ART-04-2016 - Safety and end-user acceptance aspects of road automation in the transition period

H2020-ART-2016-2017

Improved Trustworthiness and Weather-Independence of Conditionally Automated Vehicles in Mixed Traffic Scenarios

D3.2

Modular co-simulation architecture plan

This report is part of a project that has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement No 723324.

DOCUMENT INFORMATION

D3.2 - Modular co-simulation architecture plan	Public
Editors	Georg Stettinger (VIF), Pamela Innerwinkler (VIF), Lisa-Marie Schicker (VIF)
Authors	Pamela Innerwinkler (VIF), Georg Stettinger (VIF), Ralph Weissnegger (CISC), Cihangir Derse (TF), Eren Aydemir (FO), Johan Zaya (Volvo)
Responsible person	Georg Stettinger (VIF)
Nature	Deliverable
Status	Final

CHANGE HISTORY

Version	Date	Description	Issued by
1.0	14.06.2018	Initial Version	VIF
2.0	18.04.2018	Chapters 2.1, 2.2 added	VIF
3.0	25.04.2018	Chapter 2.3 Co-simulation using SHARC added	CISC
4.0	02.05.2018	First draft for chapter 3	VIF
5.0	08.05.2018	Introduction and conclusion added, V2 chapters 2.1 and 2.2	VIF
6.0	14.05.2018	Minor changes, all chapters	VIF
7.0	15.05.2018	Updates chapter 3.2 Tofaş use case	TF
8.0	18.05.2018	Updates chapter 3.1 Ford Otosan use case	FO
9.0	25.05.2018	Include changes from review	VIF
10.0	25.05.2018	Updates solver setting Tofaş use case	TF
11.0	14.06.2018	Adapting disclaimer	VIF

This report is part of a project that has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement No 723324.

REVIEWERS

Version	Date	Reviewer
8.0	23.05.2018	Steffen Metzner, Sajin Gopi (AVL)

This report is part of a project that has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement No 723324.

Table of Contents

This report is part of a project that has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement No 723324.

This report is part of a project that has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement No 723324.

List of Figures

List of Tables

This report is part of a project that has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement No 723324.

Executive Summary

Deliverable D3.2 deals with the setup of a modular architecture for the TrustVehicle use cases. The basis for such a modular architecture is co-simulation, which is addressed in the first part of this deliverable.

An overview of the used co-simulation platform Model.CONNECT™ is given. This platform can be used in different stages of the development V-cycle, in our case especially for testing on a pure software level (SiL), as well as on the driving simulator (Driver in the Loop). The basics of co-simulation within Model.CONNECT™ are outlined and a short overview of real-time vs. non-real-time as well as special coupling algorithms is given. Also CISC's co-simulation framework SHARC, which is focused on verification of safety-critical electric/electronic systems, is introduced and an example use case is given.

Based on the information already collected in deliverable D3.1, the use cases including their used tools and interfaces were further defined. After updating the use case information, a preliminary architecture plan for each use case was set up. Using a connection matrix, which depicts all interfaces in a well-structured way, it was possible to easily check feasibility of the data transfer and find missing connections. These findings led to the current use case setups presented within this deliverable. Finally for each of the respective subsystems tool-, solver- and hardware specifications are given.

Deliverable D3.2 now forms the basis for the actual setup of the co-simulation framework, which will be done in the next steps within the TrustVehicle project.

Key Words

Automated Driving, Co-simulation, Modular architecture, interfaces, solver settings

This report is part of a project that has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement No 723324.

1 Introduction

1.1 Overview of the Report

This document extends the basics for the co-simulation in the TrustVehicle use cases already described in D3.1. As mentioned therein, co-simulation can be used in different stages of the development V-cycle, in our case especially for testing on a pure software level (SiL), as well as on the driving simulator This document on the one hand gives a clearer picture on how co-simulation can be used and on the other hand forms the basis for the actual use case setup using Model.CONNECT™, which is applicable in both stages of testing mentioned above.

To broaden the knowledge on co-simulation, a second tool is described within this document as well. SHARC is CISC's co-simulation framework, developed further within the TrustVehicle project. Since it is focused on verification of safety-critical electric/electronic systems, it meets TrustVehicle's aspirations to address the increase in reliability for automated vehicles.

The specification sheets already collected in Task 3.1 are updated to include new specifications done within the past few months. Based on these specification sheets, a modular use case specific architecture is assembled for each use case. A more detailed view on the interfaces is given in form of a connection matrix, which depicts the connections between each of the respective subsystem blocks. Using this connection matrix, the entire system can be set up in Task 3.4 using the Model.CONNECT™ framework. Also it provides a clear scheme of the interfaces that can be used in case singular subsystems are replaced by hardware or other simulation tools. This will be important in WP 5 and 6, where tests on the driving simulator will be performed and demonstrator vehicles will be built up.

1.2 Structure of the report

Section 2 gives an introduction to co-simulation. In section 2.1 general information on co-simulation using Model.CONNECT™ is given. Section 2.2 further briefly describes the challenges of real-time co-simulation. In section 2.3 the co-simulation framework SHARC is introduced, giving the main ideas and an example for application. Section 3 then focuses on the TrustVehicle specific use cases. There is a subsection for the Ford Otosan, the Tofaş and the Volvo use case, where for each the respective model architecture, a connection matrix and the tool-solver settings are given. Finally, the report finishes with a conclusion wrapping up the main content of this deliverable.

This report is part of a project that has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement No 723324.

2 Co-simulation

This section gives a brief overview of the co-simulation platform Mode.CONNECT™, which is used for co-simulation of the TrustVehicle use cases. First a general introduction is given, then real-time co-simulation is introduced. The final section gives an overview of CISC's co-simulation framework SHARC, which is further developed within this project.

2.1 Co-simulation using Model.CONNECT™

The co-simulation platform Model.CONNECT™ enables setting up and executing entire mechatronic system simulations that are composed of subsystem and component models from multiple model authoring environments. Either standardized interfaces (e.g. Functional Mockup Interface, FMI) or specific interfaces to simulation tools can be used to integrate models into the framework. There is already a wide range of specific interfaces available for the inclusion of many well-known simulation tools, especially concerning tools common in the automotive industry. A list of supported tools can be found in the Model.CONNECT user manual [1]. As an example for such a co-simulation set-up, a possible co-simulation for an automated driving use case is depicted in Figure 1.

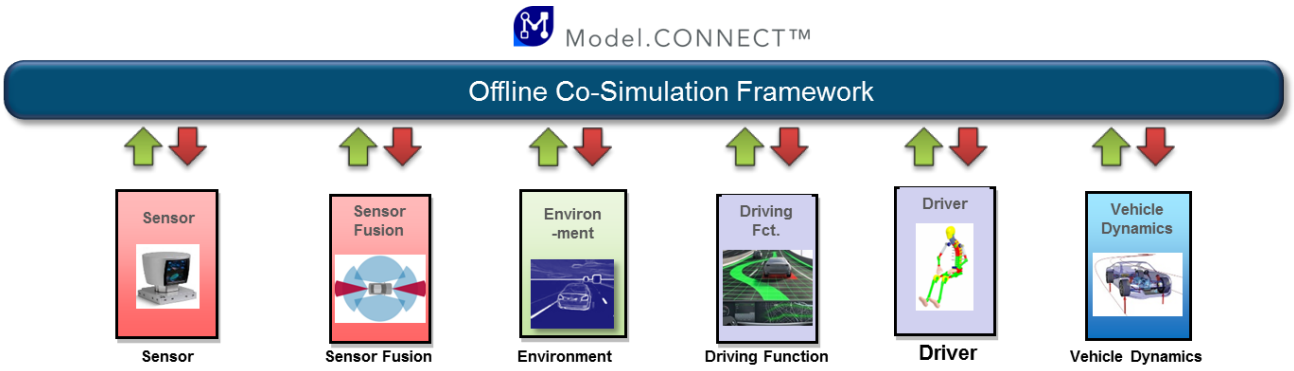

Figure 1 Possible automated driving co-simulation setup[1]

A huge advantage of Model.CONNECT™ is that it supports the user in organizing system model variants. Different configurations of the system under investigation as well as different testing scenarios and testing environments can be managed.

For coupling of the models two options are available. Both options can also be combined if needed. The first option is model integration based on models that are provided as executable libraries. In this case (FMI for Co-Simulation or Model Exchange, as well as compiled MATLAB/Simulink models) the model configurations can be executed in one process. Supported model interfaces are: AVL

[1] Image sources: http://insidercarnews24.com/2018/04/25/automotive-lidar-sensors-market-review/; www.elektrobit.com; http://www.vehicledynamicsinternational.com

This report is part of a project that has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement No 723324.

fmi.LAB, AVL BOOST™, AVL FIRE™, AVL CRUISE™, AVL CRUISE™ M, FMU, AVL VSM™, Vires VTD, IPG CarMaker/TruckMaker. The second option is tool-coupling based on the ICOS technology. ICOS is a distributed co-simulation platform included in Model.CONNECT™ 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. Interacting via inter-process communication, all tool-interface elements are running as individual processes. Supported ICOS tool interfaces are: ICOS Custom, ICOS Real Time, ICOS CAN, MSC Adams, LMS Amesim, IPG CarMaker4SL, Mechanical Simulation Corp. CarSim/TruckSim, 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. The lists of model interfaces as well as supported ICOS tool interfaces as stated above may not be complete and are also extended constantly.

Apart from local co-simulation, Model.CONNECT™ also allows the connection of different operating systems on distributed resources, see Figure 2 Distributed co-simulation. A running remote server on each host is required to perform a distributed co-simulation, i.e. connecting different simulation tools on different host computers, where every one of them may have their individual operating system.

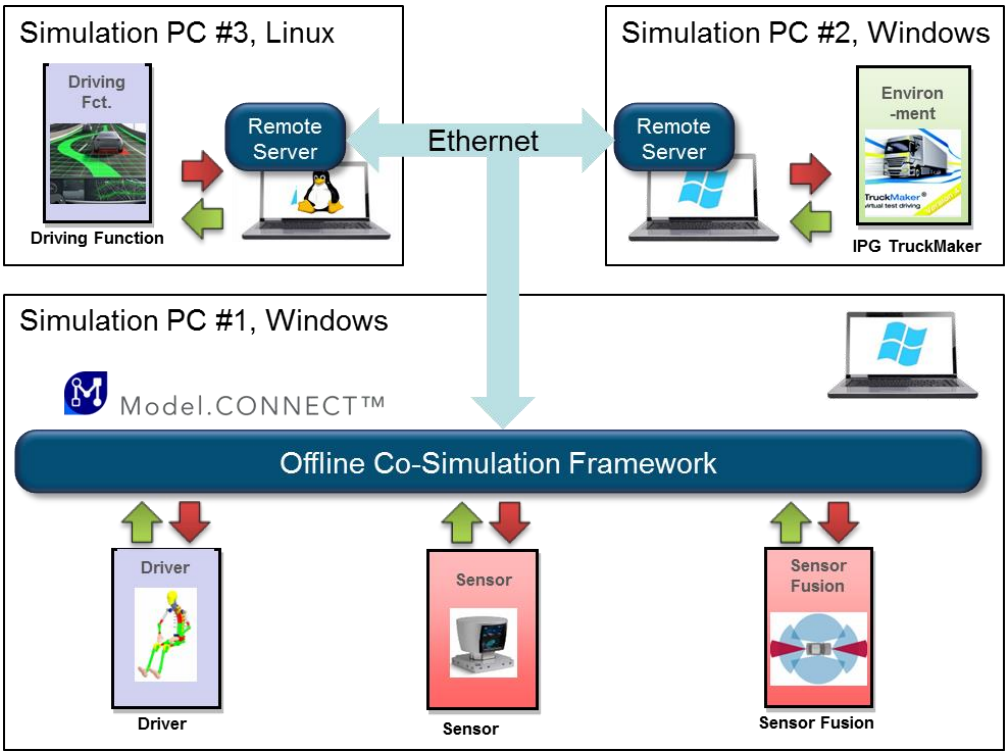

Figure 2 Distributed co-simulation

This report is part of a project that has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement No 723324.

2.2 Real-time co-simulation using Model.CONNECT™

Models in Model.CONNECT™ split in real-time and non-real-time models. Real-time systems (RT) have to satisfy the so-called hard real-time conditions (e.g. guaranteed response-time, deterministic runtime behavior). Real hardware is considered a RT system. Non-real-time systems (non-RT) in general are not required to satisfy these conditions, but for synchronization purposes often need to be executed faster than real-time [2]. Offline simulation models are considered non-RT systems.

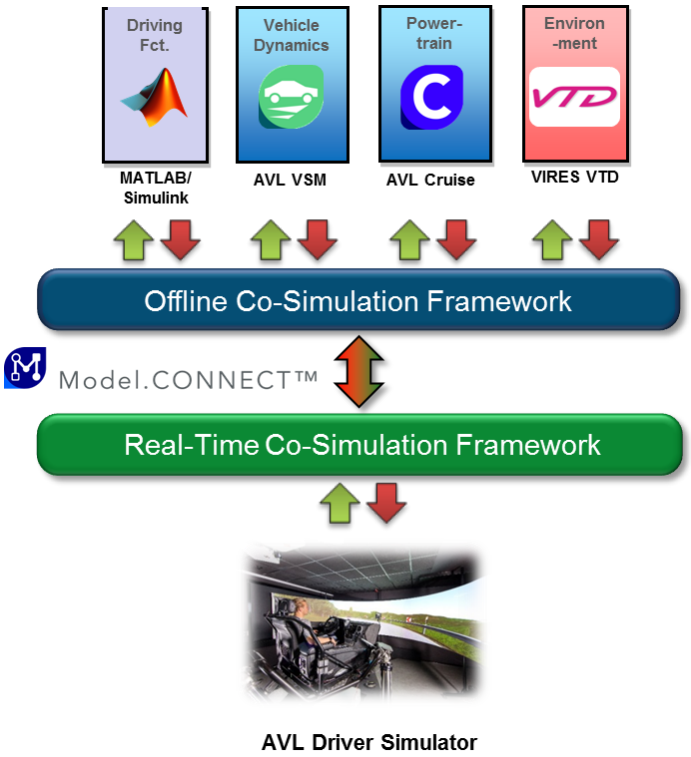

Figure 3 Exemplary real-time co-simulation setup for the AVL Driver Simulator

Figure 3 presents a real-time co-simulation setup as it could be used for example at the AVL Driver Simulator: the offline co-simulation part is depicted in the upper area of the figure and can be compared to the offline co-simulation in Figure 1. Here the simulation tools are coupled via the offline co-simulation framework of Model.CONNECT™. As mentioned above, only subsystems that can be executed faster than real-time can be used for the real-time co-simulation. The coupling of real-time systems (real hardware) via the Real-Time Co-Simulation framework of Model.CONNECT™ can be found in the lower part of the figure, where here the AVL Driver Simulator is used as an example. Special coupling- and error-correction methods have to be used in order to guarantee a smooth interaction between both co-simulation levels and to ensure 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.

This report is part of a project that has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement No 723324.

A short introduction to the AVL Driver Simulator can be found in the TrustVehicle deliverable D2.2 [3].

2.3 SHARC Framework

SHARC is a design and simulation environment for digital and analogue embedded systems. The design of (hierarchical) systems can be verified through simulation-based tests. To facilitate these simulations, SHARC provides a wide component library containing models from different domains, e.g. automotive, energy, CPS. This makes it easy to design a first system design in early development phases and eliminates the redundant design slash development costs. The behavior of the system is tested through automatically generated tests, which are derived from system requirements. The SHARC environment also includes a trace viewer to analyze the results of the simulations in a fast and easy way to derive further requirements for hardware and software design.

SHARC has a strong focus on verification of safety-critical systems in compliance with the functional safety standard ISO26262 for electric/electronic systems in the automotive domain. This includes features such as:

- Development in line with the automotive safety-lifecycle
- High traceability to requirements, design and tests
- Supporting analysis methods (FMEA, FTA) and simulation-based verification (Monte-Carlo, fault-injection)

2.3.1 Eclipse-based

SHARC is based on Eclipse, which allows easy configuration of the environment and extension through the plugin mechanism. Through relying on open standard such as SystemC for simulation, UML for design and UVM for verification, models are highly interoperable with other design and verification tools. Furthermore, it allows easy integration into existing design flows.

Support for Automotive Functional Safety – ISO26262

- Simulation-based Verification (Monte Carlo)
- Reliability Analysis (FTA, FMEA)
- Safety Requirements (Functional, Technical)
- Hardware Architecture Evaluation (Design Space Exploration)
- High Traceability from Requirements and Design to Test-results
- Generation of Safety Cases

2.3.2 Co-Simulation including Virtual Prototyping

The complexity nowadays within automotive industry increases continuously. Beside pure mechanical elements, the growing amount of assembled cyber-physical-systems, supplied from different vendors, become an unhandy issue for developers. Heterogeneity is time consuming and error prone.

This report is part of a project that has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement No 723324.

Nevertheless, the combination of electrical, mechanical and programmable system parts are responsible for innovations and compulsory to be competitive on the market.

Planning and design become therefore an essential part for the development of an entire heterogeneous system. Finding ways to establish a good design-flow for the development of new products can decrease the costs significantly. One common approach is the so-called top down approach, where utilized components can be modeled in an abstract manner first.

SystemC, a C++ library, was designed to cover these issues and allows to increase the granularity of systems continuously through the whole development process, without changing simulation nor development environments. Refinement of HW/SW system can generally be done in different ways, beginning from synchronization via communication, known as transaction-level modeling (TLM), down to cycle-accurate behavior on register-transfer level (RTL). The interaction between embedded hardware/software systems and their environment is commonly known under the term of embedded analog/mixed-signal (E-AMS) and became an essential part in the system design and simulation process. The original SystemC simulation kernel was not designed to simulate analog/continuous-time signals. The SystemC AMS (analog mixed system) was defined upon the needs of the described issues. It was built upon the SystemC standard, defined in IEEE 1666-2005 . SystemC AMS can be used for virtual prototypes, i.e. to create and simulate the entire behavior of so-called system-level models. With the UVM (Universal Verification Methodology) extension, designers have furthermore the possibility to verify their system design, through the entire development process.

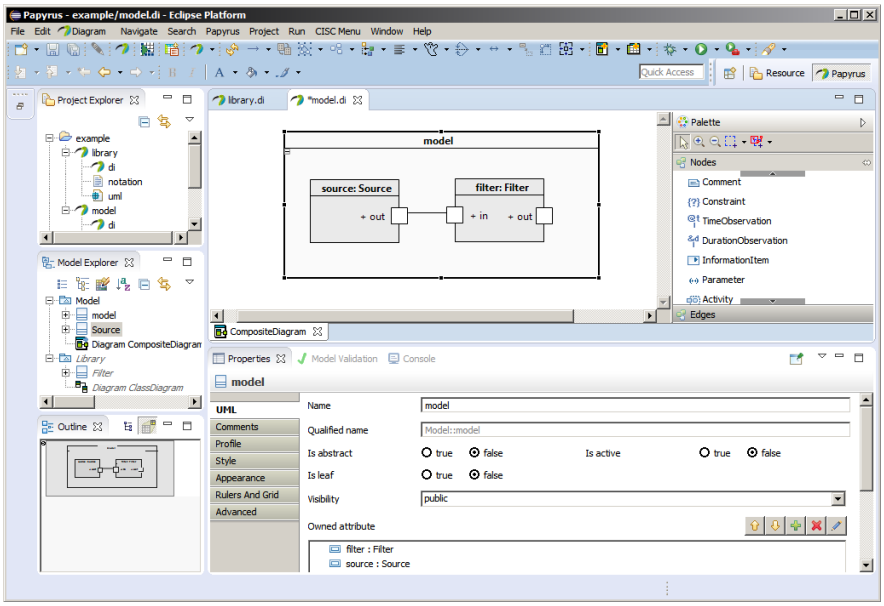

Figure 4 UML editor showing the modelling of simple system.

This report is part of a project that has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement No 723324.

2.3.3 Open Virtual Platforms (OVP)

Embedded software is often written in a desktop environment, on a general purpose operating system of the host system. This approach differs most often significantly from the target platform and parts of the written software have to be adjusted. One way to deal with this problem is the usage of an instruction set simulator (ISS) and hardware visualization. Due to a vendor variety of components within SoC design, the simulation can be very difficult. OVP can circumvent this issue, since it comes with a growing model library, which offers processor cores, memory and various peripherals. The idea is to use modular design with the combination of components in a so-called virtual platform. The generated platform can be simulated with the OVPsim API. OVP was generally founded by Imperas, its commercial brother, and can be used freely for non-commercial / academic use.

The outlined goal is to develop software within the hardware development phase, on a realistic hardware model/prototype of the target system. Hardware emulators are very popular for this issue, but require the detailed RTL description of the developed system, which is in contradiction to the outlined goal of a homogeneous development environment. OVP makes it possible to create virtual platform models, with SystemC TLM2.0 support. OVP models can be executed much faster than their according counterparts developed in RTL, since their level of abstraction is higher but still appropriate for modeling purpose.

The instruction set simulator OVPsim is released for 32 bit Windows and Linux and is a just-in-time Code morphing simulator engine, which means that the target instructions are translated to x86 host instructions. This causes a significant speed-up, since the simulation can be highly optimized thereafter.

OVPs model Generator iGen was written to build simulation models through a TCL (Tool Control Language) script, which contains used platform components and their connection. TCL files can be compared with IP-Xact hardware description, which mainly relies on the VLNV principle, a standardization of the Spirit consortium. VLNV establishes a unique identification for models, by providing the parameters Vendor, Library, Name and Version. The directory structure of the Imperas model library was designed in a similar way, such that the iGen converter, retrieves the necessary information for the instantiated models of the platform and generates a SystemC description.

2.3.4 Testbench Model in UML

The user starts by creating a graphical testbench in UML. This is done with the help of our automatic test-bench creator, built as Plugin into the Eclipse-Environment. It helps the designer to build very fast verification environments by reusing UVM verification components. The user selects the top-level DUT and an automatic verification environment is built around it by choosing components from a TestbenchWizard. This TestbenchWizard also defines the duration and time step of the simulation, as well as the number of sequences. Since there is a need to simulate different scenarios depending on

This report is part of a project that has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement No 723324.

the use-case, the user has to define a preferred amount of sequences. The utilization of these scenarios is done within an override file, which contains a specific amount of override elements. The result is a UML model that consists of the outlined components on command layer, such as drivers and monitors, and their according connections to the chosen DUT. However, from that point of view, the created test instance is only capable to simulate one certain scenario, which has been predefined through the default values in the UML description. Component-library, UML for DUT, TB and library.

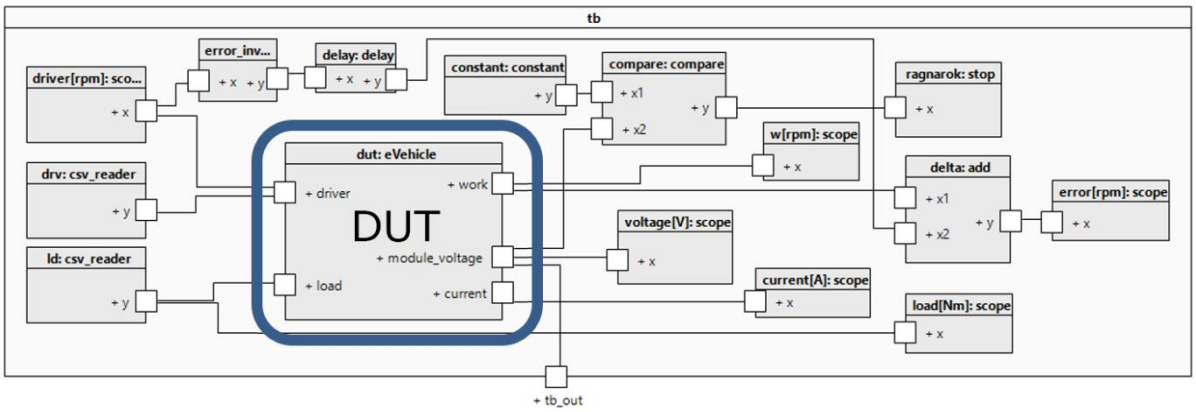

Figure 5 Automatic testbench generation around the DUT

2.3.5 Use case

The use case described here is one of an electric vehicle, where the overall system model of the eVehicle is depicted in Figure 6. This model gives an early view of the system on preAA design level with little to no assumption about the actual hardware. It is composed of the battery, controller, inverter, dc-motor and the battery management unit (BMU). The BMU is included in the battery model. The driver provides the desired speed for the eVehicle. This can be set according to standardized maneuvers such as the New European Drive Cycle (NEDC). The controller is a model for a PI state-space controller and maintains a constant speed based on the information about the state variables, motor armature current and motor-speed. The inverter model implements an inverter function for a PM-DC motor driving stage. It compares the actual battery voltage and the requested controller voltage to maintain the PM-DC motor terminal voltage. The battery model simulates the behavior of a Li-ion battery pack composed of a defined set of single cell Li-ion batteries. The appropriate number of single cells is connected in parallel and series to obtain the necessary capacity, maximum current and terminal voltage. The battery pack's terminal voltage is calculated based on the defined parameter and the battery current. A BMU is connected to the battery to measure voltage, current and temperature of the cells/modules. The BMU computes the SOC, State-Of-Health (SOH) and is responsible for cell balancing, cell protection and demand management of the battery. These computed values can then processed via a CAN controller as digital values to the power train controller. In addition, the external load environmental conditions, such as temperature, can be changed during the simulation.

This report is part of a project that has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement No 723324.

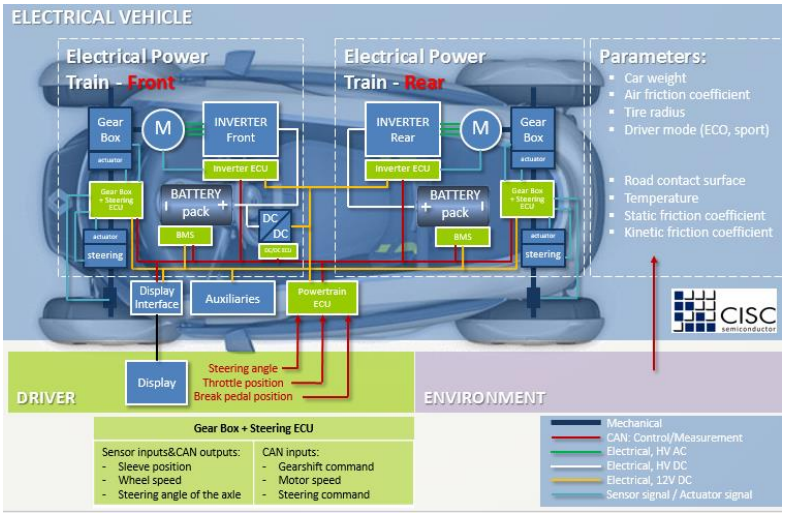

Figure 6 Electric Vehicle Co-Simulation

This report is part of a project that has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement No 723324.

3 Use case architecture

For the use cases already established in deliverable D3.1, Model.CONNECT™ is used for the co-simulation. For once using ModelCONNECT™ it is easy to exchange singular subsystems and also it is a prerequisite for the AVL driving simulator, where the TrustVehicle use cases will be tested. In order to set up the use case architecture for the co-simulation the subsystem specification sheets for each model have already been collected in a first step and presented in deliverable D3.1. In this section, the information collected in these sheets is updated and processed further so that the basis for the actual co-simulation build-up is formed. For each use case a connection matrix depicting the interfaces between the individual models is assembled. This connection matrix not only forms the basis for the co-simulation setup, but is also used to check the compatibility of the respective subsystems. Already occurring mismatches have been eradicated for this deliverable and the specification sheets have been updated accordingly. Also an updated subsystem block structure was established and can be found in each use case subsection. Further specifications of tool-solver settings are addressed.

This report is part of a project that has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement No 723324.

3.1 Ford Otosan use case

The Ford Otosan use case focuses on truck/trailer parking scenarios, especially backing to docking stations, where exact positioning is crucial, and backing into construction sites, where reliable and save motion planning and path tracking is important. The subsystem setup for pure simulation is depicted in Figure 7. The environment model as well as the vehicle model is done in TruckMaker ("Truck/Trailer & Environment"), whereas path planning ("Trajectory Planner") and the respective controller ("Trajectory Controller") are MATLAB/Simulink functions. In case of application in real world, namely using a real truck and trailer, the "Truck/Trailer & Environment" block is exchanged for an environment model provided by the respective sensors mounted on the truck and trailer as well as a truck/trailer model provided by the truck sensors.

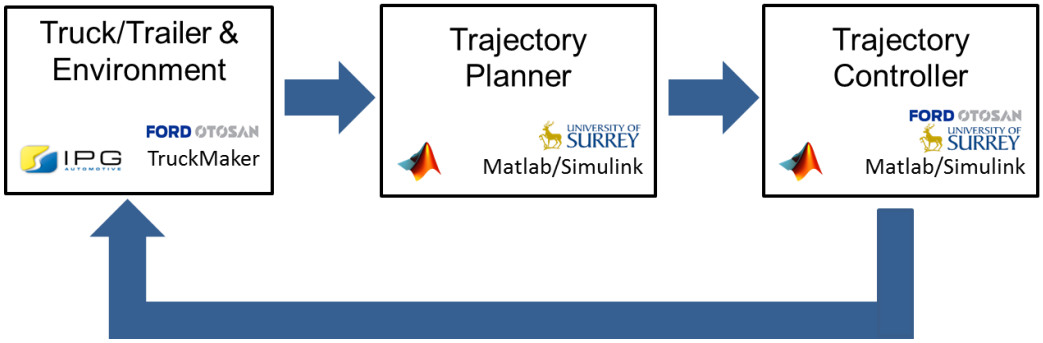

Figure 7 Ford Otosan use case simulation setup

This report is part of a project that has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement No 723324.

3.1.1 Connection matrix

A preliminary connection matrix for the Ford Otosan use case is given in Table 1. A more detailed signal list, especially concerning the information collected from the environment, will be established during the course of the project.

Table 1 Ford Otosan use case connection matrix

				Inputs: Truck/trailer Environment	Inputs: Trajectory Planner	Inputs: Trajectory Controller	Unit
Output	Truck/trailer Environment		Bounadries (Road)		X		m
			Vehicle States		X		m, m/s, rad
			States of the obstacles		X		m, rad, m/s
	Trajectory Planner		Desired Deceleration			X	m/s²
			Desired Acceleration			X	m/s²
			Desired Yaw Rate			X	rad
	Trajectory Controller		Steering Angle	X*			rad
			Braking Torque	X*			Nm
			Acceleration Torque	X*			Nm

*In case of real-world testing these signals are sent to the truck.

Signals of "Boundaries (Road)" message are:

- width of free manoeuvre space,
- length of free manoeuvre space,
- parking lane width.

Signals of "Vehicle States" message are:

This report is part of a project that has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement No 723324.

- world coordinate truck x position,
- world coordinate truck y position,
- world coordinate truck orientation,
- current speed,
- relative angle of truck-trailer combination.

Signals of “States of the obstacles” message are:

- world coordinate obstacle x position,
- world coordinate obstacle y position,
- world coordinate obstacle orientation,
- obstacle current speed.

3.1.2 Tool solver settings

Tool and solver specifics for the simulation in this use case are given in Table 2. In all subsystems there is interaction between Matlab/Simulink and IPG CarMaker. Hardware specifics are listed in

This report is part of a project that has received funding from European Union’s Horizon 2020 research and innovation programme under grant agreement No 723324.

Table 3. Information on how to integrate the trajectory planners and controllers in real hardware is already available, but truck/trailer and environment sensor specifics could not yet be given.

Table 2 Ford Otosan use case tool solver settings - Simulation

	Truck Trailer & Environment	**Trajectory Planners**	**Controlling algorithms**
Simulation tool & version	Matlab / Simulink 2016a, IPG TruckMaker 6.0	Matlab / Simulink 2016a, IPG TruckMaker 6.0	Matlab / Simulink 2016a, IPG TruckMaker 6.0
Numerical solver	Ode3	Ode45/Ode3	Ode45/Ode4
Kind of numerical step-sizes	Fixed	Fixed	Fixed
Numerical step-size	0.01s	0.01s	0.01s
Real-time capable	Yes	Yes	Yes

This report is part of a project that has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement No 723324.

Table 3 Ford Otosan use case tool solver settings - Hardware

	Truck Trailer & Environment	Trajectory Planners	Controlling algorithms
Hardware description	NA	ECU (dSpace MicroAutoBox)	ECU (dSpace MicroAutoBox)
Communication step-size	NA	fixed step-size	fixed step-size
Communication medium	NA	CAN	CAN

3.2 Tofaş use case

The Tofaş use case aims to automate door to door delivery for light commercial vehicles (LCVs). The main focus therefore is on urban scenarios, such as maneuvering in narrow streets and backing scenarios. The subsystems needed for this use case are depicted in Figure 8. The first block "Environment" integrates real-world measurements collected with a mule vehicle. The collected data is used for development and testing purposes within this use case and serves as an input for the environment simulation done in dSpace ASM simulation tool. Vehicle dynamics are added via the block "LCV Vehicle Model" and electrified powertrain of light commercial vehicle will be parameterized in the simulation tool. Motion planning for the automated LCV is done in "Trajectory Planners" and the planned trajectory is followed using the control generated in "Controlling Algorithms". The latter two subsystems are implemented using MATLAB/Simulink.

In this use case, it is planned to perform simulations on the AVL driving simulator. The "LCV Vehicle Model" as well as the "Environment" block for this purpose will be changed to an AVL VSM model, modeling the LCV vehicle, and a Vires VTD model for the environment simulation, since both are prerequisites for the used driving simulator.

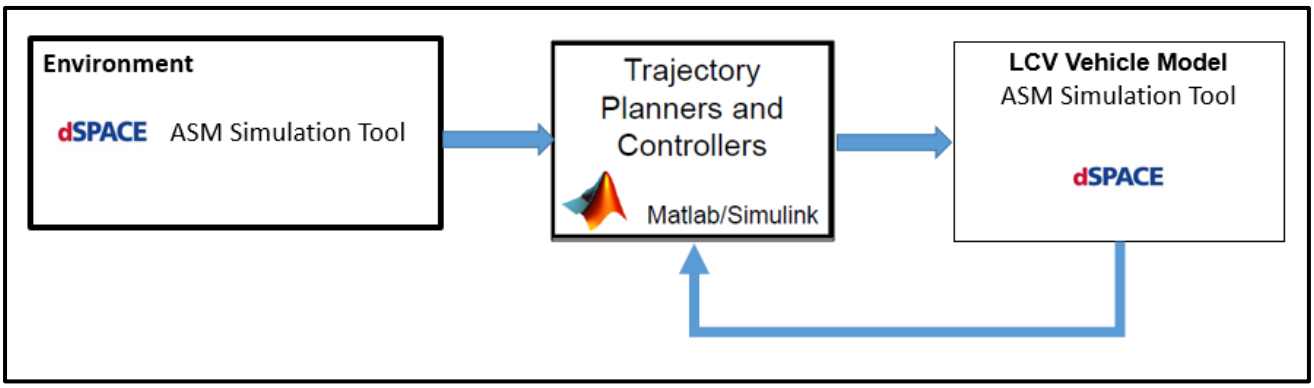

Figure 8 Tofaş use case setup

This report is part of a project that has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement No 723324.

3.2.1 Connection matrix

Table 4 shows the current version of the connection matrix for the Tofaş use case. As in the Ford Otosan use case, the subsystems are still in development and interfaces will be updated during the course of the project.

Table 4 Tofaş use case connection matrix

			Inputs: Environment	Trajectory Planning	Controlling Algorithms	LCV Dynamics	Unit
Outputs: Environment		Object lists from sensors		X			
		Ego vehicle position		X			
		Environmental Map (Real or Simulation Based)		X			
Outputs: Trajectory Planning		Automated system status			X		-
		Lateral offset request			X		m
		Vehicle speed			X		m/s
		Obstacle detected			X		-
		distance to target vehicle			X		m
		look-ahead distance (LAD)			X		m
		lateral offset at LAD			X		m
		angular deviation at LAD			X		rad
		road curvature at LAD			X		1/m
		Target vs Ego vehicle velocity difference			X		m/s
		Max acc @longitudinal			X		m/s^2
		Min acc @longitudinal			X		m/s^2

This report is part of a project that has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement No 723324.

		desired time gap to target vehicle			X		s
		desired ego velocity			X		m/s
		max. ego velocity			X		m/s
		actual ego velocity			X		m/s
		minimum cleareance to target			X		m
		Steering Wheel Angle			X		deg
Controlling Algorithms		Shifter lever				X	-
		upshift request				X	-
		downshift request				X	-
		Throttle				X	%
		Brake Force				X	Nm
		Steering Wheel Angle				X	deg
		actual lateral offset				X	m
		desired longitudinal acceleration				X	m/s^2
		actual angular deviation				X	rad
LCV Dynamics		Longitudinal acceleration	X	X			m/s^2
		Vehicle Speed	X	X			m/s
		Vehicle position	X	X			m
		Vehicle orientation	X	X			deg
		wheel travel	X	X			mm
		wheel speed	X	X			rpm
		e-motor speed	X	X			rpm

This report is part of a project that has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement No 723324.

3.2.2 Tool solver settings

The specified tools and respective solver settings are listed in Table 5. As mentioned above, measurements for the setup of the environment are collected using a mule vehicle that will be built up during the project runtime. Other than that, to this point there is no hardware information available since the use case is still in the early development phase, therefore hardware settings for this use case are neglected within this deliverable.

Table 5 Tofaş use case tool solver settings

	Environment	**Trajectory Planners**	**Controlling algorithms**	**LCV Dynamics**
Simulation tool & version	dSpace ASM	Matlab / Simulink	Matlab / Simulink 2017a	dSpace ASM
Numerical solver	Euler	Ode3/ode45	Ode3/ode45	Euler
Kind of numerical step-sizes	Fixed	Fixed	Fixed	Fixed
Numerical step-size	0.01s	0.01s	0.01s	0.01s
Real-time capable	Yes	Yes	Yes	Yes

This report is part of a project that has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement No 723324.

3.3 Volvo use case

In this use case the advantages of a sensor monitoring system shall be illustrated. The aim is to increase the availability of L3AD functions by closely monitoring the sensor availabilities and reacting accordingly and by that increase the acceptance of the user. The use case subsystem setup is depicted in Figure 9. This use case will first be performed on the driving simulator. Therefore the environment and sensor information will be provided by Vires VTD, which is the tool for environment simulation on the AVL driving simulator. Sensor fusion and the used advanced driving functions are provided in a fused Matlab/Simulink block and will further be referred to "AD functions". Sensor monitoring will either be provided in Matlab/Simulink or C++. For the first test on the driving simulator this subsystem will not yet be used since it is still in development. In this case the subsystem can just be disregarded and the use case setup without the sensor monitoring can be used as a reference for the later tests. In order to be able to inform the user on the driving simulator about possible take-over processes, the AD functions are connected to a simple HMI, which will be displayed to the user on a tablet PC. Also specific AD functions can be chosen by the driver input via HMI. Vehicle dynamics on the driving simulator are simulated using AVL VSM, which represents the last subsystem in this use case.

In the simulation performed on the driving simulator the sensor fusion and automated driving functions will be provided by VIF. Both will for the real world testing in the Volvo XC90 demonstrator vehicle in workpackage 6 be replaced by Volvo internal functions. Obviously the environment will then be provided by environment sensors mounted on the vehicle and the vehicle dynamics are coming from the vehicle itself.

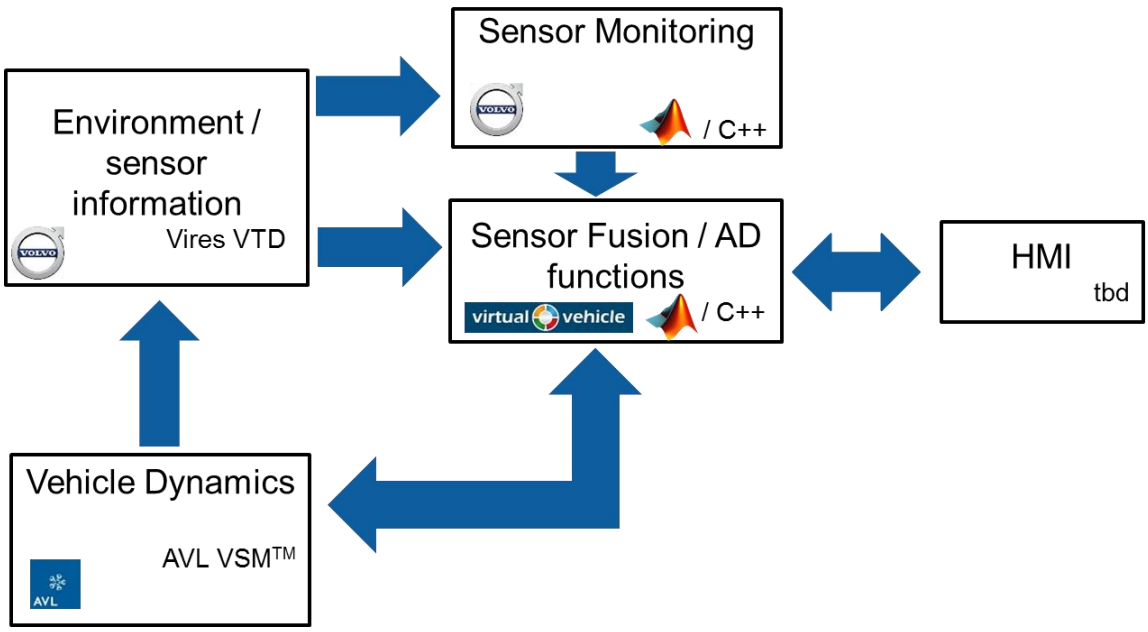

Figure 9 Volvo use case simulation setup

This report is part of a project that has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement No 723324.

3.3.1 Connection matrix

In Table 6 a preliminary connection matrix for the Volvo simulation use case can be found. In- and outputs for the sensor monitoring system are still vague since the development process is still in its beginnings.

Table 6 Volvo use case connection matrix

			Inputs					
			Environment Simulation	Sensor Monitoring	AD functions	HMI	Vehicle dynamics	Unit
Outputs	Environment Simulation		desired lane			X			-
			lane polynomial coefficients			X			m
			lane polynomial domain			X			m
		Camera	Liquid occulsion		X				-
			Solid occlusion		X				-
			Low light		X				-
			Blooming		X				-
			Reduced visibility		X				-
		Lidar	Liquid occulsion		X				-
			Solid occlusion		X				-
			Reduced visibility		X				-
		Radar	Solid occlusion		X				-
			Reduced visibility		X				-
			Interference		X				-
		US	Occlusion		X				-

This report is part of a project that has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement No 723324.

		Reduced visibility		X				-
		GPS/Cloud - Signal loss		X				-
		extreme temperature		X				-
	sensor i	sensor timestamp			X			ms
		isActive			X			-
		long. dist. ego-object			X			ms
		lat. dist. ego-object			X			ms
		rel. velocity longitudinal			X			m/s
		rel. velocity lateral			X			m/s
		lat. position of object			X			m
		object width			X			m
Sensor Monitoring		Trusted detection area front short sensor i			X			-
		Trusted detection area front long sensor i			X			-
		Trusted detection area rear short sensor i			X			-
		Trusted detection area rear long sensor i			X			-
		trusted detection area right side short sensor i			X			-
		Trusted detection area right side long sensor i			X			-
		Trusted detection area left side short sensor i			X			-
		Trusted detection area left side long sensor i			X			-
		trusted localization data			X			-
		critical road - construction			X			-
		critical road - low friction			X			-
AD function		desired ego velocity				X		m/s
		desired time gap				X		s
		suggested lane for ego veh.				X		-

This report is part of a project that has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement No 723324.

		enabled AD function				X		-
		is lateral tracking enabled					X	-
		SW torque					X	Nm
		is long. tracking enabled					X	-
		vx					X	
		brake					X	-
		gas					X	-
HMI		desired time gap			X			s
		desired ego velocity			X			m/s
		AD flag enabled			X			-
Vehicle Dynamics		vehicle rotational speeds	X					
		Vehicle position	X					
		vehicle orientation	X					
		wheel travel	X					
		wheel speeds	X					
		engine/motor speed	X					
		tyre forces	X					
		turn signal			X			-
		brake			X			-
		gas			X			-
		steering wheel angle			X			rad
		sw angular velocity			X			rad/s
		sw torque			X			Nm
		v			X			m/s
	>	vx			X			m/s

This report is part of a project that has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement No 723324.

		vy			X			m/s
		vz						m/s
		yaw rate			X			rad/s
		yaw rate measuring time			X			
	a	ax			X			m/s²
		ay			X			m/s²
		az						m/s²

3.3.2 Tool solver settings

Table 7 and Table 8 show the simulation and hardware specifics for the Volvo use case. As mentioned above the HMI will be shown on a tablet PC for the driving simulator. Information for this subsystem is not available at the moment and will be neglected at this point. Hardware specifics for sensor monitoring and AD functions are related to real world testing and are not relevant for the driving simulator tests.

Table 7 Volvo use case tool solver settings - Simulation

	Environment	**Sensor Monitoring**	**AD Functions**	**HMI**	**Vehicle Dynamics**
Simulation tool & version	Vires VTD	Matlab / Simulink, C/C++	Matlab / Simulink 2012a	NA	AVL VSM
Numerical solver	Ode3	NA	Ode3	NA	Euler
Kind of numerical step-sizes	Fixed	NA	Fixed	NA	fixed
Numerical step-size	1ms	NA	1ms	NA	0.0005s
Real-time capable	Yes	NA	Yes	NA	yes

This report is part of a project that has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement No 723324.

Table 8 Volvo use case tool solver settings - Hardware

	Environment	Sensor Monitoring	AD Functions	HMI	Vehicle Dynamics
Hardware description	NA	ECU	dSpace MicroAutoBox	Tablet PC	AVL VSM
Communication step-size	NA	NA	fixed step-size	NA	fixed step-size
Communication medium	NA	UDP, CAN, Ethernet	CAN	NA	UDP

This report is part of a project that has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement No 723324.

4 Conclusion

In this deliverable the foundation for the co-simulation setup is outlined. First the basics for co-simulation in general are given, using examples possible in the TrustVehicle context. Also, a second co-simulation framework (SHARC) further developed within the TrustVehicle project is presented. Using this knowledge and the specifications for each subsystem in the respective TrustVehicle use case, a modular architecture for each of these use cases is set up. A connection matrix depicts the interfaces between the singular subsystems and finally tool and solver settings are given as far as they are known at this point.

Especially the depiction of the use case architectures and the connection matrices proved to be useful in verifying the feasibility of the planned use case setup. Missing connections could be created and the needed interfaces and signals could be provided in order to ascertain an executable use case, in simulation as well as on the driving simulator and in real world conditions.

Next steps will be the adaption and collection of the respective models and the actual setup of the co-simulation framework for the use cases.

This report is part of a project that has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement No 723324.

5 References

[1] „Model.CONNECT™ User Manual“.

[2] G. Stettinger, M. Benedikt, N. Thek und J. Zehetner, „On the difficulities of real-time co-simulation,“ in *V International Conference on Computational Methods for Coupled Problems in Science and Engineering, COUPLED PROBLEMS 2013*, Ibiza, Spain, 2013.

[3] M. Troglia Gamba , „Specification of Traffic Scenarios and Questionnaires,“ TrustVehicle Deliverable, 2017.

This report is part of a project that has received funding from European Union’s Horizon 2020 research and innovation programme under grant agreement No 723324.