



Optimization of scalaBle rEaltime modeLs and functiOnal testing for e-drive COnceptS

EUROPEAN COMMISSION

Horizon 2020

GV-07-2017

GA # 769506

Deliverable No.	OBELICS D3.1	
Deliverable Title	Standardized model integration	
Deliverable Date	2017-12-31	
Deliverable Type	REPORT	
Dissemination level	Public (PU)	
Written By	Matthieu Ponchant (SIE-SAS) Alberto Barella (CRF) Georg Stettinger (VIF) Hellal Benzaoui	2017-10-26 2017-12-07 2017-12-12 2017-12-20
Reviewed by	Horst Pfluegl (AVL) Noshin Omar (VUB) Mathieu Sarrazin (SIE-NV) Anish Patil (UNR)	2017-12-18 2017-12-18 2017-12-22 2017-12-27
Approved by	Horst Pfluegl (AVL) – Project Coordinator	2017-12-27
Status	Final	2017-12-22



Change log:

No	Who	Description	Date
1	Matthieu Ponchant	D3.1 report creation	2017-10-26
1.1	Matthieu Ponchant	Content refinement	2017-11-27
1.2	Alberto Barella	High performance computing	2017-12-07
1.3	Georg Stettinger	Model.CONNECT, FMI/FMU description, co-simulation heterogeneous integration	2017-12-12
1.4	Matthieu Ponchant	Final update with comments from all WP3 partners	2017-12-20
1.5	Hellal Benzaoui	GSP tool	2017-12-20
1.6	Matthieu Ponchant	Add conclusion	2017-12-22
1.7	Hellal Benzaoui	Add Functional Mock-up For Matlab & Simulink description	2017-12-22



Contents

Contents	3
Figures	4
Tables.....	5
Publishable Executive Summary	7
1 Introduction.....	8
2 Simulation tools interfaces.....	9
2.1 Software platform.....	9
2.1.1 LMS Amesim	9
2.1.2 Model.CONNECT™	12
2.1.3 Volvo Global Simulation Platform (GSP)	13
2.2 FMI/FMU	15
2.2.1 Introduction	16
2.2.2 FMI for Model Exchange.....	17
2.2.3 FMI for Co-Simulation.....	17
2.2.4 FMI properties	18
2.2.5 FMU properties.....	18
2.2.6 FMI use cases.....	18
2.2.7 Comparison of FMI 1.0 and FMI 2.0.....	19
2.2.8 FMU example.....	20
2.3 RT capability.....	21
2.3.1 Description.....	21
2.3.2 RT model examples.....	22
3 Port typing conventions and model identity.....	26
3.1 Model identity.....	26
3.2 Components.....	26
3.2.1 E-motor	27
3.2.2 Inverter	28
3.2.3 Battery	29
3.2.4 Converter DC/DC.....	30
3.2.5 Body builder.....	31
3.2.6 OnBC	31
3.2.7 Powertrain	31
3.2.8 Braking system.....	32
3.2.9 Cooling system.....	32
3.2.10 Heating, Ventilation Air Conditioning (HVAC)	33
3.2.11 Energy Management control	34



3.2.12	Braking blending control	34
3.2.13	Driver	35
3.2.14	Powertrain control.....	35
3.2.15	HVAC control	35
3.2.16	Predictor supervisor	36
4	Computation approaches and heterogeneous couplings stability.....	37
4.1	Model reduction strategies & Model scalability.....	37
4.2	Co-simulation Heterogeneous integration	38
4.2.1	Scheduling.....	39
4.2.2	Coupling step size	39
4.2.3	Input signal extrapolation	40
4.2.4	Coupling Algorithms.....	40
4.3	HPC.....	40
4.3.1	Parallel computing	41
4.3.2	Constraints: simulation strategy and operating system	41
5	Conclusions.....	44
6	Abbreviations and definition.....	45
7	Risk Register	46
7.1	Risk register.....	46
8	Bibliography.....	47
9	Acknowledgement.....	48

Figures

Figure 2-1:	LMS Imagine.Lab Amesim simulation platform.....	9
Figure 2-2:	Example of Electric Vehicle thermal management model	10
Figure 2-3:	power flow chart of Electric Vehicle model	11
Figure 2-4:	Summary of the interfaces and processes supported by LMS Amesim	12
Figure 2-5:	Co-Simulation example.....	12
Figure 2-6:	Distributed Co-Simulation.	13
Figure 2-7:	VOLVO Global Simulation Platform content	14
Figure 2-8:	Architecture for vehicle subsystem modeling.....	14
Figure 2-9:	Integration of different domain specific subsystems from different suppliers (Blochwitz & Otter, The Functional Mockup Interface for Tool independent Exchange of Simulation Models, 2011).....	16
Figure 2-10:	Model integration using FMI (Blochwitz & Otter, The Functional Mockup Interface for Tool independent Exchange of Simulation Models, 2011).....	16



Figure 2-11: FMI for Model Exchange (Blochwitz & Otter, The Functional Mockup Interface for Tool independent Exchange of Simulation Models, 2011).	17
Figure 2-12: FMI for Co-Simulation (Blochwitz & Otter, The Functional Mockup Interface for Tool independent Exchange of Simulation Models, 2011).	17
Figure 2-13: Data flow between the environment and an FMU (The Functional Mock-up Interface Standard).	18
Figure 2-14: Standalone configuration (Blochwitz & Otter, The Functional Mockup Interface for Tool independent Exchange of Simulation Models, 2011).	18
Figure 2-15: Tool-based configuration (Blochwitz & Otter, The Functional Mockup Interface for Tool independent Exchange of Simulation Models, 2011).	19
Figure 2-16: Distributed configuration (Blochwitz & Otter, The Functional Mockup Interface for Tool independent Exchange of Simulation Models, 2011).	19
Figure 2-17: FMI example in Model.CONNECT™ (Model.CONNECT™ User Manual).	21
Figure 2-18 temporal stability in Real time hardware platform.....	21
Figure 2-19 documentation directory	22
Figure 2-20 Simulink driver control model	22
Figure 2-21 LMS Amesim powertrain model.....	22
Figure 2-22 LMS Amesim engine model	23
Figure 2-23 Dashboard interface	23
Figure 2-24 Engine interface.....	24
Figure 2-25: Real-Time Co-Simulation example.	24
Figure 3-1: Subsystem identity card.	26
Figure 3-2: Generic electric architecture	27
Figure 3-3: E-motor.....	27
Figure 3-4: inverter	28
Figure 3-5: Nissan leaf battery module	29
Figure 3-6: buck converter topology	30
Figure 3-7: boost converter topology	30
Figure 4-1 level of detail of simulation	37
Figure 4-2 model reduction process used during IMROVE project (Thielboger, 2017).....	38
Figure 4-3: Different possibilities of simulator scheduling for a coupled co-simulation (Benedikt, Zehetner, Watzenig, & Bernasch, 2011)	39
Figure 4-4: Exchange of subsystem data at coupling time instants and definition of time steps.	40
Figure 6-1: OBELICS model based development concept to reduce development and testing efforts.	44

Tables

Table 2-1: Different initialization and semantics of event handling in FMI 2.0 (Blochwitz, New Features of FMI 2.0 and beyond, 2014).....	19
Table 2-2: Overview of variable classification possibilities (Blochwitz, New Features of FMI 2.0 and beyond, 2014).....	20
Table 3-1 Inputs/outputs of Emotor	28
Table 3-2 Inputs/outputs of inverter	29
Table 3-3 Inputs/outputs of battery	29
Table 3-4 Inputs/outputs of converter DC/DC	30
Table 3-5 Inputs/outputs of body builder	31
Table 3-6 Inputs/outputs of OnBC.....	31
Table 3-7 Inputs/outputs of Powertrain.....	31
Table 3-8 Inputs/outputs of braking system	32
Table 3-9 Inputs/outputs of cooling system.....	32



Table 3-10 Inputs/outputs of HVAC.....	33
Table 3-11 Inputs/outputs of energy management control.....	34
Table 3-12 Inputs/outputs of braking blending control	34
Table 3-13 Inputs/outputs of thermal control.....	35
Table 3-14 Inputs/outputs of Powertrain control	35
Table 3-15 Inputs/outputs of HVAC control	35
Table 3-16 Inputs/outputs of predictor supervisor	36



Publishable Executive Summary

This deliverable (D3.1) gives the description of generic EV components model connectivity, simulation software tools and interfaces, computation and coupling strategies.

Providing new industrial tools and methods enabling the support at industrial level of new fully integrated EV architectures (electric, electronic, thermal, chassis) and designs, OEMs and tier 1 suppliers will be able to push beyond investigation of another generation of even efficient and affordable electric vehicles. This will enable new co-engineered optimizations of multiple combined components and controls to achieve higher overall vehicle performance, for conventional and automated operations

After reviewing available simulation tools interfaces and especially their capabilities to communicate between them, the FMI 2.0 have been selected to ensure good communication between models. Real-time capability has also been highlighted because some models must contain scalable components running on real-time platform for control calibration and validation.



1 Introduction

Electric Vehicle (EV) powertrain design are driven by vehicle power and range goals satisfying regulatory and real operations, while limiting components costs and insuring competitive production. It investigates key powertrain component sizing for a dedicated vehicle configuration (usually single motor FWD or dual motor AWD) but addresses chassis integration and auxiliaries in a later stage. These operations are supported by processes legacy of 100 years of engine-driven automotive design, leading to significant performances and acceptances issues when applied to electric vehicles. First, it faces limitation in addressing, at industrial level, innovative powertrain designs involving more sizing dimensions and operation degrees of freedoms, for a full line-up of different vehicles configurations. Secondary, late consideration of strongly coupled systems leads to lower vehicle performance and comfort. Best examples are: a HVAC (impacting vehicle range over 50%) and chassis integration (brake blending, damping). To set up new processes adapted to EV design, the automotive industry needs to be supported by a new generation of industrial modelling and simulation tools allowing the studies of innovative configuration combined with all the relevant systems impacting performances and comfort.

Providing new industrial tools and methods enabling the support at industrial level of new fully integrated EV architectures (electric, electronic, thermal, chassis) and designs, OEMs and tier 1 suppliers will be able to push beyond investigation of another generation of even efficient and affordable electric vehicles. This will enable new co-engineered optimizations of multiple combined components and controls to achieve higher overall vehicle performance, for conventional and automated operations

2 Simulation tools interfaces

2.1 Software platform

2.1.1 LMS Amesim

2.1.1.1 General description

LMS Amesim offers engineers an integrated simulation platform to accurately predict the multi-disciplinary performance of intelligent systems. **LMS Amesim** enables engineers to model, simulate and analyze multi-physics, controlled systems, and offers capabilities to connect to controls design, helping to assess and validate control strategies.

LMS Amesim comes with a set of standard and optional libraries of predefined and validated components from different physical domains: fluids, thermodynamics, electrics, electro-mechanical, mechanics and signal processing—as well as application libraries—cooling system, air-conditioning, internal combustion engine, aerospace, etc. Components in the libraries are based on the analytical representation of physical phenomena (see Figure 2-1).

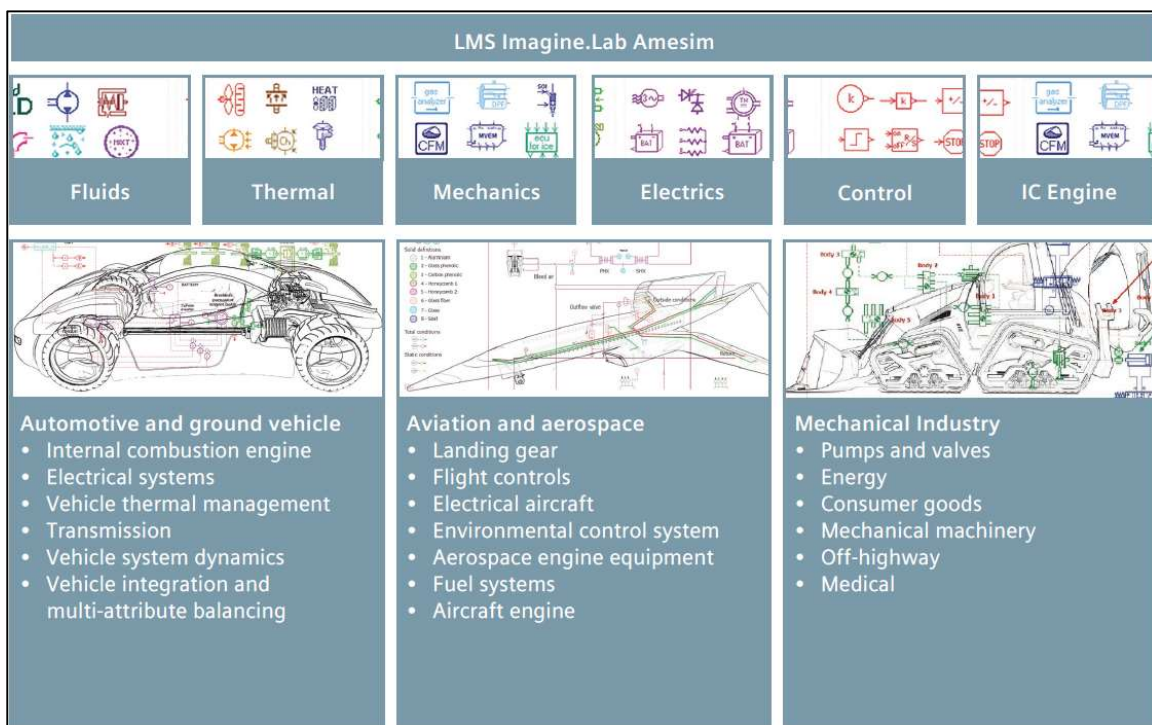


Figure 2-1: LMS Imagine.Lab Amesim simulation platform

For automotive application, a whole range of component is available for the modeling of all major vehicle subsystems as well as their integration: internal combustion engines, transmissions, vehicle thermal management systems, vehicle system dynamics, fluid systems related to engines as well as electrical systems. It enables to simulate the system performance across the complete mechatronics design cycle process, in order to optimally balance conflicting key performance attributes (fuel economy / range, drivability, thermal passenger comfort, etc.). An example of an electrical vehicle model aimed at studying the impact of electric component temperature during a driving cycle is shown in Figure 2-2

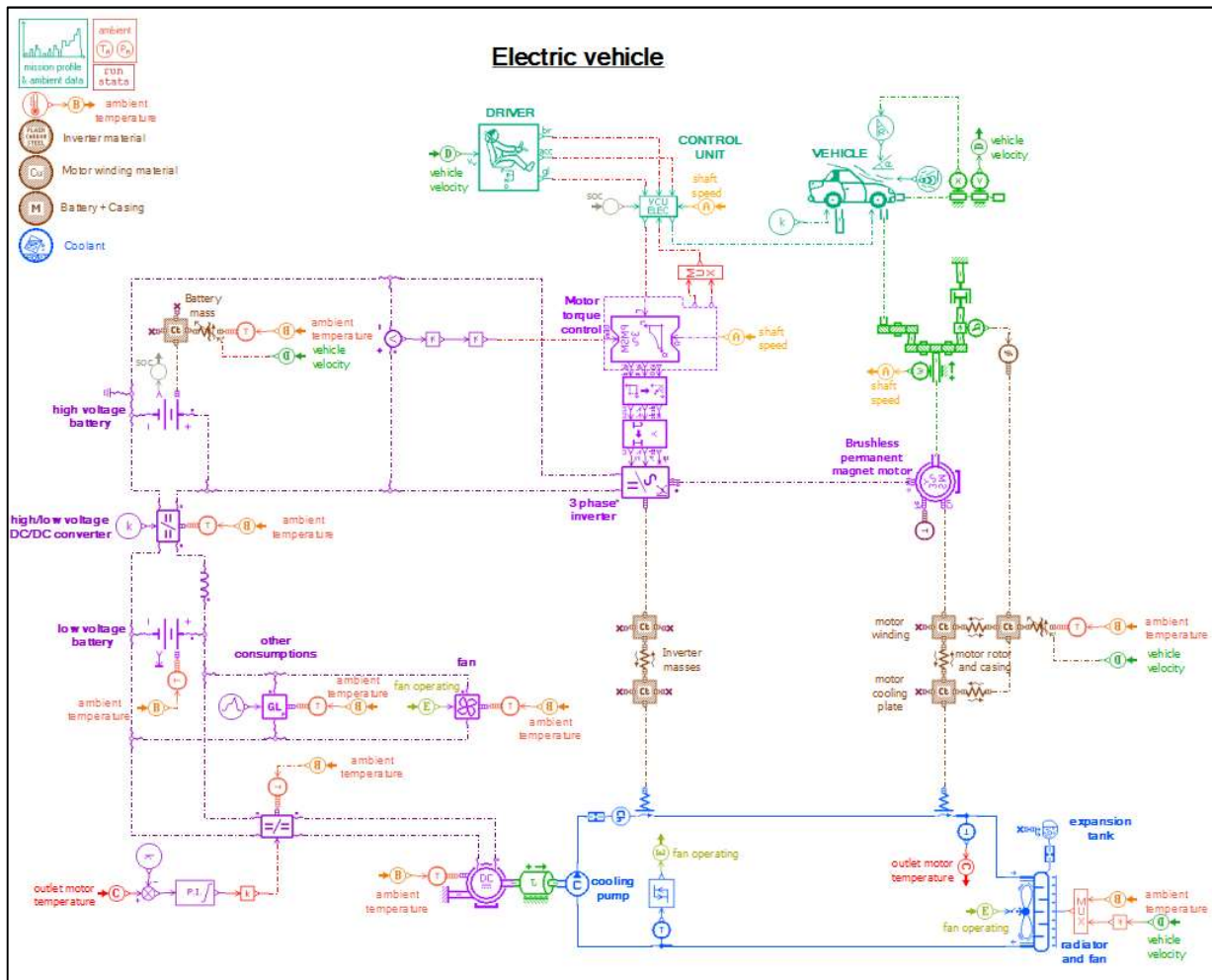


Figure 2-2: Example of Electric Vehicle thermal management model

Performance attribute can be highlighted with customized post processing tool. In this example powertrain efficiency and power flow can be directly analyzed as shown in Figure 2-3 along driving scenarios. All heat losses are transferred to components thermal masses, then to cooling circuit. We can also notice that the battery state of charge estimation is also displayed.

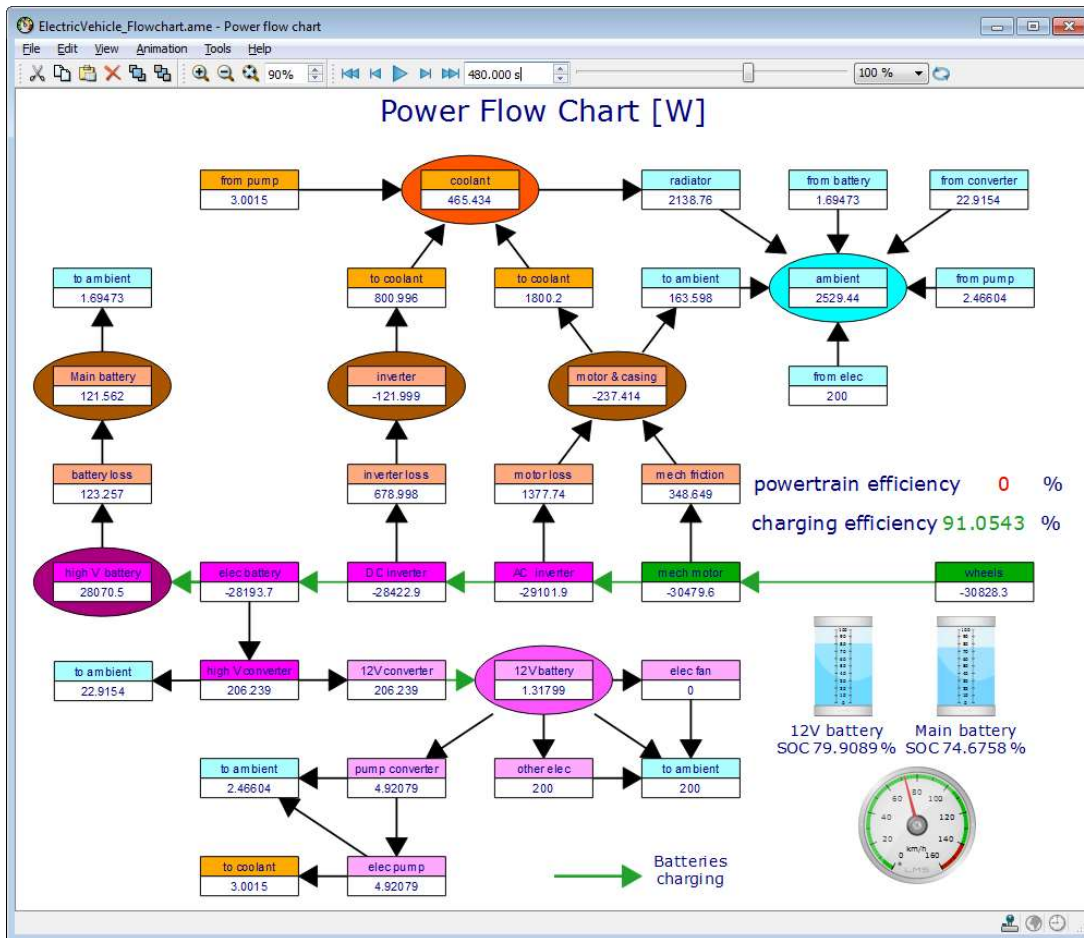


Figure 2-3: power flow chart of Electric Vehicle model

Co-simulation with any software is made possible thanks to the **LMS Amesim** generic co-simulation capability, as well as dedicated interfaces and support of Functional Mock-up Interface (FMI).

2.1.1.2 Functional Mock-Up Interface

Since rev13, LMS Imagine.Lab Amesim is able to export (create) and import Functional Mock-up Units for both Model Exchange and Co-simulation. The current release (rev15) has the following capabilities:

- export FMU for Model Exchange in version 1.0 and 2.0
- import FMU for Model Exchange in version 1.0 and 2.0
- export FMU for Co-simulation in version 1.0 and 2.0
- import (as a master) FMU for Co-simulation in version 1.0 and 2.0

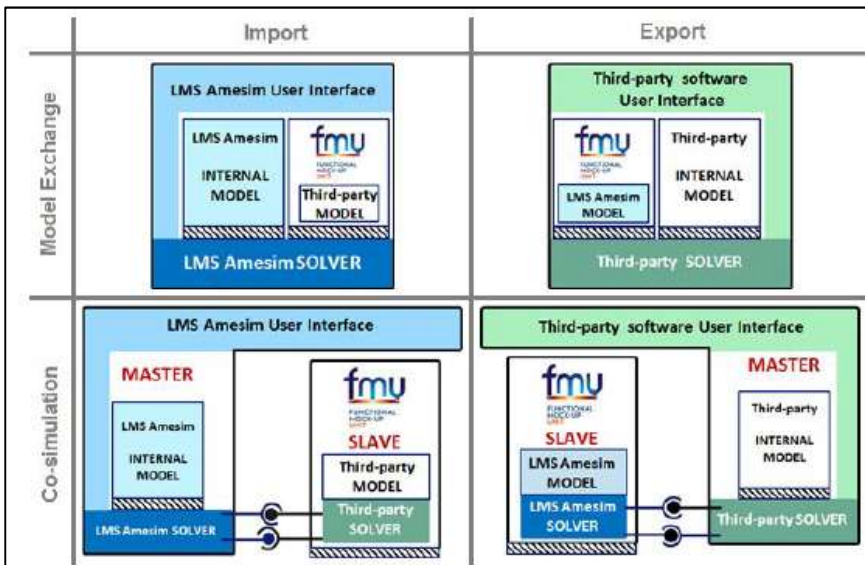


Figure 2-4: Summary of the interfaces and processes supported by LMS Amesim

2.1.2 Model.CONNECT™

2.1.2.1 Co-Simulation

Model.CONNECT™ is a platform to set up and execute entire mechatronic system simulations, which are composed of subsystem and component models from multiple model authoring environments. Exemplarily, a co-simulation setup of a hybrid electric vehicle is shown in Figure 2-5. Models can be integrated based on standardized interfaces (e.g. Functional Mockup Interface, FMI) as well as specific interfaces to a wide range of well-known simulation tools especially for the automotive industry (Model.CONNECT™ User Manual).

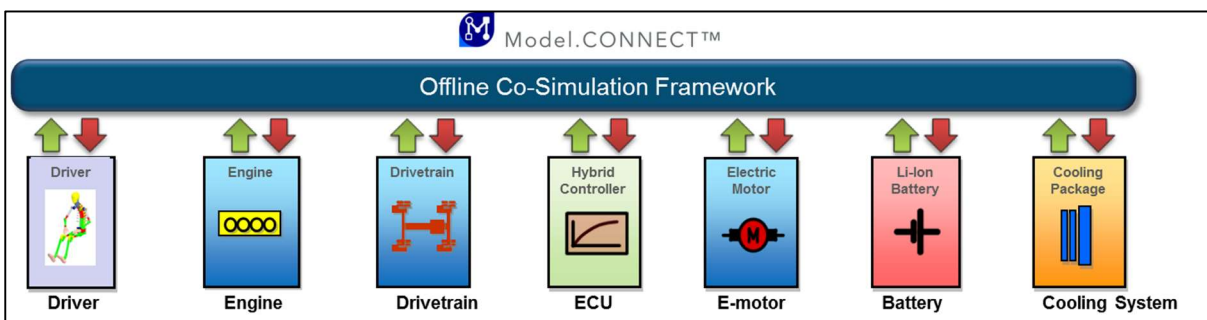


Figure 2-5: Co-Simulation example

Model.CONNECT™ supports the user in organizing system model variants. These variants may describe different configurations of the system under investigation as well as different testing scenarios and testing environments.

The model execution is supported in two flavors, which can also be combined:

- Model integration based on models that are provided as executable libraries (FMI for Co-Simulation or Model Exchange, as well as compiled MATLAB/Simulink models). Such model configurations can be executed in one process.



Currently the following *model interfaces* are supported: AVL fmi.LAB, AVL BOOST™, AVL FIRE™, AVL CRUISE™, AVL CRUISE™ M, FMU, AVL VSM™, Vires VTD, IPG CarMaker.

- Tool-coupling based on the ICOS technology, which is a distributed co-simulation platform with a wide variety of supported simulation tools, industry-leading co-simulation algorithms and the possibility to connect real-time systems to the co-simulation. All tool-interface (ICOS) elements are running as individual processes, which interact by using inter-process communication.

Currently, the following *ICOS tool interfaces* are supported: ICOS Custom, ICOS Real Time, ICOS CAN, MSC Adams, LMS Amesim, IPG CarMaker4SL, Mechanical Simulation Corp. CarSim, Dassault Systems Dymola, Mentor Graphics Flowmaster, Gamma Technologies GT-SUITE, Java, Magna KULI, National Instruments LabVIEW, Mathworks MATLAB/Simulink, Microsoft Excel, Modelica Association Open Modelica, Synopsys SaberRD, Dassault Systems SIMPACK, ESI SimulationX.

Model.CONNECT™ supports local and distributed co-simulation i.e. this allows the connection of different operating systems on distributed resources, see Figure 2-6. To perform such a distributed co-simulation a running remote server on each host is required to connect different simulation tools on different computers. Furthermore, all involved host computers can have their individual operating system (Model.CONNECT™ User Manual).

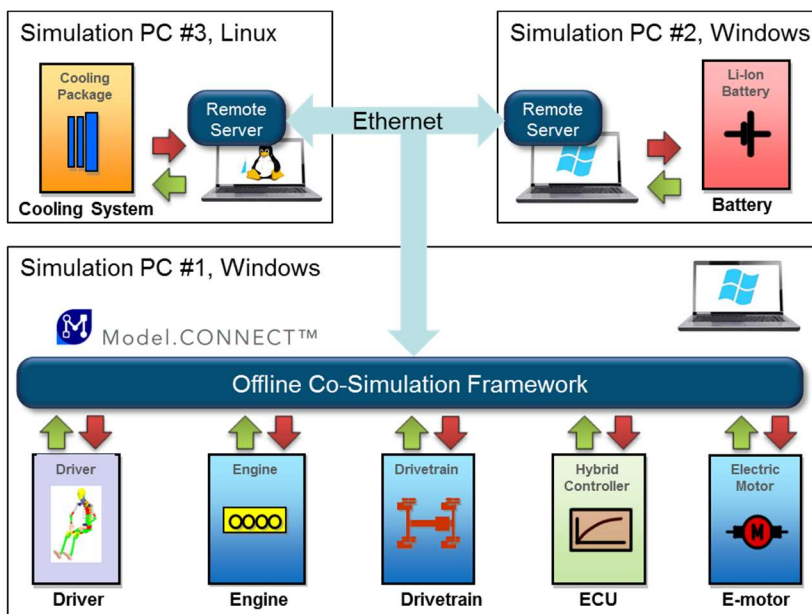


Figure 2-6: Distributed Co-Simulation.

2.1.3 Volvo Global Simulation Platform (GSP)

2.1.3.1 General description

Volvo Global Simulation Platform is the Volvo Group common repository of vehicle models, component simulation model, utilities and tools (see Figure 2-7). The common database (GSPDB) has a high standard to ensure traceability and quality assurance. GSP uses common guidelines and unified model structure in order to facilitate sharing and reuse of model components and data. It is based on MATLAB/SIMULINK which gives transparency and efficient integration with other systems and processes.



Global Simulation Platform (GSP)

GSPDB

Database, Matlab/Simulink

TOOLS

Stand Alone with User Interface

Figure 2-7: VOLVO Global Simulation Platform content

Vehicle system models developed in this simulation platform are designed with respect to the vehicle modular architecture standard (VMA). Moreover, modeling of vehicle subsystems (engine, transmission ...) is also performed with respect to a generic structure, similar to the one described in the Figure 2-8.

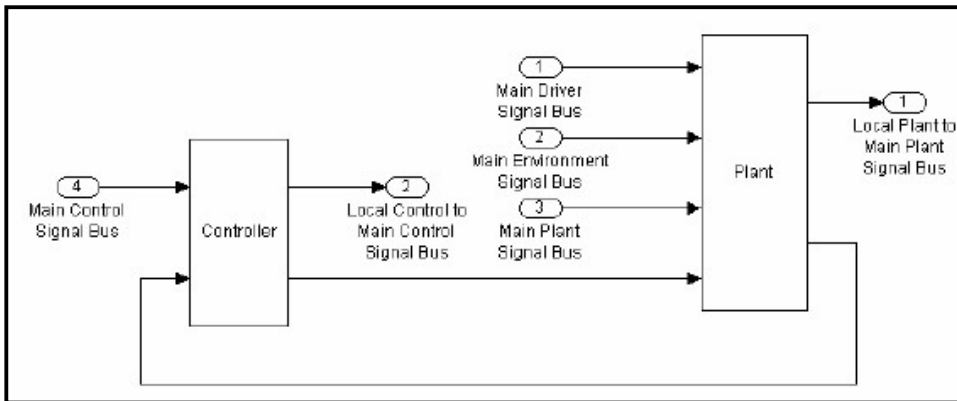


Figure 2-8: Architecture for vehicle subsystem modeling

With GSP, the Volvo Group has one common interface for analysis and evaluation of the product attributes; fuel consumption and performance of vehicles, emissions, etc. and also common standards and guidelines to allow share and reuse of models and tools. The Global Simulation Platform is divided into following main parts:

- GSP database
- Tools (standalone) with user interface

2.1.3.2 GSP Database

This GSP database core is able to manage all files need to the GSP and consists of different libraries of vehicle and system models, data files, simulation interface and tools.

Model library: this library contains vehicle sub-system models and complete vehicle system model organized in three parts:

- Road and environment models
- Driver models
- Vehicle system model: This one consists of the integration of the following key sub-systems:
 - Engine
 - Clutch
 - Transmission



- Mechanical auxiliaries
- Axle
- Wheel
- Chassis
- Battery (LV)
- Energy Storage system (HV battery, ...)
- Motor drive system
- Cooling system
- Etc. ...

Simulation interfaces: these interfaces enable following applications:

- Vehicle application definition and system configuration
- Road and environment specification
- Simulation execution

2.1.3.3 TOOLS

The GST tool (Global Simulation Tool) is one example of standalone tool that is developed from GSPDB. This tool, dedicated to complete vehicle simulation for feature analysis (fuel economy, performances, etc.) is available to everybody inside Volvo, easy to use regarding its Human-Machine Interface (HMI).

2.1.3.4 Functional Mock-up Interface

From release 2017b, Mathworks' solutions support directly the import of simulation models compiled with the FMI standard. The FMU block from Simulink automatically selects the FMU mode based on the existing FMU you want to import:

- Co-simulation to integrate FMUs that implement an FMI Co-Simulation Interface. These FMUs may contain local solvers be used for tool coupling
- Model exchange to integrate FMUs that implement an FMI Model Exchange Interface. These FMUs do not contain local solver. Instead, these FMUs inherit solvers from Simulink.

This FMU block supports FMI versions 1.0 and 2.0. For FMI version 2.0, if an FMU contains both Co-Simulation and Model Exchange elements, the block detects this and prompts the user to select the mode to operate in. An FMU block in Simulink can be used as other Simulink blocks. These blocks support Normal and Accelerator modes.

For previous Matlab® releases, a specific toolbox is necessary to import FMU components in Simulink® environment or to build FMU component from Simulink® models. The FMI toolbox for MATLAB® and SIMULINK® can manage FMU components within Simulink environment. This product is distributed by the Modelon Company and this release supports the FMI standard specification 1.0 and 2.0. The FMI library once installed in the Simulink environment is made of two blocks that manage the import of FMU models for model exchange and co-simulation. The FMI toolbox for MATLAB® and SIMULINK® supports HIL simulations on DSPACE DS 1006 systems.

2.2 FMI/FMU

2.2.1 Introduction

The interface of a third party simulation tool is typically a very specific implementation. For that reason the so-called *Functional Mock-up Interface (FMI)* was designed within the Modelisar project¹. This FMI represents a tool independent standard, which supports data exchange between different simulation tools. By using a combination of xml-files and compiled C-code a *black-box* exchange is possible without sharing knowledge of the system/model implementation (The Functional Mock-up Interface Standard).

A complex mechatronic system like a vehicle is often separated in the several subsystems. Typically, these subsystems are modelled in different domain specific simulation tools by different suppliers e.g.: LMS Amesim for the vehicle dynamics and Dymola for the cooling system. The big challenge of the OEM is to integrate many different tools to simulate the entire vehicle, see Figure 2-9.

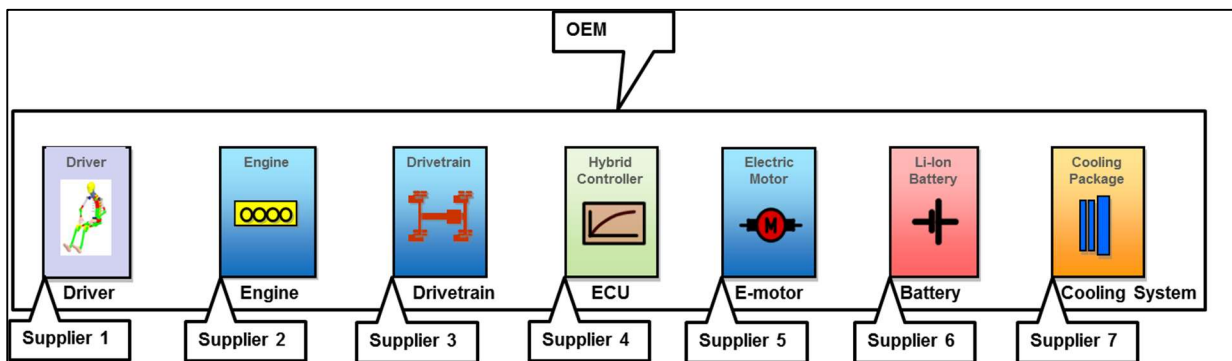


Figure 2-9: Integration of different domain specific subsystems from different suppliers (Blochwitz & Otter, *The Functional Mockup Interface for Tool independent Exchange of Simulation Models*, 2011).

A possible solution for this problem is to use FMI for model definition and data exchange. In this case the OEM only needs to know the model interfaces so the protection of model IP of supplier is still guaranteed see Figure 2-10.

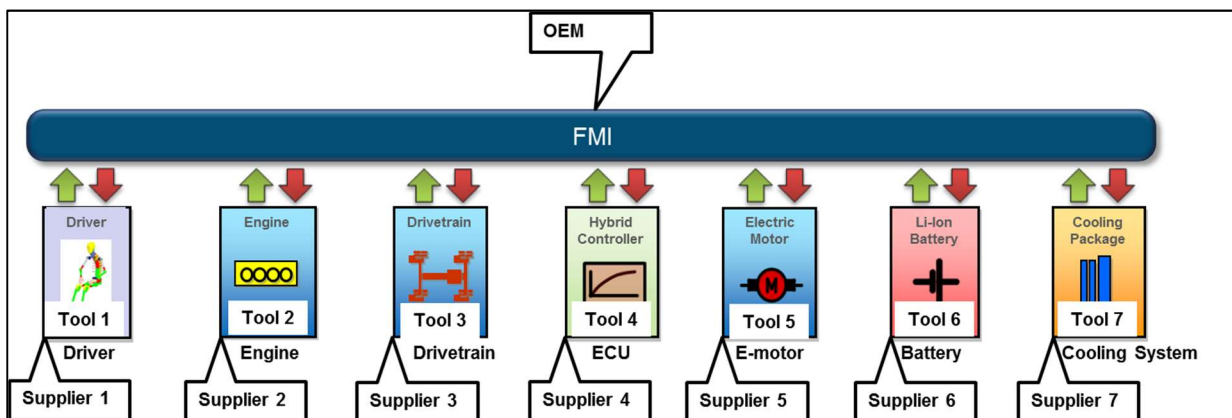


Figure 2-10: Model integration using FMI (Blochwitz & Otter, *The Functional Mockup Interface for Tool independent Exchange of Simulation Models*, 2011).

Due to a missing standard for model exchange and Co-Simulation the *FMI 1.0 for Model Exchange* was published in January 2010. Later in October 2010 *FMI 1.0 for Co-Simulation* followed. Both were developed within MODELISAR (ITEA2 project). In July 2014 *FMI 2.0 for Model Exchange and Co-Simulation* was published which is not backwards compatible to FMI 1.0. Currently there is an ongoing development for *FMI 2.1*.

¹ ITEA 2 (Information Technology for European Advancement)

The FMI defines an interface to be implemented by an executable called FMU (Functional Mock-up Unit). The FMI functions are used by a simulation environment to create one or more instances of the FMU and to simulate them, typically together with other models. A FMU can have its own solvers (FMI for Co-Simulation) or require the simulation environment to perform numerical integration (FMI for Model Exchange) (The Functional Mock-up Interface Standard).

2.2.2 FMI for Model Exchange

The FMI for Model Exchange interface defines an interface to the model of a dynamic system described by differential, algebraic and discrete-time equations and to provide an interface to evaluate these equations as needed in different simulation environments with explicit or implicit integrators and fixed or variable step-size. The interface is designed to allow the description of large models. Figure 2-11 shows the principle structure.

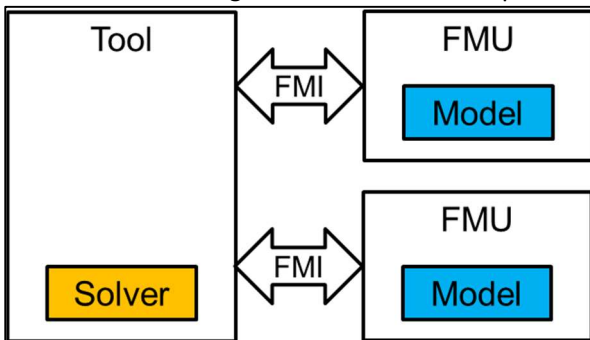


Figure 2-11: FMI for Model Exchange (Blochwitz & Otter, The Functional Mockup Interface for Tool independent Exchange of Simulation Models, 2011).

2.2.3 FMI for Co-Simulation

The FMI for Co-Simulation interface is designed both for the coupling of simulation tools (simulator coupling, tool coupling), and the coupling with subsystem models (which have been exported by their simulators together with its solvers as runnable code). The goal is to compute the solution of an entire time dependent coupled systems consisting of subsystems that are continuous in time (model components that are described by differential-algebraic equations) or time-discrete (model components that are described by difference equations, for example discrete controllers). From co-simulation point of view the coupled overall system consists of several subsystems represented by blocks with (internal) state variables $x(t)$ that are connected to other subsystems (blocks) of the coupled problem by subsystem inputs $u(t)$ and subsystem outputs $y(t)$. During time integration, the simulation is performed independently for all subsystems restricting the data exchange between subsystems to discrete communication points. The principle structure is depicted in Figure 2-12.

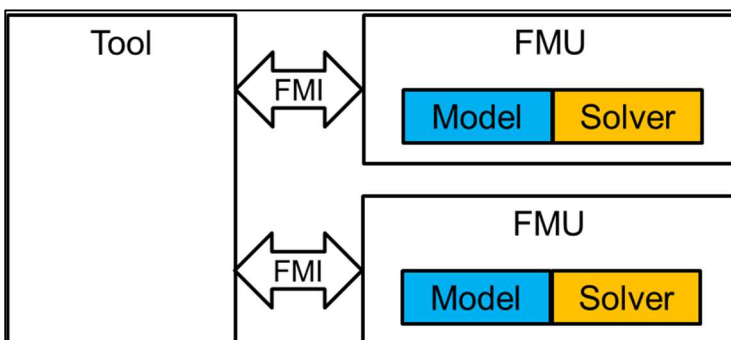


Figure 2-12: FMI for Co-Simulation (Blochwitz & Otter, The Functional Mockup Interface for Tool independent Exchange of Simulation Models, 2011).

2.2.4 FMI properties

The two interfaces have large parts in common (The Functional Mock-up Interface Standard), in particular:

- **FMI Application Programming Interface (C):** All needed equations or tool-coupling computations are evaluated by calling standardized C-functions.
- **FMI Description Schema (XML)**
The schema defines the structure and content of an XML file generated by a modeling environment. This XML file contains the definition of all variables of the FMU in a standardized way. It is then possible to run the C code in an embedded system without the overhead of the variable definition.
- **Note:** Depending on the used simulation tool a license is maybe required to run/simulate the FMI (this means FMI does not necessarily solve the tool licensing issue).

2.2.5 FMU properties

An FMU is distributed in one zip file which contains:

- The FMI Description File (in XML format).
- The C sources of the FMU, including the needed run-time libraries used in the model, and/or binaries for one or several target machines, such as Windows dynamic link libraries (.dll) or Linux shared object libraries (.so). The latter solution is especially used if the FMU provider wants to hide the source code to secure the contained know-how or to allow a fully automatic import of the FMU in another simulation environment.
- Additional FMU data (like tables, maps) in FMU specific file formats.

A schematic view of an FMU is shown in Figure 2-13:

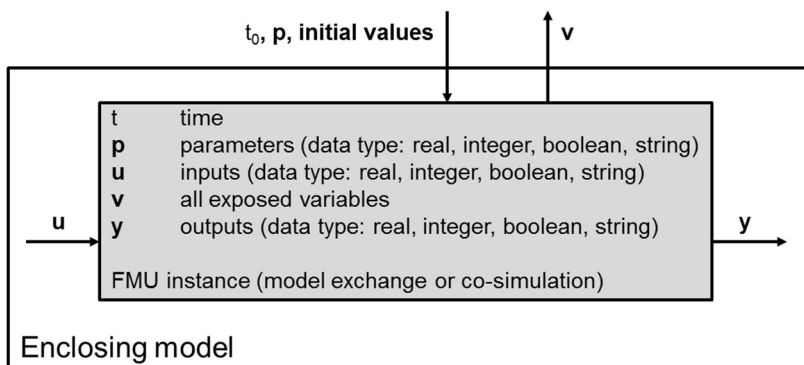


Figure 2-13: Data flow between the environment and an FMU (The Functional Mock-up Interface Standard).

2.2.6 FMI use cases

Figure 2-14 shows a use case in *standalone* configuration i.e. all elements of the simulation are loaded as shared objects and run within one process (in-process). In Model.CONNECT™ all FMUs are co-simulated within one process by default.

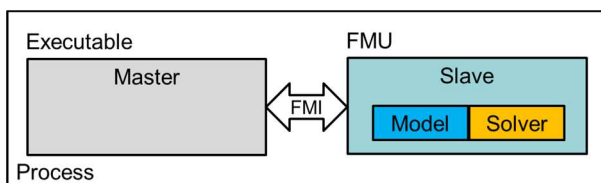


Figure 2-14: Standalone configuration (Blochwitz & Otter, The Functional Mockup Interface for Tool independent Exchange of Simulation Models, 2011).

Figure 2-15 shows a use case in *tool-based* configuration i.e. all elements are running as individual processes (multi-process). In Model.CONNECT™ all tool-interface (ICOS) elements are running as individual processes, which interact by using inter-process communication (IPC). FMUs can also be assigned to the ICOS execution group i.e. they are handled in the same way as the tool-interface elements.

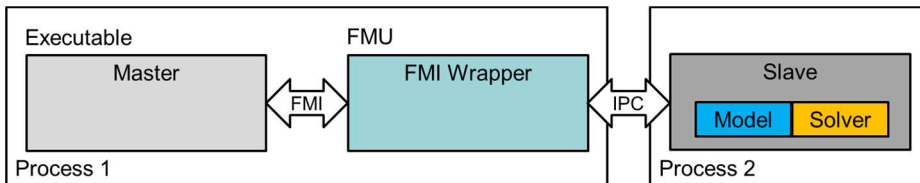


Figure 2-15: Tool-based configuration (Blochwitz & Otter, The Functional Mockup Interface for Tool independent Exchange of Simulation Models, 2011).

Figure 2-16 shows a use case in *distributed* configuration, which represents an extension of the *tool based* configuration. Since all elements are running as individual processes, they can be executed on different machines (distributed multi-process). In Model.CONNECT™ all tool-interface (ICOS) elements can be executed in this way. To make us of this configuration the FMUs has to be assigned to the ICOS execution group.

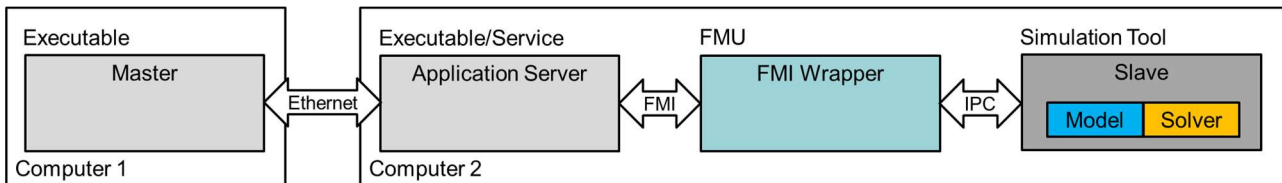


Figure 2-16: Distributed configuration (Blochwitz & Otter, The Functional Mockup Interface for Tool independent Exchange of Simulation Models, 2011).

2.2.7 Comparison of FMI 1.0 and FMI 2.0

An overview of all new FMI 2.0 features is outlined in detail in (Blochwitz, New Features of FMI 2.0 and beyond, 2014). The most important project related differences to FMI 1.0 are discussed in this section like:

- improved initialization,
- semantics of event handling,
- tunable parameters and classification of variables,
- save and restore FMU state,
- detailed dependency information (inputs, outputs, derivatives) and
- an efficient interface to Jacobian matrices.

2.2.7.1 Initialization and Semantics of event handling

Table 2-1 summarizes the improvements of FMI 2.0 concerning initialization and semantics of event handling compared to FMI 1.0.

Table 2-1: Different initialization and semantics of event handling in FMI 2.0 (Blochwitz, New Features of FMI 2.0 and beyond, 2014).

FMI 1.0	FMI 2.0
Only one function call for initialization	Introduction of <i>initialization mode</i>
No iteration during initialization possible	Solution of algebraic loops in the same way as in <i>continuous-time mode</i>
Critical, if initial conditions depend on variables in algebraic loops	Introduction of <i>event-mode</i> and a semantics for enter, exit and setting of new discrete state

Initialization Mode:



The initialization mode is used to compute at start time t_0 initial values for continuous-time states, (t_0) , and for the previous (internal) discrete-time states, $\mathbf{x}_d(t_0)$.

Continuous-Time Mode:

The introduced continuous-time mode is used to compute the values of all (real) continuous-time variables between events by numerically solving ordinary differential and algebraic equations. All discrete-time variables are fixed during this phase and the corresponding discrete-time equations are not evaluated.

Event Mode:

The event-mode is used to compute new values for all continuous-time variables, as well as for all discrete-time variables that are activated at the current event instant t , given the values of the variables from the previous instant. This is performed by solving algebraic equations consisting of all continuous-time and all active discrete-time equations. In FMI 2.0 there is no mechanism that the FMU can provide the information whether a discrete-time variable is active or is not active (is not computed) at an event instant. Therefore, the environment has to assume that at an event instant always all discrete-time variables are computed, although internally in the FMU only a subset might be newly computed.

2.2.7.2 Tunable parameters and classification of variables

The combination of *causality* and *variability* allows a clear classification of all kinds of variables, see Table 2-2. FMI 2.0 supports a distinction between tunable and fixed parameters.

Table 2-2: Overview of variable classification possibilities (Blochwitz, New Features of FMI 2.0 and beyond, 2014).

Causality	Variability
<i>Parameter</i>	<i>Constant</i>
<i>Input</i> : output of another model	<i>Fixed</i> : constant after initialization
<i>Output</i> : input of another model	<i>Tunable</i> : constant between events
<i>Local</i> : not to be used by other models	<i>Discrete</i> : changes at event instances
	<i>Continuous</i>

2.2.7.3 Save and restore FMU state

FMI 1.0 is based on an implicate save and restore mechanisms while FMI 2.0 uses explicate function calls for model exchange and co-simulation. Therefore FMI 2.0 is capable to support iterative co-simulation algorithms, and model predictive control schemes.

2.2.7.4 Detailed dependency information

Using FMI 1.0, only dependencies of outputs on inputs can be indicated while FMI 2.0 also considers dependencies of outputs on continuous states and dependencies of derivatives on continuous states and inputs. This improvements support the detection of algebraic loops and the definition of sparsity pattern of Jacobian matrices.

2.2.7.5 Efficient interface to Jacobian matrices

FMI 2.0 offers an efficient interface to Jacobian matrices which are needed for e.g. implicit integration methods, solution of systems of equations resulting from algebraic loops, linearization of FMU and Extended Kalman filters. The calculation of these Jacobians may be expensive for large models.

2.2.8 FMU example

Model.CONNECT™ supports FMI for model exchange as well FMI for co-simulation (for more information about the FMI types see section 2.2.3). Figure 2-17 shows an example with both FMI types (Model.CONNECT™ User Manual): Solver0 and Solver 1 handles FMU Model Exchange 1 and FMU Model Exchange 2 respectively, which are not directly connected to other Model Exchange FMUs. Solver 2 handles FMU Model Exchange 3 and FMU Model Exchange 4 which are interconnected and share the same settings. On the contrary the last remaining

FMU Co-Simulation 1 uses its own solver. In conclusion, the Model.CONNECT™ platform ensures maximum flexibility in using different FMUs with user specific settings.

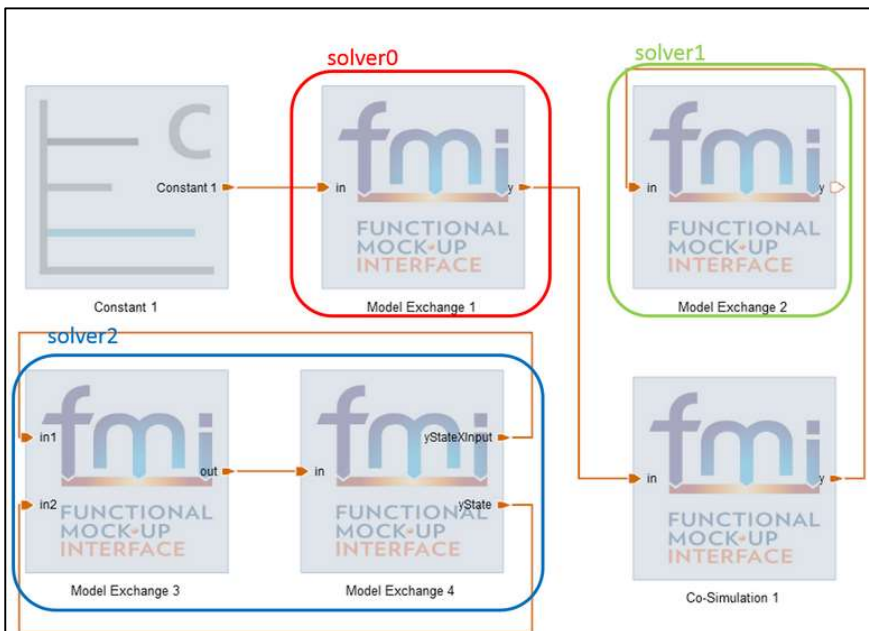


Figure 2-17: FMI example in Model.CONNECT™ (Model.CONNECT™ User Manual).

2.3 RT capability

2.3.1 Description

The classical conception is that in a hard real-time or immediate real-time system, the completion of an operation after its deadline is considered useless - ultimately, this may cause a critical failure of the complete system (see Figure 2-18). A soft real-time system on the other hand will tolerate such lateness, and may respond with decreased service quality (e.g., omitting frames while displaying a video). Hard real-time systems are used when it is imperative that an event is reacted to within a strict deadline. Such strong guarantees are required of systems for which not reacting in a certain interval of time would cause great loss in some manner, especially damaging the surroundings physically or threatening human lives (although the strict definition is simply that missing the deadline constitutes failure of the system). For example, a car engine control system is a hard real-time system because a delayed signal may cause engine failure or damage.

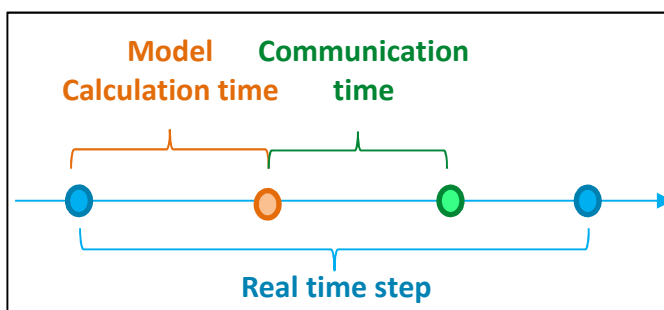


Figure 2-18 temporal stability in Real time hardware platform

Most of real-time models are executed with Simulink. Nevertheless, a model can be exported to a compatible real-time target by means of an FMU. This specific type of FMU is called “source code FMU” and contains (Figure 2-19):



- the necessary source code to compile
- a real-time target-specific pre-compiled library to be linked during the build phase
- a text file containing information about what needs to be compiled



Figure 2-19 documentation directory

2.3.2 RT model examples

2.3.2.1 LMS Amesim model

FMU for real-time in LMS Amesim are compatible with the following real-time target platforms:

- Concurrent Simulation Workbench, tested on version 6.10.1 gold
- dSPACE SCALEXIO, tested on version R2015b
- ETAS LABCAR (32-bit), tested on version V5.3.1

Below we can find an example of co-simulation with FMU (Amesim models), split into three parts:

- The driving control part is a Simulink model; it is therefore not exported from LMS Amesim:

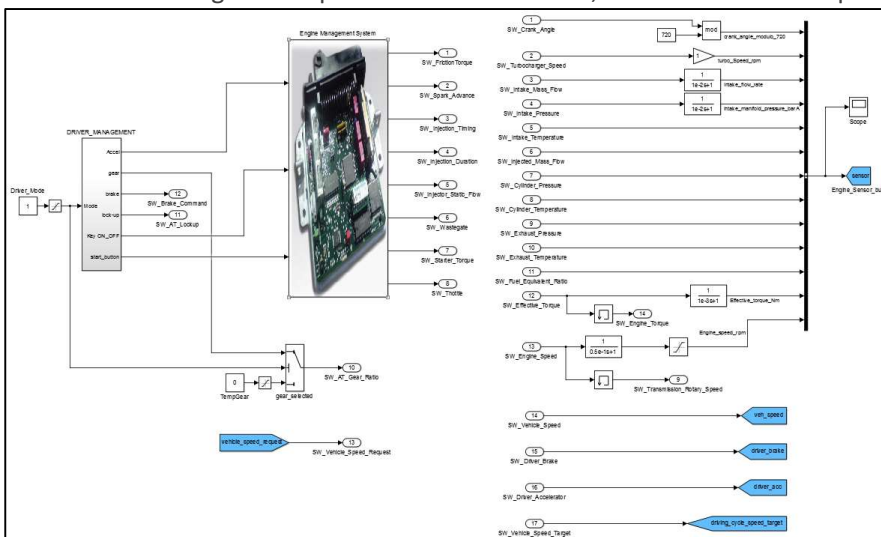


Figure 2-20 Simulink driver control model

- The powertrain part is an LMS Amesim model that has been exported as a 2.0 FMU for co-simulation:

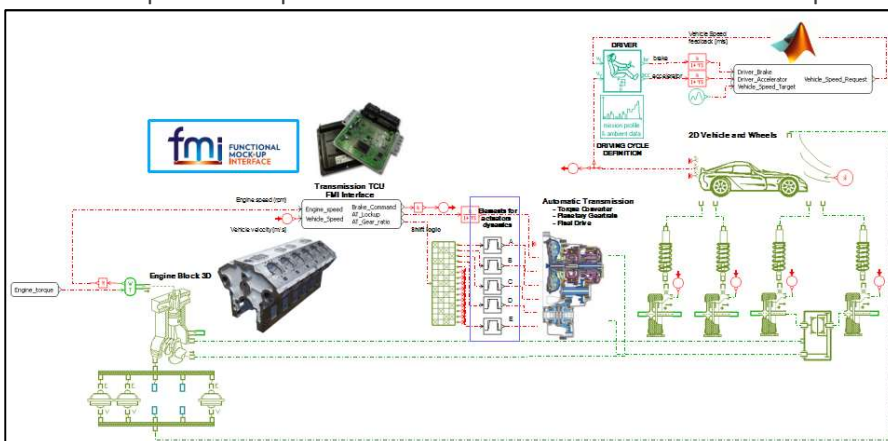


Figure 2-21 LMS Amesim powertrain model

- The engine part is an LMS Amesim model that has been exported as a 2.0 FMU for co-simulation:

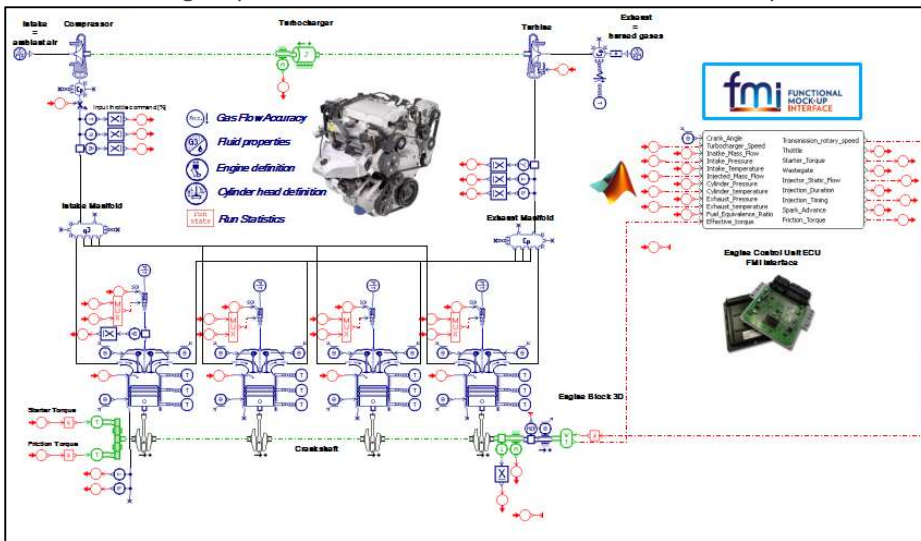


Figure 2-22 LMS Amesim engine model

The co-simulation time steps are set to 10-3s for all models whereas their solver settings are given hereafter:

- Driving control model: first-order Euler method at a step size of 10-3s.
- Powertrain model: first-order Euler method at a step size of 5.10-4s.
- Engine model: first-order Euler method at a step size of 10-4s.

All these models have been exported to a dSPACE SCALEXIO target. Here below are some examples of the user interface with results:

- Dashboard interface with vehicle speed, gear, engine rpm (Figure 2-23)

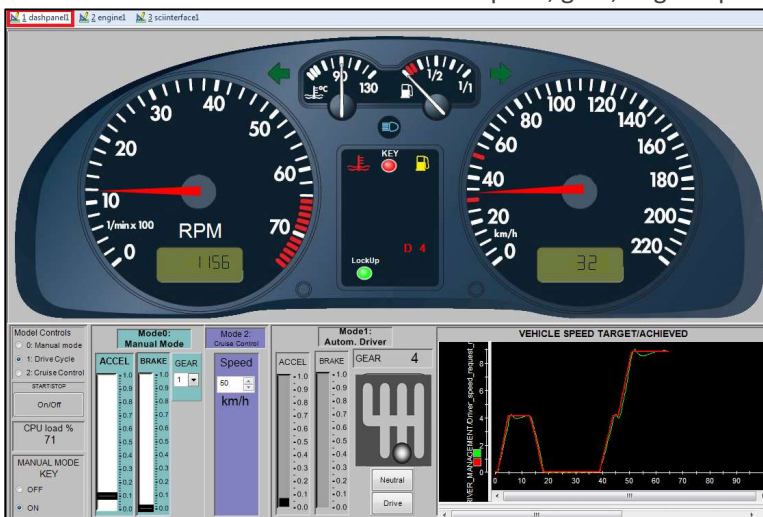


Figure 2-23 Dashboard interface

- engine interface with actuator positions, cylinder pressure (Figure 2-24)

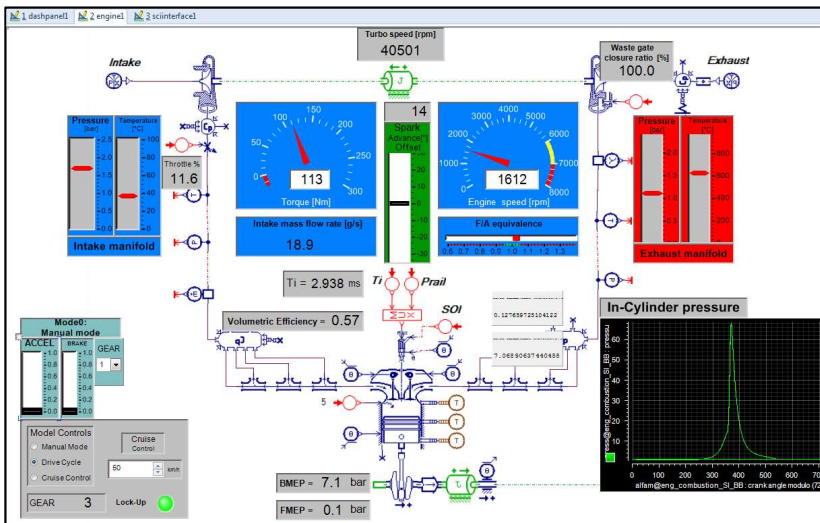


Figure 2-24 Engine interface

2.3.2.2 Model.CONNECT™ model

In the context of real-time co-simulation, a distinction between real-time and non-real-time systems is commonly used. Real-time systems (RT), e.g. in form of real hardware, have to satisfy the so-called hard real-time conditions (e.g. guaranteed response-time, deterministic runtime behavior). Non-real-time systems (non-RT) in form of offline simulation models do not satisfy these conditions in general, but have to be executed faster than real-time for synchronization purposes (Stettinger, Benedikt, Thek, & Zehetner, 2013).

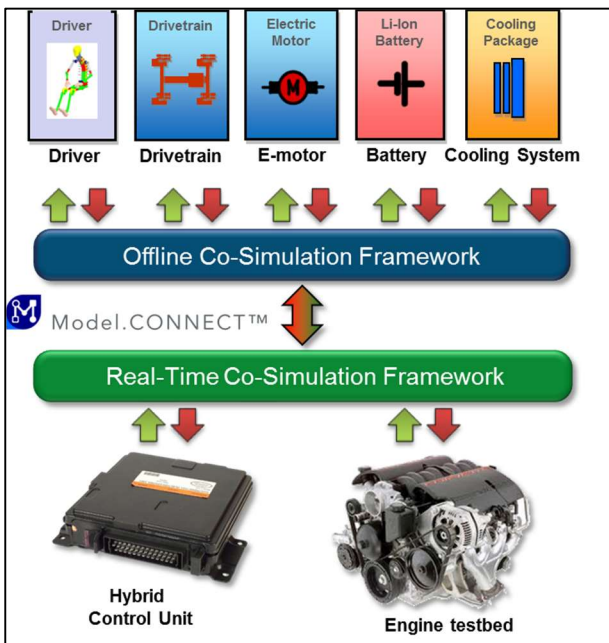


Figure 2-25: Real-Time Co-Simulation example.

Figure 2-25 shows an exemplary setup for a real-time co-simulation configuration: The upper area of Figure 2-25 represents the offline co-simulation part. In this area, the simulation tools are coupled via the offline co-simulation framework of the Model.CONNECT™ co-simulation platform. Only subsystems which can be executed faster than real-time can be used for the real-time co-simulation problem due to synchronization purposes. The lower area of Figure 2-25 shows the coupling of real-time systems (real hardware) via the Real-Time Co-Simulation framework of Model.CONNECT™. For a real-time co-simulation both levels of interaction



have to be connected via special coupling- and error-correction methods. This ensures the required hard real-time behavior of the resulting overall system. Currently, Model.CONNECT™ supports CAN as well as UDP communication to connect real-time systems to the offline co-simulation part.

The main problems of real-time co-simulations are the so-called round trip-times. They occur in closed-loop systems due to finite communication-, computation- or data-processing-times. From a control engineering point of view these dead-times can cause critical problems concerning the stability of the system. They lead to oscillations and in the worst case to an unstable closed loop behavior. In either case a distorted picture of the real system as a whole is generated. Furthermore data-losses (e.g. caused by data-collisions) as well as noisy coupling signals (originating from real sensors) complicate the coupling process. For that reason special coupling algorithms are available which can compensate this disturbing effects (Stettinger, Benedikt, Thek, & Zehetner, 2013), see section 4.2.4.

3 Port typing conventions and model identity

3.1 Model identity

Figure 3-1 shows a kind of template to share the most information for each subsystem, which is needed to setup a co-simulation. So, this template acts as a kind of subsystem-identify card (SIC). For interconnecting different subsystems from different domains, a clear interface definition is mandatory i.e. this interface definition includes all present in- and outputs of the specific subsystems to ensure a correct interaction with other subsystems in terms of signal flow.

Apart from that, simulation tool specific information like numerical solver information is important for the correct configuration of the co-simulation. This information is used to define correct coupling time-steps for interaction with other involved subsystems. For real-time co-simulations, the required real-time capability must be ensured. Furthermore, the target hardware information is essential to setup the real-time co-simulation. Especially, the used communication medium and the communication step-size need to be specified.

Subsystem Identity Card						
Subsystem model specific information						
Name of subsystem	e.g. LilonBattery					
From Partner	ViF, AVL, FO, US, ...					
Use Case Cluster	1, 2, 3					
Use Case Number	2.1, 2.3, ...					
Short description	purpose, usage, additional information, subsystem dynamics ...					
Known limitations	for specific simulations only - valid in specific operating conditions, ...					
...	individually list extensions					
Inputs of the subsystem						
Name / Description	Unit	Lower Limit	Upper Limit	Data Type	Dimension	Init Value
Outputs of the subsystem (at least the relevant ones)						
Name / Description	Unit	Lower Limit	Upper Limit	Data Type	Dimension	
Simulation specific information						
Simulation tool & version	Matlab/Simulink 2010a, MSC-Adams 2010, ...					
Numerical solver	Ode45, Dassi, Euler, ...					
Kind of numerical step-sizes	fixed, variable, both, ...					
Numerical step-size	fixed step-size - max step-size in case of variabel step-size solver					
Real-time capable	yes/no					
Project licence available	yes/no					
...	individually list extensions					
Hardware specific information						
Hardware description	dSPACE MikroAutoBox, Testbed, HiL, ECU, ...					
Communication step-size	fixed step-size					
Communication medium	UDP, CAN, ...					
...	individually list extensions					

Figure 3-1: Subsystem identity card.

The focus in this project is on the electric components. Nevertheless, the complete interface has to be described in order to check the interaction between all subsystems. 16 different subsystems have been defined. For each of them dedicated inputs and outputs have been specified to insure the consistency between several levels of the model (scalability) but also to anticipate integration in the complete vehicle model.

3.2 Components

Figure 3-2 shows a generic electric vehicle architecture. In this project the focus is on the electrical components (inverter, Emotor and battery), but other subsystem have also to be taken into account:

- the cooling circuit,
- the auxiliaries,
- the driveline,

- the controls

We can observe a lot of physical connections between all these subsystems:

- Electrical connections
- Mechanical connection
- Thermal connection (heat transfer, heat exchangers)
- Control signals

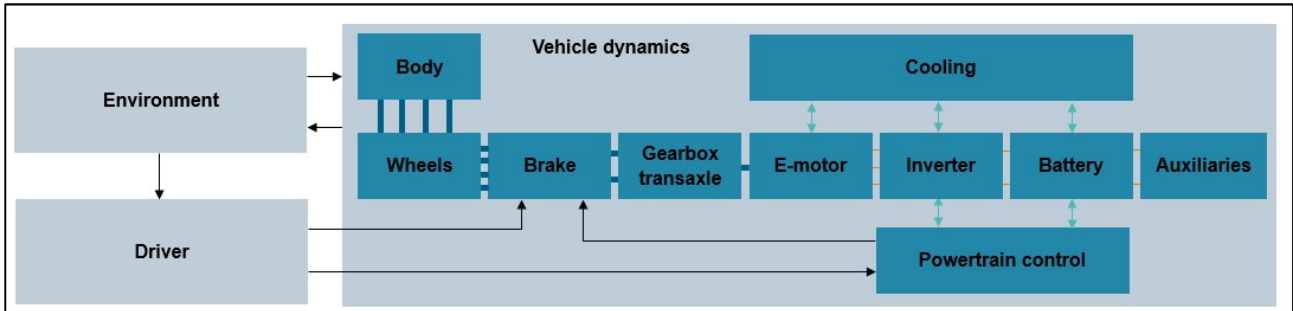


Figure 3-2: Generic electric architecture

All these connections have to be identified for each subsystem in order to ensure the consistency during the vehicle integration. Furthermore, several level of modeling at least for electrical components must be developed. Indeed, a lot of physical effects are required in detailed models implying longer simulation time, which could not be real-time compatible. Faster models have so to be created to be real-time compatible based on first models.

3.2.1 E-motor

3.2.1.1 Level of modeling

Three technologies of E-motor (Figure 3-3) are considered in this project:

- Wound Rotor Synchronous Motor (WRSM)
- Permanent Magnet Synchronous Machines (PMSM)
- Induction machine (IM)



Figure 3-3: E-motor

2 levels of model have to share the maximum common parameters but also inputs/outputs:

- Multi-physical approach coupling magnetic, electric, thermal and mechanical domains being implemented for a detailed physical inverter
- Real-time compatible model sharing most of the information with the physical model and running with real-time compatible inverter model for vehicle simulation

The second model is derived from the first one by following the model reduction process (4.1).

3.2.1.2 Port typing convention

Table 3-1 Inputs/outputs of Emotor

Variables	Name	I/O	Type	Unit	dimension
Motor speed	Omega_EM	Input	physical	rad/s	1
Motor coolant temperature	T_cool_EM	Input	physical	degC	1
Motor voltage	U_EM	Input	physical	V	n ²
Motor torque	Tq_EM	Output	physical	Nm	1
Motor power losses	P_loss_EM	Output	physical	W	1
Motor heat rejection to coolant	P_cool_EM	Output	physical	W	1
Motor temperature	T_EM	Output	physical	degC	n ³
Motor current	I_EM	Output	physical	A	n ⁴
Motor current phase	I_phase_EM	Output	physical	deg	3
Motor maximum generative torque	Tq_max_rege_EM	Output	signal	Nm	1
Motor maximum boost torque	Tq_max_boost_EM	Output	signal	Nm	1

3.2.2 Inverter

3.2.2.1 Level of modeling

The inverter (Figure 3-4: inverter) has two main functions:

- Convert DC current to AC current
- Control motor torque

This system is mainly made of semiconductor components based on Silicon (Si). New semiconductor material could be investigated in this project like Silicon Carbide (SiC) and Gallium Nitride (GaN).



Figure 3-4: inverter

2 levels of models have to share the maximum common parameters but also inputs/outputs:

- detailed multi-domain and multi-physical Power Electronics Converters (inverter, converter) being connected with detailed physical motor
- Real-time compatible model sharing most of information with physical model and running with real-time motor model for vehicle simulation

² highly dependent of level of modeling and also the inverter (see detailed model) -->d/q , abc ... usually 2 or 3

³ if motor thermal masses are split (e.g. stator and rotor considered separately) or not.

⁴ highly dependent of level of modeling and also the inverter (see detailed model) -->d/q , abc ... usually 2 or 3

The second model is derived from the first one by following the model reduction process (4.1).

3.2.2.2 Port typing convention

Table 3-2 Inputs/outputs of inverter

Variables	Name	I/O	Type	Unit	dimension
Inverter AC Voltage request	U_AC_Inv_rq	Input	signal	V	1
Inverter AC current	I_AC_Inv	Input	physical	A	n ⁵
Inverter DC voltage	U_DC_Inv	Input	physical	V	1
Inverter coolant temperature	T_cool_Inv	Input	physical	degC	1
Inverter AC voltage	U_AC_Inv	Output	physical	V	n ⁶
Inverter DC current	I_AC_Inv	Output	physical	A	1
Inverter power losses	P_loss_Inv	Output	physical	W	1
Inverter heat rejection to coolant	P_cool_Inv	Output	physical	W	1
Inverter temperature	T_Inv	Output	physical	degC	n ⁷

3.2.3 Battery

3.2.3.1 Level of modeling

Li-Ion battery (Figure 3-5) is a very complex system with different scale level phenomena. Current electrochemical models are based on Newman's approach, which is not able to consider such phenomena (R. Malik, 2013) (W. Dreyer, 2010) (J. Lim, 2016).



Figure 3-5: Nissan leaf battery module

2 levels of models have to share the maximum common parameters but also inputs/outputs:

- Electro-chemical model based on a new approach with thermal influence and ability to predict thermal runaways
- Real-time compatible model sharing most of information with physical model and running with for vehicle simulation

The second model is derived from the first one by following the model reduction process (4.1).

3.2.3.2 Port typing convention

Table 3-3 Inputs/outputs of battery

Variables	Name	I/O	Type	Unit	dimension
-----------	------	-----	------	------	-----------

⁵ highly dependent of level of modeling and also the motor (see detailed model) -->d/q , abc ... usually 2 or 3

⁶ highly dependent of level of modeling and also the motor (see detailed model) -->d/q , abc ... usually 2 or 3

⁷ if inverter thermal masses are split (e.g. each semiconductor considered separately) or not.

Battery current	I_Bat	Input	physical	A	1
Battery maximum power	P_Bat_max	Input	signal	W	1
Battery minimum power	P_Bat_min	Input	signal	W	1
Battery coolant temperature	T_cool_Bat	Input	physical	degC	1
Battery voltage	U_Bat	Output	physical	U	n ⁸
Battery state of charge	SOC_Bat	Output	signal	%	1
Battery state of health	SOH_Bat	Output	signal	%	n ⁹
Battery power losses	P_loss_Bat	Output	physical	W	1
Battery heat rejection to coolant	P_cool_Bat	Output	physical	W	1
Battery temperature	T_Bat	Output	physical	degC	n ¹⁰

3.2.4 Converter DC/DC

3.2.4.1 Level of modeling

DC/DC converter is used to transfer current between two electric circuits with different voltages. Buck and boost converters (Figure 3-6, Figure 3-7) are widely used to, respectively, decrease and increase the output voltages.

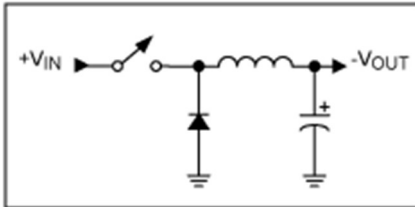


Figure 3-6: buck converter topology

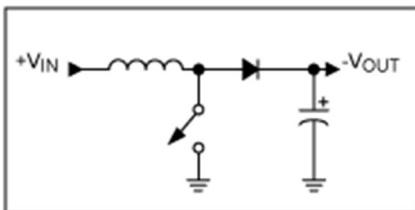


Figure 3-7: boost converter topology

2 levels of models have to share the maximum common parameters but also inputs/outputs:

- detailed multi-domain and multi-physical Power Electronics Converters (inverter, converter)
- Real-time compatible model sharing most of information with physical model and running for vehicle simulation

The second model are derived from the first one by following the model reduction process (4.1).

3.2.4.2 Port typing convention

Table 3-4 Inputs/outputs of converter DC/DC

Variables	Name	I/O	Type	Unit	dimension
DC/DC coolant temperature	T_cool_DC	Input	physical	degC	1
DC/DC coolant flow	Q_cool_DC	Input	physical	l/min	1

⁸ Several voltage could be considered (e.g. maximum/minimum/average)

⁹ Several SOH could be considered (e.g. pack/module/cell)

¹⁰ Several temperature could be considered (e.g. pack/module/cell)



DC/DC power losses	P_loss_DC	Output	physical	W	1
DC/DC heat rejection to coolant	P_cool_DC	Output	physical	W	1
DC/DC temperature	T_DC	Output	physical	degC	1

3.2.5 Body builder

3.2.5.1 Level of modeling

Only one level of modeling supplying additional electrical load to the electrical circuit is considered.

3.2.5.2 Port typing convention

Table 3-5 Inputs/outputs of body builder

Variables	Name	I/O	Type	Unit	dimension
Body Builder coolant temperature	T_cool_BB	Output	physical	degC	1
Body Builder power losses	P_loss_BB	Output	physical	W	1
Body Builder temperature	T_BB	Output	physical	degC	1

3.2.6 OnBC

3.2.6.1 Level of modeling

Only one level of modeling supplying additional electrical load to the electrical circuit is considered.

3.2.6.2 Port typing convention

Table 3-6 Inputs/outputs of OnBC

Variables	Name	I/O	Type	Unit	dimension
OnBC DC voltage	U_DC_OnBC	Input	physical	V	1
OnBC coolant temperature	T_cool_OnBC	Output	physical	degC	1
OnBC DC current	I_AC_OnBC	Output	physical	A	1
OnBC power losses	P_loss_OnBC	Output	physical	W	1
OnBC heat rejection to coolant	P_cool_OnBC	Output	physical	W	1
OnBC temperature	T_OnBC	Output	physical	degC	1

3.2.7 Powertrain

3.2.7.1 Level of modeling

Powertrain component deals with mechanical subsystem of the vehicle from the chassis to the transmission. Connection with the E-motor is connected to the powertrain as torque supplier. Furthermore, the vehicle speed and acceleration are transferred to the driver component.

3.2.7.2 Port typing convention

Table 3-7 Inputs/outputs of Powertrain

Variables	Name	I/O	Type	Unit	dimension
Motor torque	Tq_EM	Input	physical	Nm	1
Gear demand (if transmission)	Gear_ratio	Input	signal	-	1
K1 state (if transmission)	K1	Input	signal	-	1
Vehicle speed	Vveh	Output	physical	m/s	1
Motor speed	Omega_EM	Output	physical	rad/s	1
Travel distance	distance	Output	physical	m	1
Vehicle acceleration	accel	Output	physical	m/s**2	1



Transmission loss (if transmission)	P_loss_Trans	Output	physical	W	1
Transmission heat rejection to oil	P_cool_Trans	Output	physical	W	1

3.2.8 Braking system

3.2.8.1 Level of modeling

Only one level of modeling supplying braking system for the vehicle simulation is considered, allowing focus on brake blending.

3.2.8.2 Port typing convention

Table 3-8 Inputs/outputs of braking system

Variables	Name	I/O	Type	Unit	dimension
Dissipative Braking Torque	Tq_Br_d_i	Input	Physical	Nm	1
External Torque	Tq_E_r_i	Input	Physical	Nm	1
Wheel Speed	W_w_i	Input	physical	rad/s	1
Ref. Cooling Temp	T_cool	Input	physical	degC	1
Powertrain Speed	W_w_g_i	Output	Physical	rad/s	1
Measured Speed	W_w_s_i	Output	Signal	rad/s	1
Wheel Torque	Tq_W_i	Output	physical	Nm	1
Pad Temp	T_pad_i	Output	signal	degC	1
Disc Temp	T_disc_i	Output	signal	degC	1
Consumed Pad Volume	V_pad_i	Output	physical	m3	1

3.2.9 Cooling system

3.2.9.1 Level of modeling

The two main functions of the cooling system are:

- the thermal management of all electrical and mechanical component as well as the battery
- the heat supply for cabin comfort

Heat rejection from loss component is transferred to the coolant. Fan and pump are controlled to optimize the thermal behavior within the cooling loop.

Only one level of modeling supplying cooling circuit compatible for Real-time application is considered for the vehicle simulation.

3.2.9.2 Port typing convention

Table 3-9 Inputs/outputs of cooling system

Variables	Name	I/O	Type	Unit	dimension
Battery heater loss	P_loss_Bat	Input	physical	W	1
Inverter loss	P_loss_Inv	Input	physical	W	1
Motor loss	P_loss_EM	Input	physical	W	1
Radiator airflow	Airflow-Rad	Input	physical	Kg/s	1
Radiator air inlet temperature	Tair-in-Rad	Input	physical	degC	1
Fan AC current	I_fan	Input	physical	A	1
Fan AC voltage	U_fan	Input	physical	V	1
Fan inlet temperature	T_in_fan	Input	physical	°C	1
Fan inlet pressure	Pres_in_fan	Input	physical	kPa	1
Pump AC current	I_pmp	Input	physical	A	1
Pump AC voltage	U_pmp	Input	physical	V	1



Pump Coolant mass flow	M_cool_pmp	Output	physical	Kg/s	1
Pump coolant flow	Q_cool_pmp	Output	physical	l/min	1
Radiator coolant flow	Q_cool_Rad	Output	physical	l/min	1
Coolant temperature in radiator	T_cool_in_Rad	Output	physical	degC	1
Coolant temperature out radiator	T_cool_out_Rad	Output	physical	degC	1
Coolant temperature in inverter	T_cool_in_inv	Output	physical	degC	1
Coolant temperature out inverter	T_cool_out_inv	Output	physical	degC	1
Coolant temperature in motor	T_cool_in_motor	Output	physical	degC	1
Coolant temperature out motor	T_cool_out_motor	Output	physical	degC	1
Coolant pressure in radiator	Pres_cool_in_Rad	Output	physical	kPa	1
Coolant pressure out radiator	Pres_cool_out_Rad	Output	physical	kPa	1
Coolant pressure in inverter	Pres_cool_in_inv	Output	physical	kPa	1
Coolant pressure out inverter	Pres_cool_out_inv	Output	physical	kPa	1
Coolant pressure in motor	Pres_cool_in_motor	Output	physical	kPa	1
Coolant pressure out motor	Pres_cool_out_motor	Output	physical	kPa	1
Coolant pressure in pump	Pres_cool_in_pmp	Output	physical	kPa	1
Coolant pressure out pump	Pres_cool_out_pmp	Output	physical	kPa	1
Radiator air Pressure drop	DP-air-rad	Output	physical	kPa	1
Radiator Cooling capacity	P_rad	Output	physical	W	1
Fan speed	Speed-fan	Input	physical	rpm	1
Fan outlet pressure	Pres_out_fan	Output	physical	kPa	1
Fan flow	Q-cool-fan	Output	physical	l/min	1
Fan Power	Pow-fan	Output	physical	kW	1
Pump speed	Speed_pmp	Output	physical	rpm	1
Pump Power	Pow-pmp	Output	physical	kW	1

3.2.10 Heating, Ventilation Air Conditioning (HVAC)

3.2.10.1 Level of modeling

Several models have been investigated from functional to detailed model (especially with cabin). Cabin internal flows are complex and must be managed with 3D flow. Advanced cabin heating technology will be investigated like heat pump.

3.2.10.2 Port typing convention

Table 3-10 Inputs/outputs of HVAC

Variables	Name	I/O	Type	Unit	dimension
Blower position	Pos_Blower	Input	signal	-	1
recirculation	Recir	Input	signal	%	1
Compressor speed	Omega_Comp	Input	physical	rad/s	1
Chiller Inlet coolant temperature	Ch_Inlet_coolTemp	Input	physical	degC	1
Chiller Inlet coolant flowrate	Ch_Inlet_fr	Input	physical	kg/s	1
Air HV Heater Command	Air_Heat_cmd	Input	signal	-	1
Chiller Txv command	Ch_cmd	Input	signal	-	1
Evaporator Txv command	EV_cmd	Input	signal	-	1
Condenser Inlet air Temperature	Cond_in_AirTemp	Input	physical	degC	1
Condenser air flowrate	Cond_Air_fr	Input	physical	kg/s	1



Cabin temperature	T_Cab	Output	physical	degC	1
Cabin humidity	Rh_Cab	Output	physical	%	1
Compressor torque	Tq_Comp	Output	physical	Nm	1
HVAC dissipated thermal power	W_HVAC	Output	Physical	W	1
Chiller Outlet coolant temperature	Ch_Outlet_coolTemp	Output	physical	degC	1
Chiller Outlet coolant flowrate	Ch_Outlet_fr	Output	physical	kg/s	1
Condenser Outlet air Temperature	Cond_out_AirTemp	Output	physical	degC	1
Head Pressure	pHead	Output	physical	bar	1
Evaporator Air Outlet temperature	EV_out_AirTemp	Output	physical	degC	1
Air HV Heater Outlet Temperature	Heat_out_AirTemp	Output	physical	degC	1

3.2.11 Energy Management control

3.2.11.1 Level of modeling

Only one level of modeling supplying energy distribution between different motors compatible for Real-time application is considered for the vehicle simulation. Battery state of charge as well maximum and minimum power are considered to limit the power flow from the battery.

3.2.11.2 Port typing convention

Table 3-11 Inputs/outputs of energy management control

Variables	Name	I/O	Type	Unit	dimension
Battery maximum power	P_Bat_max	Input	signal	W	1
Battery minimum power	P_Bat_min	Input	signal	W	1
Battery state of charge	SOC_Bat	Input	signal	%	1
Vehicle speed	Vveh	Input	signal	m/s	1
Motor speed	Omega_EM	Input	signal	rad/s	1
Power req driver	P_req	Input	signal	W	1
Power req auxiliary systems	P_aux_req	Input	signal	W	1
Power command to the Battery	P_batt	Output	signal	W	1
Power command to the EM	P_em	Output	signal	W	1
Power command to the Auxiliary	P_aux	Output	signal	W	1

3.2.12 Braking blending control

3.2.12.1 Level of modeling

Only one level of modeling supplying control for the braking system is considered, compatible for Real-time application for vehicle simulation. Brake blending between applied torque (e-motor) and conventional actuation is optimized by also considering the dynamic limitation for applied torques.

3.2.12.2 Port typing convention

Table 3-12 Inputs/outputs of braking blending control

Variables	Name	I/O	Type	Unit	dimension
Brake Demand	BRK_REF_i	Input	Signal	%	1
Electrical Limits	ELE_lim_i	Input	Signal	Nm, W	2
BBCPARAM	BBC_i	Input	signal	-	4
Functionality Mode	FCM_i	Input	signal	-	1
Measured Speed	W_w_s_i	Input	signal	rad/s	1



Measured Wheel Normal Load	Fn_w_i	Input	signal	N	1
Brake Feedback	BRK_FED_i	Output	Signal	%	1
Ext. Torque Demand	Tq_bre_i	Output	signal	Nm	1
Dis. Braking Torque	Tq_Br_d_i	Output	physical	Nm	1

3.2.13 Driver

3.2.13.1 Level of modeling

Only one level of modeling supplying the overall power request is considered, compatible for Real-time application for vehicle simulation. Acceleration and braking request are calculated based on vehicle speed and requested vehicle power.

3.2.13.2 Port typing convention

Table 3-13 Inputs/outputs of thermal control

Variables	Name	I/O	Type	Unit	dimension
Accelerator pedal	Acc_pdl	Input	signal	%	1
Brake pedal	B_pdl	Input	signal	%	1
Power request	P_req	Output	signal	W	1

3.2.14 Powertrain control

3.2.14.1 Level of modeling

Only one level of modeling supplying control for powertrain is considered, compatible for Real-time application for vehicle simulation. If transmission is implemented in the vehicle, gear demand and clutch closing have to be calculated.

3.2.14.2 Port typing convention

Table 3-14 Inputs/outputs of Powertrain control

Variables	Name	I/O	Type	Unit	dimension
Vehicle speed	Vveh	Input	physical	m/s	1
Travel distance	distance	Input	physical	m	1
Vehicle acceleration	accel	Input	physical	m/s**2	1
Gear demand (if any)	Gear_ratio	Output	signal	-	1
K1 state (if transmission)	K1	Output	signal	-	1

3.2.15 HVAC control

3.2.15.1 Level of modeling

Only one level of modeling supplying control for HVAC is considered, compatible for vehicle simulation. Thermal comfort (cabin temperature and humidity) is controlled by adjusting the compressor speed and blower position. Air recirculation could be taken into account.

3.2.15.2 Port typing convention

Table 3-15 Inputs/outputs of HVAC control

Variables	Name	I/O	Type	Unit	dimension
Cabin temperature	T_Cab	Input	physical	degC	1
Cabin humidity	Rh_Cab	Input	physical	%	1
Compressor torque	Tq_Comp	Input	physical	Nm	1



HVAC dissipated thermal power	W_HVAC	Input	Physical	W	1
Chiller Outlet coolant temperature	Ch_Outlet_coolTemp	Input	physical	degC	1
Chiller Outlet coolant flowrate	Ch_Outlet_fr	Input	physical	kg/s	1
Condenser Outlet air Temperature	Cond_out_AirTemp	Input	physical	degC	1
Head Pressure	pHead	Input	physical	bar	1
Evaporator Air Outlet temperature	EV_out_AirTemp	Input	physical	degC	1
Air HV Heater Outlet Temperature	Heat_out_AirTemp	Input	physical	degC	1
Blower position	Pos_Blower	Output	signal	-	1
recirculation	Recir	Output	signal	%	1
Compressor speed	Omega_Comp	Output	physical	rad/s	1
Air HV Heater Command	Air_Heat_cmd	Output	signal	-	1
Chiller Txv command	Ch_cmd	Output	signal	-	1
Evaporator Txv command	EV_cmd	Output	signal	-	1

3.2.16 Predictor supervisor

3.2.16.1 Level of modeling

Only one level of modeling supplying information about environment is considered, compatible for vehicle simulation.

3.2.16.2 Port typing convention

Table 3-16 Inputs/outputs of predictor supervisor

Variables	Name	I/O	Type	Unit	dimension
Travel distance	distance	Input	physical	m	1
Reference vehicle speed xx kms ahead	Vveh	Output	physical	m/s	1
Road altitude xx kms ahead	Altitude	Output	physical	m	1
Stop time xx kms ahead	Time	Output	physical	s	1
Estimated road cycle power demand xx kms ahead	Power	Output	physical	W	1

4 Computation approaches and heterogeneous couplings stability

4.1 Model reduction strategies & Model scalability

Design phase can be shortened by using same architecture whatever the level of modeling of each component will be. Conclusion from ASTERICS (FP7) about integrated approach (C. Ricci, 2014) are:

- Interoperability between models at every modeling stage
- System has to be easy upgradeable

The consistency has to be ensured between different levels of modeling. Furthermore numerical stability has to be guaranteed especially for HiL application.

Physical models are commonly used during design phase of the “V design cycle” (Left side) and are the first step of modeling activity in model-based development. Based on these system integrated models, some subsystem has to be reduced, meaning having the capability to run in fixed step with a limited CPU time consumption to be integrated in real-time platform (HiL). Generally, 3 levels can be found in simulation (see Figure 4-1):

- Quasi-static approach: only steady state outputs are considered in the component model. Such models are useful for Real time application, but level of accuracy is poor and some physical output could be missing.
- Low transient approach: quasi-static output and main dynamic are considered. This level offers better accuracy especially in term of transient behavior. The maximum frequency must comply with communication time in real time hardware platform (~100 Hz)
- High frequency approach: these models are generally very accurate but require longer CPU time due to high frequency involved. Such level cannot be directly used in HiL approach.

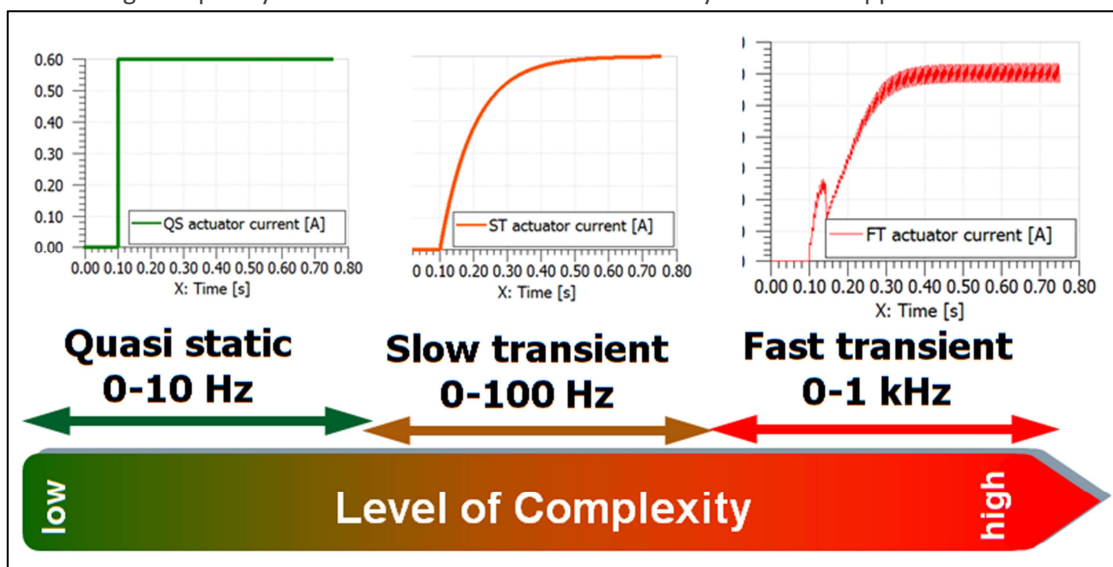


Figure 4-1 level of detail of simulation

The main objective of the model reduction strategies is to limit the frequency domain of the model to ensure the numerical stability with higher integration step. Generally, the Euler criteria should be respected:

$$T_{max} < \frac{1}{2 \cdot \pi \cdot \min(f_{system})}$$

Physical models with high dynamics require very low integration step due to their high frequencies. The computation takes more time than real time and as a consequence must be replaced by low dynamic or quasi-static models. Nevertheless the consistency between models have to be preserved meaning the loss of accuracy should not be critical.

According to the complexity of each subsystem several strategies could be applied:

- Quasi-static model based on physical model could be applied on electrical component (see 3.2.1.1, 3.2.2.1, 3.2.3.1)
- Response Surface Methodology (RSM) (see Figure 4-2)
- Neural Network (RNN)

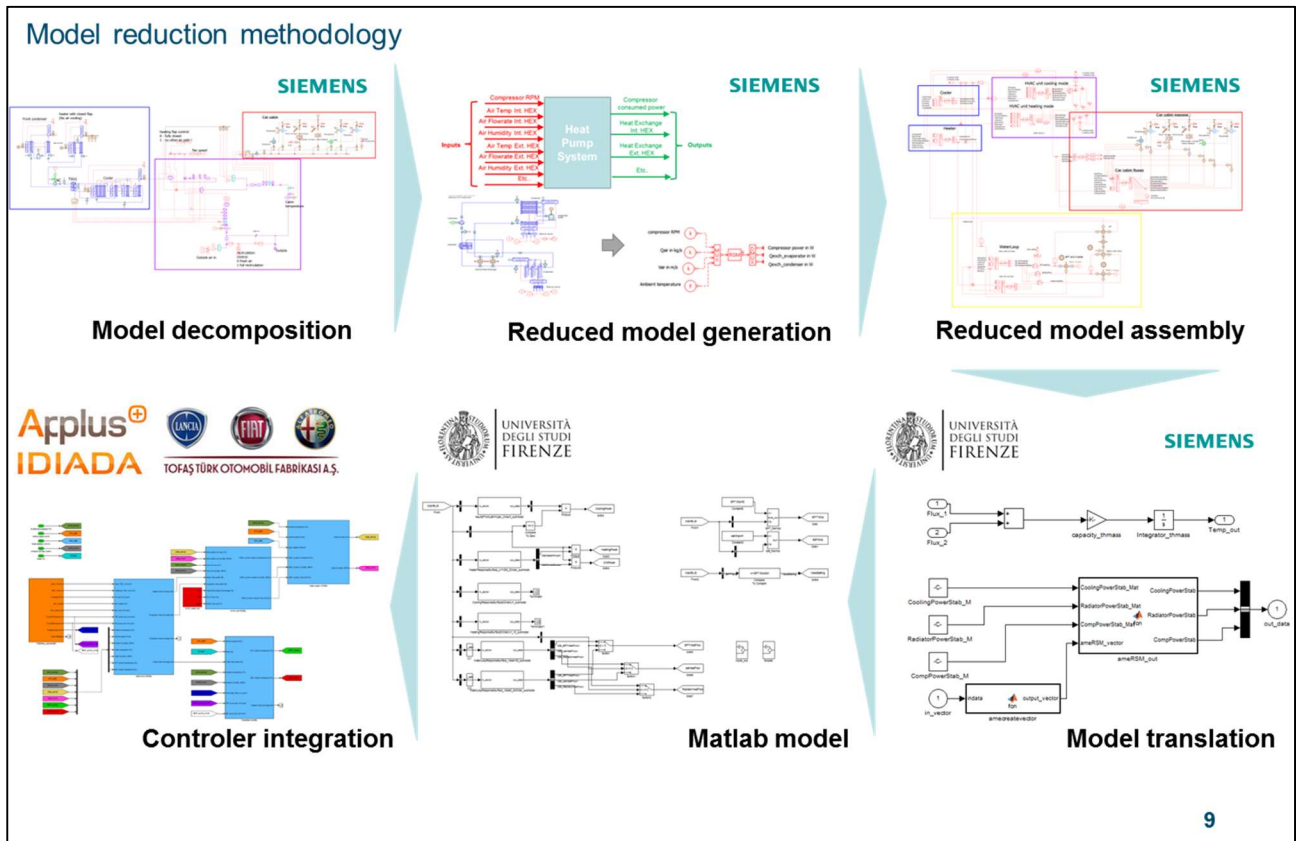


Figure 4-2 model reduction process used during IMROVE project (Thielboger, 2017)

4.2 Co-simulation Heterogeneous integration

The main task of co-simulation is the holistic simulation of an overall system to determine the global characteristics of the system. It consists of several subsystems, which are simulated in their domain specific simulation tools, see Figure 2-5. From an abstract point of view the subsystems, can be considered as black-box systems with inputs and outputs. Thereby, the overall system is assembled by interconnecting the subsystems via the inputs and outputs, which ensures the interaction of the involved subsystems in a collaborative manner. For the simulation of the overall system typically a co-simulation platform is used. In this case the subsystems run independently from each other and only exchange values at discrete points in time. The two main tasks of the platform are to define an effective subsystem scheduling for the simulation tools and to handle the occurring coupling data at specific points in time (Benedikt, Zehetner, Watzenig, & Bernasch, 2011). The co-simulation user has to define the following settings for each subsystem:

- Subsystem scheduling
- Coupling step-size (for data exchange between subsystems)
- Input signal extrapolation / interpolation

4.2.1 Scheduling

In terms of scheduling, several rules must be obeyed to obtain an efficient configuration of subsystems. The subsystem scheduling influences the simulation time as well as the accuracy. Furthermore, it is necessary to simulate the subsystems in signal flow direction and, in the case of internal loops, only the correct scheduling guarantees the solvability of the whole system. Nevertheless, it is important to reduce the amount of required extrapolation steps because each extrapolation is associated with a coupling error. Figure 4-3 shows exemplarily two interconnected subsystems with different scheduling policies.

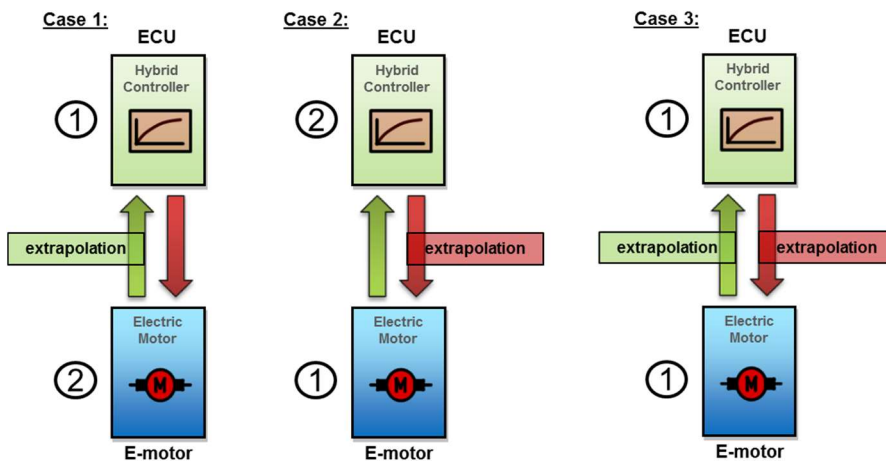


Figure 4-3: Different possibilities of simulator scheduling for a coupled co-simulation (Benedikt, Zehetner, Watzenig, & Bernasch, 2011)

In case 1, subsystem 1 (hybrid controller) is solved first leading to an extrapolation of its unknown input signal. In case 2, the electric motor subsystem is solved first and its input signal has to be extrapolated. In both cases the subsystem models are solved sequentially in time and only one coupling signal has to be extrapolated. For the parallel case (see case 3), a simultaneous extrapolation of all subsystem inputs is required and thus a larger coupling error is introduced with distorts the entire system behavior (Benedikt, Zehetner, Watzenig, & Bernasch, 2011).

4.2.2 Coupling step size

In the context of co-simulation, the step-size regarding the internal solver is called micro time-step and is denoted by the symbol δT , see Figure 4-4. The coupling time instants, where coupling data is exchanged between the simulation tools, are defined by the specified coupling time-step ΔT . The coupling data exchange between subsystems is performed by the co-simulation platform. In practice, an efficient coupling time-step is mostly determined using domain-specific experts or by executing numerous trial and error tests. In fact, by non-iterative co-simulation applications the coupling time-step is the most important factor to achieve accurate simulation results (Benedikt, Zehetner, Watzenig, & Bernasch, 2011).

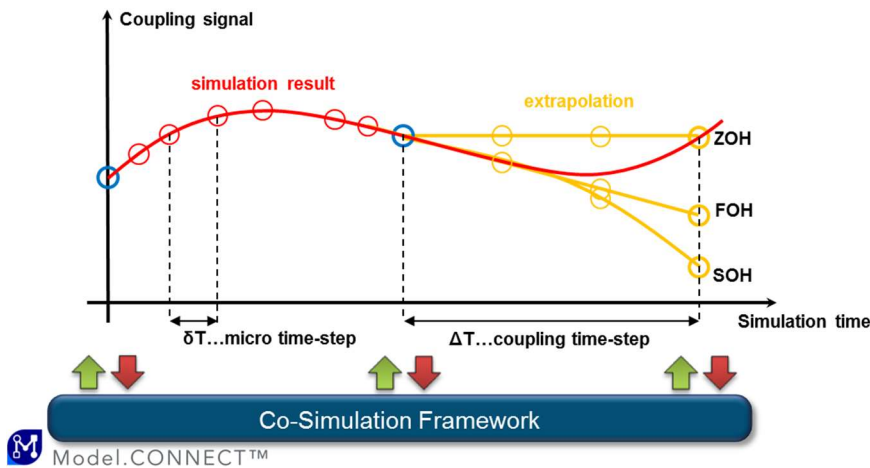


Figure 4-4: Exchange of subsystem data at coupling time instants and definition of time steps.

4.2.3 Input signal extrapolation

According to the specified subsystem scheduling some subsystem inputs have to be extrapolated to perform the co-simulation. Extrapolation of coupling quantities represents a prediction of simulation results over the subsequent coupling time-step at defined coupling time instants, see Figure 4-4. In this context extrapolation is a kind of estimation and thus a coupling error is introduced. The introduced error depends strongly on the extrapolation technique and on the applied coupling step size. In co-simulation applications typically polynomial extrapolation techniques of low order are used such as zero order hold (ZOH), first order hold (FOH) or second order hold (SOH) extrapolation (Benedikt, Zehetner, Watzenig, & Bernasch, 2011), see Figure 4-4.

4.2.4 Coupling Algorithms

The co-simulation platform Model.CONNECT™ also contains algorithms to compensate distorting coupling errors introduced by extrapolation. The so-called NEPCE (Nearly Energy Preserving Coupling Element) for non-iterative co-simulation compensates such extrapolation errors. This approach produces some kind of energy preservation. Furthermore it is possible to eliminate high frequent signal components in the coupling signals with predefined filters (in general low-pass filter) (Benedikt & Hofer, Guidelines for the Application of a Coupling Method for Non-iterative Co-simulation, 2013).

Besides NEPCE, especially for real-time co-simulations special model-based coupling algorithms (ACoRTA coupling) can be used to ensure stable entire system simulations. These model-based coupling algorithms are designed to cope with typical coupling imperfections caused by the incorporation of real-time systems like communication time-delays, data-losses and noisy coupling signals (Stettinger, Benedikt, Thek, & Zehetner, 2013).

4.3 HPC

In the last decades, the design world has been deeply transformed by computer science. Many industries, including automotive, rely on this technology to develop new products and test processes virtually, thus the need of physical prototypes has been reduced.

The virtual validation is overtaking the physical one in many areas and this trend is constantly accelerating. The need of more reliable simulations means more complex virtual models and in turn this means that the computational power needed for each simulation is drastically increasing. The solution to limit the time needed by numerical simulations has been increased the computational power by parallelization.

The aim of High Performance Computing (HPC) is to provide the computational power for the simulations needed for virtual design and validation phases. It is possible to connect together many ordinary computers to obtain a single system, called 'cluster'. The computational power of the all the computers, called 'nodes', is aggregated, making it possible to solve much more complex problems than one could solve on a typical desktop



computer or workstation. The components of a cluster are connected to each other through a network (connection high speed is a key requirements) and each node runs on its own operating system. In most applications, all of the nodes use the same hardware and the same operating system.

Traditionally, the computational power needed by the numerical simulations is handled by the central processing unit (CPU) of a node. A new and very promising approach, is to handle all or part of these computations through the graphical processing unit (GPU). CPUs are very efficient for sequential calculation but creates difficulties parallelized computations with high numbers of CPUs, whereas GPUs are attractive due to their high floating-point operation capability and their high energy-efficiency for highly parallel computations. Some clusters are built combining GPUs and CPUs so one can optimize the computation in a way that exploits both the GPU and CPU strengths while avoiding their weaknesses. More and more computational clusters are equipped with GPUs or uses hybrid technologies.

A large company that heavily relies on numerical simulations for its design and validation process will typically need a cluster made up by hundreds of nodes with each node having a number of CPU cores of the order 32

4.3.1 Parallel computing

To fully exploit the computational power provided by a modern HPC cluster, the software that will perform the numerical simulation needs to be designed using parallel computing paradigms. Parallel computing is a type of computation in which many calculations are executed at the same time. Large simulations can be partitioned into smaller ones that can be run simultaneously. A parallel software normally manages different processes that are computed, independently, but continually share data with each other. The communication is granted by a properly designed network. Parallel programming relies on two main pillars: efficient use of the computational power available on single nodes and efficient communications of inter-dependent data among different nodes (and within the cores).

Ideally the performance of a parallel program should be linearly dependent on the available computational power: doubling the computational power, the performance of the software should be doubled. However, this is hardly the case. The reason must be searched in the cost of communication among the nodes that is extremely high. The scalability (also referred to as the scaling efficiency) is a common measure of the parallel performance of a software. This indicator shows how efficient an application is when the number of parallel processing elements (CPUs, cores, etc.) is increased. Typical software for industrial simulation will exhibit good scalability also when running on hundreds of cores.

The use of HPC is limited to applications that require high computing performances; focusing the attention to the scope of OBELICS, the main application that will need such amount of resources will be the 3D thermal simulation because it relies on data obtained from Computational Fluid Dynamics analysis. The 0D-1D solver can partially benefit from the parallelization but the few processors of a high-end personal computer are already enough. There will be a significant gap between the time needed from 1D and 3D simulation so specific simulation strategies will be developed. Increasing the number of processors used by the CFD solver could not be a solution because the constraints on the maximum number of available CPUs, the scalability (80-95% of efficiency should be expected till 500 cores but, further increasing the number of cores, downgrades it) and the time needed for initialization, I/O writing and saving time.

Despite the promising properties of the GPU based calculations, this technology will not be used during this project due the limited number of software that are fully compatible.

4.3.2 Constraints: simulation strategy and operating system

The HPC use is necessary for the 3D simulations only so the compatibility of the different software with this kind of technology is relevant for the subtask 3.2.2. How the software will communicate and in which way would mostly depend on the 0D-1D-3D coupling developed strategy.

To determine the possible synergies among the different kind of simulation, hereunder a brief summary to highlight the main roles of each kind of analysis in a thermal simulation is given:

- 0D-1D analysis: effectively simulate components and subsystems of an electric vehicle (battery, electric motors, HVAC), replicating how they work and which I/O are required or provided. They could be



mutually linked to simulate the behavior of a whole vehicle in real driving conditions. Focusing on the thermal management, this kind of numerical simulations can provide information about the temperatures of each component and its thermal emission

Where the 0D-1D simulations lack is the interaction with the external environment that can only be considered extensively simplified (this includes also the mutual interaction between components). The 3D simulations will, on one hand, fulfill this shortage and, on the other, benefit of more reliable data (most of all in dynamic condition typical of unsteady simulations, like in a drive-cycle) coming from the mono-dimensional analysis. The 3D thermal simulations usually relies on different kinds of technologies in order to describe all the mechanism of the thermal exchange:

- Computational Fluid Dynamics (CFD): it is used to determine the main parameters that rule the heat exchanged by convection. Simulating the flow field around a vehicle and its parts provides the value of heat transfer coefficient (H) and the local fluid temperature around the surface of the components (normally the simulation is limited to the external part of each component to limit both the computational cost and the complexity: no internal structure is modeled)
- Thermal solver: performs the full thermal analysis simulating the heat exchanged by conduction and radiation. It exploits the data coming from the CFD to compute the heat exchanged by convection. For the same reasons seen for the CFD also in this case the internal structure of components is ignored or approximated (detailed geometry, materials and other properties are needed). Many thermal solver includes 1D codes to simulate the internal fluid. The thermal solver can be embedded in the CFD one, Conjugate Heat Transfer, or stand alone, exploiting technologies as the Boundary Element Method.

Both CFD and thermal solver need the use of HPC, but, between the two, the former has the most important computational weight and requires more time to run.

The benefits originated by the 1D-3D coupling, the kind/volume of data exchanged depends on the integration strategy as well as the working constraints.

Supposing to simulate a drive-cycle, while the subsystems simulations and the 3D thermal one can be fully transient (even with different discretization times, Δt), the same approach cannot be used by the CFD because of the resources (and time) needed by unsteady analysis. The most efficient approach is the pseudo-transient. All the other software will simulate the whole-time domain while the CFD will analyze only few operating points supposing the flow-field (and the thermal one) as stationary. This approach, although simplified, is justified by the different time-responses of the fluid behavior and the thermal behavior of solid parts: the evolution of the flow-field is far way more rapid than the temperature change of a component, so, till the scope of the simulation is the thermal management, the high-frequency aerodynamic changes can be neglected (the optimization of the number of CFD analysis is in the scope of the project).

Two kinds of pseudo-transient approaches are reasonable,

- Co-simulation (On-the-fly): 1D and the 3D thermal solver perform transient analysis exchanging data every time-step (e.g. heat loss of the battery from 1D to 3D, coolant fluid temperature and mass-flow at the inlet of the battery from 3D to 1D). The convection properties are updated at specific times (the CFD model receives the temperatures of all the surfaces coming from the 3D thermal simulation and after the run returns H and T of the fluid). All the solvers have to be frozen waiting for the CFD solution to be computed. This approach to be successful, all the software must be run on the HPC so they have to be compatible with Linux.

Another strategy would be to manage the workflow through an external software (e.g. ModeFrontier) that controls the exchange of data among the different software and their execution. In this way a multiple environment could be supported (e.g. PC window for 1D and HPC Linux for 3D).



- Off-line simulation: this is a more approximated approach and need to have at least one loop. The main advantage is an easier interaction among software, a limitation of constraints and the 1D-3D decoupling. The counterpart is an increase of approximation of the physics of the problem and a reduced accuracy. A possible workflow could start with the execution of the 0D-1D model of the whole vehicle, data (as power electronics heat dissipation and coolant mass flow) are exported and used by the 3D thermal analysis. At the end of the 3D analysis some data could be exported to the mono-dimensional model to update and re-run the simulation

5 Conclusions

After reviewing available simulation tools interfaces and especially their capabilities to communicate between them, the FMI 2.0 have been selected to ensure good communication between models. Real-time capability has also been highlighted because some models must contain scalable components running on real-time platform for control calibration and validation.

Then port typing convention have been established for electric components (E-motor, inverter and battery), which will be modeled in detailed in the WP2 in OBELICS project, and also for additional components to be fully integrated in EV architectures. Model management have to be correctly ensured. So a subsystem identity card has been proposed to fasten model sharing.

Then description of different computation approaches have been described especially in term of model reduction, co-simulation heterogeneous integration or HPC to validate complex simulation integration during the complete design cycle.

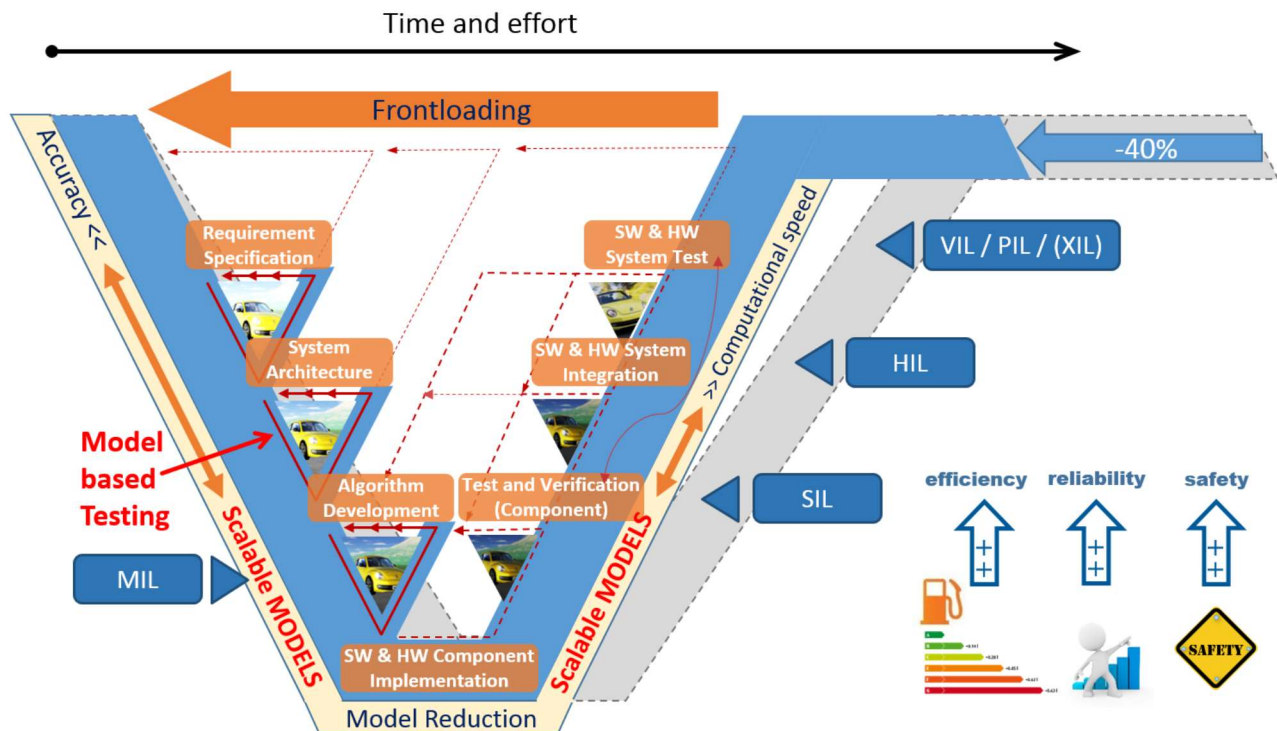


Figure 5-1: OBELICS model based development concept to reduce development and testing efforts.

Common interface, standard component input /output and dedicated computation approaches will allow the virtual integration standardization to help reducing design process by up to 40 % (see Figure 5-1).



6 Abbreviations and definition

AC	Alternative Current
DC	Direct Current
EV	Electric Vehicle
FMI	Functional Mock-up Interface
FMU	Functional Mock-up Unit
GSP	Global Simulation Platform
GSPDB	Global Simulation Platform DataBase
HiL	Hardware In the Loop
HMI	Human-Machine Interface
HPC	High Performance Computing
HV	High Voltage
HVAC	Heating, Ventilation & Air Conditioning
IM	Induction machine
ITEA	Information Technology for European Advancement
LV	Low Voltage
MiL	Model In the Loop
NN	Neural Network
PMSM	Permanent Magnet Synchronous Machines
RSM	Response Surface Methodology
SiL	Software In the Loop
VMA	Vehicle Modular Architecture
WRSM	Wound Rotor Synchronous Motor



7 Risk Register

7.1 Risk register

No new risks were identified.



8 Bibliography

- Benedikt, M., & Hofer, A. (2013). Guidelines for the Application of a Coupling Method for Non-iterative Co-simulation. *2013 8th EUROSIM Congress on Modelling and Simulation*, (pp. 244-249). Cardiff.
- Benedikt, M., Zehetner, J., Watzenig, D., & Bernasch, J. (2011). Modern coupling strategies – is co-simulation controllable? In NAFEMS (Ed.), *The Role of CAE in System Simulation*. Wiesbaden, Germany.
- Blochwitz, T. (2014). New Features of FMI 2.0 and beyond. *10th International Modelica Conference*. Lund Sweden.
- Blochwitz, T., & Otter, M. (2011). The Functional Mockup Interface for Tool independent Exchange of Simulation Models. *8th International Modelica Conference 2011*. Dresden, Germany.
- C. Ricci, P. M. (2014). *ASTERICS D1.2: methodology to identify Design goals based on real life data through preliminary model building*.
- J. Lim, a. (2016). Origin and hysteresis of lithium compositional spatiodynamics within battery primary particles,. *Science*. 353,, 566-571.
- Model.CONNECT™ User Manual. (n.d.).
- R. Malik, A. A. (2013). *Journal of The Electrochemical Society*., A3179-A3197.
- Stettinger, G., Benedikt, M., Thek, N., & Zehetner, J. (2013). On the difficulties of real-time co-simulation. *V International Conference on Computational Methods for Coupled Problems in Science and Engineering, COUPLED PROBLEMS 2013*. Ibiza, Spain.
- The Functional Mock-up Interface Standard. (n.d.). Retrieved from <http://fmi-standard.org/downloads/>
- Thielboger, H. (2017). IMPROVE: Digital modelling for EV optimization. *ASIM*. Kassel.
- W. Dreyer, a. (2010). *Nature Materials*. 9,, 448-453.



9 Acknowledgement

The author(s) would like to thank the partners in the project for their valuable comments on previous drafts and for performing the review.

Project partners:

Partner no.	Partner organisation name	Short Name
1	AVL List GmbH	AVL
2	Centro Recherche Fiat SCpA	CRF
3	FORD Otomotiv Sanayi Anonim sirketi	FO
4	Renault Trucks SAS	RT-SAS
5	AVL Software and Functions GmbH	AVL-SFR
6	Robert Bosch GmbH	Bosch
7	SIEMENS INDUSTRY SOFTWARE NV	SIE-NV
8	SIEMENS Industry Software SAS	SIE-SAS
9	Uniresearch BV	UNR
10	Valeo Equipements Electroniques Moteurs	Valeo
11	Commissariat à l'Energie Atomique et aux Energies Alternatives	CEA
12	LBF Fraunhofer	FhG-LBF
13	FH Joanneum Gesellschaft M.B.H.	FHJ
14	National Institute of Chemistry	NIC
15	University Ljubljana	UL
16	University Florence	UNIFI
17	University of Surrey	US
18	Das Virtuelle Fahrzeug Forschungsgesellschaft mbH	VIF
19	Vrije Universiteit Brussel	VUB



Copyright ©, all rights reserved. This document or any part thereof may not be made public or disclosed, copied or otherwise reproduced or used in any form or by any means, without prior permission in writing from the OBELICS Consortium. Neither OBELICS Consortium nor any of its members, their officers, employees or agents shall be liable or responsible, in negligence or otherwise, for any loss, damage or expense whatever sustained by any person as a result of the use, in any manner or form, of any knowledge, information or data contained in this document, or due to any inaccuracy, omission or error therein contained.

All Intellectual Property Rights, know-how and information provided by and/or arising from this document, such as designs, documentation, as well as preparatory material in that regard, is and shall remain the exclusive property of the OBELICS Consortium and any of its members or its licensors. Nothing contained in this document shall give, or shall be construed as giving, any right, title, ownership, interest, license or any other right in or to any IP, know-how and information.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 769506.

The information and views set out in this publication does not necessarily reflect the official opinion of the European Commission. Neither the European Union institutions and bodies nor any person acting on their behalf, may be held responsible for the use which may be made of the information contained therein.