condition:

On going/executed simulation of specified MST simulator configuration.

Main Scenario:

- 1, The actor triggers the simulation start
- 2, The MST orchestrates the execution of the FMUs according to the predefined co-simulation parameters and FMU parameters (communication step size, start time, stop time and master algorithm strategy)

Alternatives, Exceptions, Extensions:

N/A

Associated actors:

Extended Use Case: Execute Co-Simulation,

1.2.1.9 Debug Simulation:

Description:

Be able to debug a simulation both in terms of execution time performance and robustness/initialization aspects.

Pre-Condition:

Debugger is started prior to simulation start

Debugger code profiler is started prior to simulation start

Erroneous (crashed or slow) simulation.

Post-Condition:

N/A

Main Scenario:

- 1, The actor utilizes the debugger to deduce the source to the simulation crash. Typical sources that the debugger needs to identify are: un-physical input boundary conditions, erroneous FMU connections rendering un-physical inputs,

failure for solver to converge as a result of stiff systems of equations.

2, The actor uses the code profiler to monitor and identify slow FMUs

3, The actor implements the profiler to detect and log real time overruns (for simulations where real time performance is required)

4, The actor utilizes the debugger to identify reasons for the simulation to be unexpectedly slow, such as clattering.

5, The actor utilizes the debugger to identify the reason for erroneous initialization.

The debugger makes suggestions as to which initialization values that need to be adjusted

Alternatives, Exceptions, Extensions:

N/A

Associated actors: Systems Engineer, Simulation Model Integrator, Hardware/Physical Systems Model Developer, Software Model Developer, Test Engineer,

Extended Use Case: Execute Co-Simulation, Set Breakpoint,

1.2.1.10 Perform variables logging:

Description:

Log simulation results.

Pre-Condition:

On going/executed simulation of specified MST simulator configuration.

Post-Condition:

Logged simulation results.

Main Scenario:

1, The actor specifies which variables/parameters to log during simulation.

2, The actor is able to choose resolution of output data and if to store data at the occurrence of events.

3, The simulation results are stored in convenient machine readable format that MST and other platforms can read (ex: *.csv or *.mat)

Alternatives, Exceptions, Extensions:

1, If no particular variables/parameters to be stored are selected, then all variables/parameters are stored in the results.

Associated actors:

Extended Use Case:

1.2.1.11 Dynamically monitor Variables:

Description:

Dynamically monitor variables.

Pre-Condition:

MST is configured in debug mode.

Post-Condition:

On going/executed simulation of specified MST simulator configuration.

Main Scenario (Sunny day):

- 1, The main actor selects the FMUs variables (with causality =inputs, outputs, and local) where he/she wants to monitor the values
- 2, The MST simulation tool displays the values during the simulation time or at least after each simulation stop (stop time, after each breakpoint occurrence)

Alternatives, Exceptions, Extensions:

- 1/ The MST may plot the variables graph $y(t)$ dynamically to support visualization
- 2/ The MST may log in a file the variables values at each communication step time

Associated actors:

Extended Use Case:

1.2.1.12 Set Breakpoint:

Description:

Set break point

Pre-Condition:

Available FMU composition.

Post-Condition:

N/A

Main Scenario:

- 1, The Actor specifies a simulation break condition.
- 2, The actor investigates model variables and parameters at the break.
- 3, The actor continues the simulation from the break point via an actor command.

Alternatives, Exceptions, Extensions:

N/A

Associated actors: Software Model Developer, Systems Engineer, Test Engineer, Simulation Model Integrator,

Extended Use Case:

1.2.1.13 Execute Co-Simulation:

Description:

Execute co-simulation.

Pre-Condition:

One or more FMUs incorporated into the simulation model used by the MST.

Post-Condition:

Successfully executed simulation.

Main Scenario:

1, The actors simulates the specified MST simulator configuration, with the applied test and solver settings, via an actor command.

Alternatives, Exceptions, Extensions:

N/A

Associated actors: Systems Engineer, Hardware/Physical Systems Model Developer, Software Model Developer, Simulation Model Integrator,

Extended Use Case: Compose Simulation Models,

1.2.1.14 Define FMU Unit Test:

Description:

Be able to define a test of a single FMU for functional and regression testing.

Pre-Condition:

Available FMU as unit under test.

Post-Condition:

Stored test script and FMU unit test MST setup.

Main Scenario:

1. The actor imports an FMU for co-simulation or model exchange and the system displays it on a canvas.
2. The actor opens a test script editor and the system displays mandatory inputs and outputs serving as aid to the user in the test specification procedure.
3. The test script appears on the drawing Canvas to be connected with a FMU of interest.
4. The test script is saved for later re-use.
5. The FMU unit test MST setup is stored for later re-use.

Alternatives, Exceptions, Extensions:

N/A

Associated actors: Hardware/Physical Systems Model Developer, Simulation Model Integrator, Software Model Developer,

Extended Use Case: Import FMU,

1.2.1.15 FMU Unit Test Simulation Setup:

Description:

Simulation Setup.

Pre-Condition:

Existing FMU unit test.

Post-Condition:

Solver and initialization settings stored along with the FMU unit test MST setup.

Main Scenario:

1. The actor modifies the FMU solver settings. If the incorporated FMU is for model exchange than the central solver settings are modified. Also the type of central solver is specified (implicit/explicit, fixed/variable step).
If the incorporated FMU is for co-simulation then the included solvers settings are modified. Solver tolerance and step length are settings that shall be available to the user
2. The MST provides information of which variables that need to be initialized.
3. The actor modifies available default initialization values.
4. The actor stores the solver and initialization settings along with the simulation configuration.

Alternatives, Exceptions, Extensions:

N/A

Associated actors: Simulation Model Integrator, Software Model Developer, Hardware/Physical Systems Model Developer,

Extended Use Case: Configure co-simulation parameters,

1.2.1.16 Simulate FMU Unit Test:

Description:

Be able to Simulate a test of a single FMU.

Pre-Condition:

Available FMU unit test and FMU loaded and ready for simulation in Master Simulation Tool. All simulation settings are set.

Post-Condition:

Stored unit test results.

Main Scenario:

1. The actor simulates the single FMU by the press of a button in a graphical user interface.
2. The simulation results are stored.

Alternatives, Exceptions, Extensions:

1. Single FMU is broken down and simulated in parallel if possible.
2. The FMU unit test is simulated via command line interface.

Associated actors: Software Model Developer, Simulation Model Integrator, Hardware/Physical Systems Model Developer,

Extended Use Case: Execute Co-Simulation, Perform variables logging,

1.2.1.17 Post Process of Unit Test:

Description:

Post processing of data.

Pre-Condition:

FMU simulation results.

Post-Condition:

Exported plotted figures and plot setup.

Main Scenario:

1. The actor imports previous unit test results.
2. The actor plots simulated FMU unit test results.
3. The actor plots earliest unit test results.
4. Compare results (the possibility of defining difference, relative errors, absolute errors, etc. is beneficial).

Alternatives, Exceptions, Extensions:

N/A

Associated actors: Software Model Developer, Hardware/Physical Systems Model Developer,

Extended Use Case: Display Simulation Results,

1.2.1.18 Define Batch Simulation Boundary Conditions:

Description:

Be able to define batch simulation boundary conditions for multiple connected FMUs.

Pre-Condition:

Multiple FMUs developed from FMI 2.0 compliant tools.

Post-Condition:

Defined and stored multiple simulation boundary conditions.

Main Scenario:

1. The actor imports several FMUs (for example from: OpenModelica, Dymola, UML-tools, Simulink) and the system displays them on a canvas.
The incorporated FMUs can be a mixture of FMUs for model exchange and FMUs for co-simulation as well as a mixture of continuous and discrete-events models.
2. The actor connects the FMUs graphically via the canvas or through manipulation of code.
3. The actor saves the simulation model.
4. The actor opens a test script editor and the system displays mandatory inputs and outputs.
5. The actor defines multiple sets of boundary conditions through the test script editor.
6. The actor imports existing boundary conditions into MST.
7. The defined boundary conditions are presented as objects on a canvas.
8. The boundary conditions are stored in convenient format (ex: *.csv, *.mat) for import into MST and other tools (ex: Matlab, Dymola etc.)

Alternatives, Exceptions, Extensions:

1. The actor imports a saved existing simulator configuration and the loaded configuration is displayed on a canvas.

Associated actors: Test Engineer, Systems Engineer,

Extended Use Case: Import FMU,

1.2.1.19 Small Scale Simulator Simulation Setup:

Description:

Small scale simulator simulation setup.

Pre-Condition:

Defined set of boundary conditions for batch simulations.

Post-Condition:

Saved simulator settings.

Main Scenario:

1. The MST automatically identifies which of the incorporated FMUs are FMUs for model exchange and which are FMUs for co-simulation and presents this information to the actor (at the request of the actor?)
2. The solver settings of each incorporated FMU is individually modified. Solver tolerance and step length are solver settings that shall be available to the user. both for the central solver (FMUs for model exchange) as well for co-simulation FMUs individual solver settings. The user can choose between different central solvers, solvers with variable step length as well as fixed step lengths.
3. The actor specifies values for simulator initialization via the user interface. The tool aids in this process by visualization of default initialization values.
4. The actor selects which model parameters and variables to store. Store all is available as an option.
5. The actor selects the resolution in which the simulation output results are to be stored, storing results at every integration step shall be available as an option.
6. The MST automatically schedules the execution of the simulation model; however, the actor is able to modify scheduling via the user interface
7. The actor stores the solver and initialization settings along with the simulation configuration.
- 8, The actor places the simulator configuration under configuration management in order to document and track simulator changes.

Alternatives, Exceptions, Extensions:

N/A

Associated actors: Systems Engineer,

Extended Use Case: Configure co-simulation parameters,

1.2.1.20 Simulate Small Scale Simulator:

Description:

Be able to simulate small scale simulator.

Pre-Condition:

Defined set of boundary conditions for batch simulations and specified simulator simulation settings.

Post-Condition:

Stored simulation results.

Main Scenario:

1. The FMUs are simulated in parallel for each set of boundary conditions implementing a numerically stable method without compromising the validity of the simulation results.
2. The sets of boundary conditions are simulated in parallel.

Alternatives, Exceptions, Extensions:

N/A

Associated actors: Test Engineer, Systems Engineer,

Extended Use Case: Execute Co-Simulation, Perform variables logging,

1.2.1.21 Post Process of Batch Simulation Results:

Description:

Post processing of data.

Pre-Condition:

Successfully executed batch simulation of small scale simulator.

Post-Condition:

Stored plotted results.

Main Scenario:

1. The actor browses through all simulated results.
2. The actor defines steady state or specific dynamic events and the MST finds relevant points or time segments in the simulation results.
3. The actor plots and compares results from all run simulations.

Alternatives, Exceptions, Extensions:

N/A

Associated actors: Test Engineer, Systems Engineer,

Extended Use Case: Display Simulation Results,

1.2.1.22 Load Existing Simulator Configuration:

Description:

Load existing simulator configuration.

Pre-Condition:

Stored existing simulator configuration.

Post-Condition:

Existing simulator configuration loaded into MST, the loaded configuration is ready for simulation provided solver and initialization settings were set in the stored configuration.

Main Scenario:

1. The actor opens stored simulator configuration.
2. The MST poses the question of access rights to the actor. The actor is only allowed to open the specific simulator configuration if he/she has appropriate access rights.

Alternatives, Exceptions, Extensions:

N/A

Associated actors: Test Engineer, Systems Engineer,

Extended Use Case:

1.2.1.23 Trace and Replay Simulation:

Description:

Be able to trace a simulation and replay the simulation for only one FMU for debugging.

Pre-Condition:

Available simulation results.

Post-Condition:

N/A

Main Scenario:

1. For a long simulation is desired to be able to store all inputs / outputs.
2. If the simulation crashes one should be able to replay the simulation for only one FMU that crashed.
3. The debugger can be used to debug the FMU.

Alternatives, Exceptions, Extensions:

N/A

Associated actors: Systems Engineer,

Extended Use Case: Perform variables logging,

1.2.1.23.1 Debug Simulation:

Description:

Be able to debug a simulation both in terms of execution time performance and robustness/initialization aspects.

Pre-Condition:

Debugger is started prior to simulation start.

Debugger code profiler is started prior to simulation start.

Erroneous (crashed or slow) simulation.

Post-Condition:

N/A

Main Scenario:

1. The actor utilizes the debugger to deduce the source to the simulation crash.

Typical sources that the debugger needs to identify are: un-physical input boundary conditions, erroneous FMU connections rendering un-physical inputs, failure for solver to converge as a result of stiff systems of equations.

2. The actor uses the code profiler to monitor and identify slow FMUs.

3. The actor implements the profiler to detect and log real time overruns (for simulations where real time performance is required).

4. The actor utilizes the debugger to identify reasons for the simulation to be unexpectedly slow, such as clattering.

5. The actor utilizes the debugger to identify the reason for erroneous initialization.

The debugger makes suggestions as to which initialization values that need to be adjusted.

Alternatives, Exceptions, Extensions:

N/A

Associated actors:

Extended Use Case: Execute Co-Simulation, Set Breakpoint,

1.2.1.24 Load Existing System Model:

Description:

Load an existing system model (created in any tool).

Pre-Condition:

The system model is stored in an OPENCPS standard format. The OPENCPS standard format is later to be proposed for standardization in FMI.

Post-Condition:

A simulator configuration ready for simulation.

Main Scenario:

1. The actor opens the system model in the MST.

2. The actor imports FMUs to represent the components in the system model.

3. The actor allocates the FMUs into the system model components.
4. The MST verifies the allocated FMUs according to the system model specification.
5. The actor prepares further solver and initialization settings to complete the simulation configuration.
6. The simulation configuration is saved for later reuse.

Alternatives, Exceptions, Extensions:
Report validation errors

Associated actors: Systems Engineer,

Extended Use Case: Import FMU,

1.2.1.25 Display Simulation Results:

Description:
Display Simulation Results.

Pre-Condition:
Available simulation results.

Post-Condition:
Saved simulation plots.

Main Scenario:

1. The actor defines outputs and inputs to investigate, the MST presents only this information.
2. The selected results are plotted via an actor command.
3. The plot setup is saved for later re-use via an actor command.
4. The actor exports and saves selected figures.

Alternatives, Exceptions, Extensions:
N/A

Associated actors: Systems Engineer, Hardware/Physical Systems Model Developer,
Simulation Model Integrator, Software Model Developer,

Extended Use Case: Execute Co-Simulation,

1.2.1.26 Define Architectural rules for UML/FMI models:

Description:
Specify formal architectural rules for UML/SysML models including FMI components.

Pre-Condition:
System Requirements have been defined and analysed.

Post-Condition:
System Architecture rules are described in a class diagram based meta model.

Alternatives, Exceptions, Extensions:

N/A

Main Scenario:

1. The system architect opens a GUI and creates a separate UML model including Class diagrams.
2. The system architect defines stereotypes for the different types of components, including FMI-type.
3. The system architect specify allowed relations and required attributes etc for the FMI-components.
4. The architects save the model and connect it to the system model that should conform to the architectural rules.

Associated actors: Systems Engineer,

Extended Use Case:

1.2.1.27 Validate Model conformance to Architectural rules:

Description:

Support the system engineer by continuous or batch-wise validation that the UML/SysML model including FMI components, conforms to architectural rules.

Pre-Condition:

System architectural rules meta-model has been defined.

Post-Condition:

System model conforms to Architecture.

Main Scenario:

1. The system/software engineer opens a GUI and creates a connection between the development model (including FMI/FMU components) and the rules meta model.
2. The System/software engineer develops system/software model and continuously receives remarks or instructions (from e.g. ArCon) on how to define and connect FMI/FMU.
3. The system/software engineer can stop continuous validation and instead activate validation batchwise.

Alternatives, Exceptions, Extensions:

N/A

Associated actors: Systems Engineer,

Extended Use Case:

1.2.2 Master Simulation Tool Use Cases Diagrams

This section presents UML diagrams containing graphical representation of use cases, actors and relationships between them are presented in this section.

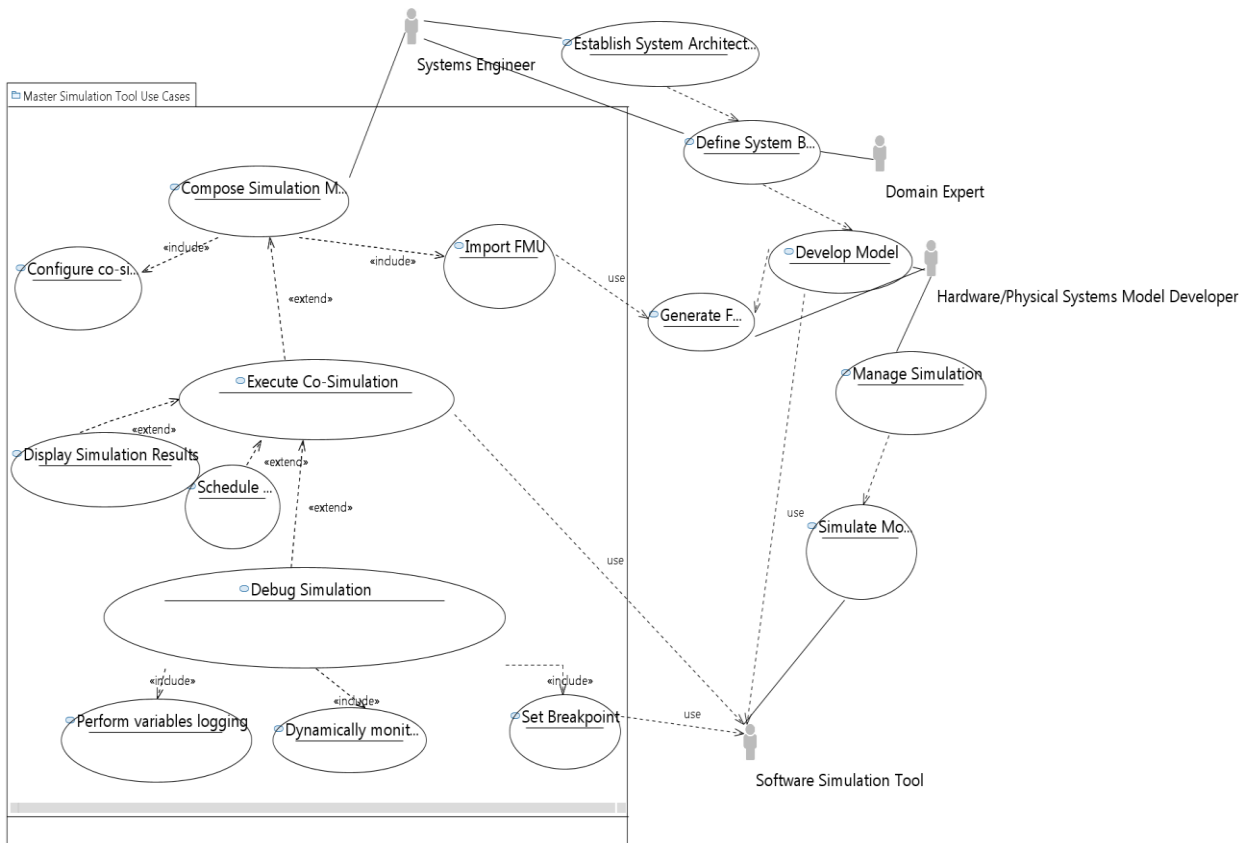


Figure 1 Master Simulation Tool Use Cases Diagram 1

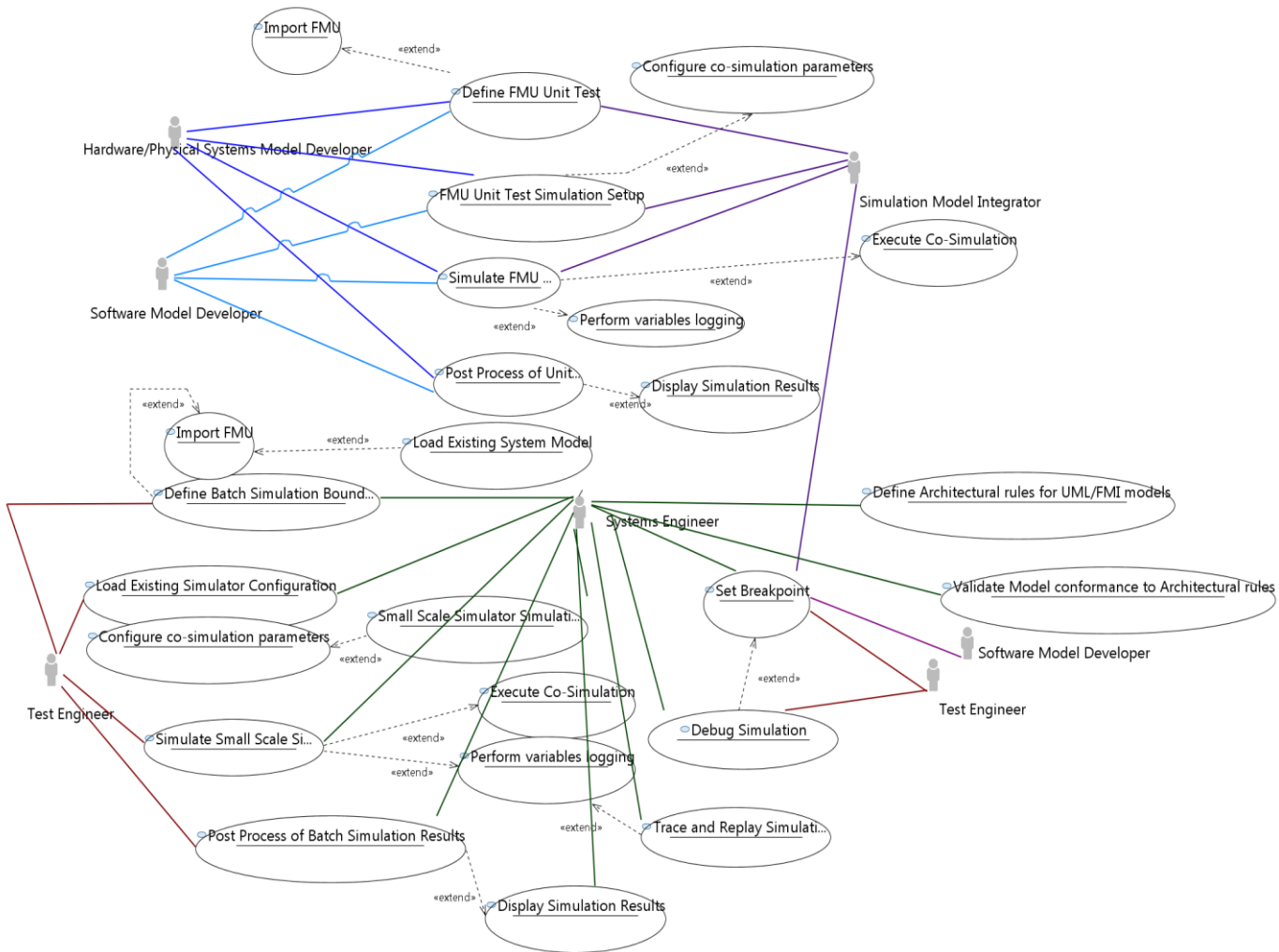


Figure 2 Master Simulation Tool Use Cases Diagram 2

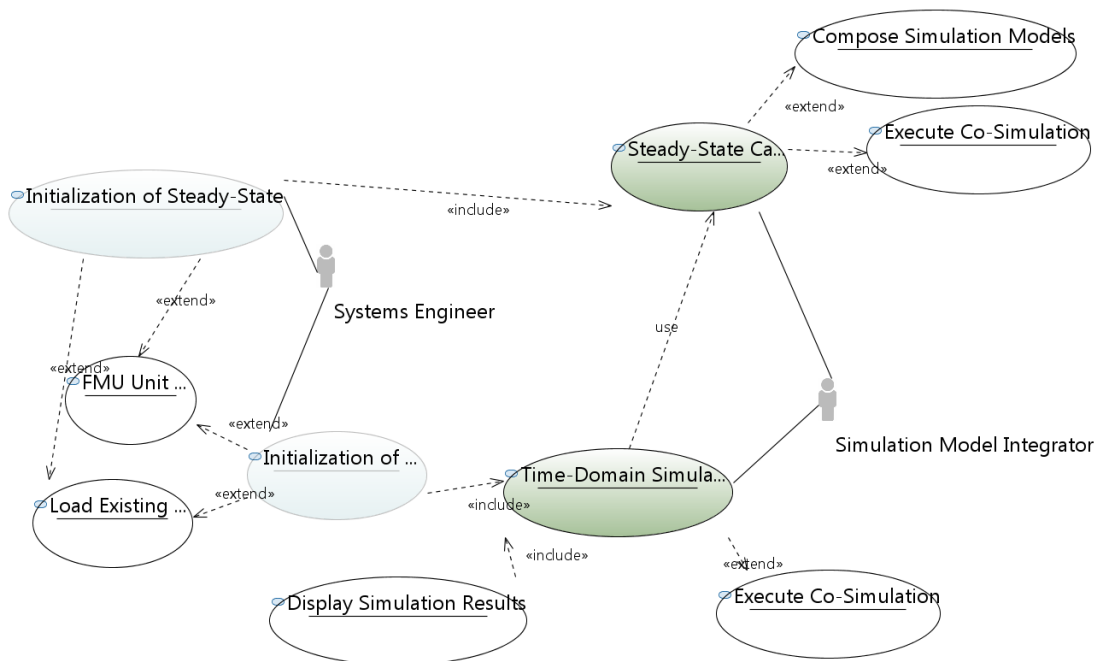


Figure 3 Master Simulation Tool Use Cases Diagram 3

1.2.3 Simulation Tools

This section contains simulation tool specific use cases relevant in the context of OPENCPS.

1.2.3.1 Generate FMU:

Description:

Generate FMU from a specific Simulation Tool.

Pre-Condition:

Simulation model has been developed and verified in the Domain specific simulation tool.

Post-Condition:

FMU is generated for Co-Simulation, Model Exchange or Both.

Main Scenario:

- 1, The Main actor verifies the simulation results of the individual model.
- 2, The Main actor selects the model components to export as FMU.
- 3, The Main actor chooses the way to export the FMU (CS, ME or both).
- 4, The Main actor selects the generation properties of the FMU such as platform (win32, win64, ...), include source codes or not, select local variables and parameters to export, compiler and associated compiler options, ...
- 5, The Main actor selects the type of exported FMU according to Simulation tool capabilities (FMU as a DLL, FMU wrapper generation).
- 6, The Main actor verifies the resulting behavior of the exported FMU (comparison between initial model and generated model).

Alternatives, Exceptions, Extensions:

N/A

Associated actors: Hardware/Physical Systems Model Developer,

Extended Use Case:

1.2.3.1.1 Generate FMU from Matlab/Simulink:

Description:

Generate FMU from Matlab/Simulink model.

Pre-Condition:

The simulation model has been developed and verified in Matlab/Simulink.

The main Actor has a Matlab/Simulink to FMI converter available.

Post-Condition:

FMU is generated for CoSimulation, ModelExchange or Both.

Main Scenario:

1. The Main actor verify the simulation results of the individual model.
2. The Main actor select the model components to export as FMU.
3. The Main actor choose the way to export the FMU (CS, ME or both).

4. The Main actor select the generation properties of the FMU such as platform (win32, win64, ...), include source codes or not, select local variables and parameters to export, compiler and associated compiler options, ...
5. The Main actor select the type of exported FMU according to Simulation tool capabilities (FMU as a DLL, FMU wrapper generation).
6. The Main actor verify the resulting behavior of the exported FMU (comparison between initial model and generated model).

Alternatives, Exceptions, Extensions:
N/A

Associated actors: Simulation Model Integrator, Hardware/Physical Systems Model Developer, Systems Engineer, Software Model Developer,

Extended Use Case:

1.2.3.1.2 Generate FMU from SimulationX:

Description:
Generate FMU from SimulationX model.

Pre-Condition:
Simulation model has been developed and verified in SimulationX.

Post-Condition:
FMU is generated for CoSimulation, ModelExchange or Both.

Main Scenario:

1. The Main actor verify the simulation results of the individual model.
2. The Main actor select the model components to export as FMU.
3. The Main actor choose the way to export the FMU (CS, ME or both).
4. The Main actor select the generation properties of the FMU such as platform (win32, win64, ...), include source codes or not, select local variables and parameters to export, compiler and associated compiler options, ...
5. The Main actor select the type of exported FMU according to Simulation tool capabilities (FMU as a DLL, FMU wrapper generation).
6. The Main actor verify the resulting behavior of the exported FMU (comparison between initial model and generated model).

Alternatives, Exceptions, Extensions:
N/A

Associated actors: Simulation Model Integrator, Hardware/Physical Systems Model Developer, Systems Engineer, Software Model Developer,

Extended Use Case:

1.2.3.1.3 Generate FMU from OpenModelica:

Description:

Generate FMU from a Modelica model using OpenModelica.

Pre-Conditions:

Simulation model has been developed and verified in OpenModelica.

Post-Condition:

FMU is generated for CoSimulation, ModelExchange or Both.

Main Scenario:

1. The Main actor verify the simulation results of the individual model.
2. The Main actor select the model components to export as FMU.
3. The Main actor choose the way to export the FMU (CS, ME or both).
4. The Main actor select the generation properties of the FMU such as platform (win32, win64, ...), include source codes or not, select local variables and parameters to export, compiler and associated compiler options, ...
5. The Main actor select the type of exported FMU according to Simulation tool capabilities (FMU as a DLL, FMU wrapper generation).
6. The Main actor verify the resulting behavior of the exported FMU (comparison between initial model and generated model).

Alternatives, Exceptions, Extensions:

N/A

Associated actors: Simulation Model Integrator, Hardware/Physical Systems Model Developer, Systems Engineer, Software Model Developer,

Extended Use Case:

1.2.3.2 Develop Model:

Description:

Develop a model.

Pre-Condition:

Domain expert has defined the expected behavior and associated requirements/specifications of sub-system or component.

Post-Condition:

Model content is developed (graph, parameters, variables, ...)

Main Scenario:

1. The Main actor develop the model by drag and drop library components in main sheet or creating models from a design palette.
2. The Main Actor connect the selected components in the library together by drawing connections between components.
3. The Main actor defines the variables, signals, parameters types.

Alternatives, Exceptions, Extensions:

N/A

Associated actors: Hardware/Physical Systems Model Developer,

Extended Use Case:

1.2.3.3 Establish System Architecture:

Description:

Develop a System Architecture from System Requirements Analysis.

Pre-Condition:

System Requirements have been defined and analysed.

Post-Condition:

System is described as a set of functions, logical components and/or physical components in a result model (example: in SysML diagrams like Internal Block Diagrams, Activity Diagrams, ...).

Main Scenario:

1. The Main actor develop an architectural model of the system which describes the several components/functions of the system and their relations.

Alternatives, Exceptions, Extensions:

N/A

Associated actors: Systems Engineer,

Extended Use Case:

1.2.3.4 Define System Behavior:

Description:

Establish the system behaviors

Pre-Condition:

System Requirements have been defined and analysed

Post-Condition:

System Behaviors are identified and documented in specifications

Main Scenario (Sunny day):

1, The Main actor document the system behaviors according to its field of expertise

Alternatives, Exceptions, Extensions:

N/A

Associated actors: Domain Expert, Systems Engineer,

Extended Use Case:

1.2.3.5 Simulate Model:

Description:
Simulate the model.

Pre-Condition:
Simulation model is available.

Post-Condition:
Model has been successfully simulated and simulation data report is available.

Main Scenario (Sunny day):

1. The Main actor select the expected solvers and configure solver parameters.
2. The Main actor execute the simulation.
3. The Main actor examine the simulation results and store the simulation data in simulation data report.

Alternatives, Exceptions, Extensions:

N/A

Associated actors: Software Simulation Tool,

Extended Use Case:

1.2.3.6 Manage Simulation:

Description:
Manage the simulation results.

Pre-Condition:
Simulation model is available for simulation.

Post-Condition:
Model has been successfully simulated once.

Main Scenario (Sunny day):

1. The Main actor selects model to simulate.
2. The Main actor executes the simulation.
3. The Main actor examines the simulation results and store the simulation data in simulation data report.
4. The Main actor selects and stores several simulation results and documents the results efficiently.

Alternatives, Exceptions, Extensions:

N/A

Associated actors: Hardware/Physical Systems Model Developer,

Extended Use Case:

1.3 Requirements

This section contains the list of functional requirements for the MST. Functional requirements describe the expected functionalities of the system. These requirements are derived from the use cases described in the previous section. A subset of requirements include verification test cases.

In the requirements definition phase, the ambition has been to obtain as a complete picture as possible regarding required MST functionality, ensuring industrial relevance in terms of simulation efficiency, robustness, and scalability. However, development of an MST compliant with all identified requirements is not in the scope of the OPENCPS project. Rather, the project is focused on the most essential requirements enabling realization of the planned industrial demonstrators. Therefore, a priority has been defined for each requirement, describing the need for implementation during the project. Priority definition:

- 1) Mandatory for demonstrator realization, needed latest by M24
- 2) Mandatory for demonstrator realization, needed latest by M33
- 3) Optional for demonstrator realization, may be utilized if available
- 4) Relevant MST functionality, but will probably not be utilized in demonstrators

Depending on the progress of MST and demonstrator development, and on the new knowledge gained during these development efforts, the MST requirements as well as priorities are subject to continuous improvement/reconsideration during the project.

Requirements can be decomposed into sub-requirements for better readability. In the following table high level requirements are highlighted with the darker colour and sub-requirements are highlighted with lighter colours.

id	Requirement Name	Text	Derived From (Use Cases)	Verified By (Test Cases)	Priority
Req1	FMU Import	The MST shall enable composition of FMUs coming from various Modeling tools.	[Define FMU Unit Test], [Import FMU],	[test-connectivity-one-model, test-connectivity-one-direction, test-connectivity-two-direction,]	1
Req1.1	SimulationX FMU Import	The MST should enable import of FMUs produced by SimulationX			3
Req1.2	Simulink FMU Import	The MST should be able to import of FMUs generated from Matlab/Simulink			1

Req1.3	Import Other SysML	The MST should enable import of FMUs for co-simulation from other graphical editing tool for UML/SysML			1
Req1.4	Import Other Modelica	The MST should enable import of FMUs for co-simulation from other editing tools for Modelica			1
Req1.5	Import Papyrus	The MST should enable import of FMUs for co-simulation produced by Papyrus			1
Req1.6	Import OpenModelica	The MST should enable import of FMUs for co-simulation produced by OpenModelica			1
Req1.7	Co-simulation or Model Exchange	The MST identifies if the imported FMU is for co-simulation or if it is for model exchange. This information is presented to the user			1
Req1.8	Unit extension import	The Master Simulation Tool may import additional unit information for the variables of an FMU.		[Unit Extension Import,]	2
Req2	Co-Simulation Composition	The MST shall allow composition of FMUS. The MST should enable connecting FMU models	[Define Batch Simulation Boundary Conditions],		1
Req2.1	Communication Steps Configuration	The MST shall allow to configure the Communication Steps between FMUs	[Compose Simulation Models],		1
Req2.2	UML events between models	The Master Simulation Tool should support sending events between models that can handle them (e.g. UML or Modelica models).		[UML events between models, UML events with attributes,]	3
Req2.2.1	Event attributes	The Master Simulation Tool should support sending basic type values as attributes of events.			3
Req2.3	UML Operation calls between models	The Master Simulation Tool may support operation calls between models that can handle them (e.g. UML models).		[UML operation calls between models,]	3
Req2.4	Automatic unit conversion	The Master Simulation Tool may perform automatic unit conversion between compatible measuring units. This should be optional to the user.		[Automatic unit conversion,]	2
Req2.5	Basic type connection	The Master Simulator Tool shall ensure that basic type values can be passed between models (Boolean, Integer, Real, String)		[Basic type connection,]	1
Req2.6	Variable groups	The Master Simulator Tool should allow to manually create groups of existing variables (input/output signals of FMUs)		[Variable groups,]	1
Req2.7	Variable group connection	The Master Simulator Tool should allow to connect compatible (type, unit, causality, etc.)		[Variable group	2

		groups of variables, which should be the same as connecting variable pairs individually		connection,]	
Req2.8	Automatic variable group detection	The Master Simulation Tool may detect and create groups of variables automatically based on naming or additional model information		[Automatic variable group detection, Automatic variable group creation based on external information,]	2
Req3	Configuration of the solver	The MST should support specification of the type of central solver (implicit/explicit, fixed/variable step) for simulation of FMUs for model exchange. Solver tolerance should also be available to the user. The MST should support specification of integrator method (fixed/variable step) for simulation of FMUs for model exchange. Stop time, tolerances, and number of intervals (steps for the integration method) should also be available to the user. These settings should be available via the Master Simulation Tool graphical user interface, the command line interface, and the scripting interface.	[Configure co-simulation parameters], [FMU Unit Test Simulation Setup], [Time-Domain Simulation],		1
Req4	FMU Configuration	The MST should enable configuring FMUs parameters. It must be possible to control start time, stop time, maximum step size, and I/O for FMUs.	[Configure co-simulation parameters],		1
Req5	FMU Composition Validation	The MST should enable FMUs composition validation. Validation rules should be defined. For example: Types and units of connected ports should be compatible, An output port should be connected to an input port. The MST shall allow connections between compatible ports only (compatible causality, compatible units, compatible types, continuous vs discrete). In case of incompatible variables, the MST shall display an information message. Validation rules should be executed in the GUI in batch or live mode.	[Compose Simulation Models],		2
Req5.1	Type definition consistency	The Master Simulation Tool must analyze the platform dependent type definitions of the imported FMUs (fmi2TypesPlatform.h) and, in case of differences, either report incompatibilities or take care of the necessary conversions. Properties to consider are, for example, the size of integer, character, byte and pointer types, the precision of the Real type.			2

Req5.2	Unit compatibility validation	The Master Simulation Tool should display an error message if variables with incompatible units are connected.		[Unit compatibility validation,]	2
Req5.3	Ports Consistency Check	The MST shall allow connections between compatible ports only (compatible causality, compatible units, compatible types, continuous vs discrete). In case of incompatible variables, the MST shall display an information message.			2
Req6	Scheduling	The MST shall schedule in an optimal way the execution of each FMU (by analysis of dependency graph, communication steps sizes, ...)	[Small Scale Simulator Simulation Setup], [Schedule FMUs execution],		1
Req6.1	Select Master Algorithm	The MST may have the capability to propose to user to select Master Algorithm			3
Req6.2	Import External Master Algorithm	The MST may have the capability to import or develop external master scheduling algorithm			3
Req7	Optimizations	The MST should propose various optimizations capabilities. For example, support for multiple solvers in case of FMU-ME, multi-core simulation, distributed simulation.	[Simulate Small Scale Simulator], [Simulate FMU Unit Test],		3
Req8	Distributed Simulation	It shall be possible to run simulations locally, on a computer cluster or over the internet. The MST may have the capability to execute co-simulation in distributed environment (several computer or ECU).	[Configure co-simulation parameters], [Schedule FMUs execution],		3
Req8.1	Thread safety	The Master Simulation Tool must use the FMUs in a thread-safe way, because the FMI standard does not require FMUs to be thread-safe (section 2.1).			3
Req9	Debugging	The MST shall propose an interface for debugging	[Debug Simulation], [Trace and Replay Simulation],		2
Req9.1	Visualize Active States of FMU	The MST should enable to visualize graphically the active states of each FMUs when the FMU is modelled as a state machine	[Debug Simulation],		3
Req9.2	Monitor Exchanged Variables	The MST should enable to monitor exchanged variables between FMUs (including at least time step, variable name and value)			2

Req11.2.1	Monitor_Variables_Selection	The MST shall propose an interface to select variables to monitor for each FMU	[Perform variables logging], [Dynamically monitor Variables],		3
Req9.3	Master Breakpoints	The MST should enable to set breakpoints during the simulation.			2
Req11.3.1	Breakpoint_Effect	When breakpoint is set, the MST shall stop the time stamp, collect on-going results and wait for user action to restart the simulation	[Set Breakpoint],		2
Req9.4	Step Execution	The MST shall allow Step Execution and automatic break	[Debug Simulation], [Debug Simulation],		3
Req9.5	Monitor Local Variables	The MST should enable to monitor the "local"(FMU internal) variables into the log file	[Perform variables logging], [Dynamically monitor Variables],		2
Req9.6	Clattering	The MST should identify the presence and location of clattering. Only for FMU ME, special cases of FMU Co-sim, and maybe for the top-level scheduler. In the general case, we cannot access the "internal life" of a FMU Co-sim.	[Debug Simulation],		3
Req9.7	Real-Time Overrun	The MST should be able to detect real time overrun if specified by the user	[Debug Simulation],		3
Req9.8	Slow FMUs	The MST should detect FMUs that are executing unreasonably slow and present the information to the user.	[Debug Simulation],		2
Req9.9	Log handling	The Master Simulation Tool must handle log messages of the FMU's (i.e. calls to the logger function) and present them to the user as appropriate, for example in log files, console or, in case of errors, pop-up windows.			2
Req9.9	Log handling	The Master Simulation Tool must handle log messages of the FMU's (i.e. calls to the logger function) and present them to the user as appropriate, for example in log files, console or, in case of errors, pop-up windows.			2
Req9.10	Log categories	The Master Simulation Tool shall enable the user to choose which category of messages to log from the categories exposed by the FMU.			2
Req10	Execute Simulation Tools Instances	The MST shall allow to perform co-simulation of FMUs for CS using FMU Wrapper to external tool instance	[Configure co-simulation parameters],		3
Req11	Co-Simulation Papyrus Only	The MST may allow to execute FMUs for CS as imported DLLs (black box approach)	[Configure co-simulation parameters],		1

Req12	Hybrid Co-Simulation	The MST shall enable to perform Hybrid simulation between continuous and discrete models	[Schedule FMUs execution],		2
Req12.1	Experiment parameters for UML	It should be clearly defined what is meant by the tolerance, startTime, stopTime arguments of the fmi2SetupExperiment function in case of an FMU exported from a UML model.			2
Req12.2	Initialization mode and reset for UML	It should be clearly defined what is meant by the initialization mode for an FMU exported from a UML model and how to reset these FMUs.			2
Req12.3	FMU state handling for UML	It should be clearly defined if the canGetAndSetFMUstate parameter (see section 2.1.8 of the FMI 2.0 standard) can be true for FMUs exported from UML models, and, if yes, what should the FMU state consist of.			2
Req12.4	UML exposed variables	It should be clearly defined what can the exposed variables be of an FMU exported from a UML model, and which causalities and variabilities can these variables have (see section 2.2.7 of the FMI 2.0 standard).			2
Req13	Scripting	A scripting editor shall be available via the MST. The editor shall allow the user to formulate and store simulation boundary conditions.	[Define Batch Simulation Boundary Conditions], [Define FMU Unit Test],		3
Req14	Store FMU Composition	The user shall be able to store a composition of connected FMUs along with solver-, simulation, and data logging-settings via the Master Simulation Tool user interface.	[FMU Unit Test Simulation Setup], [Small Scale Simulator Simulation Setup], [FMU Unit Test Simulation Setup],		1
Req15	Command Line Interface	The MST shall have a command line interface from which a simulation of an existing composition of FMUs can be simulated. The user shall be able to modify solver settings (solver type and tolerance etc.), simulation settings (simulation time etc.), and data logging settings (output resolution etc.) from the command line interface	[FMU Unit Test Simulation Setup], [Small Scale Simulator Simulation Setup], [Simulate Small Scale Simulator],		3

			[Simulate FMU Unit Test], [Simulate FMU Unit Test],		
Req16	Post Processing	The MST shall have basic functionality for post processing and storing data	[Post Process of Unit Test], [Post Process of Batch Simulation Results], [Post Process of Unit Test],		2
Req16.1	Save To	The MST shall store the variables and parameters specified by the user, of an executed simulation. All available variables and parameters are to be stored as default. The simulation results are to be stored with a resolution specified by the user in machine readable format.			2
Req16.2	Import Results	The MST shall enable import of simulation results stored during earlier MST simulations and initialize the simulation with stored parameters			3
Req17	Load Simulator Configuration	The MST shall enable the user to load a stored composition of FMUs	[Simulate Small Scale Simulator],		1
Req18	Visualization	The MST should display the co-simulation results in a time scope. The MST shall provide basic plotting functionality both of current and imported results.			2
Req19	Common core	The MST should serve as a common core for multiple different applications			3
Req19.1	Embed MST Into Other Applications	It should be possible to embed the MST to different applications user interfaces (UIs)			3
Req20	Connect Sub-Models	It must be possible to connect multiple sub-models from different simulation tools.			1
Req21	Robustness	Given that all individual FMUs of a simulator configuration are numerically stable, the MST shall guarantee numerical robustness for the simulator configuration as a whole.	[Simulate FMU Unit Test],		2
Req21.1	Numerical Errors generated by the MST	In addition to the numerical errors related to finite machine precision, delays in the communication between FMUs shall not introduce any numerical errors.		[test-numerical-stability-identity, test-numerical-stability-overflow,]	2

Req22	Asynchronous	Asynchronous continuous-time communication must be supported.			1
Req22.1	One-Step Mode Execution	One-step mode execution of slaves shall be supported for good resolution in interpolation tables.			1
Req22.2	Model Step-Size	Each sub-model shall have an independent step-size, with a maximum limit depending on the communication interval.			1
Req23	Independent Solver Method	Each FMU-ME shall have an independent solver method.			2
Req23.1	Singlestep and Multistep Integration	Both singlestep and multistep integration methods must be supported.			2
Req23.2	Explicit and Implicit Integration	Both explicit and implicit integration methods must be supported.	[Simulate Small Scale Simulator],		2
Req23.3	Inputs at Given Time Instance	The MST must be able to provide the FMU access to interpolated variables for any given time instance within the duration of the step, or with interpolation tables for internal interpolation inside the FMU			2
Req24	FMI	Current version of Functional mock-up interface standard should be supported.		[test-fmi-compliance -best-case, test-fmi-compliance -callbacks, fmi-compliance -init-null,]	1
Req24.1	Model Exchange and Co-simulation	The MST shall support both co-simulation and model exchange specs	[Import FMU],		1
Req24.2	FMI standard test cases	The Master Simulation Tool must handle the example FMUs provided by https://www.fmi-standard.org			2
Req24.3	Status information handling	The Master Simulation Tool should handle correctly the status information returned by FMU functions. These are defined in section 2.3.1 in the FMI 2.0 standard (OK, Warning, Discard etc.) together with the required behavior.			2
Req25	Access Control	The MST shall support user access control with respect to information confidentiality and export control.	[Load Existing Simulator Configuration], [Small Scale Simulator Simulation Setup],		4
Req26	Architecture rules	The MST GUI shall support setting up architectural rules	[Define Architectural		3

			rules for UML/FMI models],		
Req26.1	UML2 Class diagrams	The MST GUI shall be capable to model UML2 Class diagrams			3
Req26.2	UML2 elements	The MST GUI shall at least handle UML2 stereotypes, Packages, Classes, Associations, Dependencies, Attributes, Operations			3
Req27	Architecture validation	The MST GUI shall support model validation against architecture rules	[Validate Model conformance to Architectural rules],		4
Req27.1	GUI API	The MST GUI shall have an API to reach UML2/SysML models			4
Req27.2	GUI error decorators	The MST GUI shall enable marking of architecture errors as decorators in the model browser and diagrams			4
Req27.3	Error console text	The MST GUI shall be able to present textual explanations of identified architectural errors			4

1.4 Requirements Diagram

This section presents SysML diagrams containing graphical representation of requirements and their relationships with use cases and test cases.

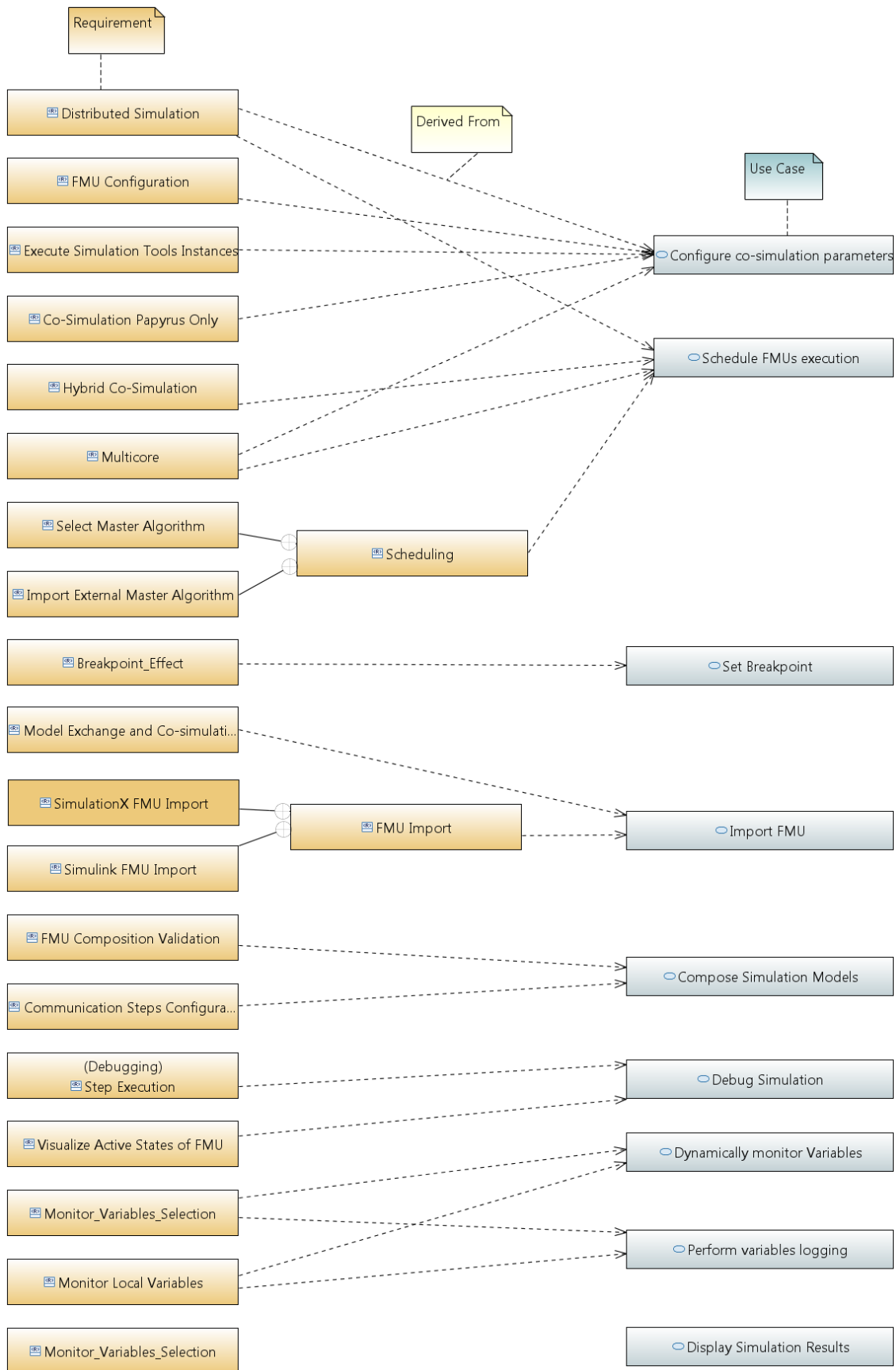


Figure 4 Requirements Diagram 1

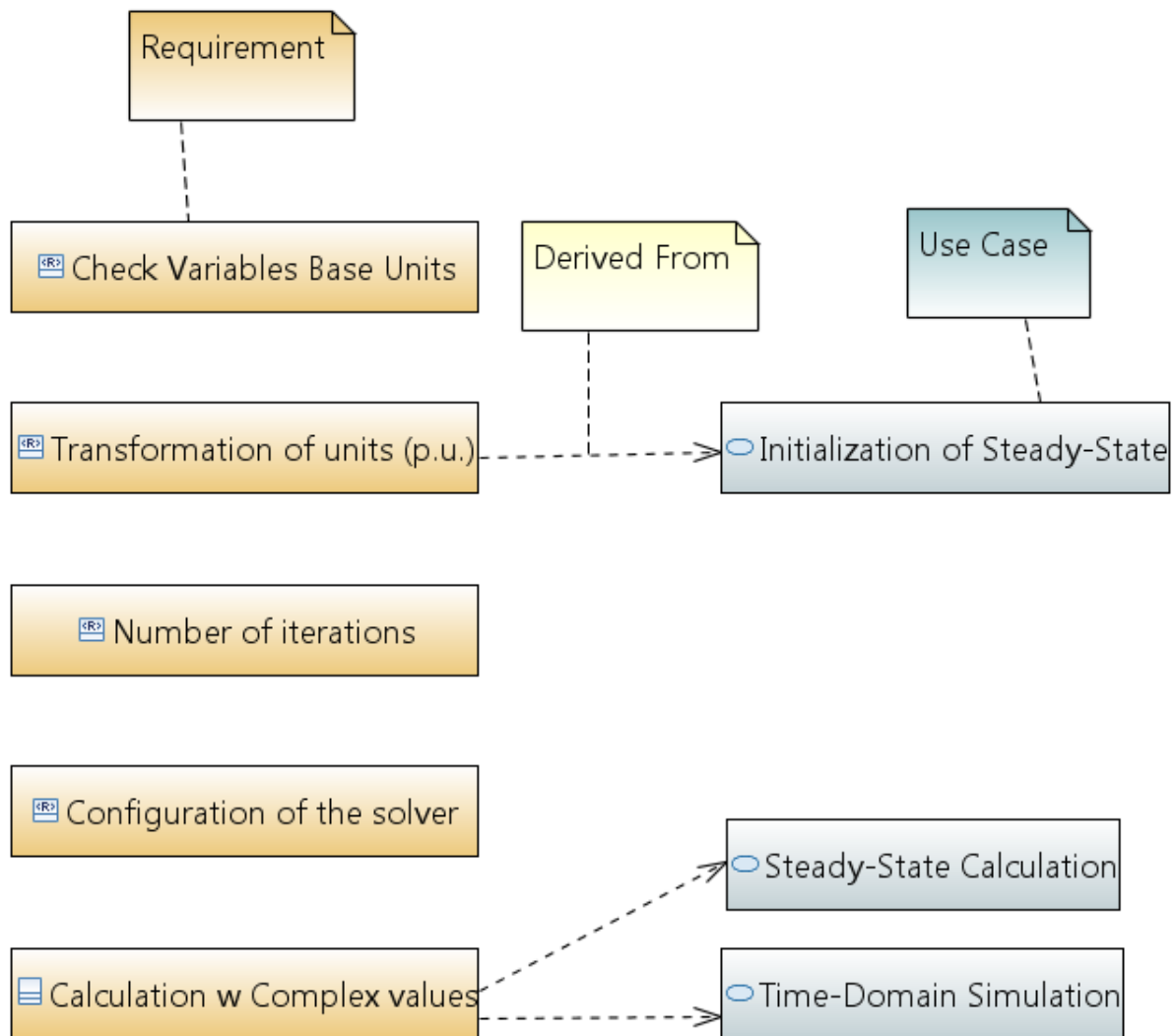


Figure 5 Requirements Diagram 2

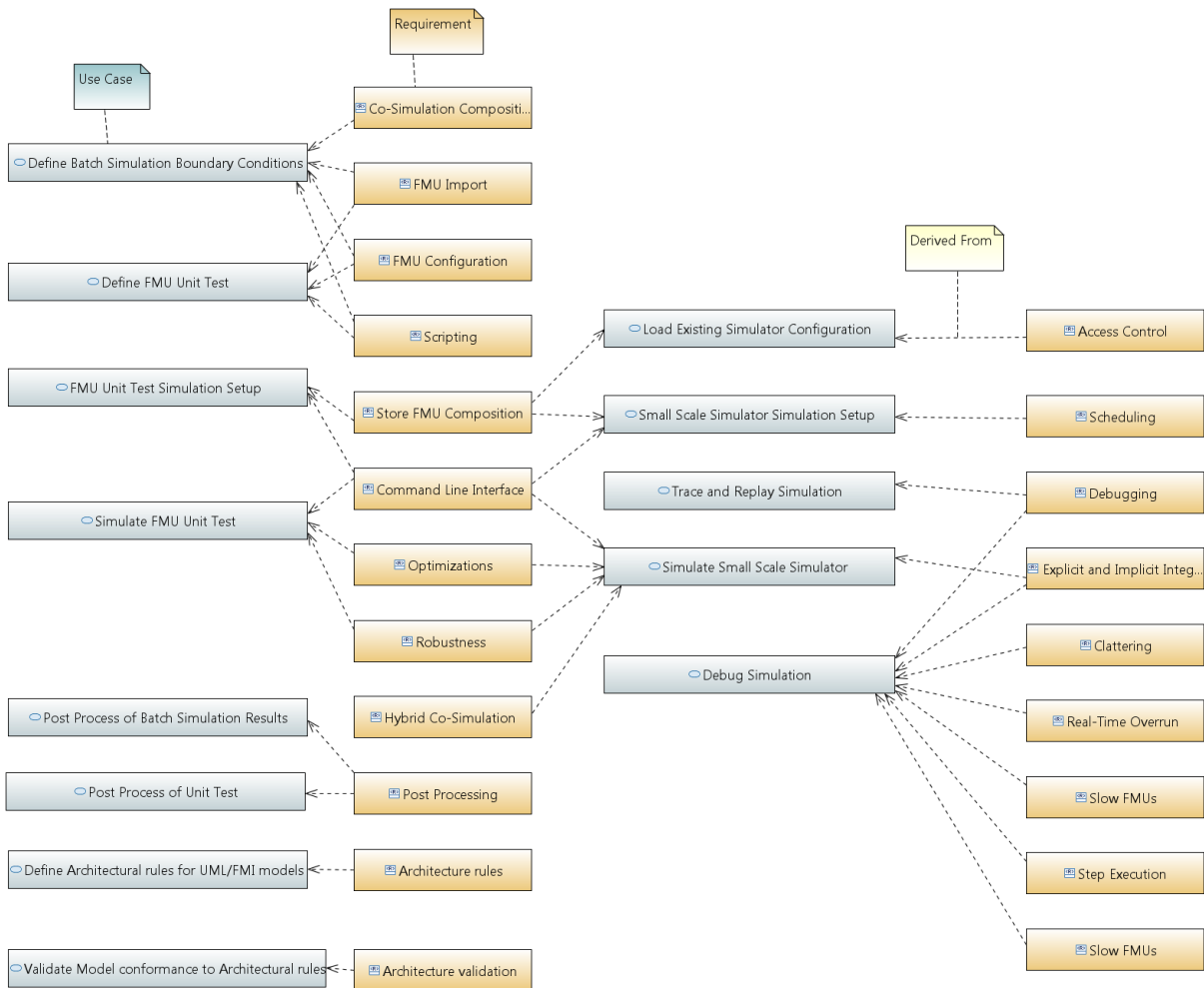


Figure 6 Requirements Diagram 3

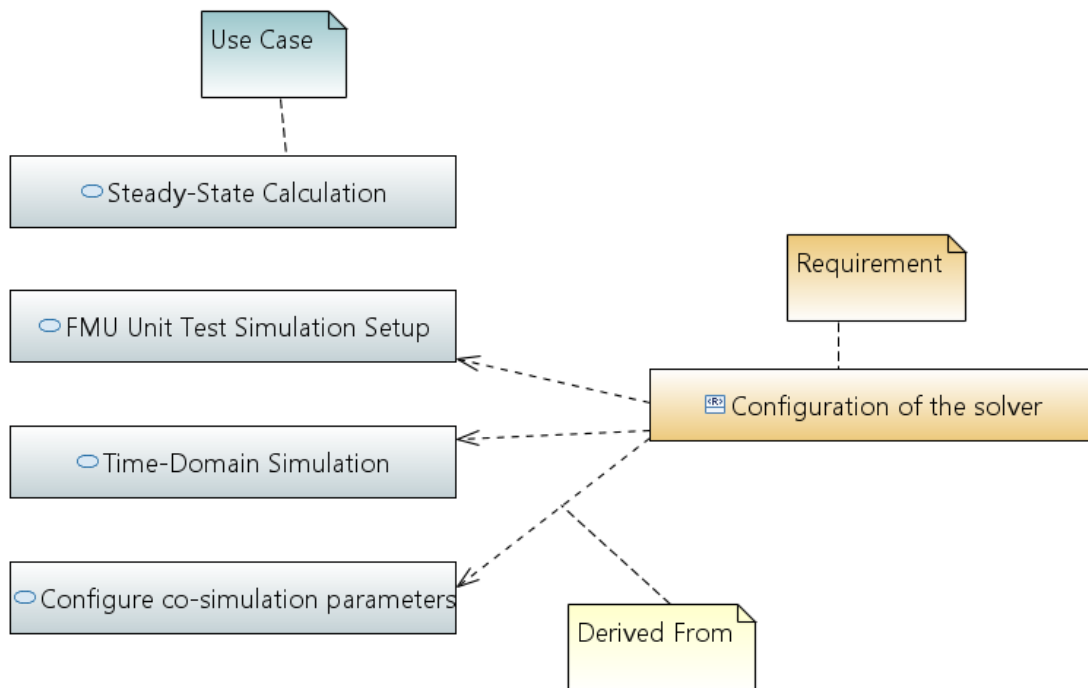


Figure 7 Requirements Diagram 4

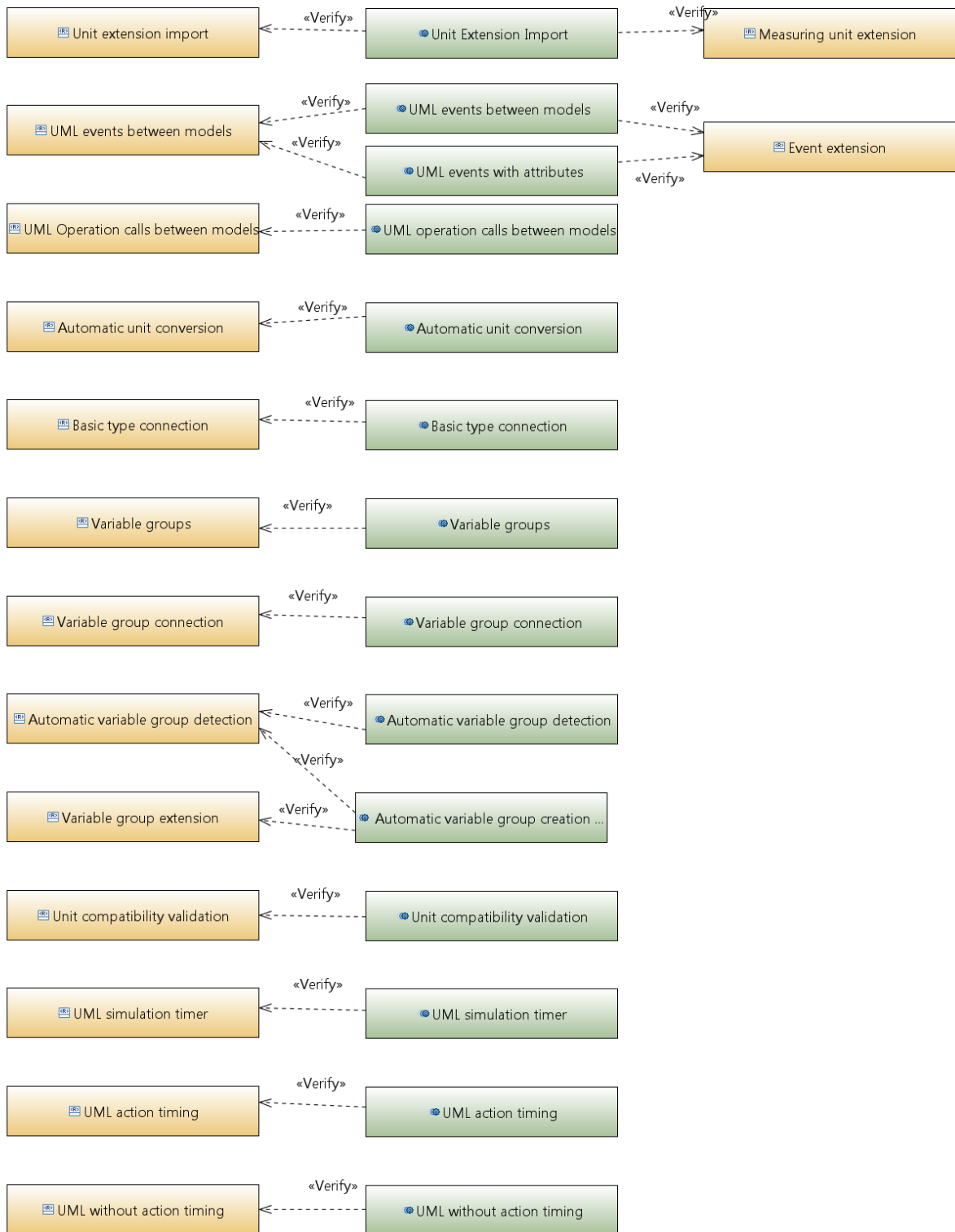


Figure 8 Requirements Diagram 5

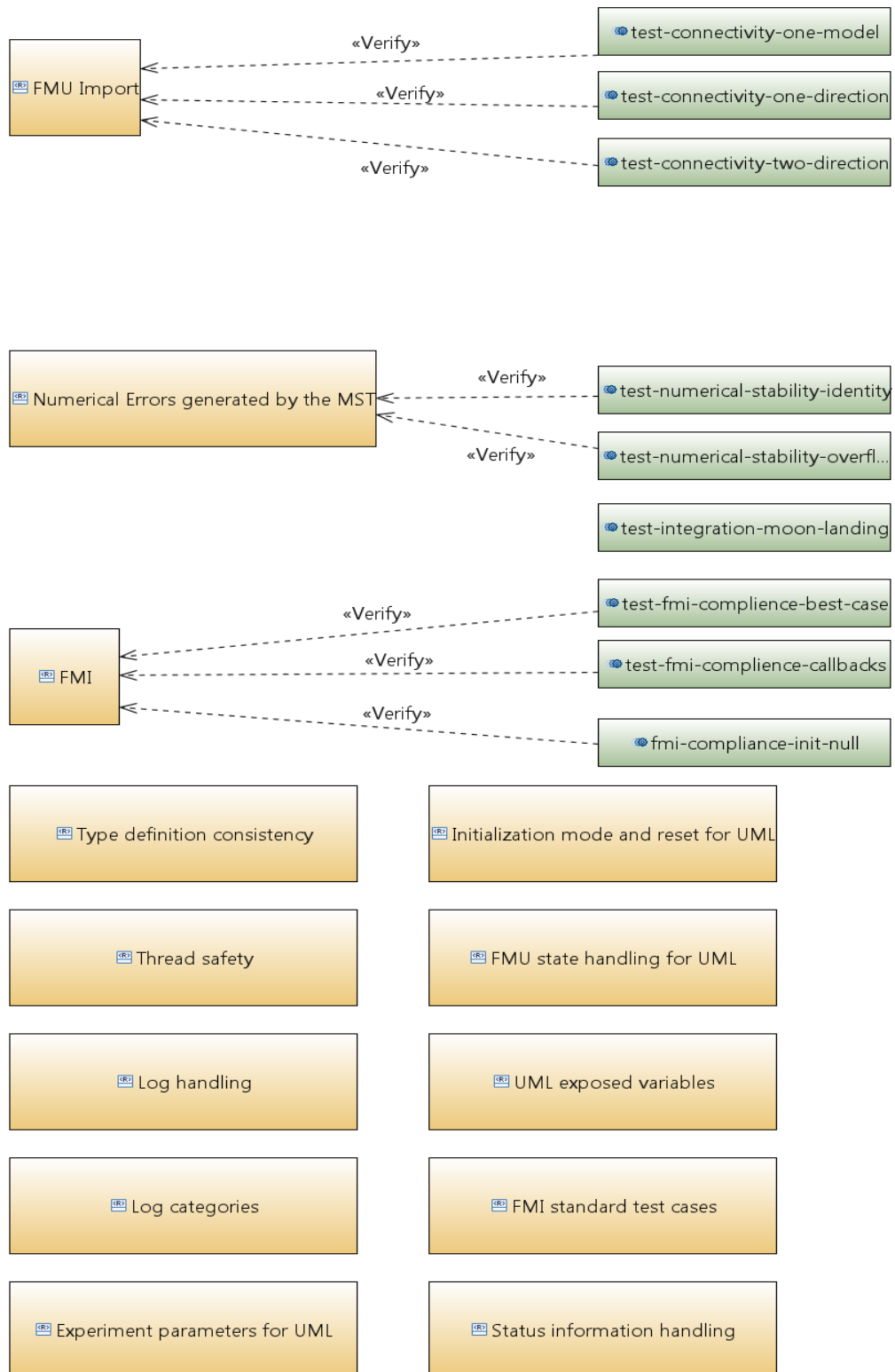


Figure 9 Requirements Diagram 6

2 NON FUNCTIONAL REQUIREMENTS

Non functional requirements describe emergent system properties such as safety, security, reliability, performance and response time.

id	Requirement Name	Text	Derived From (Use Cases)	Verified By (Test Cases)
Req28	Transformation of units (p.u.)	The MST should normalize all the model values to a per unit system.	[Initialization of Steady-State],	
Req29	Number of iterations	A solver may do a large number of computations until it reaches a solution point within a threshold limit. To avoid infinite loop breaking the threshold condition, a maximum number of iterations should be defined		
Req30	Platform	The MST shall fulfill all requirements when operating on Linux, Windows and MacOS X platforms		[Run all use cases,]
Req31	Scalability	The MST shall handle single FMUs as well as up to 1000 FMUs of varying complexity. Ability to perform FMU simulations in the cloud is desired.	[Post Process of Unit Test], [Simulate Small Scale Simulator], [Define Batch Simulation Boundary Conditions], [Small Scale Simulator Simulation Setup],	[Demonstrator Simulation,]
Req32	Efficiency	The MST shall be able to simulate connected FMUs efficiently using both fixed step size and variable step size methods. If possible MST shall run simulations in parallel.		
Req32.1	Multicore	The MST may (optimization) enable to handle simulation using distribution over CPU cores	[Configure co-simulation parameters], [Schedule	

			FMUs execution],	
Req33	Data Storage	The MST shall store data in machine readable formats (ex *.mat compatible with Dymola/OpenModelica formats, and *.csv formats)	[Post Process of Batch Simulation Results], [Define Batch Simulation Boundary Conditions], [Define FMU Unit Test],	[Demonstrator Simulation,]
Req34	Compatability	The MST shall support simulation model configurations stored by earlier versions of the tool		
Req35	Simulator Load Time	It should not take longer than 30s to load an existing simulator configuration	[Load Existing Simulator Configuration],	[Demonstrator Setup,]
Req36	FMU Load Time	An FMU shall be loaded for use in the MST less than 1s post the actor command	[Define Batch Simulation Boundary Conditions], [Define FMU Unit Test],	[Performance-Load single FMU,]
Req37	Data Interpretation	The MST shall be able to interpret data in machine readable format (ex: *.mat compatible with Dymola/OpenModelica formats, and *.csv)	[Post Process of Unit Test], [Post Process of Batch Simulation Results], [Define Batch Simulation Boundary Conditions],	[Demonstrator Simulation,]
Req38	Check Variables Base Units	Verify that each variable has the correspondent unit defined.		
Req39	Calculation w Complex values	The Master Simulation Tool must support arithmetic operation with Complex values	[Steady-State Calculation], [Time-Domain Simulation],	

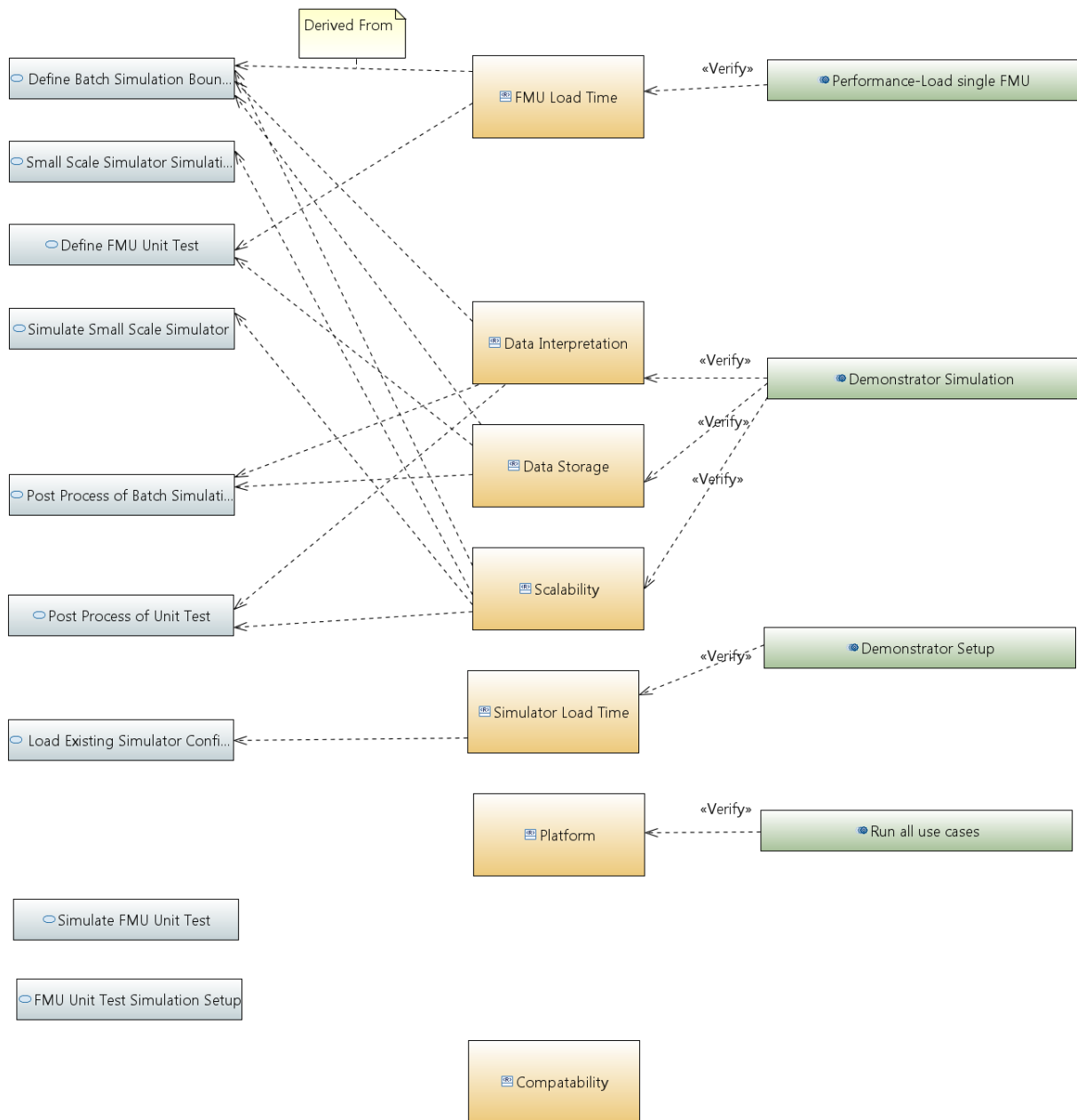


Figure 10 Non functional Requirements

3 VERIFICATION

Requirement Verification is typically the responsibility of Systems Engineers, who establish that the systems engineering product is conformant to its Requirement specification.

SysML primarily supports Requirement Verification via two complementary relationships:

- Satisfy («satisfy») dependency: The client model element fulfills a supplier Requirement.
- Verify («verify») dependency: The client TestCase («testCase») determines whether a supplier Requirement has been fulfilled.

In the next subsection, test cases for the verification of the MST requirements will be presented.

3.1 Test Cases

3.1.1 Unit Extension Import:

Given an FMU imported to the MST and an additional descriptor containing the measurement unit information for the input/output signals of the FMU (e.g. in JSON or XML).

When the descriptor is imported into the MST then it is expected to assign the correct measurement units to input/output signals of the FMU.

3.1.2 UML events between models:

Given:

- an UML interface with an UML event (signal) defined on it
- and a model expecting and a model providing that interface
- and these models imported to the MST as FMUs and connected to each other

When an event is sent to the second model during simulation, then the event arrives to the second model.

3.1.3 UML events with attributes:

Given a model that can receive an event with a real attribute when such an event is sent to this model during simulation then both the event and the sent real value arrive to the model.

3.1.4 UML operation calls between models:

Given an UML interface with an UML operation defined on it and a model expecting and a model providing that interface and these models imported to the MST as FMUs and connected to each other when one of the models calls the operation then the operation is called in the other model.

3.1.5 Automatic unit conversion:

Given two FMUs imported to the MST both with an exposed variable with distance units and one of those using metric, the other using imperial units and these variables are connected in the MST when simulating the models then the FMUs can work together with correct unit conversion by the MST.

3.1.6 Basic type connection:

Given two FMUs with exposed Int, Real, Boolean and String variables and compatible type variables connected in the MST when the model is simulated then the variable values are correctly passed between the FMUs .

3.1.7 Variable groups:

Given an FMU imported to the MST with three variables when the variables are selected in the MST then these variables can be grouped together.

3.1.8 Variable group connection:

Given two FMUs imported to the MST and these FMUs having compatible (count, types) variable groups created when connecting the variables then it can be done with a single connection between the variable groups.

3.1.9 Automatic variable group detection:

Given an FMU with an integer array with the variables named $i[0]$, $i[1]$, ... when the FMU is imported to the MST then the array is recognized as a possible variable group and the user can choose to accept it and create a variable group from it.

3.1.10 Automatic variable group creation based on external information:

Given an FMU with multiple variables with different types and an additional information describing variable groups (e.g. in JSON format), when the FMU is imported to the MST then the array is recognized as a possible variable group and the user can choose to accept it and create a variable group from it.

3.1.11 Unit compatibility validation:

Given two FMUs imported to the MST both with an exposed variable, one with a distance unit and the other with a speed unit assigned when these variables are connected in the MST then an error/warning message is displayed.

3.1.12 UML simulation timer:

Given an FMU exported from UML and a timing event or a timed action execution when the model is simulated then the timing event/timed action executes at the expected step.

3.1.13 UML action timing:

Given an UML state machine with an entry action when editing the UML model then execution duration can be set for the action and after exporting to an FMU and simulating in steps, the action execution takes the expected amount of steps.

3.1.14 UML without action timing:

Given an UML model without action durations set and this model exported as an FMU when the model is simulated then all possible actions are executed in a single simulation step.

3.1.15 test-connectivity-one-model:

Given a Modelica 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.

3.1.16 test-connectivity-one-direction:

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.

3.1.17 test-connectivity-two-direction:

Given two 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.

3.1.18 test-numerical-stability-identity:

Given an Open Modelica model and a Papyrus FUMML model and these models are exported as an FMU, when the models are simulated, then the value of the output parameters will remain the same.

3.1.19 test-numerical-stability-overflow:

Given a Modelica model and a Papyrus FUMML model and these models are exported as FMUs, when the models are simulated, then the value of the output parameters will increase and overflow together.

3.1.20 test-fmi-compliance-best-case:

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

3.1.21 test-fmi-compliance-callbacks:

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

3.1.22 fmi-compliance-init-null:

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

3.1.23 test-integration-moon-landing:

Given an FMU defined using xtUML and an FMU defined using OpenModelica, when the models are simulated, then the model will behave correctly.

3.1.24 Run all use cases:

Purpose:

Execute all use cases.

Main Scenario:

Execute all use cases.

Outcome:

Successful execution of all use cases.

3.1.25 Demonstrator Simulation:

Purpose:

Demonstrator simulation.

Main Scenario:

1, A single continuous FMU unit test is simulated in the MST implementing external boundary conditions expressed in machine readable format (ex *.mat, *.csv).

2, A single discrete events based FMU unit test is simulated in the MST. implementing external boundary conditions expressed in machine readable format (ex *.mat, *.csv).

3, A demonstrator consisting of a mix of discrete and continuous FMUs (both for model exchange and co-simulation) is scheduled, initialized, and a unit test simulated in the MST. implementing external boundary conditions expressed in machine readable format (ex *.mat, *.csv).

Outcome:

Stored simulation results from all 3 simulations specified in the main scenario.

Successful execution of all use cases.

3.1.26 Performance-Load single FMU:

Purpose:

Load single FMU complying with FMI 2.0 or later

Main Scenario:

Load a single FMU into the MST and monitor the time it takes for it to load

Outcome:

Single FMU loaded (ready for simulation setup) into MST

3.1.27 Demonstrator Setup:

Purpose:

Load demonstrator configuration.

Main Scenario:

- 1, Actor sets up demonstrator configuration consisting of atleast 5 different discrete and continous FMUs f(co-simulation and model exchange).
- 2, The actor saves and stores the configuration.
- 3, The stored simulation configuration is loaded into the MST and the loading time is monitored.

Outcome:

Saved, stored, and loaded simulator configuration.

Measured load time.

4 NON MST REQUIREMENTS

4.1 FMU export tool requirements

id	Requirement Name	Text	Derived From (Use Cases)	Verified By (Test Cases)	Priority
Req40	FMI compliance	FMUs exported from Papyrus, OpenModelica and other tools involved in the project have to pass the test with the "FMI compliance checker" provided by https://www.fmi-standard.org			
Req41	FMU size	The size of the FMUs generated from the tools involved in the project should be small, in accordance with the "Small footprint" point of section 1.1 ("Properties and Guiding Ideas") of the FMI 2.0 standard, in order to facilitate the deployment on Electronic Control Units with limited memory.			
Req42	UML simulation timer	A simulation-step based timer may be included in the FMU exported from an UML model.		[UML simulation timer,]	
Req42.1	UML action timing	The simulation duration of UML actions may be adjustable individually		[UML action timing,]	
Req42.2	UML without action timing	The simulation duration of UML actions should be considered zero if not specified otherwise.		[UML without action timing,]	

4.2 FMI standard extension requirements

id	Requirement Name	Text	Derived From (Use Cases)	Verified By (Test Cases)	Priority
Req43	Measuring unit extension	There may be an extension of the FMI standard to allow assigning measuring units to variables.		[Unit Extension Import,]	
Req44	Variable group extension	There may be an extension of the FMI standard to specify groups of model variables.		[Automatic variable group creation based on external information,]	
Req45	Event extension	There may be an extension of the FMI standard to allow sending events (e.g. UML state machine events) between FMUs.		[UML events between models, UML events with attributes,]	

5 REFERENCES

- [1] Soares, M. S.; Vrancken, Jos L. M., "Model-driven User Requirements Specification using SysML"; Journal of Software; 2008
<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.523.5486&rep=rep1&type=pdf>
- [2] Peter Fritzson, Principles of Object Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach. 1250 pages. ISBN 9781-118-859124, Wiley IEEE Press, 2014.

Appendix

This document results from the automatic documentation generation of models designed in SysML via the Papyrus tool.



www.eclipse.org/papyrus

www.omg.org

This document has been generated using **gendoc**. More information available here:

<https://www.eclipse.org/gendoc/>