INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

FMI PLM Interface

Specification for Product Lifecycle Management (PLM) of modeling, simulation and validation information

MODELISAR (ITEA 2 - 07006)

Document version: V1.0

31 March 2011

••

History

Version	Date	Remarks
	April 2010	Document creation
Draft	Sept 28th 2010	1st Draft version for consortium comments
Draft	Nov 26th 2010	Presentation to partners
V1.0	Mar 28th 2011	Specifications revised according partners comments and proposals: 1. Specifications versions is a Draft , not V1 2. Green data flow do not go through the application tolls , but more often in the network 3. These specification rather define generic processes instead of Interfaces with APIs 4. Need to understand how the configuration information is defined (which tools to launch where they are …) ; a description is needed 5. In some cases, can it be automatically launched? 6. Possibly several directories to store FMU & related data would be needed 7. Added chapters *4.2 Network description and 4.3 deployment description* 8. Added appendix *Remote Control By PLM* for optional deployment and execution 9. Diagrams correction

License of this document

Abstract

This document defines how to handle in a "Product Lifecycle Management" system of all FMI related data needed in simulation of systems:

(1) Functional Mock-up Units data needed for: edition, documentation, simulation, validation

(2) Co-simulation data needed for: edition, simulation, and Results management.

(3) Result valuation data needed for: Post-processing, analysis, Report

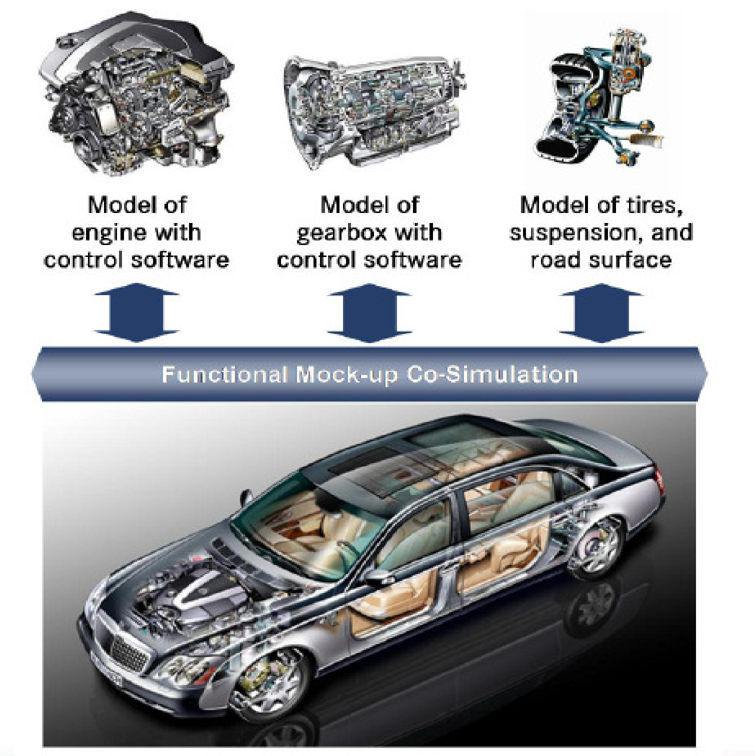
Contents

1. Purpose of the document

The Modelisar project is engaged for standardizing simulation of multi-domain physical models with control system models.

The Modelisar major goals are:

- Delivering easy use of simulation technology for virtual integration of multi-domain systems
  - Joint simulation of model being developed with different tools
  - Interface for simulation with AUTOSAR control units in the loop
- Developing a proposal for an open standard, the FMI-standard
- Proving the proposed standard by means of several real world use-case scenarios

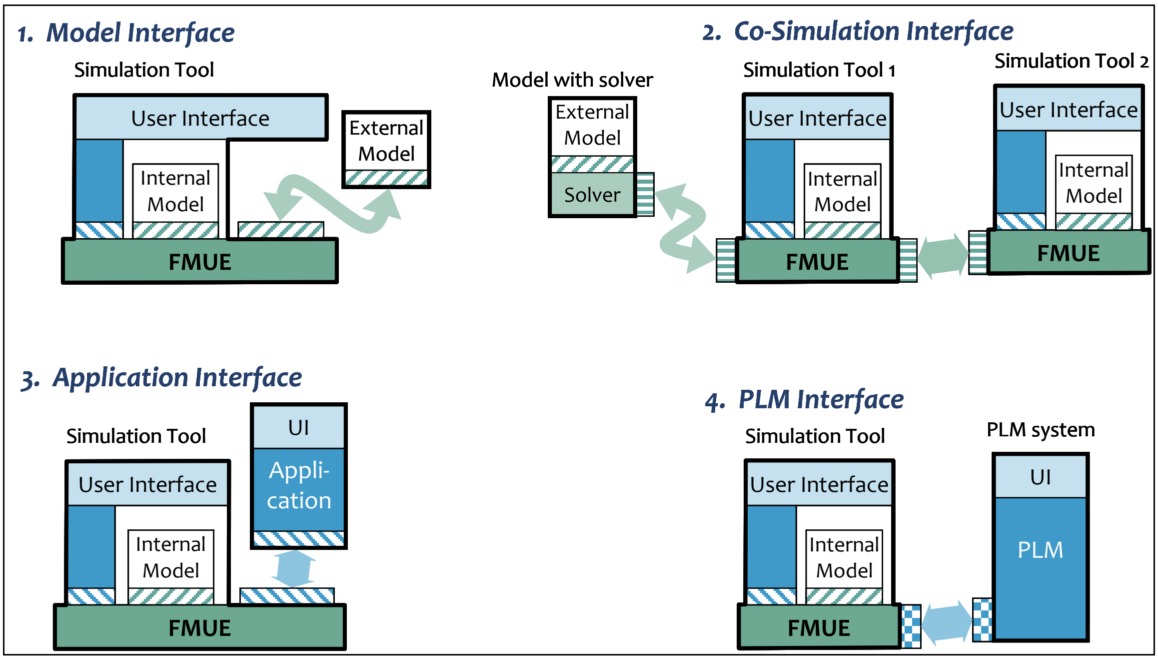


Modelisar Vision of automotive system co-simulation

To standardize simulation or co-simulation, Modelisar project defines FMI composed of 4 parts:

- *Model interface.* This standardizes the exchange file format: "**.fmu**", which is the Functional mock-up unit for model exchange.
- *Co-simulation interfaces.* This standardizes solver coupling during co-simulation.
- *Application interface.* This standardizes communication API between tools during Co-simulation.
- *PLM specifications* as a set of generic processes & a deployment format provided by PLM services to manage application data and co-simulation files.

The Purpose of this document is to specify the PLM interface part of FMI. As it will appear in this document, FMI aspects are described by an exchange format between the PLM system and the authoring (simulation, modeling ...) tools.



The different parts of the FMI standard [source: Functional Mock-up Interface (FMI) - Concept Description]

2. Objectives of FMI applied to PLM

These FMI PLM interface standard specifications define simulation and co-simulation integration methods, and requirements for the supporting PLM system, in order to guarantee that the other components implementing FMI such as simulation and co-simulation servers, editors or other applications may properly store and retrieve the relevant data into the PLM.

The PLM server provides services to Manage Lifecycle of:

- Executable units and the associated files (.fmu)
- Co-simulation Configuration files
- Results files
- Executable unit's data for Scenario: parameters, input, initialization…

This document leverages the specifications from the following other Modelisar FMI documents:

- Functional Mock-up Interface for Model Exchange
- Functional Mock-up Interface for Co-Simulation

3. Uses Cases scenarios

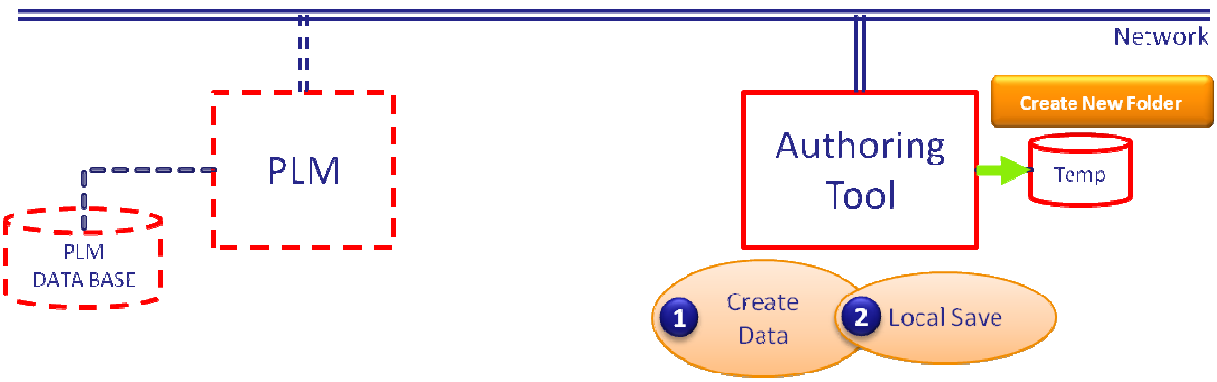
The following section present the usages scenarios supported by the FMI PLM Interface.

3.1. Scenario 1: Create a new executable unit in PLM

Creation of simulation data with authoring tool is independent from PLM, to create a new executable unit the user creates data in a working directory and stores it into the PLM data base via the Check in Action of the PLM Interface.

3.1.1. Scenario 1 - Step 1: Create executable unit with authoring tool

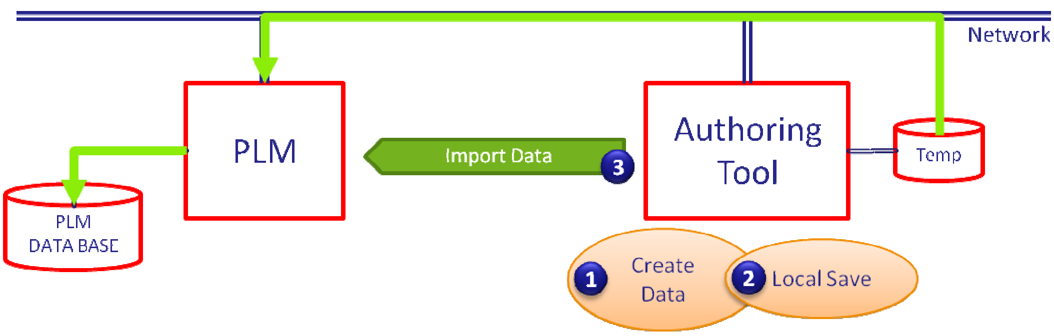
Using a specific authoring tool, the user creates data and saves on a local temp directory.



Data flow for Create data with authoring tool

3.1.2. Scenario 1 - Step 2: Import the new executable unit into the PLM

Whenever necessary the user save data into the PLM.



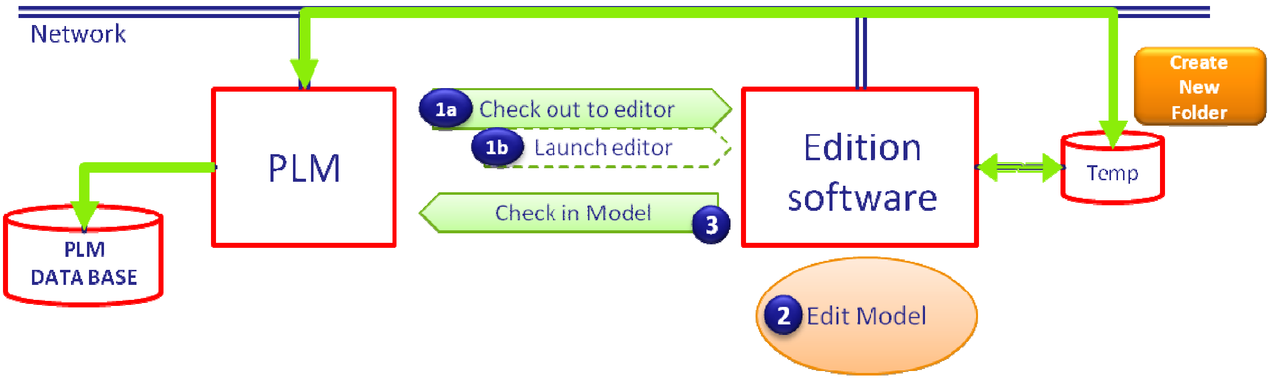
Data flow for Import data to new PLM container

All useful files are grouped into a new PLM executable unit and data are located with network path on authoring tool station.

3.2. Scenario 2: Edit a model

The FMI PLM interface assumes that the PLM user interface has the followings capabilities:

- Allowing user to extract executable unit files to editor working directory
- Launch editor in order to edit the extracted files, and wait end of edition process.
- Upload the updated files into PLM data base.



Data flow for Edit a model

To edit a model, the PLM tool extract Data to host with edition tool define by user, the files are checked-out to local working directory in a new folder.

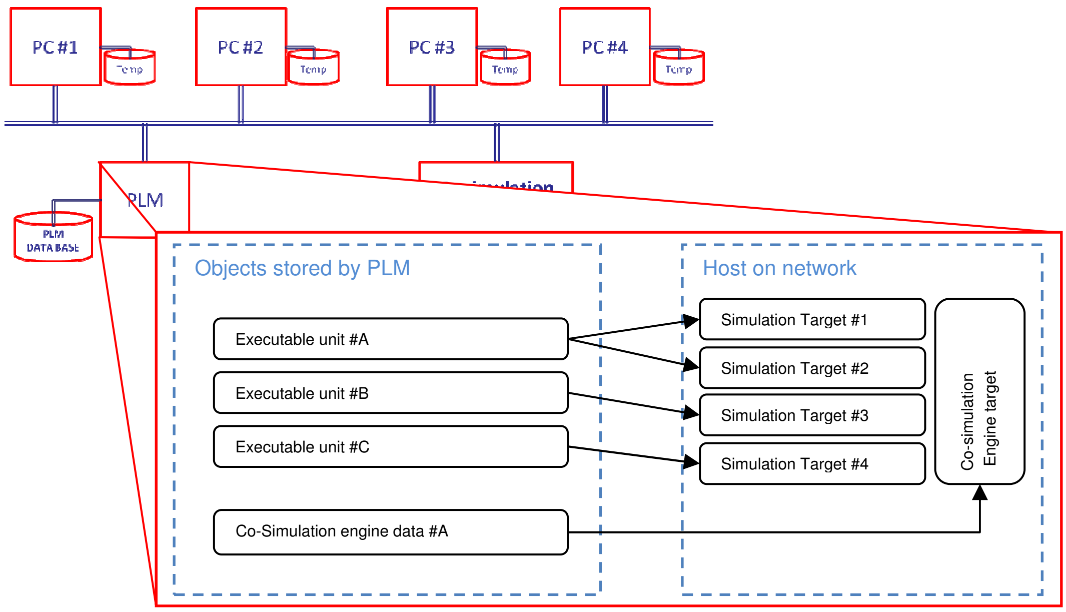
When all files are transferred, the corresponding tool can be remotely launched by PLM to open editor user interface.

Edition task is done from working directory, the PLM is not mandatory during this step.

When edition process is finished, the PLM check-in all modified data into PLM data base, this action can be launch automatically by PLM or manually on user request in PLM interface.

3.3. Scenario 3: Create co-simulation configuration data

A co-simulation typically involves several executable unit on several simulation targets, PLM place files on target according user define configuration.



Definition of a configuration for co-simulation

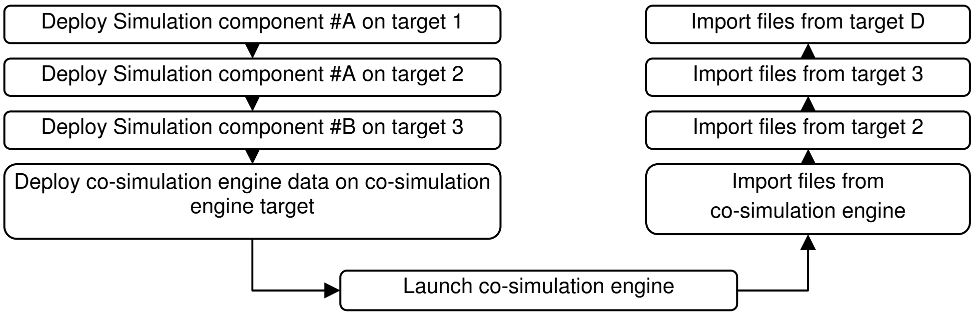
In PLM tool, the simulation manager defines association between executable unit and simulation software on network host.

The user define simulation deployment by a sequence of Data management task: Deploy on target, launch application or import data.

The FMI PLM interface assumes that the PLM user interface has the capabilities to define:

- Export rules to define how executable unit will be deployed on targets.
- Launch tool options for remote application on targets.
- Import rules to set which files will be imported as result files and how store it in PLM data base.
- Sequences for deploying, launching and importing.

Hereafter is a deployment sequence for a complex co-simulation example:



Sequence for deployment and import for simulation data

3.4. Scenario 4: Simulate a Co-Simulation configuration

To illustrate data management in co-simulation context, **two variants** of the simulation workflow will be presented with placement either by PLM tool or by Co-simulation engine.

Variant 1: The **PLM places data directly on each simulation target**. In this case the simulation activity is split in 3 steps:

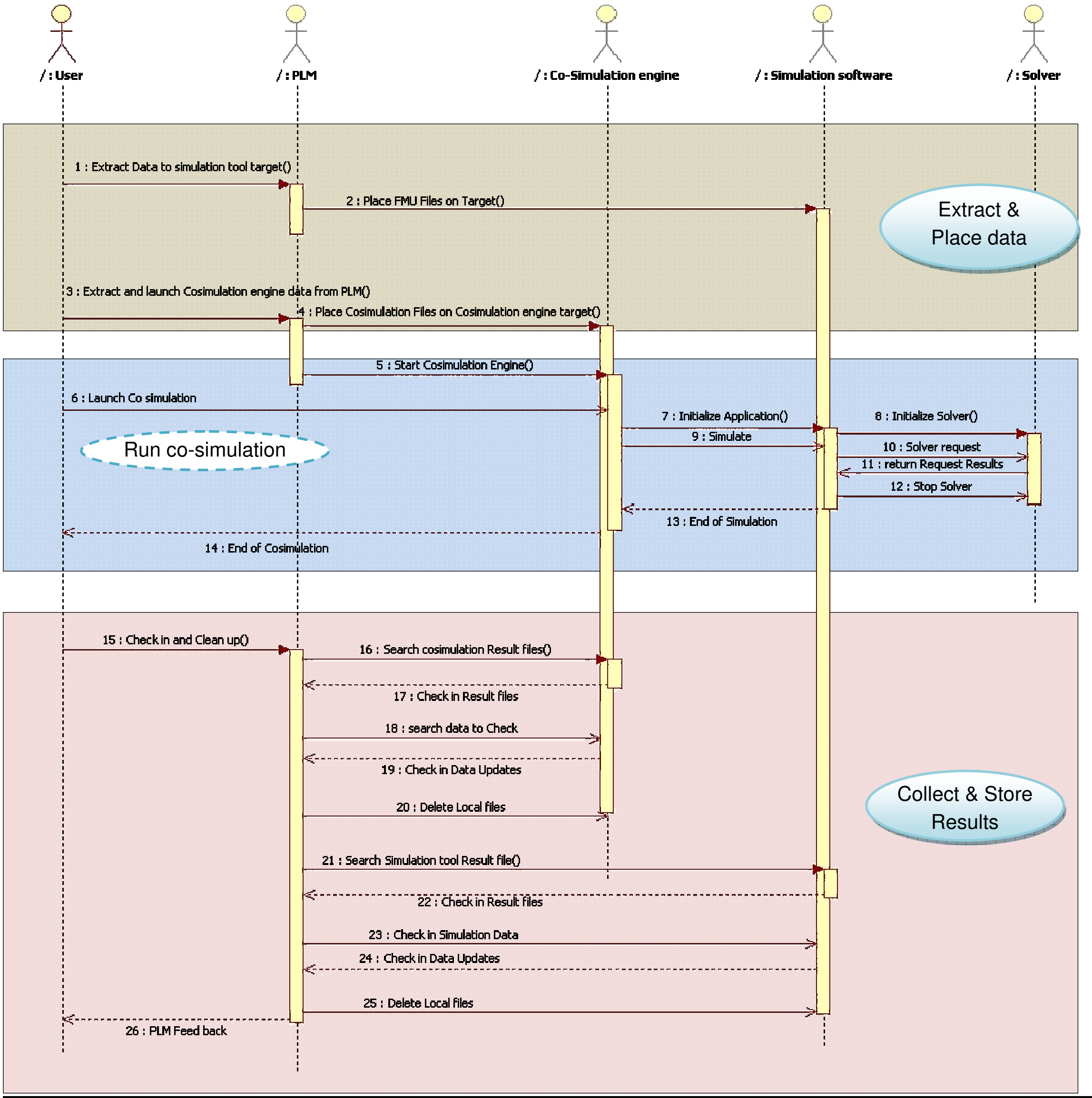
- Extract and deploy Data
- Run co-simulation
- Collect and store results to PLM data base

Variant 2: The PLM tool allows **Co-simulation engine to place Data on simulation targets**. The simulation activity is split in 5 steps:

- Extract data from PLM data base
- Place data on targets
- Run co-simulation
- Collect results
- Store results to PLM data base

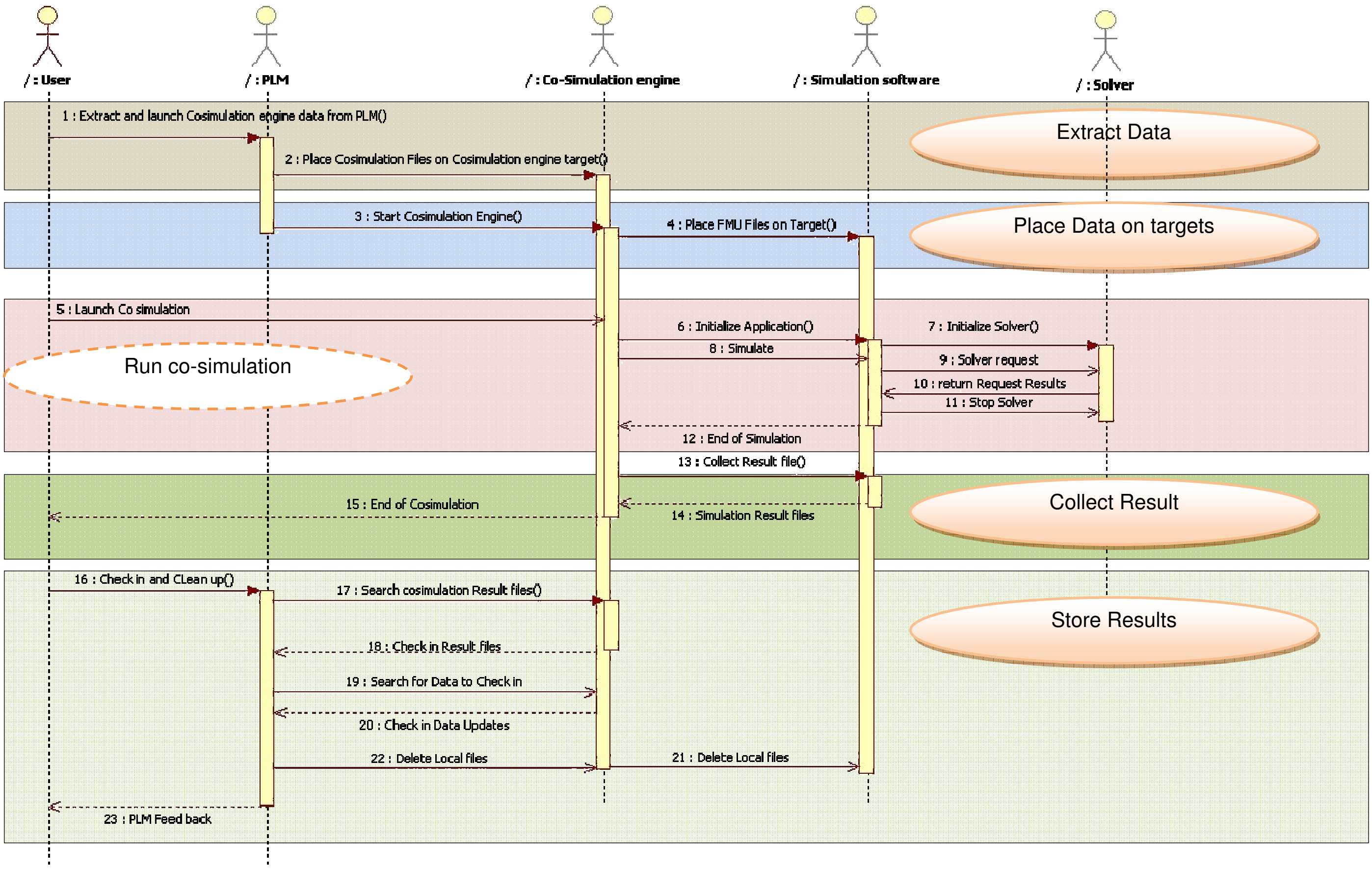
The followings diagrams provide the details of each variant:

Variant 1:



Execute a co-simulation – Variant 1

Variant 2:



Execute a co-simulation – Variant 2

3.4.1. Scenario 4 - Step 1: Extract and place data on co-simulation targets

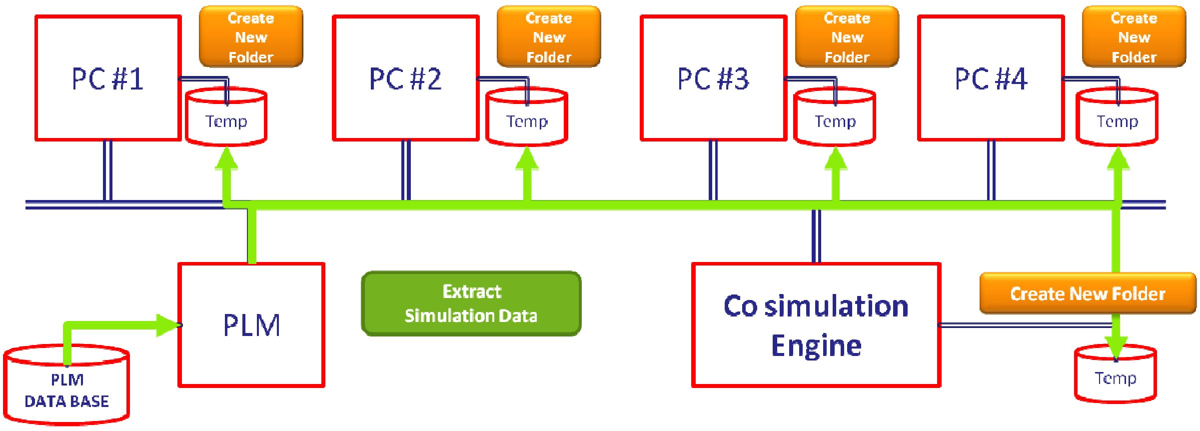
In this scenario all data are stored in PLM data base, the simulation user has to extract executable unit (.fmu) and automatically place them on each target in order to run the co-simulation.

The FMI PLM interface assumes that the PLM user interface has the followings capabilities:

1. Allow the PLM simulation manager to specify how stored files will be deployed on each target.
2. Extract from PLM and place either on each target in one step, or place it solely on co-simulation engine working directory. The co-simulation engine will then place it on second a wave to the simulation targets.

3.4.1.1. Scenario 4 - Step 1 / Variant 1: Placement on Simulation targets by PLM

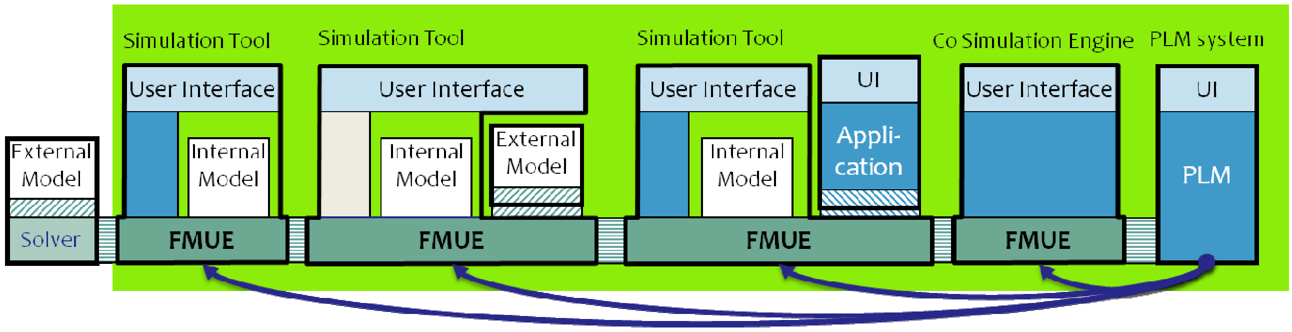
The user request deployment task with PLM interface



Data flow for Placement on Simulation targets by PLM

Data are extracted by PLM to new folder created on simulation targets' working temp directory.

At this step, local initialization can be processed on each target by remote launch of local application by PLM tool as preprocessing actions.



Involved items for Placement on Simulation targets by PLM

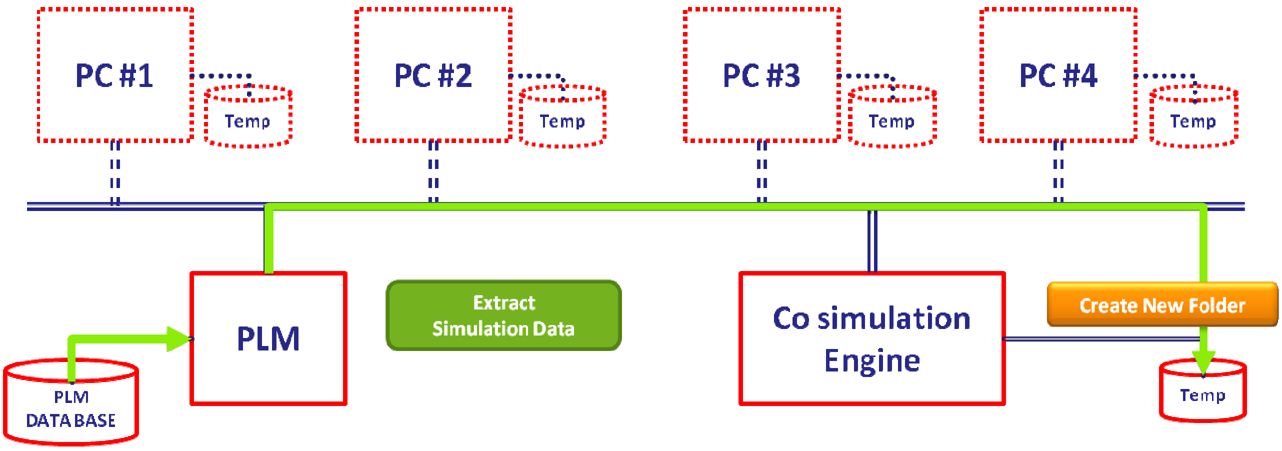
3.4.1.2. Scenario 4 - Step 1 / Variant 2: Placement by Co-simulation engine

In this case, the placement task is done in 2 steps:

- Extract files on co-simulation engine station
- Place Data on each target by co-simulation engine

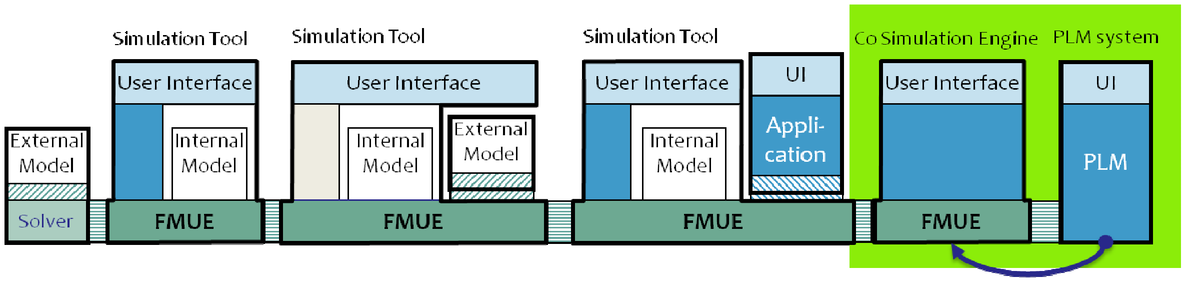
Co-simulation engine need remote control on each co-simulation targets, and need to manage placement by itself.

At first time, PLM tool extract all files in a new folder on Co-simulation engine working directory.



Data flow for Placement on Co-simulation engine

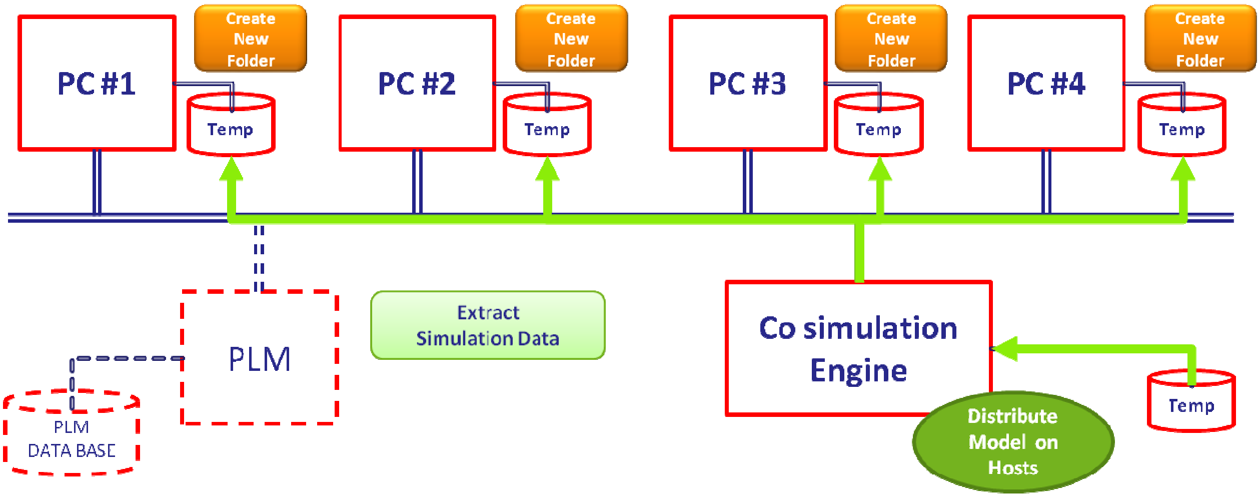
Only PLM and Co-simulation engine station are engaged in this step.



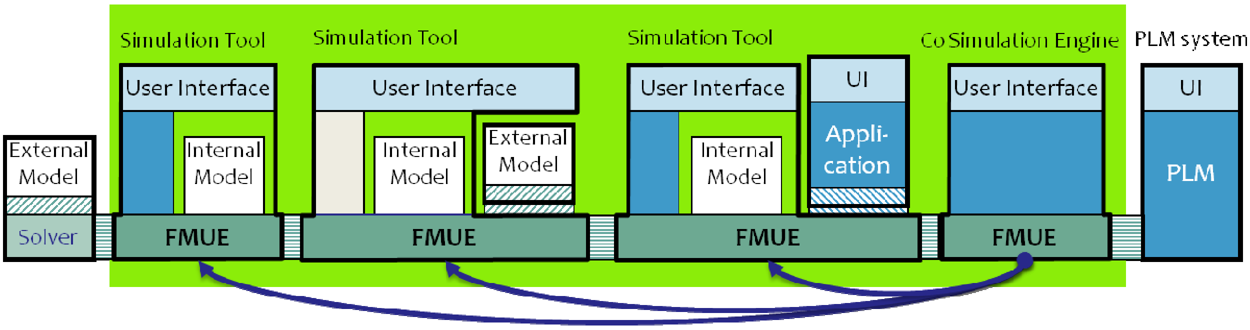
Involved items for Placement on Co-simulation engine

The Co-simulation engine is in charge of place, launches and controls Simulation tools; placement on target is done from Co-simulation temporary working directory.

At second time, Co simulation engine is start and deployment task can proceed:



Data flow for deployment by Co-simulation engine

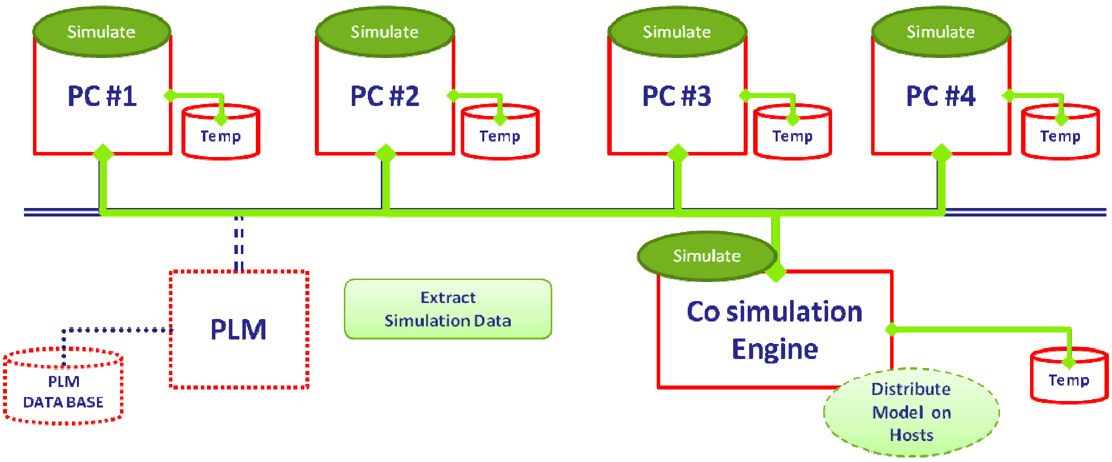


Involved items for deployment by Co-simulation engine

The co-simulation deployment task can be launch by PLM at the end of extract process.

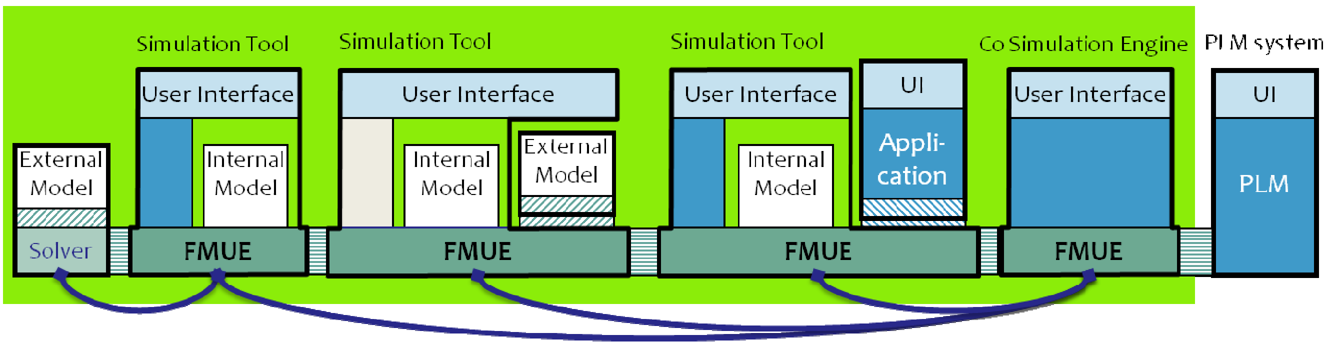
3.4.2. Scenario 4 - Step 2: Simulate

The user start simulation between all simulation targets and co simulation engine with files placed in precedes steps.



Data flow for simulation

PLM access is not managed during simulation or co-simulation.



Involved items for simulation

During simulation task, all executable unit deployed on target can locally generate result files; the user must be able to store it with PLM tool.

3.4.3. Scenario 4 - Step 3: Collect and store Results in PLM

As in Placement cases, Results import in PLM can be done either from each targets or solely from co-simulation engine working directory when all results files from each targets are already collected in it.

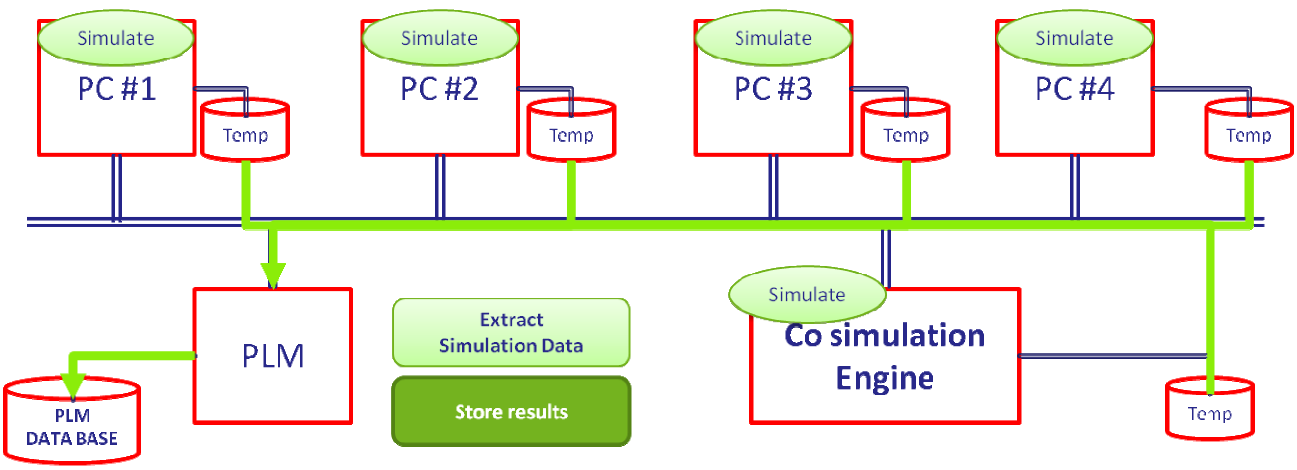
The FMI PLM interface assumes that the PLM user interface has the followings capabilities:

1. Allow the PLM simulation manager to specify how results files will be import and manage in PLM data base.
2. Import Results files from co-simulation target and simulation targets.

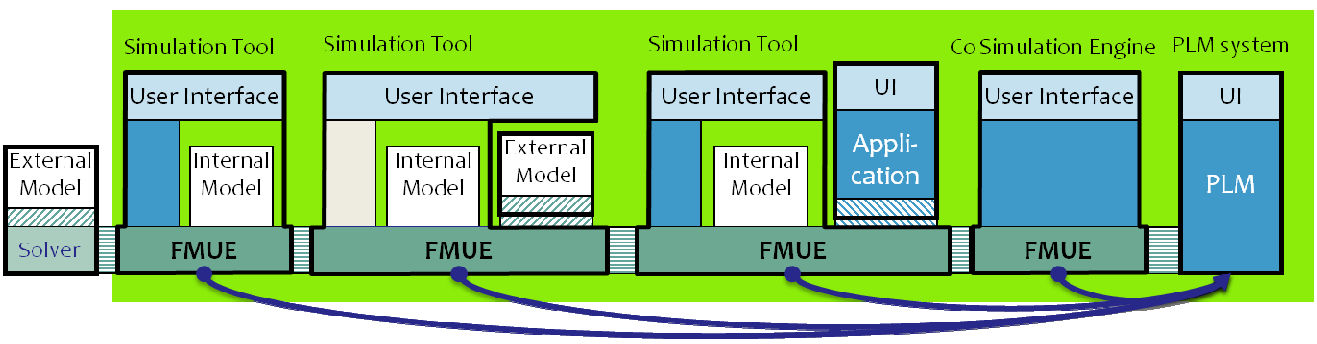
After Import, PLM can clean-up created working directory.

3.4.3.1. Scenario 4 - Step 3 / Variant 1: Collect results on each target and import by PLM

After simulation task, results files on Directory created at extract step are import to PLM data base.



Data flow for Collect results on each target and import by PLM



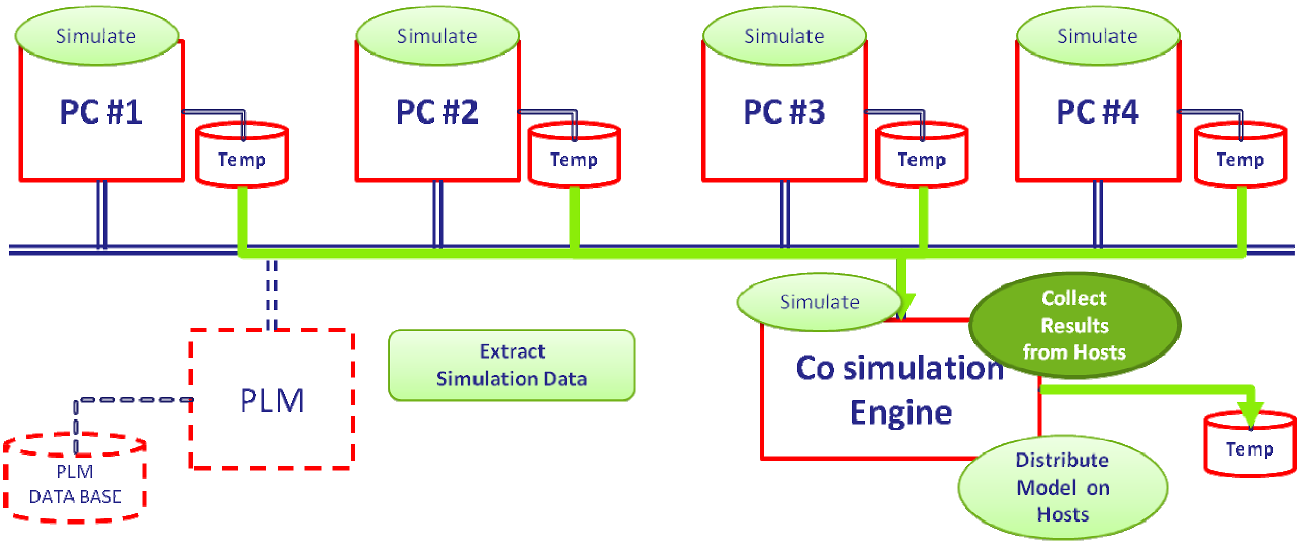
Involved items for Collect results on each target and import by PLM

3.4.3.2. Scenario 4 - Step 3 / Variant 2: Collect results on each target by Co-simulation engine and import by PLM

In this case, the Collect and import tasks are done is done in 2 steps

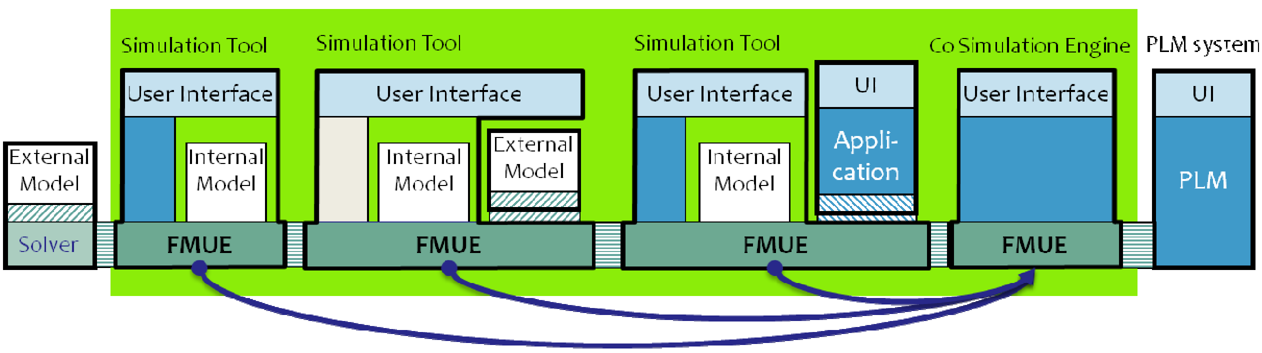
- Collect results files on co-simulation engine working directory
- Import Results from co-simulation engine working directory

After Co-Simulation task, All Results are collect by Co-simulation engine to temporary working directory.



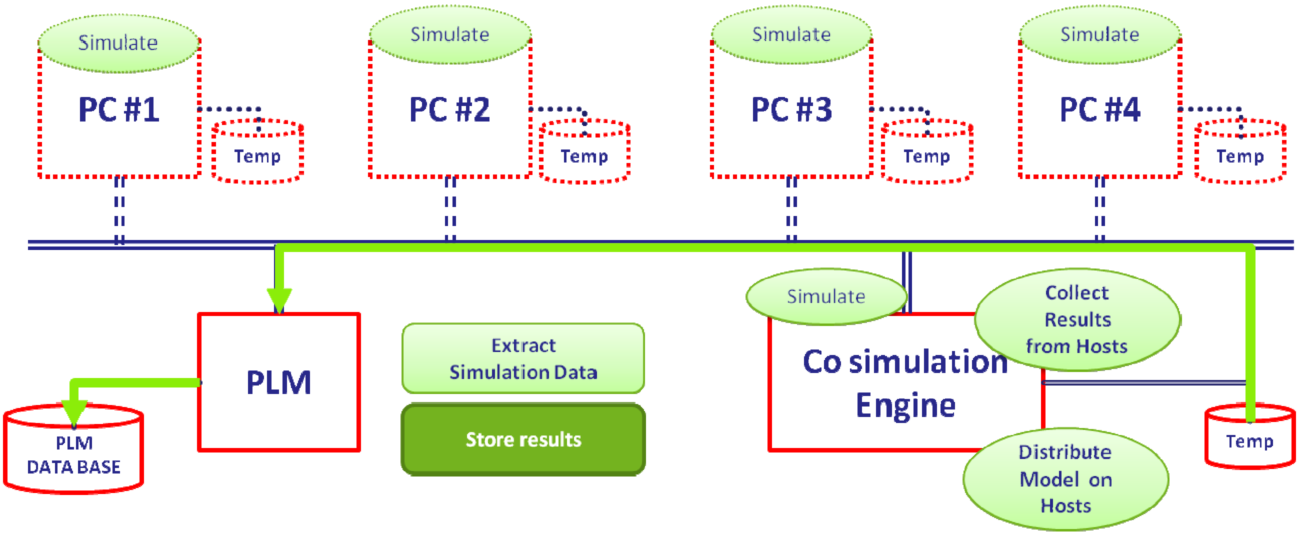
Data flow for Collect results on each target by Co-simulation engine

The Co-simulation engine can order Files into hierarchical file structure known by end user and PLM.

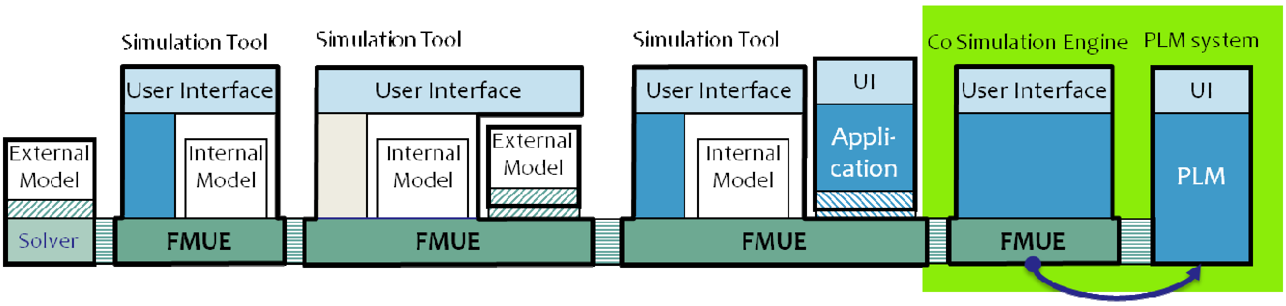


Involved items for Collect results on each target by Co-simulation engine

At the end of collect task, Co-simulation engine can run post processing task on collected data files before stop to let the PLM collect and store all files in Temp directory created at export step.



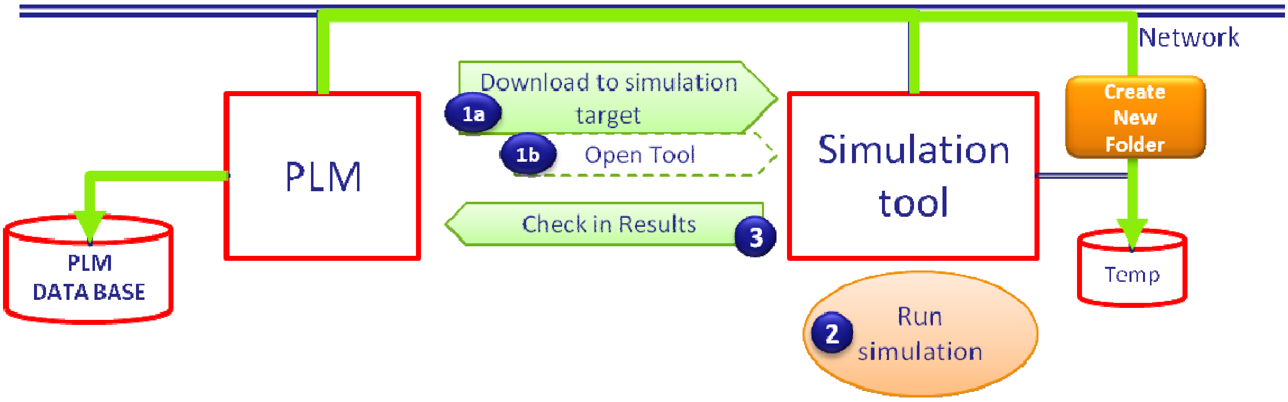
Data flow for import by PLM



Involved items for import by PLM

3.5. Scenario 5: Simulate single executable unit.

Simulate single executable unit process can be described as a simplified view of co-simulation process: with only one target.



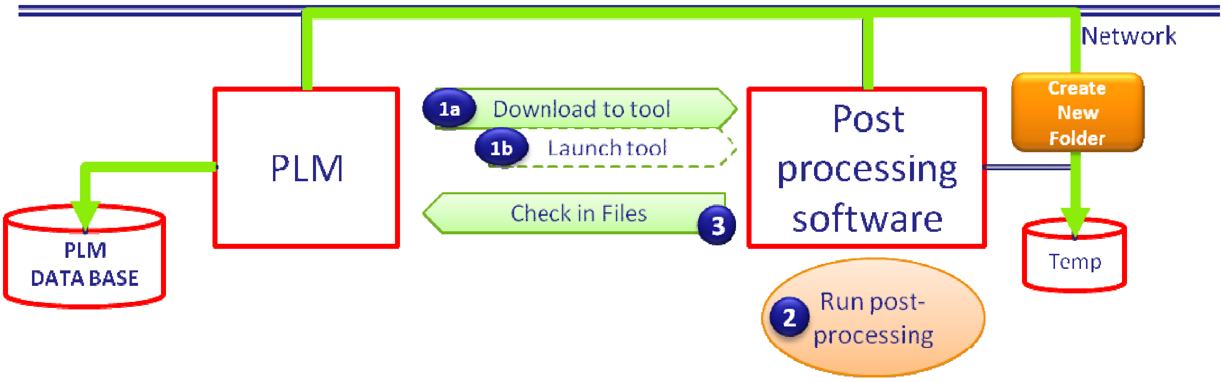
Involved items for single executable unit

The FMI PLM interface assumes that the PLM user interface has the capability to associate executable unit and simulation tool on target.

3.6. Scenario 6: Post process simulation result

To run post processing task, the PLM tool extract useful results files to host with corresponding tool define by user, the files are download to local working directory in a new folder, then the corresponding tool can be remotely launched by PLM to open user interface.

The FMI PLM interface assumes that the PLM user interface has the capability to associate data to post processing tool on target.



Involved items for post process simulation result

Edition and computations task are done from working directory, the PLM is not mandatory during this step, when finish, the PLM check-in all created files into PLM.

The FMI PLM interface assumes that the PLM user interface has the capability to associate data to post processing tool on target.

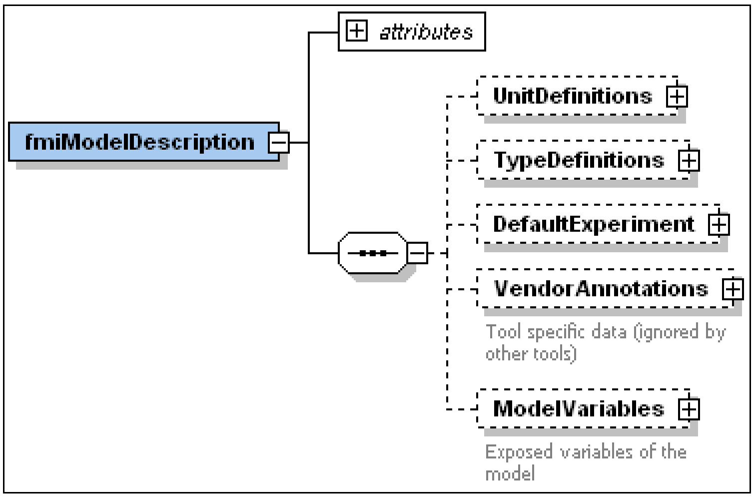
3.7. Scenario 7: Search Item in PLM

The FMI PLM interface standard assumes that the PLM user interface has the followings capabilities:

- Define relation between items co-simulation configuration, executable unit and simulation results
- Allowing user to navigate on relation between PLM items.
- Allowing user to search data with request on attributes and metadata.
- Exposing Information exposed on the FMU (model properties, input / output ports list…)
- Controlling data access according to user, role and projects.

For further details please refer to the FMI specification for Model exchange which provide XML format for model exchange.

As an example the following illustration show the top of model exchange tree.



Example of FMU properties

Followings sample of user request on PLM interface:

- Search and explore executable unit from owner in PLM
- Search results associated to simulation scenario
- List all executable units involved in a co-simulation selected by user
- Navigate on relations between theses PLM items

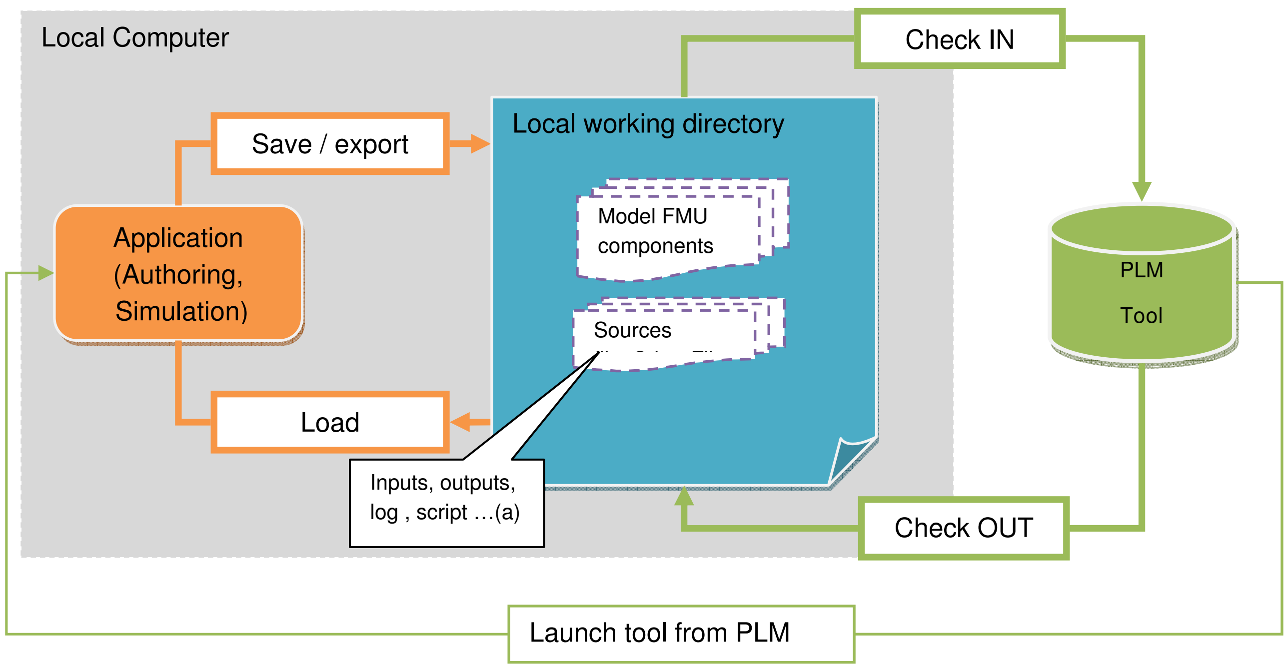
4. Functions to be provided by PLM

4.1. Summary of PLM provided functions

Summery of the PLM capability Needed to implement the FMI PLM interface standard are :

- navigation & search functions :
  - Access to the FMU properties
  - Search on Simulation content attributes ex : FMU's
  - Navigate on links between PLM items

- Basic PLM functions:
  - The PLM deploys and / collects FMUs and the associated files
  - Check Out or download on working directory of a target on network.
  - Check-in or import into PLM from working directory of a target on network.
  - Launch remotely application on a target define on network.



Services between external tools and PLM

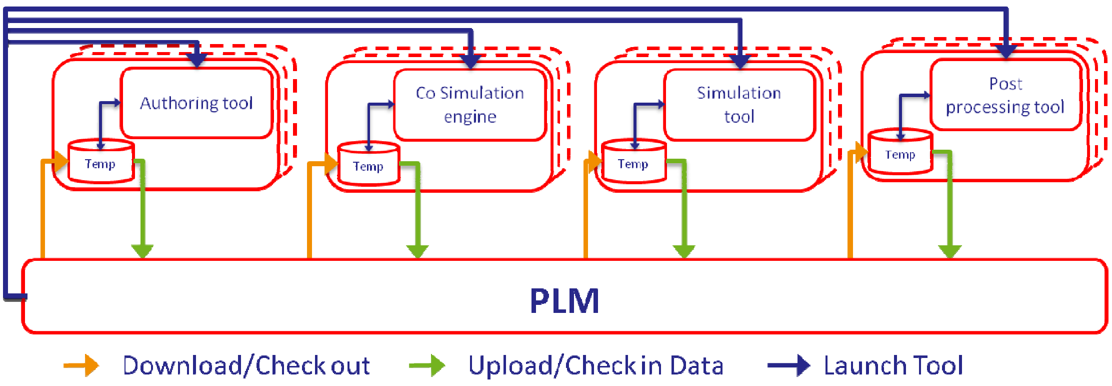
(a) additionnal information such as (input simulation data, calculated data, acquired data, generated reports, log files etc…)

- Admin functions:
  - Manage access for users to simulation objects
  - Create a new Simulation content
  - Edit/delete existing Simulation content

- Associate simulation content to simulation result or co-simulation configuration

In this generic approach, tools could be launched by PLM. In the reality, it is a matter of implementation. It also depends on the connection possibility.
The general way to proceed will be that for a Co-simulation the tools are cascade launched by Co-simulation engine.

The FMI PLM interface doesn't prescribe how the functionality should be implemented by the PLM.



List of interactions between PLM and "outside PLM" applications

The whole interest of the proposed approach is that it works without imposing specific API to be developed on top of the PLM. Basic PLM functions are adequate to support the FMI PLM interface, in a fully open approach.

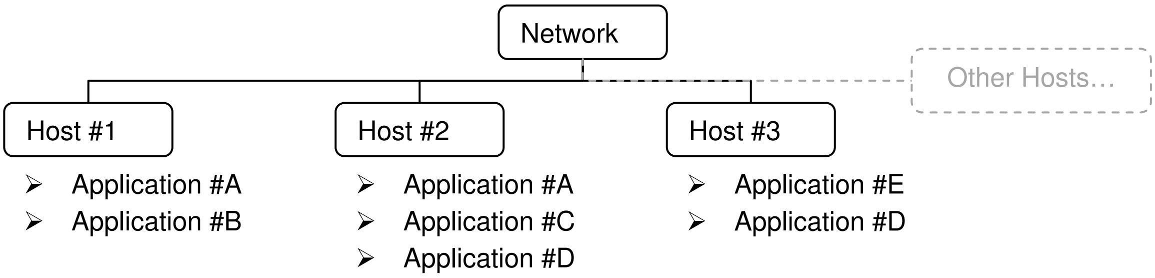
4.2. Network description

This is a NEW chapter

Purpose: describe the topology of the available resources (machines, simulation tools …) in the enterprise network.

All the files that have been identified in the previous chapter are stored in the PLM system. These files have to be extracted and deployed on specifics hosts, as defined by the PLM user.
Before this stage, as a pre requisite the Network available **hosts** and **applications** must be declared in the PLM system.

Each **Host** is identified and associated to a list of available **Applications**. These information will help user to choose the good target (Host + Application) for edition, simulation or post processing activities; for example according to their hardware and software specifications & versions.



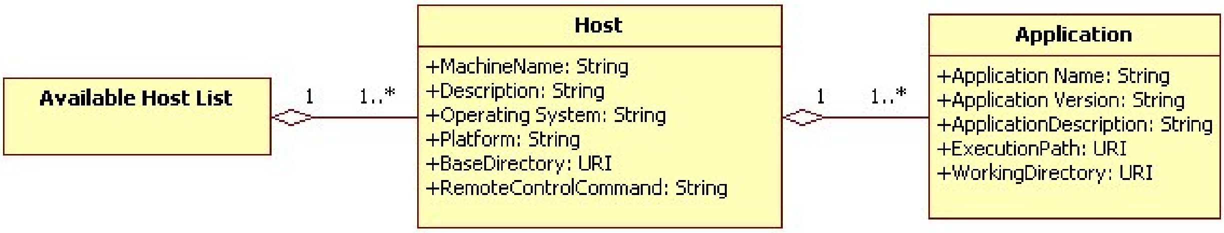
Available Hosts and applications on Network

4.2.1. Description

4.2.1.1. Host description

The available means are defined in PLM by:

- ***MachineName*** (URI) (must be unique on the network) to locate the Host.
  - *Ex : 192.168.2.1 , MyDomain/MyHostName*
- ***Description*** (string) to user-friendly identify the host
  - *Ex : hardware in the loop host*
- ***OperatingSystem*** (string) to define the Host operating system
  - *Ex : Unix, Windows XP, Windows 7, Red Hat...*
- ***Platform*** (string) to describe the Hardware of the host like processor (x86, x64...) or Memory size and help user with host choice for simulation tasks
  - *Ex : x64, 8 core with 16Go Ram*
- ***BaseDirectory*** (URI) to define the directory where files (FMU, data, outputs ...) will be placed in. This URI gives the access from local machine to a temporary directory.
  - *Ex: C:\Temp*
- **Remote control command** (string) to set the command line in order to launch. Notice this command depends of the PLM tool usage and the network security context.
  - *Ex: SSH -2*
- **ApplicationList** gives the available application on the host.
  - *Ex : Dymola 7.4,Matlab 6.2...*



UML Data Model to manage network description

4.2.1.2. Host's Application description

The PLM Application List associated to one Host represents a way to launch Application on this Host, these applications are describe by:

- **ApplicationName (String) to** identify the Application and help user to find corresponding action.
    - *ex: Dymola*
- **ApplicationVersion (string)**
    - *ex : 7.4*
- **ApplicationDescription (string)**.
    - *ex : Modelica Editor and simulation tool*
- The full **ExecutionPath (URI)** corresponding to the command line to launch locally the targeted application on the host.
    - ex: "C:\Program Files\Dymola 7.4\bin\Dymola.exe"
- The application **WorkingDirectory (URI) defines** a directory where the application has to be launched.
    - *ex: "C:\Documents and Settings\User\My Documents\Dymola\"*
      or C:\Programmes files \ToolName

4.2.2. Description format

The previous information describes the network topology through **specifications**.

The implementation format should not be defined here, for several reasons:

- Different implementations have no impact on the cooperation between the PLM system and the modeling / simulation tools.
- This part of specifications has no impact on the other FMI specifications (currently available are Model Exchange V1 & Co simulation V1 specifications).

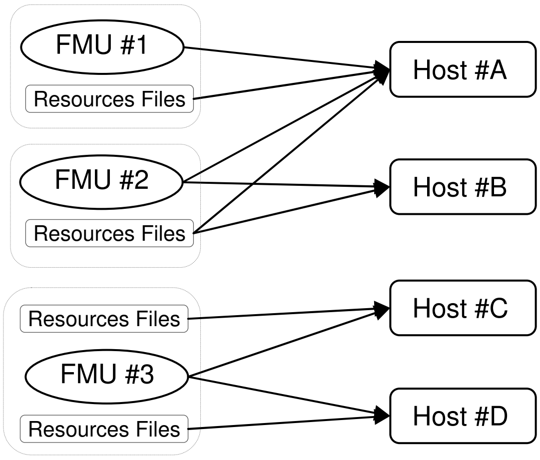
Thus the implementation will be under the responsibility of the PLM solution.

4.3. Deployment description

This is a NEW chapter

Purpose: describe the configuration that will be deployed in the network, and that will be available for the authoring tools (simulation engine, modelling tool, post processing tool …)

To prepare a simulation (or any other activity as defined in the Use Cases), the PLM provides the capability to associate the FMUs and the available means (host & applications).



FMU deployment on Hosts

Especially notice from the above example that:

- The same FMU can be used once or several times, thus several instances may be executed
- Resources files can also be used in the same way, although simulation parameters should probably be different
- From different instances of the same FMU, different results will be collected

As a consequence, additional PLM information has to be found in the deployment directories, and vice versa when Checking In the information that have been produced by the simulation. There are necessary for the PLM system, but not for the deployment description.

These information include:

- tree structure identification of the FMU and the configuration to be simulated
- FMU identification and version

4.3.1. Description

To prepare the deployment, the PLM User builds all needed associations between FMU and their targeted localisation (Host + Application).

Deployment information is composed of:

- **fmuDeploymentList** : list of fmuDeployment items

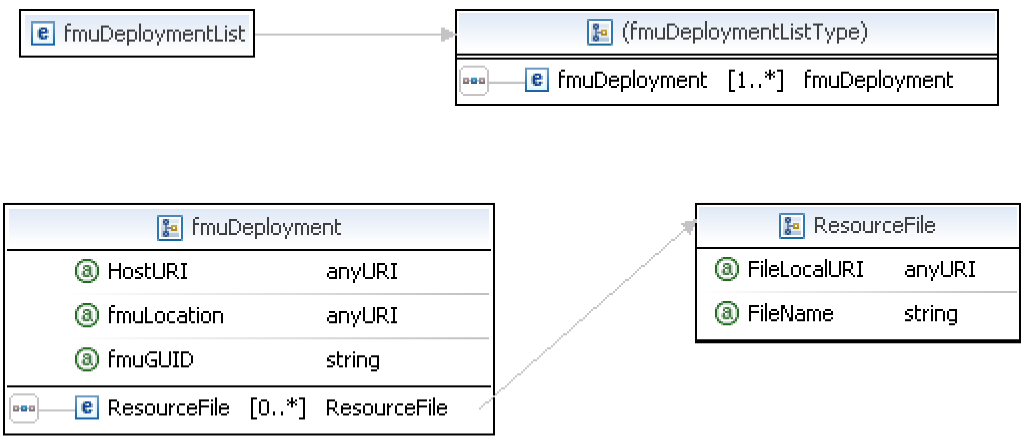
Each **fmuDeployment** is composed of

- **HostURI** (URI, mandatory) to define the **FMU targeted location**
  - ex : *Domaine\HostName*
  - Corresponds to *MachineName* of *Host* (see previous chapter)
- **fmuLocation** (URI, mandatory) to define the position of the FMU file in the Host context. fmuLocation is referenced in Co simulation specifications V1 § *3.2. Creation and Destruction of Co-Simulation Slaves*
  - ex : *\Temp\fmufiles\Myfile.fmu*
  - Corresponds to *BaseDirectory of Host* (see previous chapter)
- **fmuGUID** (string, mandatory) corresponding to the unique identification define in the FMU (see FMI Model Exchange specifications for details)
  - This is defined in the FMU model exchange XML description.

Each **fmuDeployment** will also need **resources files**.

The list of associated filed are described by:

- **FileName** (string) as file's name
  - ex : *"Script.sh"*
- **FileLocalURI** (URI) define the file location on the local Host ,
  - ex : */temp/Script.sh*



FMU deployment format

4.3.2. Description format

Both the FMU and its resources files are identified and then deployed according to this format.

Also the PLM should provide the **deployment description to the co-simulation engine**.

This description is defined in *FmiPLMDeployment.xsd* schema.

See example in Appendix.

4.4. Mapping of scenario to PLM Services

Table – Mapping of scenario to PLM services

FMI required capability	Scenario §	FMI for PLM mandatory format	Functional capability needed for PLM engine to implement FMI Methodology	Comment
Search and navigate	§3.23.2 §3.33.3 §3.4.13.4.1 §3.4.33.4.3 §3.53.500 §3.73.7	NO	• Search object in PLM • Navigate on PLM links	
Create and Edit PLM objects	§3.1.23.1.2 §3.23.2 §3.33.3 §3.4.13.4.1 §3.4.33.4.3 §3.53.5 §3.73.7	YES	• Create new executable unit • Create new Result object • Create new co-simulation configuration • Edit Object's attribute	
Check-out	§3.23.2 §3.4.13.4.1 §3.53.5	YES	• PLM search • Check out • PUT on network location	Files are extract to new directory
Launch	§3.23.2 §3.4.23.4.2 §3.53.5	NO	• Remote application start • Wait end of Process activity	May be limited to launch application Graphic user interface
Check-in	§3.1.23.1.2 §3.23.2 §3.4.33.4.3 §3.53.5	YES	• Check in file • Check in directory • Get from network location • Import metadata	
Simulation Control	§3.4.23.4.2	NO	• This is out of PLM scope	No PLM Action requested during simulation Main process can be launch by PLM

5. End user benefits from implementing the FMI PLM interface

Criteria	Benefits
Interface definition	• Interface definition is not required • Respect a reduced list of rules (network location, working directory) • No risk of increasing content and thus augmented complexity
Genericity	• Approach is generic, covers all cases • Simple solution is simple • Not constrained by the behavior and/or technology of the WP5 selected tools, and even other possible tools
Development cost	• Little of no software development effort required from tool vendors side (modelling tool, simulation tool, post process …) • Implementation is done inside the PLM Solution • No Connector to develop from PLM side (server) • No Layer to develop from each authoring tool (client) side • Easy to migrate from simulation without PLM to simulation with PLM
Openess	• Open and generic • No strong pre requ on tools
Evolutivity	• A tool can be added without additionnal development

6. Glossary

This glossary is a subset of (*MODELISAR Glossary, 2009*) with some extensions specific to this document.

Term	Description
AUTOSAR	AUTomotive Open System Architecture (www.autosar.org). Evolving standard of the automotive industry to define the implementation of embedded systems in vehicles including communication mechanisms. An important part is the standardization of C-functions and macros to communicate between software components. AUTOSAR is targeted to build on top of the real-time operating system OSEK (www.osek-vdx.org, de.wikipedia.org/wiki/OSEK). The use of the AUTOSAR standard requires AUTOSAR membership.
co-simulation	Couple several simulation programs including their numerical solvers in order to simulate a system consisting of several subsystems.
Co-simulation Configuration Data	A Simulation configuration represent a group of FMUs, simulations, backplanes, scenario data, simulation results , Parameter Sets … The configuration has lifecycle: could be versioned, updated, duplicate…
Co-simulation engine	The Co simulation engine is the master scheduler for simulations tools, it's role is to initialize and control other simulation tools during simulation
ECU	Electronic Control Unit (Microprocessor that is used to control a sub-system in a vehicle)
event	The time instant at which the integration is halted and variables may change their values discontinuously. Between event instants, all variables are continuous.
FMI	Functional Mock-up Interface: Interface of a functional mock-up in form of a model. In analogy to the term digital mock-up (see *mock-up*), functional mock-up describes a computer-based representation of the functional behaviour of a system for all kinds of analyses.
FMU	Functional Mock-up Unit: A "model class" from which one or more "model instances" can be build for simulation. A FMU is stored in one zip-file as defined in section. Consisting basically of one xml file that defines the model variables and a set of C-functions, in source or binary form, to execute the model equations.
mock-up	A full-sized structural, but not necessarily functional model built accurately to scale, used chiefly for study, testing, or display. In the context of computer aided design (CAD), a digital mock-up (DMU) means a computer-based representation of the product geometry with its parts, usually in 3-D, for all kinds of geometrical and mechanical analyses.
model	A model is a mathematical or logical representation of a system of entities, phenomena, or processes. Basically a model is a simplified abstract view of the complex reality. It can be used to compute its expected behaviour under specified conditions. In this document, "models" are described by differential, algebraic and discrete equations and are mainly used to represent physical systems and controllers.
Model Description Schema	An *XML* schema that defines how all relevant, non-executable, information about a "model class" (*FMU)* is stored in a text file in *XML* format. Most important, data for every variable is defined (variable name, handle, data type, variability, unit, etc.).

Term	Description
Model Interface	A set of C-interface definitions to access the equations of a dynamic system from an external program, e.g., to compute the state derivatives of a model, see section
parameter	A quantity within a *model*, which remains constant during simulation, but may be changed before a simulation is started. Examples: mass, stiffness, resistance, etc.
XML	eXtensible Markup Language (www.w3.org/XML, en.wikipedia.org/wiki/Xml) – An open standard to store information on text files in a structured form.
PLM	PLM (Product Lifecycle Management) is a business strategy, together with a set of methods, authoring and collaboration tools and platforms that helps companies share product data, apply common processes, and leverage corporate knowledge for the development of products from conception to retirement.
Initialization set	Initial values for state variables, Inputs and outputs that are necessary to the FMU
Observer	list of Models' variables or states follow stored in PLM at the end of Simulation
Data	Stored Values of observed variables or state during simulation
Log	Execution trace created during Co-simulation (eg. Process trace for Debug)
Executable unit	Executable unit is a file set involve in simulation or co-simulation, to represent system or environment behavior. Example: a FMU file.

7. Bibliography

MODELISAR, **Functional Mock-up Interface for Model Exchange** Version 1.0 January 26, 2010

MODELISAR**, Functional Mock-up Interface for Co-Simulation** Version 1.0 October 12, 2010

MODELISAR**, FMI Job Control Interface** Version 0.4, November 19th, 2009

MODELISAR, Glossary (2009): **MODELISAR WP2 Glossary and Abbreviations**. Version 1.0, June 9, 2009.

8. Appendix A : Contributors

A.1 Version 1.0

The Functional Mock-up Interface FMI for PLM subproject was initiated and organized by Dassault Systèmes. This work is part of WP200 of the MODELISAR ITEA2 project, organized by the WP200 work package leader Dietmar Neumerkel (Daimler).

We thank the following persons who have contributed to the elaboration of theses specifications, either writing the documents, participating to meetings and discussions, or submitting proposals, requirements or feedbacks:

M Andersson (Volvo)

R Andersson (Volvo)

M Arnold (Uni Halle)

C Bausch (Atego systems)

F Bichet (Dassault Systèmes)

T Blochwitz (ITI)

L Bondarenko (Dassault Systèmes)

P Chombart (Dassault Systèmes)

C Clauss (Fraunhofer IIS EAS)

B Engelman (Dassault Systèmes)

B Garat (Dassault Systèmes)

C Grepet (Trialog)

F Merceron (Dassault Systèmes)

J Mezger (TWT)

M R Monteiro (Atego systems)

T Neidhold (ITI)

D Neumerkel (Daimler)

O Oelsner (ITI)

M Otter (DLR)

JV Peetz (Fraunhofer SCAI)

X Remond (Dassault Systèmes)

B Rousselin (Dassault Systèmes)

M Sall (Trialog)

T Schierz (Uni Halle)

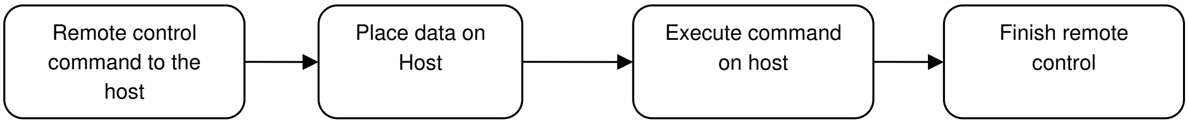
K Wolf (Fraunhofer SCAI)

9. Appendix B - Remote Control by PLM (optional)

This is a NEW chapter

As an option, a simulation could be launched without the co simulation engine. This may be useful when simulation is executed in a batch mode, or for repetitive simulation with user interaction..

In that particular case, PLM can provide Services to remotely launch the applications.



Note: The Remote control is limited to launching command; the PLM is not supposed to follow the activities. If necessary the simulation process must then be managed by the Co-Simulation engine (monitoring of the simulation applications, re launch if need to restart simulation…).

Reminder: as defined in the Use Cases, the PLM system is not expected to fully handle the simulation phase.

The Remote access is based on:

- Host' remote control command: to create a communication channel between the PLM tool and the target.
- Selected Application for execution path and working directory: the execution path will be executed in the working directory context.

For more flexible use of application on targets, Applications can be parameterized from Application's execution path with a list of arguments set by PLM according to deployed files and requested activities:

$$\underbrace{\textit{C:\textbackslash My\_Directory\textbackslash My\_Programme.exe}}_{\text{Application's execution Path}} \quad \underbrace{\textit{argument\#1} \quad \textit{argument\#2}}_{\text{Argument list}}$$

To provide Remote control command, the PLM tools have to manage a list of arguments associated to deployment step; model name of deployed files names should be present in execution path.

10. Appendix C - Deployment format

This is a NEW chapter

10.1. Schema specification


```
<?xml version="1.0" encoding="UTF-8" ?>
- <!--
edited with Eclipse by Benoit ROUSSELIN (DASSAULT SYSTEMES)
-->
- <xsd:schema elementFormDefault="qualified"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
- <xsd:annotation>
  <xsd:documentation>Copyright(c) 2008-2009, MODELISAR consortium. All rights
   reserved. This file is licensed by the MODELISAR Consortium members to third parties
   under the BSD License (http://www.opensource.org/licenses/bsd-license.html): ----
   ---------------------------------------------------------------------- Redistribution and use
   in source and binary forms, with or without modification, are permitted provided that
   the following conditions are met: - Redistributions of source code must retain the
   above copyright notice, this list of conditions and the following disclaimer. -
   Redistributions in binary form must reproduce the above copyright notice, this list of
   conditions and the following disclaimer in the documentation and/or other materials
   provided with the distribution. - Neither the name of the copyright holders nor the
   names of its contributors may be used to endorse or promote products derived from
   this software without specific prior written permission. THIS SOFTWARE IS
   PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY
   EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
   IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
   PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR
   CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
   EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
   PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
   PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
   LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
   NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
   SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. -----------------
   -----------------------------------------------------</xsd:documentation>
   </xsd:annotation>
- <xsd:element name="fmuDeploymentList">
- <xsd:complexType>
- <xsd:sequence>
  <xsd:element name="fmuDeployment" minOccurs="1" maxOccurs="unbounded"
   type="fmuDeployment" />
   </xsd:sequence>
   </xsd:complexType>
   </xsd:element>
```

```
- <xsd:complexType name="fmuDeployment">
- <xsd:sequence>
  <xsd:element minOccurs="0" maxOccurs="unbounded" name="ResourceFile"
    type="ResourceFile" />
    </xsd:sequence>
  <xsd:attribute name="HostURI" type="xsd:anyURI" use="required" />
  <xsd:attribute name="fmuLocation" type="xsd:anyURI" use="required" />
  <xsd:attribute name="fmuGUID" type="xsd:string" use="required" />
    </xsd:complexType>
- <xsd:complexType name="ResourceFile">
  <xsd:attribute name="FileLocalURI" type="xsd:anyURI" />
  <xsd:attribute name="FileName" type="xsd:string" />
    </xsd:complexType>
    </xsd:schema>
```

10.2. Deployment example

```
<?xml version="1.0" encoding="UTF-8"?>
<fmuDeploymentList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="fmiPlmDeployment.xsd">

  <fmuDeployment HostURI="MyComputer" fmuGUID="A35F89DD56"
    fmuLocation="\Temp\FMU\myFMU.fmu">
      <ResourceFile FileLocalURI="\Temp\FMU\data.txt" FileName="data.txt"/>
      <ResourceFile FileLocalURI="\Temp\FMU\data2.txt" FileName="data2.txt"/>
  </fmuDeployment>

 <fmuDeployment HostURI="MyComputer2" fmuGUID="A35F897A6"
   fmuLocation="\Temp\myFMU2.fmu">
      <ResourceFile FileLocalURI="\FMU\inputFile.txt" FileName="inputFile.txt"/>
      <ResourceFile FileLocalURI="\FMU\data2.txt" FileName="data2.txt"/>
  </fmuDeployment>

</fmuDeploymentList>
```