**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**

# Deliverable 3.4

# Final co-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.

# DOCUMENT INFORMATION

| Deliverable 3.4 - Final co-simulation setup | Confidential |
|---|---|
| **Editors** | Bernhard Hillbrand, Pamela Innerwinkler, Lisa-Marie Schicker (VIF) |
| **Authors** | Bernhard Hillbrand (VIF), Sajin Gopi (AVL), Ralph Weissnegger (CISC), Ersun Sozen (FO), Ahu Ece Hartavi Karci (US), Ali Demir (TF), Gülhan Sinem Alanya (TF), Kemal Rodoplu (TF), Talat Büyükakın (TF) |
| **Responsible person** | Bernhard Hillbrand (VIF) |
| **Nature** | Report |
| **Status** | Final |

Not yet approved by the EC.



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.

## CHANGE HISTORY

| Version | Date | Description | Issued by |
|---|---|---|---|
| 1.0 | 30.07.2019 | Initial Version | VIF |
| 1.1 | 19.04.2019 | Input section 3.1 | AVL |
| 1.2 | 16.05.2019 | Input section 2.1 and 3.4 | VIF |
| 1.3 | 21.05.2019 | Input section 2.3 | CISC |
| 1.4 | 28.05.2019 | Input section 2.2 and 3.2 | AVL, FO, US |
| 1.5 | 30.05.2019 | Input section 3.3 | TF |
| 1.6 | 11.06.2019 | Include changes from review | VIF |
| 1.7 | 18.06.2019 | Include changes in section 3.3 | TF |
| 1.8 | 30.07.2019 | Include changes in section 3.2 | FO |

## REVIEWERS

| Version | Date | Reviewer |
|---|---|---|
| 1.5 | 10.06.2019 | Ralph Weissnegger |
| | | |
| | | |
| | | |



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.

# Content





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





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 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

This deliverable concludes the build-up of the co-simulation models within the TrustVehicle project. Based on the connection matrices and information on tools and settings developed and collected within the first half of the project, detailed information about the use case setups, especially for simulation and the usage on the driving simulator are given.

The report gives a good overview of the co-simulation platforms SHARC and Model.CONNECT as well as some specifics. It also introduces the actual setup and usage of the co-simulation models. Also, additional information on the used software and hardware is given. The last part is about the use cases of the project.

**Key Words**

Automated Driving, Co-Simulation, AVL Model.CONNECT, SHARC, 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.

# 1 Introduction

## 1.1 Overview of the Report

This deliverable gives an overview of the co-simulation techniques used within the TrustVehicle project, as well as details on setup and usage. Therefore, manuals for the two co-simulation platforms already introduced briefly in the previous deliverables (D3.2 and D3.3) are given. This should help users to get an idea of the platform and help them start working with them. The two co-simulation platforms are Model.CONNECT$^{TM}$ and SHARC. Model.CONNECT is also the platform that was used in the first driving simulator study at AVL in the summer of 2018. Outcomes of this study are descripted in the deliverable D5.1.

The main part of the report is about the three use cases of Ford Otosan (FO), Tofaş (TF) and Volvo (VCC). The FO use case contains a truck-trailer combination that should be parked at the dedicated parking slot in the construction site within an environment with no obstacles (a), static obstacles (b) and dynamic obstacles (c) and in addition to that, disturbances and environmental effects can be caused by the site itself. In the second FO use case, the construction site is replaced by a docking park. There will also be the scenarios with no obstacles (a), static obstacles (b) and dynamic obstacles (c).

The TF use case is about automated door to door delivery with a light-duty vehicle. The scenarios will contain backward and forward approaching to the delivery point and fine-tuned low speed manoeuvres for narrow street delivery duties in urban mixed traffic. Hand over situations will be implemented for high density urban traffic scenarios.

In the VCC use case the vehicle has to check the availability of the sensors for AD and if necessary, reduce the AD system availability, especially in harsh weather conditions.

## 1.2 Structure of the report

In the following section 2 of the report, a short introduction about co-simulation is provided. This leads to a manual for the use of the two available co-simulation platforms in this project, AVL Model.CONNECT$^{TM}$ and the SHARC framework. Section 3 starts with a description of the co-simulation setup of the first driving simulator study and continues with the use cases of Ford Otosan, Tofaş and Volvo.



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 General overview on co-simulation platform and generated framework

## 2.1 Introduction to Co-Simulation platforms

In the past few decades, numerous specific simulation tools have been established in the automotive industry. However, these tools typically specialize on individual areas of expertise. There is very limited support for a heterogeneous simulation environment. The goal of "co-simulation" is to overcome this limitation and to merge the challenges from different areas. Common co-simulation platforms are often limited to a single area of expertise (e.g. the design of a thermal management system with a heterogeneous tool landscape). Such platforms typically address problems from a very specific, restricted dynamic range, which means that the different models exhibit similar time behavior.

However, the development of modern, mechatronic systems requires a much broader approach. The interactions between sub-systems from different areas have to be taken into account through a suitable interconnection of the parts. The coupling of existing (specific) simulation programs (and the models implemented therein) from different areas of expertise represents a promising approach for the simulation of the complete system.

With the introduction of co-simulation in the development process the task of developing complex mechatronic systems can be solved in a very efficient way. For example, the integration of Finite Element Methods (FEM) is supported for applications in the field of "Integral Safety". Another example is the integration of electrical and thermal components into existing drivetrain concepts in the field of "alternative drives", such as hybrid or electric vehicles. In all of these considerations the vehicle itself is not the exclusive focus, but rather the interactions with the vehicle's environment and the influence of the real driver are taken into account.

The task of a co-simulation platform is to take the complex interactions of the various simulation models in a suitable and correct way into account. The platform has to enable the precise co-working of different simulation tools. Two co-simulation platforms are listed in this report.

## 2.2 Model.CONNECT User Manual

### 2.2.1 Introduction

Model.CONNECT is a model integration and co-simulation platform, connecting virtual and real components. Models can be integrated based on standardized interfaces (Functional Mockup Interface - FMI) as well as based on specific interfaces to a wide range of well-known simulation tools. 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. Through the integration with multiple model execution environments, Model.CONNECT supports the continuous functional integration scenarios that form the basis of model-based development processes in particular in the automotive industry. Model.CONNECT features powerful model parametrization and batch simulation capabilities, simulation online



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.

monitoring, result analysis, and reporting functionalities. Interfaces to various optimization tools enable design studies and optimizations.

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

- Model integration based on models that are provided as executable libraries (FMU for Co-Simulation or Model Exchange). Such model configurations can be executed in one process. Such closely coupled model configurations are prepared for execution on real-time operating systems with Model.CONNECT.
- 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. The modular architecture provides the possibility for an iterative model developing process. Furthermore, influences of model accuracy, model depth, and non-linearities on the result can be determined, due to the cross-domain considerations.

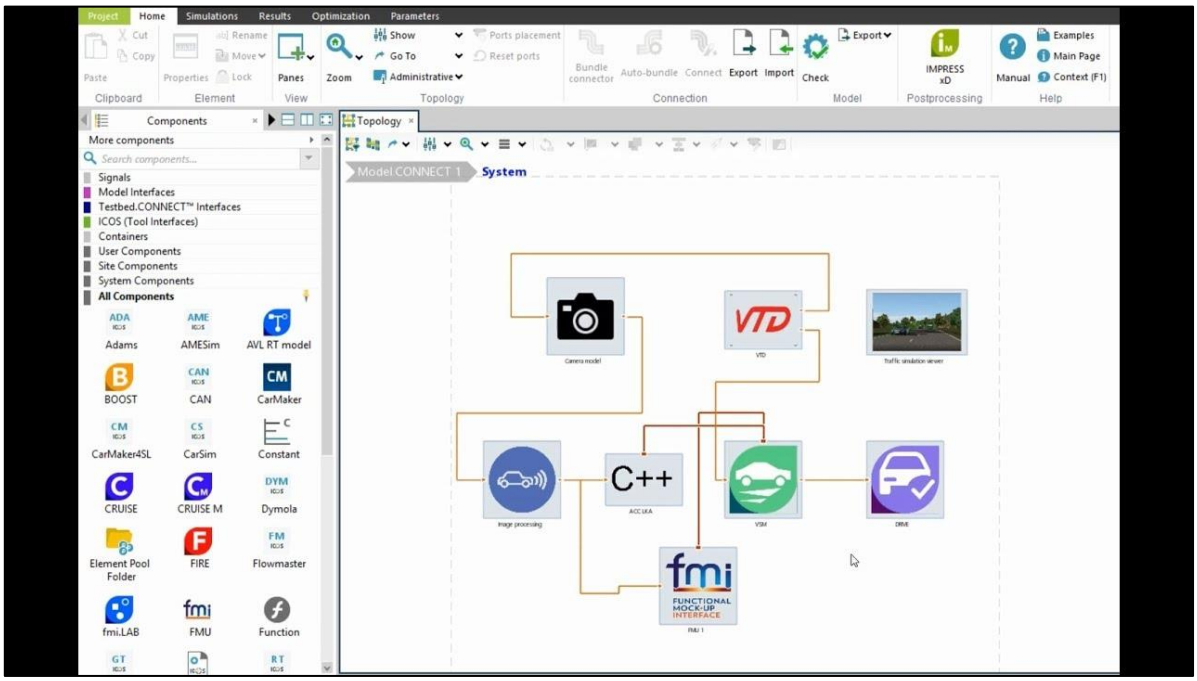


**Figure 1: AVL Co-simulation platform - Model.CONNECT**



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.

### 2.2.2 Model.CONNECT workflow

The following steps are necessary to perform a co-simulation:

1. AUTHORING

The individual sub-models of the co-simulation model can be created by various tools:

- FMUs can be created by any tool with an FMU-export capability.
- MATLAB/Simulink models can be exported by AVL fmi.LAB.
- Direct model interfaces to certain AVL tools are available: AVL VSM,AVL CRUISE.
- Direct tool interface coupling is possible by using the ICOS tool-coupling components.

2. CONNECTING

- Add the individual elements to the co-simulation model.
- Establish the connections between the elements of the model

3. EXECUTING

Depending on the actual configuration of the co-simulation, various simulation tools run to simulate the whole model:

- In-Process:

  All elements of the simulation are loaded as shared objects and run within one process. Hence, it is necessary that all binaries of the elements of the model are available for the same platform and bitness (32bit/64bit).

- Multi-process:

  ICOS components are running as individual processes. Hence, the bitness of the individual (32bit/64bit) binaries may differ.

- Distributed Multi-Process:

  Since ICOS components are running as individual processes, they can be started and run on different machines.

- The Online-Monitor can be used to inspect simulation values during simulation of the co-simulation model.

4. POSTPROCESSING

After the simulation the results can be inspected or used for further processing steps:

- Visualize with Results Tab
- Visualize with Concerto



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.

### 2.2.3 Model.CONNECT Configuration

General Model.CONNECT GUI information for layout, modelling modes are described in below section.

**Home Tab:**

The individual sub-models are integrated here and connected to each other for co-simulation. While working on a model user can inadvertently create situations which cannot be handled by the solver, such as invalid connections or uninitialized elements. For these cases Model.CONNECT provides model checking functionality which is invoked either manually ('Check' ribbon button in the 'Model' ribbon group), or immediately before a simulation run. These checks can be divided into two categories: model-level checks (connections, illegal component combinations, etc.) and component-specific checks.

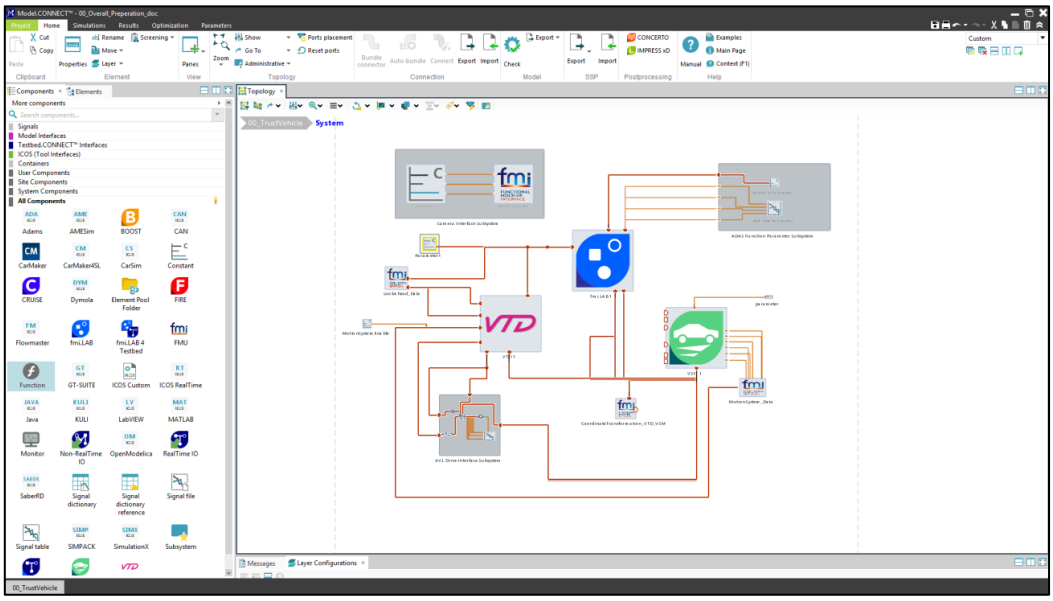

**Figure 2: Model.CONNECT Configuration for Driving Simulator**



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.

**Simulation Tab:**

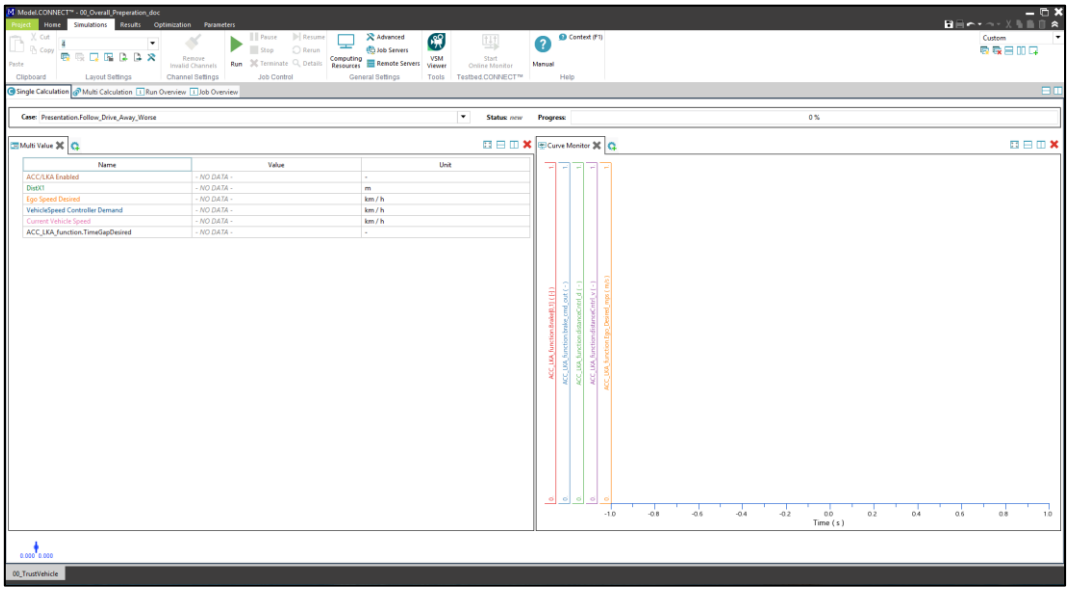

**Figure 3: Simulation Tab**

The Simulation tab used to start and monitor all simulations. The user can start a single simulation, or a multi-case simulation run from the Simulation tab. The user can simulate the prepared model here by pressing the Run button in Single Calculation tab. It also provides the facility to monitor all the simulated signals online while the simulation is running.

Figure 3 shows the simulation tab and the case 'Follow_Drive_Away_Worse' is selected from 'Presentation' case set for simulation. In Figure 4, all the available online monitors are shown.

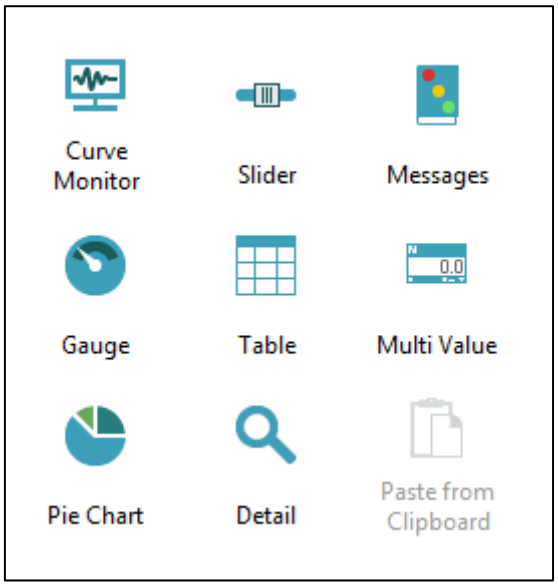


**Figure 4: Online Monitor**



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.

In the Simulation tab, the user can simulate multiple cases or case sets in Multicalculation tab. Figure 5 shows the Case sets on the left and the Cases for selected case set(s) on the right. To show the information regarding the selected case in the Job Overview pane, double click on the row number in the Cases list. This view shows all simulations started from the Run Simulation dialog or directly from the Single Calculation pane. Double click on the row number to automatically open the Job Overview pane to show all tasks and their attributes for the selected simulation run.

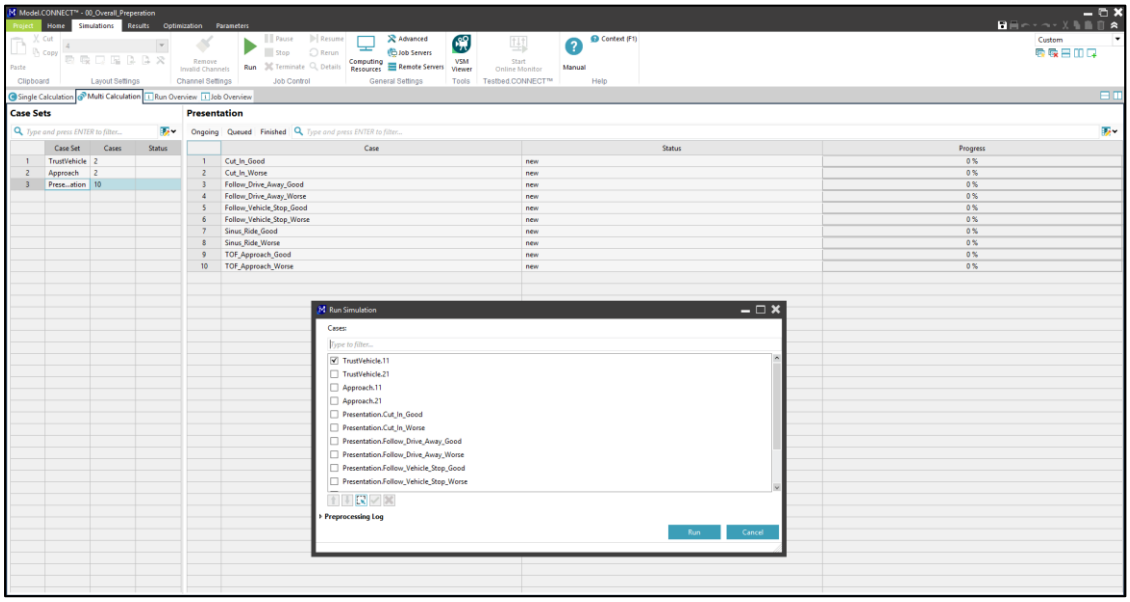


**Figure 5: Case set Definition**

**Parameter Tab:**

The Parameter tab contains the Data Pool, Parameters, Cases and Scenarios ribbon tools. The Parameters ribbon tab is used to handle several key features of the application:

*Project parameters:*

The Parameters pane contains a parameter tree and a parameters table. They are used to manage the model parameters. Element properties are defined in the property dialogs. For more efficient control over the property values, values can be parametrized. Parameters are used in simulation cases. Each simulation case can have a different parameter value. This way the user can optimize his model by varying the parameter value.



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.

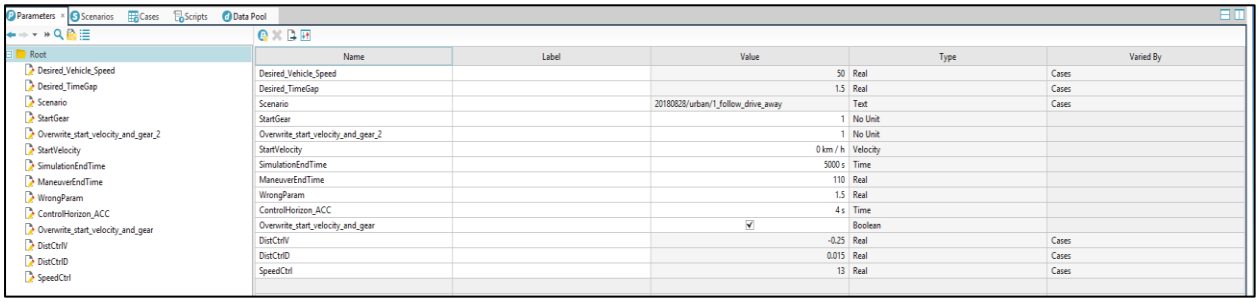


**Figure 6: Parameter definition for Driving Simulator**

Already defined parameters are managed using the parameter pane in the parameter ribbon tab.

The user can create a new parameter and assign it to a certain property, or they can reuse an already existing one. The assigned parameter is shown in the parameter state. The Parameter state is indicated by the equal sign and the parameter name in the appropriate box.

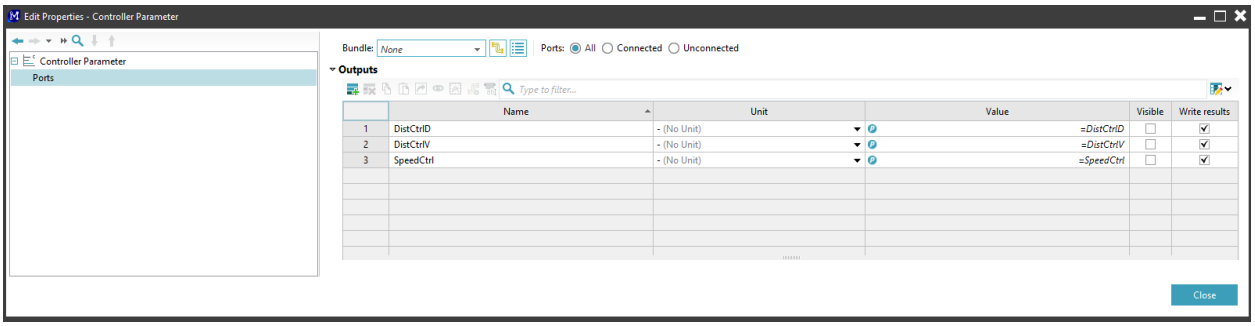


**Figure 7: Parameter Assigning**

*Project Cases:*

Parameters can be varied in each simulation run. These runs are called cases. Every case is basically a new simulation run with a new set of parameter values. In order for cases to have any meaning the user must define some parameters. After parameters are defined, they are added to the cases and varied in the same case set. Users can have multiple case sets defined, each of which can use different parameters in it. Cases are used to automatically run the simulation multiple times and have different parameter values defined each time



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.

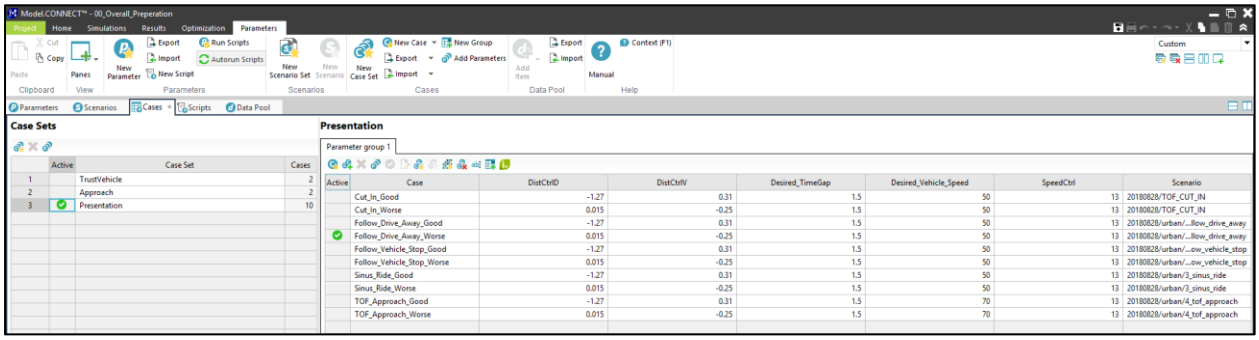


**Figure 8: Case set definition**

In the driving simulator study, different cases are defined with different controller parameter as shown in Figure 8. This provides very easy access to the user to select the predefined cases during simulation and switches in between if required.

**Results Tab:**

All the inputs and outputs of the subsystems are stored in Model.CONNECT project folder in csv format. This signal can be monitor and analysis after simulation. Figure 9 shows the result tab in Model.CONNECT.

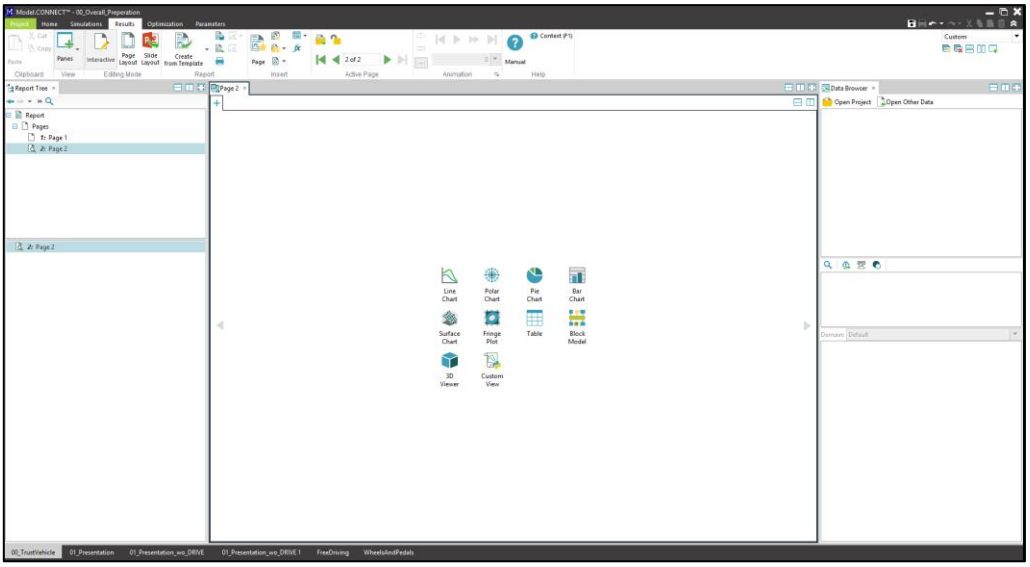

**Figure 9: Result Tab**

The Data Viewer is the main working area where charts and 3D views are presented. The Data viewer pane can be divided into multiple smaller panes (the "sections") using split buttons. Usually there is one view (chart or a 3D view) in each section. More views can be added to a section by dragging them between sections. If a section contains more than one views, then they can be shown by clicking their tabs.



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 SHARC User Manual

Since especially the automotive industry has to cope with increasing requirements - due to bigger product line - and shortened time to market, one seeks for solutions to decrease the complexity of the development phase. One way to accomplish this is the reuse of the system components, and the development on a higher abstraction level.

The approach of SHARC [9] is founded on the Unified Modeling Languages UML [3] and a standardized extension, to model embedded systems, named MARTE [4] . Individual components and communication channels can be outlined in graphically manner, which ultimately leads to a modular representation of an overall system design. The functionality of individual core components is generated by the system description language SystemC [1] , whereby the behavior of the designed system can be simulated, after parsing a given graphical description consisting of those pre-developed SHARC core components. The usage of SystemC has the main advantage that the granularity can be increased without switching to other programming language, since SystemC aims to implement Hardware description languages like VHDL or Verilog.

Verification is one of the most essential concepts in the overall design process. Since many (in)famous examples have shown that redesigns or, even worse, call-backs cause high costs, system designers have investigated many efforts to avoid faulty designs. All in all, it is essential to start system verification in early phases and reuse them through the whole development. One common way to verify components is the simulation-based approach, with the aid to use concepts from the Universal Verification Methodology (UVM). UVMs core functionalities make it possible, to generate test environment that allows a prediction, which parts of the design can be verified from a given test scenarios. This approach is available with the usage of the so-called constrained random stimulus principle. For the simulation-based verification of the electrical vehicle model this method is used to generate possible driving scenarios, which lead to different test inputs for the Device under Test (DuT). In the end it should be possible to get an overall verification, through the simulation, since corner cases can be obtained in a satisfying way.

### 2.3.1 Objectives

The developed SHARC IDE relies on the basic Eclipse framework and is meant to become a platform, which integrates the above described issues. The developer should get the possibility to defined SystemC components as well as using them in within the Eclipse UML editor Papyrus.

Beneath many free available extensions for the basic framework, the Eclipse framework can be enhanced easily in any reasonable direction through the plug-in mechanism. The stated objective of this work is therefore to implement a simulation-based approach to verify designs within a distributed environment. Whereby the methodology should rely on standards, such as the previous described Universal Verification Methodology. The main step towards this direction is the development of a graphical user interface, with the aim to create test instances programmatically and distribute them, if desired. Nevertheless, the whole platform always offers space for improvement when it comes to terms



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.

of usability. Therefore, a further goal was to deploy those graphical descriptions of the predefined core components in a centralized way and provide them for the user in a feasible manner.

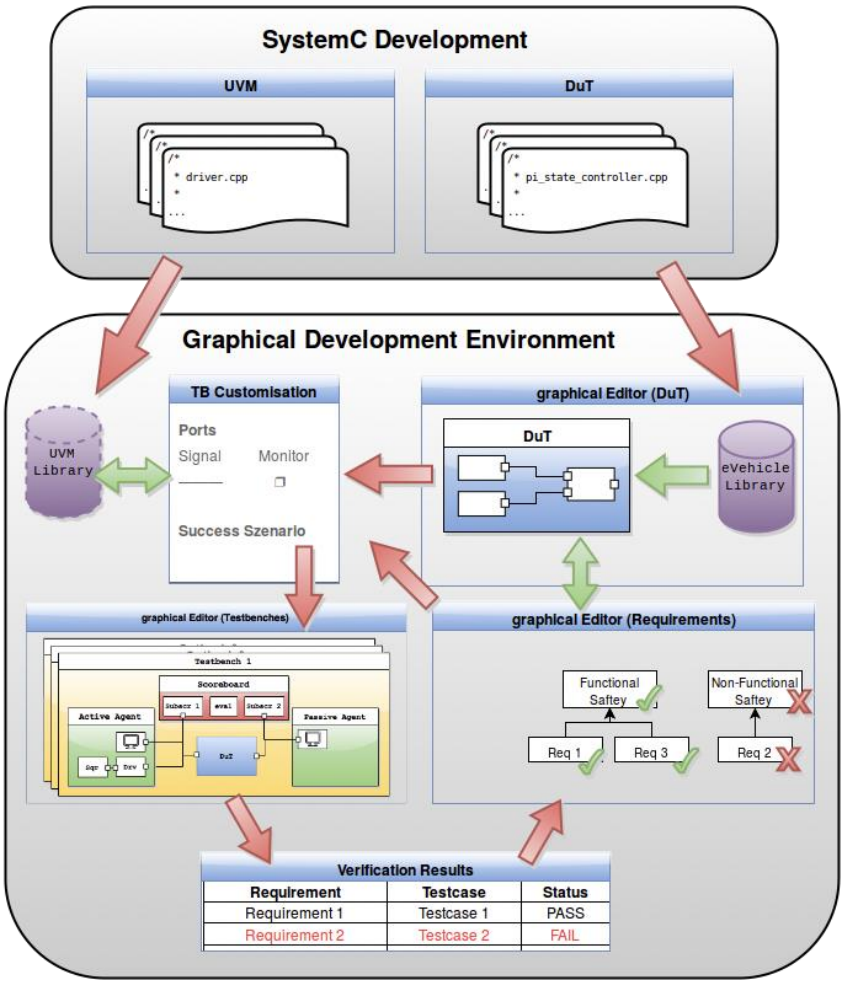


**Figure 10: Work-flow for developers using the SHARC platform.**

### 2.3.2 Model library

The CISC-SHARC Palette is one important tool, when it comes to terms of usability. The underlying simulation core has the aim to connect predefined SystemC components according to a UML description. One typical example how such a designed system might look alike can be obtained from the electrical vehicle example in figure Figure 11.

The connection between different components is established by so-called FlowPorts, which are part of the Papyrus MARTE [4] library. In general, FlowPorts distinguish between three directions: in, out and inout, whereby only the first two are supported by the simulation tool and can be parsed correctly. As mentioned, it's the UML description which provides the system behavior and therefore those



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.

available SHARC core components have to be accessible for the system designer in the graphical Editor as well.

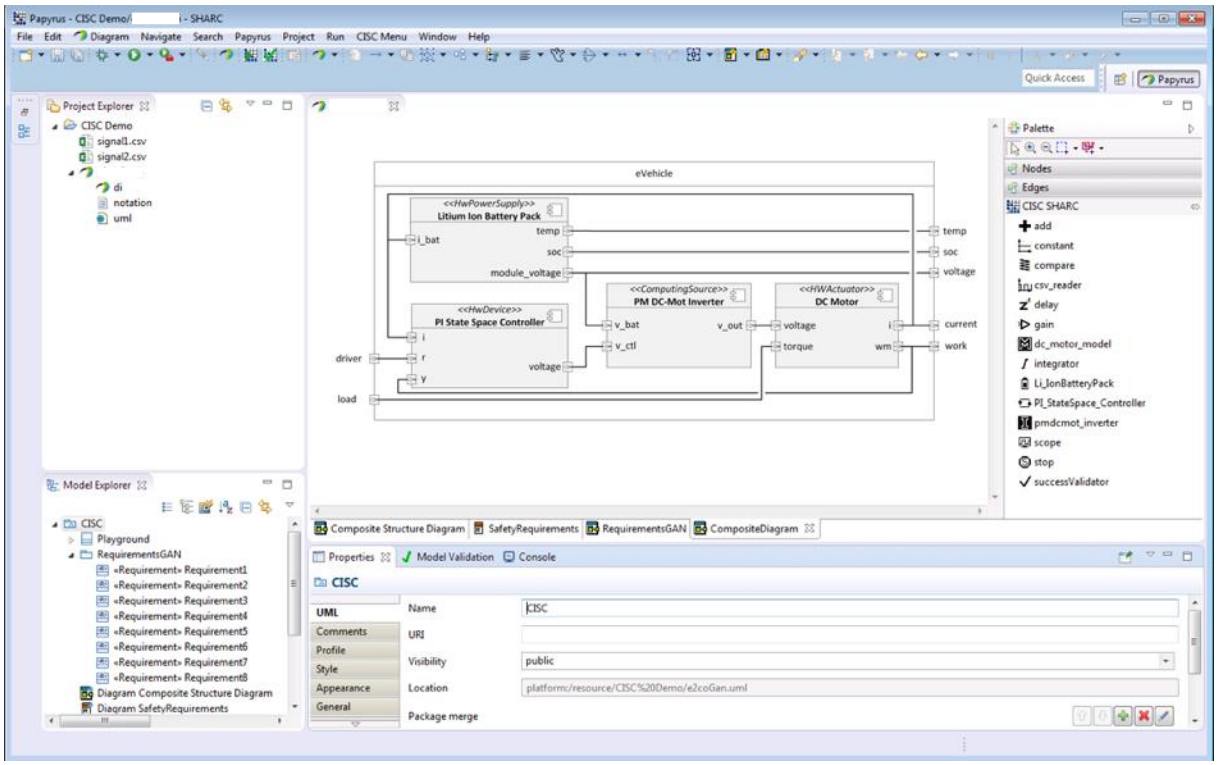


**Figure 11: Simple graphical model of an electric vehicle, where system components are plugged together via connectors.**

Inverter and battery are implemented as parts, whereas controller and motor are subsystems referring to other imported parts.

### 2.3.3 Requirements

In addition to the structure and behavior, also the requirements for a given system are crucial. Therefore, SysML enables the textual description of requirements within a diagram. By definition, they are not subject to further restrictions and only consist of a text and a unique ID. Depending on the application, requirements can be extended manually, through the UML profile mechanism.

In context of the SHARC platform, requirements for a designed system should be used in terms of verification, which means that the test instance is mainly created with respect to predefined constraints.

### 2.3.4 Modeling language

The context, in which SHARC project was launched is the development of cyber-physical-system components in the automotive sector. Since almost every mechanical component in a car is supported by embedded electronics, the outlined aim was to find ways to model these components in terms of real-time requirements and performance. The choice fell on MARTE, which is also an extended library



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.

of UML2 and implements the stated requirements of above. MARTE is an acronym for Modeling and Analysis of Real Time and Embedded systems and was also standardized by OMG.

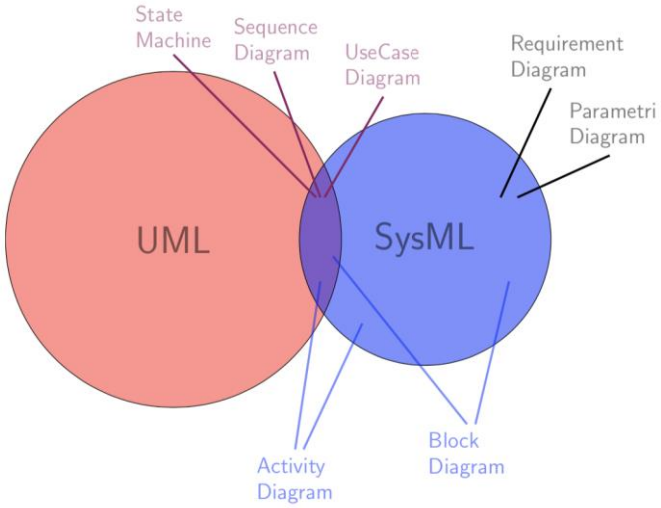


**Figure 12: Distinction between SysML und UML and their similarities [Bal07].**

### 2.3.5 Simulation language

SystemC is used as system description language for SHARC to implements the functionality of predefined SHARC core components, which can be used in the UML editor thereafter. The work on SystemC was coordinated by the Open SystemC Initiative and was a collaboration of several renowned electronics and semiconductor manufacturers. SystemC has its own IEEE standard since 2005 and is now supported by the Accellera Systems Initiative [2] . Because of this standardization, SystemC has gained more and more importance over the last decade. One of the main advantages is the widespread use of the C programming language in the area of “low level” software. Thereby developers do not have to learn a new syntax. Furthermore, the simulation of SystemC models does not need extra tools [7] .

### 2.3.6 Cloud-based Verification

Beneath the actual system description, verification is a one of the most essential aspect during any phase of the system design process. A very common way in terms of hardware development is the simulation-based approach, with the aid to use standards defined by the Universal Verification Methodology.

UVM was originally created for SystemVerilog which describes components on Register- Transfer-Level. Due to the explained issues in the design phases, a work group of different semiconductor manufacturers was initiated to realize the UVM approaches for SystemC/C++. UVM defines the utilization of the DuT by varying input signals through ports. A concrete test stimulus is defined as so-



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.

called sequence and is more or less the description of a certain test input. As mentioned above, the verification is done by the simulation of various independent scenarios. This issue definitely allows the idea of parallel processing on different CPUs in a distributed environment. For this purpose, a cloud-based concept was elaborated.

UVM provides standardized components, which must be used to build a test instance. The structural architecture is explained in [1] and introduces the layer concepts of the UVM architecture. Nevertheless, these components do not provide the parallel simulation of test sequences in a cloud-based system, which is clearly in conflict to the outlined project goal. Furthermore, UVM for SystemC was under construction during the implementation phase and did not provide a stable version [8] .

### 2.3.7 Simulation

The cisc.simulation plug-in provides a graphical user interface, which establishes multiple possibilities to customize under which conditions a simulation can be started. The simulation interface can be called in three different ways, defined in the plugin.xml. The CiscMenu, extends the ordinary Eclipse menu and includes an according procedure call. Additional there is an icon at the Eclipse toolbar and thirdly it is also callable through the short-cut CTRL+1. Each option triggers the OpenSimulationCommand, which instantiates a jface.Wizard SimulationWizard. The usage of a wizard can be reasoned in terms of usability, since it provides a user-friendly interface. The interaction is established through three WizardPages (Figure 14).

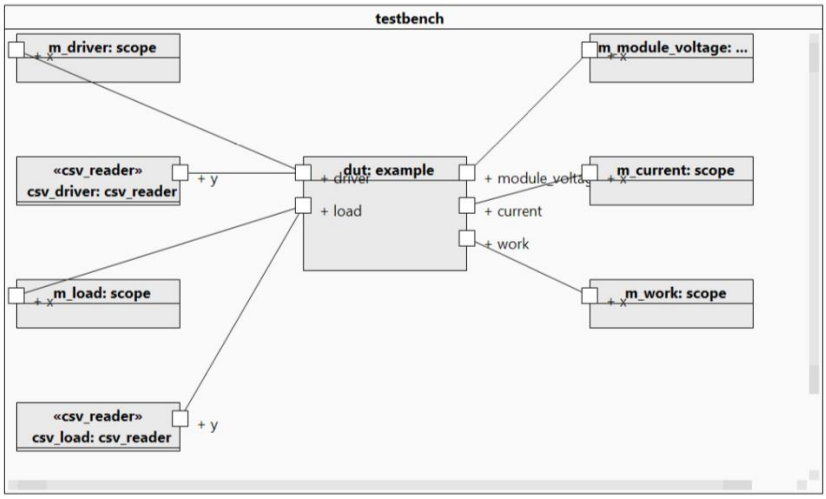


**Figure 13: Example testbench model**

The WizardPageSimulationConfig Page establishes a distinction between an online and offline mode, as well as how the information of the design can be retrieved. Nevertheless, each option leads to the same way to start the simulation itself, namely providing necessary resp. dependent files, as well as a configuration file which contains the information of the selected root model, duration, timestep and



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.

options to show the output. Figure 4.10 outlines the overview of the configuration, with various given options and the additional error reasoning mechanism.

To increase usability, a build in functionality of the IEclipsePreferences was implemented in the SimulationPreferenceManger class. The main idea is to store previous selections, to provide them for the user in case of rerunning certain simulations.

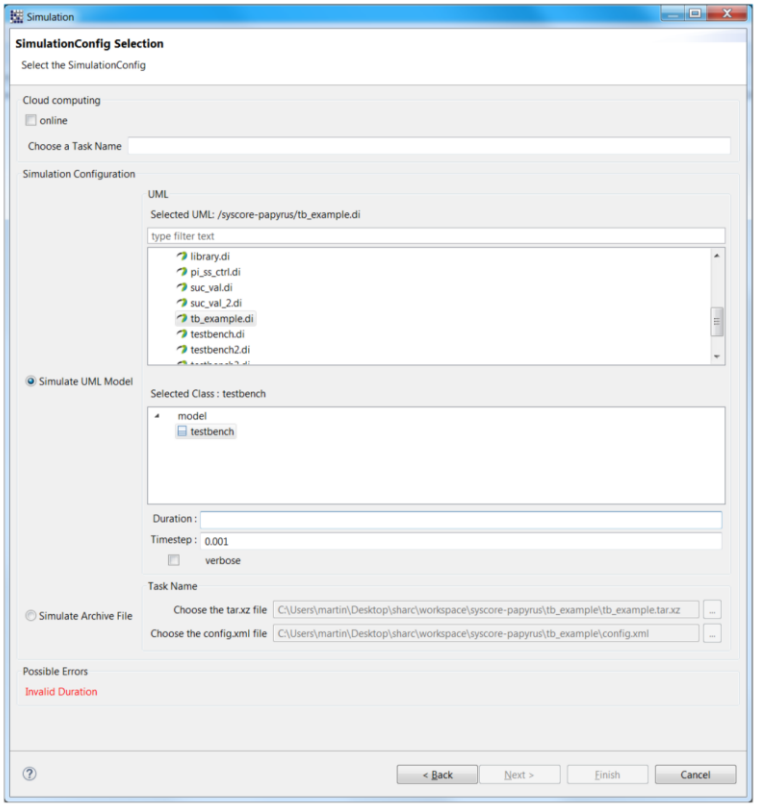


**Figure 14: The simulation can either be done in the cloud or locally. Furthermore, the user has the option to select either an UML model or an archive file.**

### 2.3.8 Archiv

For the simulation itself it might be obvious to provide the UML file of the developed system. Nevertheless, the design most often consists of various subsystems, which lead to dependencies between different elements. Therefore, it is important to provide all referenced UML files, which are relevant for the simulation. In that sense, it becomes necessary to step through the models recursively and retrieve class and file dependencies. One example can be found in the cisc.testbench/ResourceManager.java class.

Beside those UML relations, there might be some further references to .csv files, which have to be available for the simulation as well. Considering both issues, it makes sense to bundle those files in an archive tar.xz file from which they are extracted in case of an execution. Although the simulation can



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.

be executed locally in the directory of the root UML model without any further packaging, it becomes necessary in case of a cloud simulation. From that point it might be very helpful if certain simulations can be rerun once they have been executed. In that case the user can provide a certain configuration and the archive file, containing all relevant files.

### 2.3.9 Local Simulation

Referring to the previous sections, the user has the option to configure the simulation within the user interface. One can either select an UML Model from the workspace and the class which should be simulated, or simply provide an archive and configuration file. For the first option, additional information concerning duration, timestep and verbose mode has to be entered within this user interface. From that point, the configuration file is generated programmatically, and the stated model is analyzed recursively, such that all file dependencies are resolved and an archive tar.xz file can be created. Both files are written to a newly created directory in the workspace, which has the name of the selected model. The second option is straight forward and allows rerunning previous simulations.

Each local execution is done in the systems /temp directory, where the archive file gets decompressed in the SharcSimulation folder. After the successful execution of the simulation the resulting .tab files, are copied back to the origin directory of the provided archive file, from which they can be viewed with the impulse plug-in.

The according WizardPageOfflineSimulation Page provides a monitor, which shows the console log output of the simulation. This feature allows the user to observe uncertain behaviors and failures in the design. Figure 15 shows one example for the simulation of a certain scenario. The verbose mode provides additional debug information from the parsing process of the stated UML files.



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.

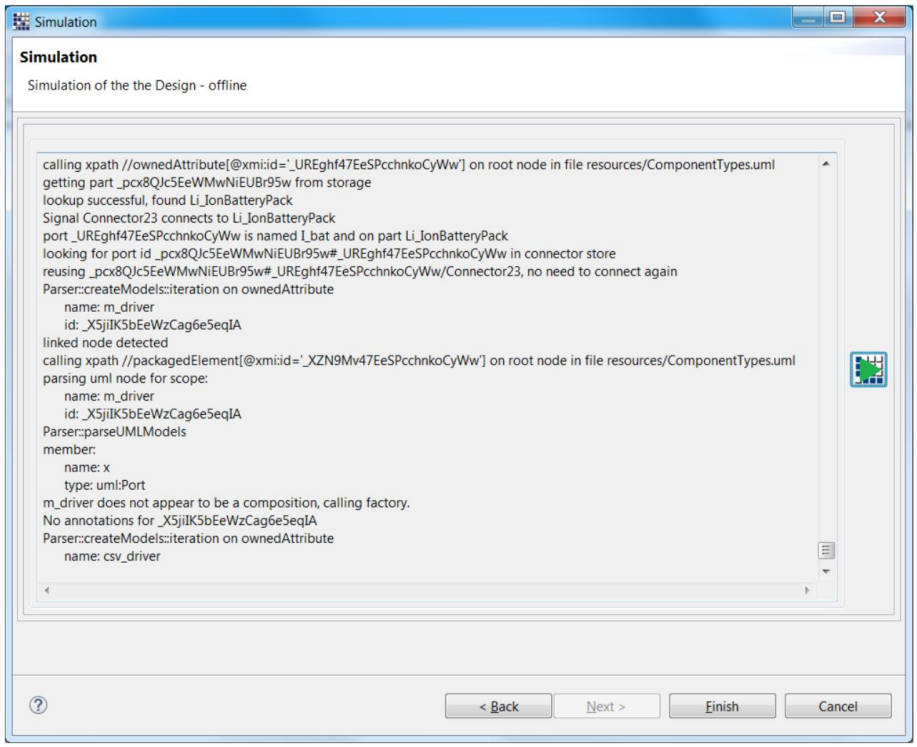


**Figure 15: The WizardPageOfflineSimulation monitors the parsing process for the given UML model. The verbose mode is on, which means that debug messages are printed as well.**

### 2.3.10 Cloud Simulation

Additionally, the graphical user interface provides further options for the simulation in the cloud. This approach was mainly developed to gain a speed-up for the system verification within the testbench simulation. This can be reasoned with possibility to process various subtasks parallel, since no synchronization has to be made between single simulations.

To establish a web configuration, the user has to provide two URLs. The broker is used to trigger the execution on its distributed worker instances within the cluster. This is done by forwarding configuration files and providing a task id, which refers to the files that are needed to process their given jobs. The web server is mainly used to provide information about the existing tasks and simulations results. Before the simulation can be executed, a task has to be created to which the simulation belongs. Therefore, the user has to specify the necessary URLs to establish cloud simulations in the OnlineConfigDialog can be seen in Figure 16.



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.

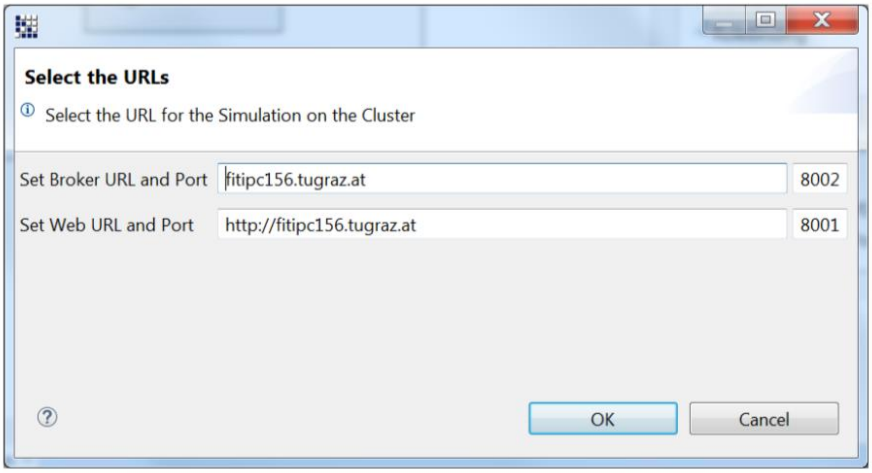


**Figure 16: The OnlineConfigDialog for the cluster configuration is the user interface, to provide the URLs for simulation in the cloud.**

### 2.3.11 Webinterface

As mentioned above, the core idea is to establish many different scenarios, to verify a certain system. From that point the underlying design should remain the same and only certain, specified parameters should change between each simulation. Therefore, the cloud-based approach was designed in such a way, that each simulation has an underlying task definition, where necessary system components are located in an archive file. The task itself has a unique id and a name. To fulfill these requirements, the user must provide a valid task name when he is doing a simulation in the cloud; the id is created by the server afterwards. To obtain validity of the task name a HTTP request of Content-Type ”application/json“ is sent to the web server with the defined user input. In case of a JSON response, the provided name is already assigned to another task and must be redefined.

The online simulation can only be achieved by providing an archive and a corresponding configuration file. Whereby the configuration file is more or less some kind of template, that can be extended in case of a simulating various scenarios resp. a testbench. Therefore, the user has to provide an override file, which is automatically generated within the testbench generation.

The specified archive file resp. compressed stream gets uploaded to the web server via http post request. Afterwards the task is created on the server with the given task name, the response includes the task id, which is necessary to assign the later sent configurations to the created task. The representative parts of this procedure can be obtained from the runTarXZSimulationOnline() method of the SimulationWizard class in the cisc.simulation plug-in.

To invoke the simulations, the distribution of the configurations to different worker instances has to be established. This is done in the createConfigsAndSendToServer() method. Therefore, a queue is filled with the configurations, which are extended with respect to the override file. The according WizardPageOnlineSimulation Page consists of two parts. The described selection for the override file is in the first group. The second Group provides an additional view of the web interface. From that



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.

point, the simulation progress can easily be seen from the user interface, without switching to a web browser.

### 2.3.12 Testbench creation

This section primarily deals with programmatically creation of a graphical UML testbench. For the main architectural design of testbenches, which obtains a link to the Universal Verification Methodology. Nevertheless, the reader should get a further short introduction to the conceptual ideas for the testbench simulation life cycle. This can be important for users, verifying their designs, on the one hand as well as for developers, who need to get familiar with the current state of the implementation.

The testbench creation for a certain DuT can be called from three different points, which are defined in the plugin.xml. Firstly, the user has the possibility to use the CiscMenu, which extends the ordinary Eclipse menu. On the other hand, there is one additional way to call it via icon at the Eclipse toolbar and thirdly through the short- cut CTRL+2. Each of these opportunities leads to the call of the OpenTestbenchCommand, which instantiates the jface.Wizard TestbenchWizard. The interaction is established through two WizardPages, which are shown in Figure 17 and Figure 18.

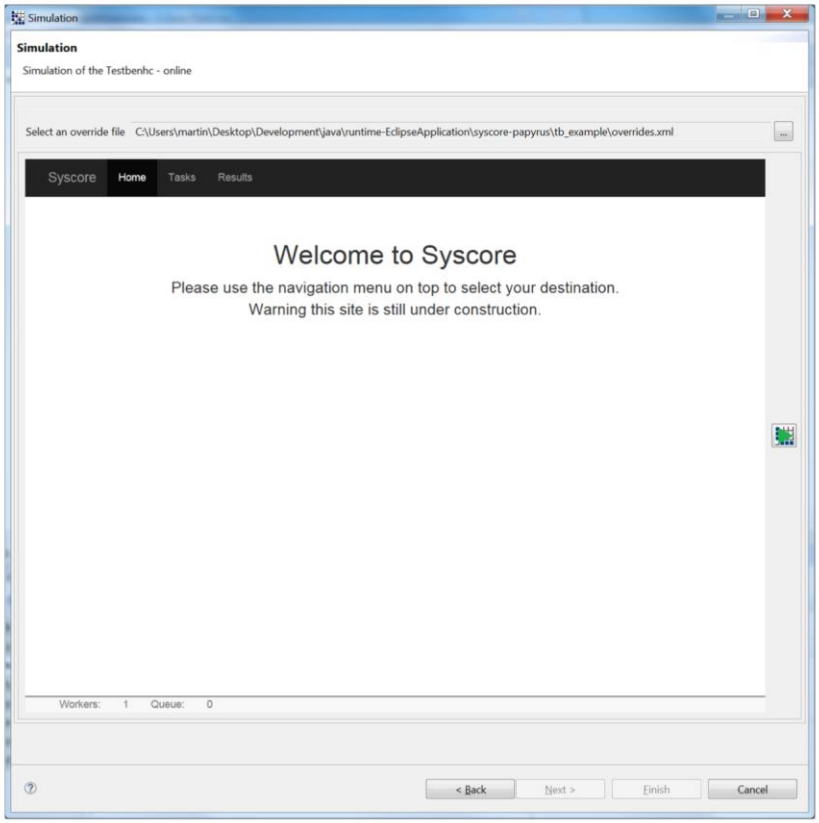


**Figure 17: The WizardPageOnlineSimulation Page provides a SWT browser widget. The user is able to explore the web interface in the same way as in an ordinary web browser.**



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.

### 2.3.13 Configuration template

The initial step for the testbench creation is implemented in the WizardPageFileSelection Page shown in Figure 17. Therefore, the user has to select a DuT from the available models of the workspace. Since models may contain multiple classes, the designer has to select the actual DuT class as well. Additionally, duration and timestep have to be defined. This step can be seen as creating a template configuration file for the testbench simulation. Therefore, the WizardPageFileSelection can be compared with the WizardPageSimulationConfig, outlined in Figure 18. Since there is a need to simulate different scenarios, the user has to define a preferred amount of sequences, which are later generated randomly according to the given user input within the WizardPageCreateTestbench Page. The utilization of these scenarios is done within an override file, which contains an according amount of override elements.

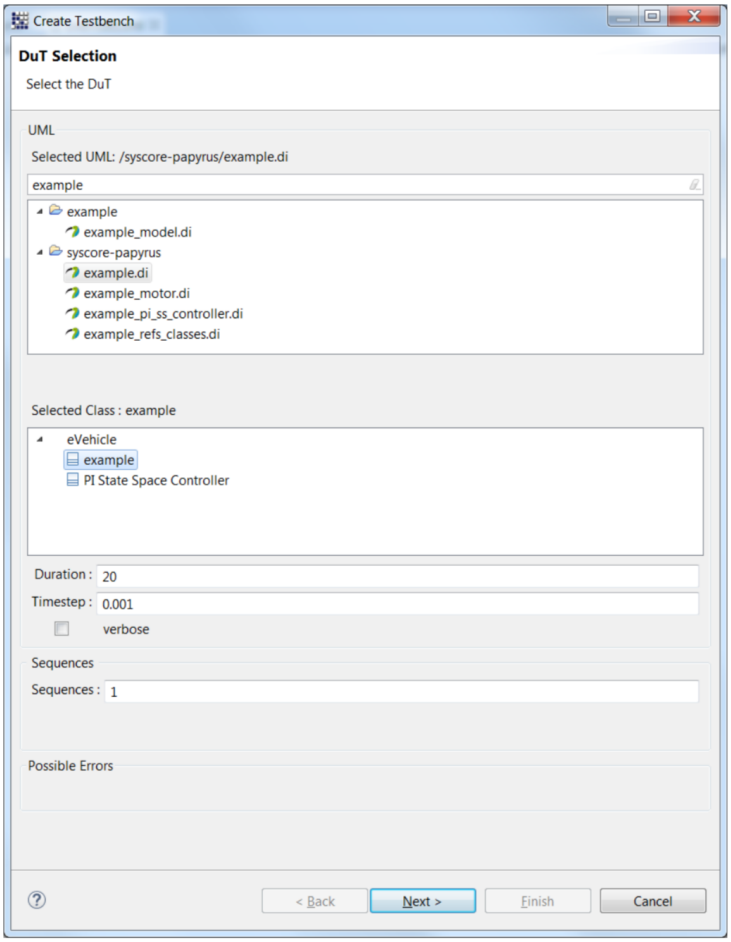


**Figure 18: The WizardPageFileSelection Page outlines the workspace in a TreeViewer. The user has to select a DuT class as well as defining timestep, duration, verbose mode and the number of sequences which should be created.**



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.

### 2.3.14 Testbench Wizard

After pressing the finish button the generateUMLTestbench() procedure is called, which firstly generates a new model consisting of the .di .notation and .uml files. The new model is named like the stated DuT file, with tb_ as prefix. The next step is the programmatically creation of an UML Composite Structure Diagram as well as changing the editor perspective to the new diagram. This is important, since the "Transactional Editing Domain", which delivers the context for any graphical insertion, can only be retrieved from an open editor. The instantiation of any UML element is done in the CreateCompositeDiagram ElementsCommand class. The first element which has to be instantiated is a basic class, named testbench, where all properties and connectors can be instantiated at. The test-bench itself needs the defined DuT class, which is instantiated as property. Moreover, for each port of the DuT, independent of its direction, one scope is instantiated, which monitors the corresponding port signal. For the driving purpose, each input port gets an additional csv reader. The filename field for the reader remains empty since those filename values get parametrized through the override file. Depending whether the user has provided an OutPortObservation, a SuccessValidtor is instantiated, with an additional constant property, defining the tolerance for the deviation as well as a csv reader defining the reference signal. As already explained in the section of the GMFNotation, the creation of a property is always accompanied by improving the default appearance with TestbenchNotation class. An example for the created UML testbench model can be obtained from Figure 19.



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.

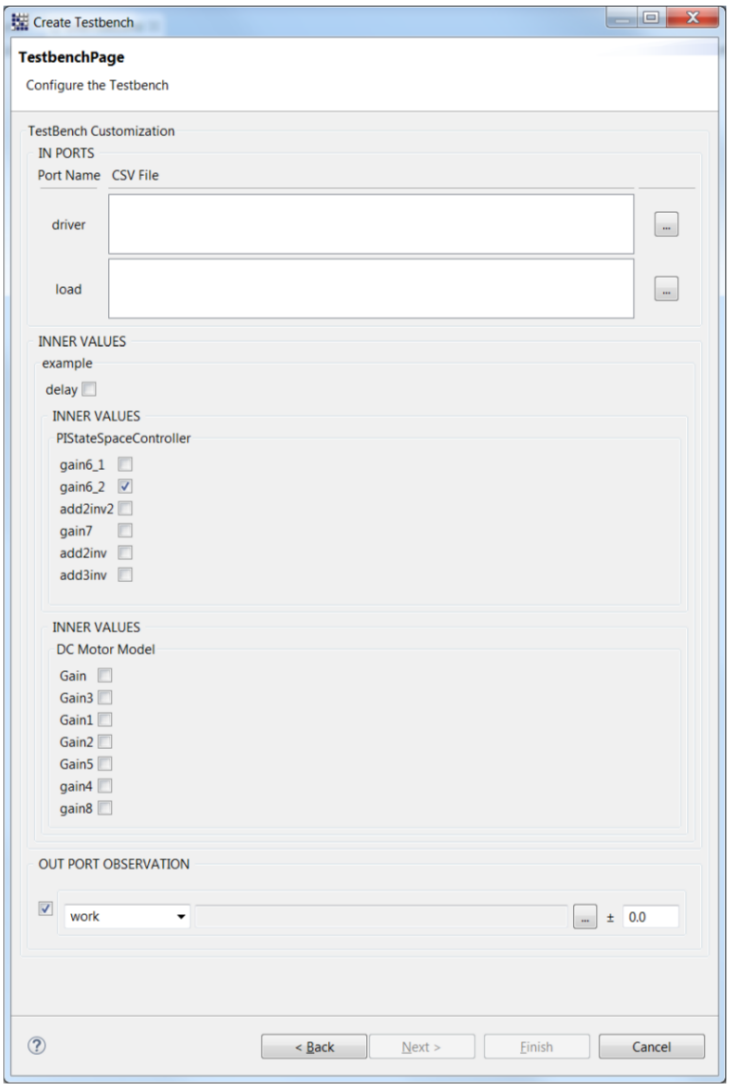


**Figure 19: Example view of WizardPageCreateTestbench for the testbench utilization of the eVehicle. Each FlowPort needs at least one signal, which drives the DuT.**

The created UML model contains all user provided information and has more or less, a satisfying appearance. Similar to the local simulation of a given UML model, the necessary UML and .csv files get compressed within an archive file. Afterwards the configuration template is created. The override file is created with respect to the stated sequence number, in such a way that for each varied value a random value is created, which is written to the override element. This is done in a loop with the range of the stated sequence number. All three files are stored within a directory named like the testbench model. If the directory already exists, the files are simply overridden.

The created UML testbench model can now be simulated in different ways. The first and most common way should be the simulation in the cloud, where the user can provide the created archive file, the configuration template and the override file. For a local simulation the user has the ability to modify



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.

either the configuration file, by extending the template with a certain override element. In that case the user can use the archive and the extended configuration file. The second possibility would be the modification of the filename values in the graphical model, since they contain empty strings by default. From that point the user can choose the UML simulation. In that case the user has to take into account, that the previous files in the directory are overwritten, which means that it might come to uncertain behaviors, since .csv files stated in the override files might be missing in the newly created archive file.



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 Implementation of co-simulation framework

In this chapter the co-simulation setup of the first driving simulator study is shown. Then the focus is on the three use cases of Ford Otosan, Tofaş and Volvo.

## 3.1 Co-simulation setup from 1st driving simulator study

### 3.1.1 General Architecture:

The modular co-simulation framework, proposed in Work Package 3, is the common validation architecture for testing all the Use Cases in simulation as well as on the Driving-Simulator platform. In the Driver-in-Loop validation the virtual driver in the co-simulation framework is replaced with the Human Driver. The Proposed architecture is shown in Figure 1.

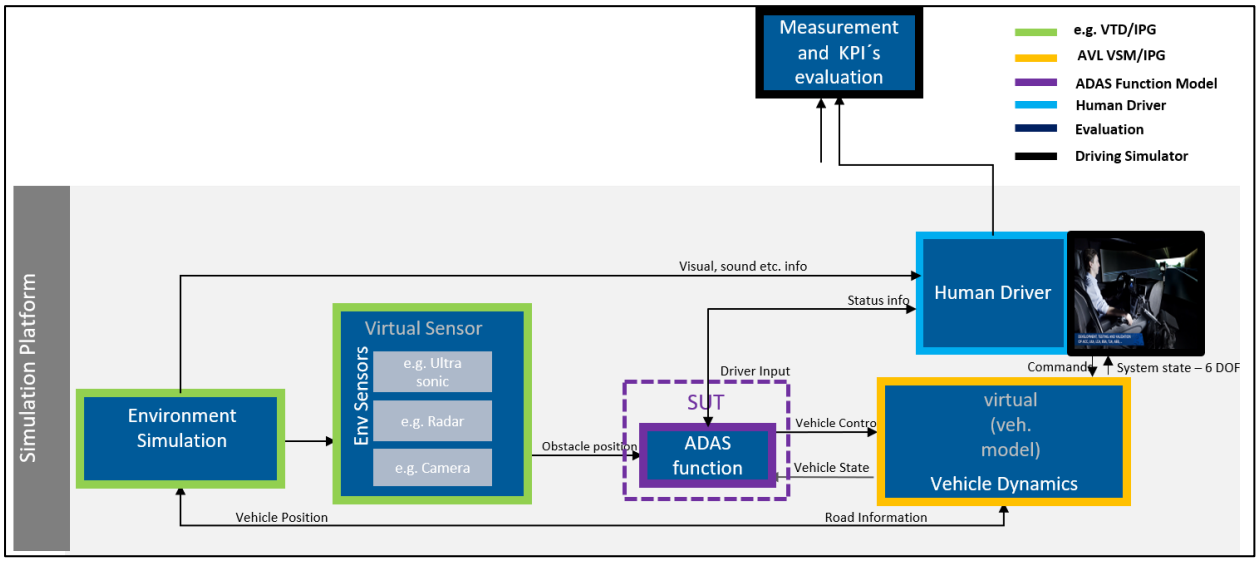


**Figure 20: Driving Simulator Architecture**

The following sub-chapters explain the components of this framework in detail.

1. Environment Simulation:
   The co-simulation framework provides the flexibility to use IPG CarMaker or Vires VTD as a simulation environment. The environment model includes the functionalities of defining scenarios, traffic objects such as obstacles, road markers or traffic signs as well as environmental conditions such as weather, lighting or temperature. The environment model also provides the road information like friction and slope to the vehicle model to calculate the dynamics behavior.

2. Sensor Model:
   Depending on the Use Case the ideal sensor which is available in the Simulation environment or custom developed sensor model can be used. The virtual sensor model is the interface



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.

between Environment Simulation and the ADAS function. The Ideal sensor in the Simulation environment provides the Object and the environment information around the Ego vehicle as Object Lists.

3. Virtual Vehicle Model / Vehicle Dynamics Model:
   The virtual vehicle model is used in this framework to simulate the ego vehicle behavior and dynamics. The model receives the inputs from either Human/Virtual Driver in terms of Throttle pedal position and Brake pedal position or from the ADAS/AD function in the co-simulation framework. The resulting vehicle positions, angles and movements are then transferred to the virtual environment model where it is used to position the Ego vehicle with respect to the surrounding.

4. Driver Model:
   Depending on the Use Case and the validation stage, the framework provides the flexibility to use either the virtual driver model from the simulation environment or human driver.

5. ADAS Function:
   The Automated Driving Function can be integrated to the framework as shown in above Figure 1. The model can be added as an FMU or Matlab/Simulink model. The model receives the environment information from the sensor model and controls the vehicle model accordingly.

### 3.1.2 Driving Simulator Architecture in Model.CONNECT

The General Architecture of Figure 1 is implemented using AVL co-simulation platform called Model.CONNECT. This is shown in Figure 2. All the components are integrated in Model.CONNECT as Functional Mock-up Units (FMUs), so that the co-simulation platform doesn't have any dependency on the external tool in which it is developed. For Example, the highway pilot function used in this Driving Simulator study is developed in MATLAB/SIMULINK, but it is integrated as FMU, so that it doesn't need MATLAB for 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.

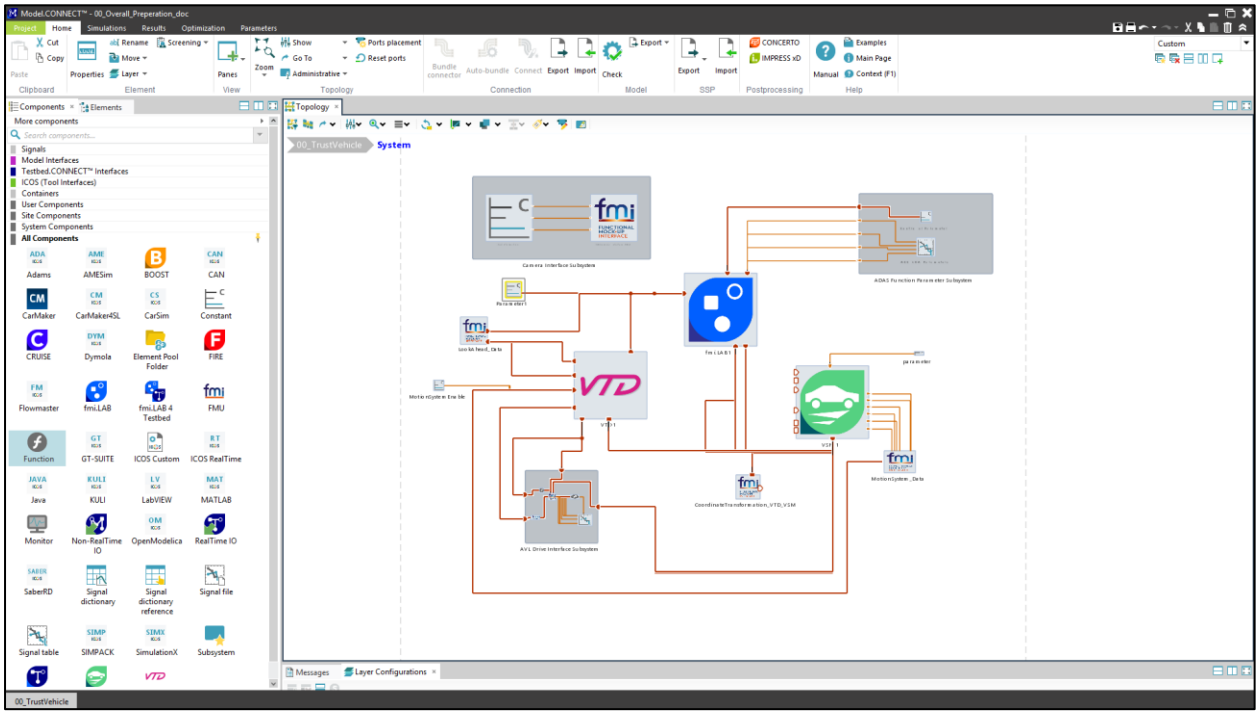


**Figure 21: Model.CONNECT Project Overview**

The following subchapters explains the different components and Subsystems used in the Model.CONNECT project.

### VIRES VTD - Environment Simulation

VIRES VTD is used in the co-simulation platform as an environment simulation models because of its real time capabilities and visualization. VTD provides high quality visualization along with motion compensation with respect to the movement of Driving Simulator motion platform. Scenarios, traffic objects such as obstacles, road markers and traffic signs as well as environmental conditions such as weather, lighting and temperature are simulated in VTD along with the virtual sensor modules. VTD runs in a separate Linux computer and Model.CONNECT provides a default wrapper to connect to the VTD computer using TCP/IP interface through ethernet cable.

### AVL VSM – Vehicle Dynamics

AVL VSM used here is to simulate the behavior and dynamics of the Ego vehicle. VTD sends the road profile information to AVL VSM and it calculates the dynamics and sends back this information to VTD for positioning the Ego vehicle in the environment. Model.CONNECT provides default VSM wrapper to communicate to AVL VSM tool which runs in background.

In this Driving Simulator study, AVL VSM with an addon feature of ‘Motion platform Interface’ is used. This feature in VSM provides UDP interface to communicate with the Driving Simulator motion platform to receive the pedal information and steering wheel information and give back the yaw, pitch and roll angles along with Accelerations in X, Y and Z directions.



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.

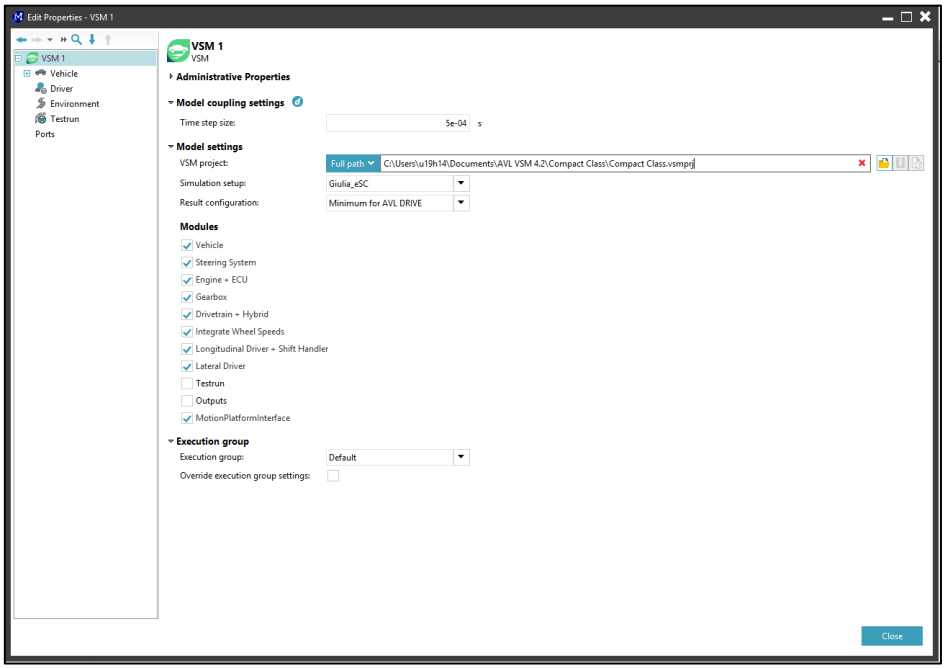


**Figure 22: AVL VSM Configuration**

### ADAS function – Highway Pilot function

The Highway pilot function tested in Driving Simulator consist of mainly two features, Automatic Cruise Control (ACC) and Lane Keep Assist. These functions are integrated in Model.CONNECT as an FMU. This ADAS function takes the outputs from the virtual sensors in VTD and calculates the necessary pedal values and steering input for the Ego vehicle in AVL VSM. The virtual sensors used in this case are 'LookAhead Sensor' for lane information and 'DefaultSensor' for traffic object information.

### ADAS Function Parameter Subsystem

All necessary parameters required for the Highway pilot function are defined in this subsystem. This subsystem consists of following components:

### Controller Parameters

The parameters required to tune the Highway pilot function, DistCntrlD, DistCntrlV and SpdCntrlV are defined here using a constant block.



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.

### ACC_LKA_Parameters

The parameters Controller Enable signal, Desired Speed and Timegap which are time dependent are defined in this component using a signal table. Using Signal tables in Model.CONNECT, user can define the values for different parameters on different simulation time.

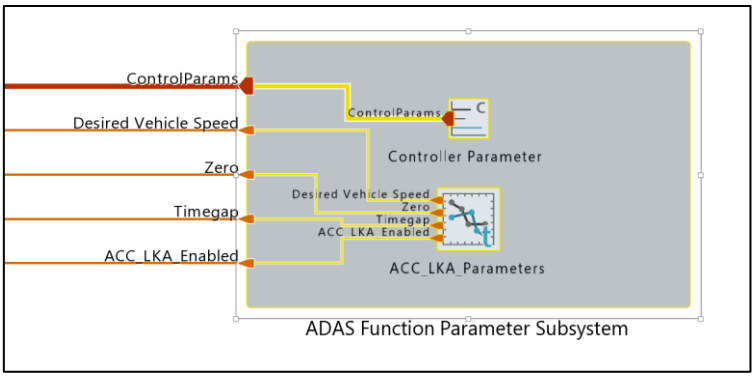


**Figure 23: Parameter Subsystem**

### CoordinateTransformation_VTD_VSM

The signals in VTD are calculated by considering the center of rear axis as an origin but at the same time VSM calculate its signals with respect to Centre of Gravity (CoG). Since in simulation these signals are exchanged between these components, an FMU is defined between them for coordinate transformation.

### Camera interface Subsystem

This subsystem is defined for sending the test information to Infineon Camera system to synchronize in test measurement with Model.CONNECT results for offline analysis.

### Test Information

Test Information like participation ID, Scenario ID and Controller Parameter are defined here using a constant block.

### TOFcamera_Interface_FMU

This FMU receives the test information and send it to Infineon cameo system through UDP interface using ethernet cable.



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.

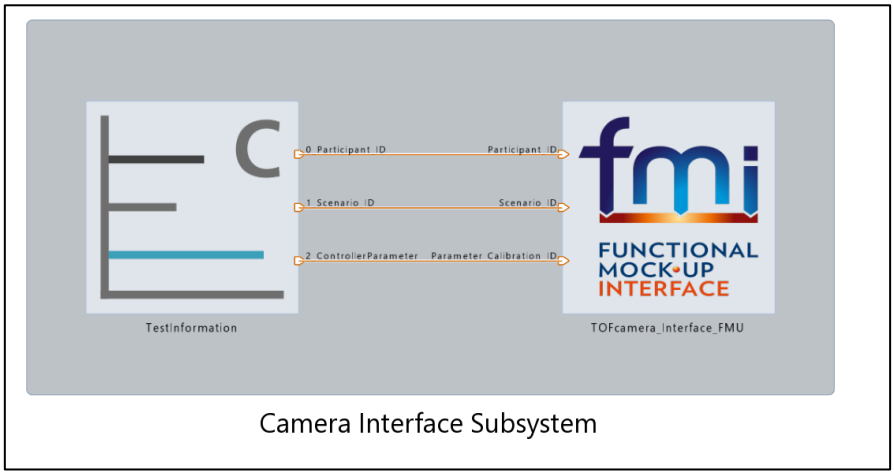


**Figure 24: Infineon Cameo interface Subsystem**

### AVL Drive Interface Subsystem

AVL Drive is integrated to the Model.CONNECT project to create an objective analysis of the simulated vehicle attributes. This subsystem provides all the necessary information to AVL Drive to calculate the rating for every simulation.

This subsystem consists of following components,

### AVL Drive_Interface_FMU:

This is a custom made FMU to communicate to AVL Drive. it sends all the necessary information to AVL Drive which run on background on the same computer via a self-developed protocol.

### ACC_LKA_Parameters1

This is the same component defined in section ACC_LKA_Parameters. AVL Drive also needs the input parameter of the Highway pilot function to calculate the rating.

### LaneInformation

For AVL Drive rating calculation, it needs the distance to the nearest left and right lane from VIRES VTD. Since it is not directly available, it is implemented using C function component in Model.CONNECT, through which the user can write his C code directly.



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.

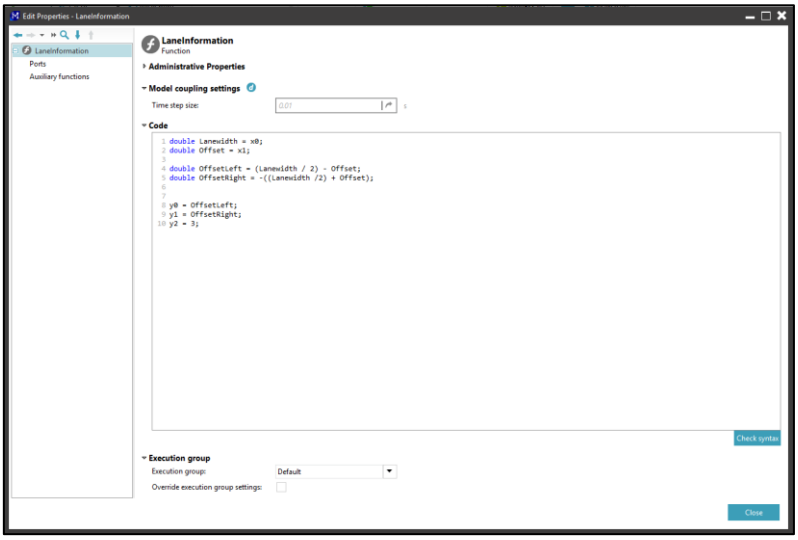


**Figure 25: C-function component in Model.CONNECT**

### ObjectInformation

This FMU extracts the nearest traffic object information, like speed, distance and acceleration in X and Y direction from the virtual sensor in VIRES VTD for AVL Drive.

### TimegapInformation

AVL Drive needs the time gap information as an integer for the rating calculation. This C function component converts the time gap parameter defined in the ACC_LKA_Parameter1 unit to nearest integer value.

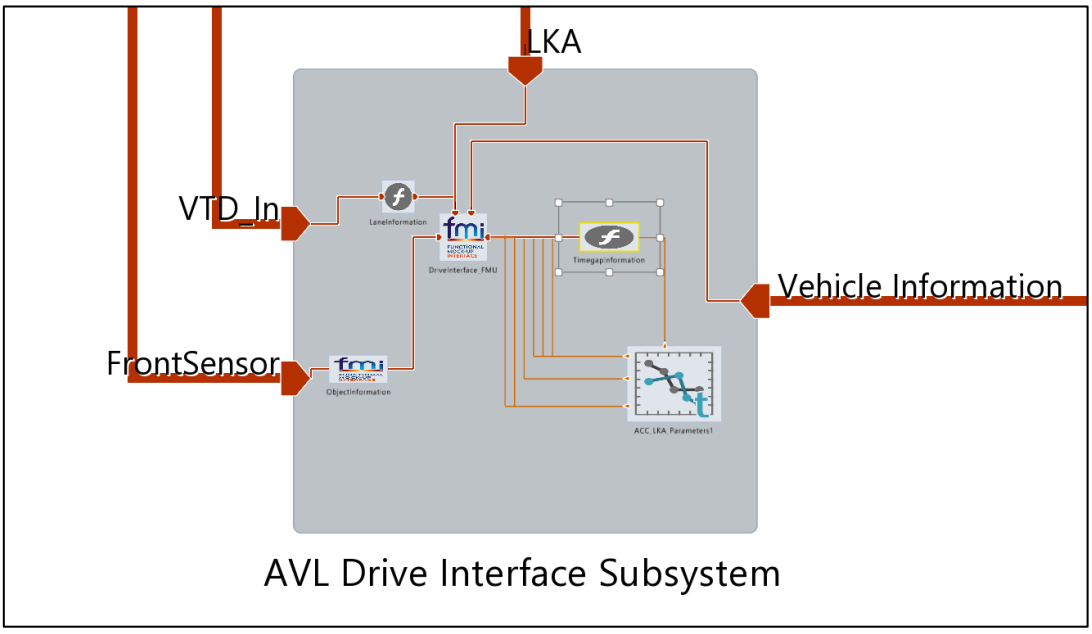


**Figure 26: AVL Drive Interface Subsystem**



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.

**LookAhead_Data**

The LookAhead sensor in VIRES VTD provides the information about lane offset and road curvature at 5m, 10m and 20m ahead of the Ego vehicle. The Highway pilot function needs this information as an array along with lane offset and curvature at the vehicle center. This FMU combines the LookAhead sensor information along with the road information at the vehicle center from VTD and provides this to the highway pilot function as a float array of size 5.

**MotionSystemEnable**

VIRES VTD provides the feature of moving the eyepoint on its visualization with respect to the movement of Driving Simulator motion platform. To activate this feature, the corresponding enabler flag must be set to 1. This constant block provides this information.

### 3.1.3 Simulation and Execution of the models

The Model.CONNECT project prepared for the Driving Simulator study consists of VIRES VTD, AVL VSM and AVL Drive along with the Driving Simulator motion platform. All the systems can be simulated from Model.CONNECT. It is responsible for synchronization and data exchange between the components and storing the measurement results.

**Model.CONNECT Simulation properties**

The simulation properties are defined as shown in Figure 8. In this setting user can define the Start time and End time of the simulation. If the End time is not defined the simulation continues to run for infinite time. The step time of this configuration is defined here as 10ms. The ‘Relative epsilon’ value is used for floating point number comparisons. For a typical co-simulation with various elements the times of the individual elements will be compared within this epsilon range. The default value is 1e-8.



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.

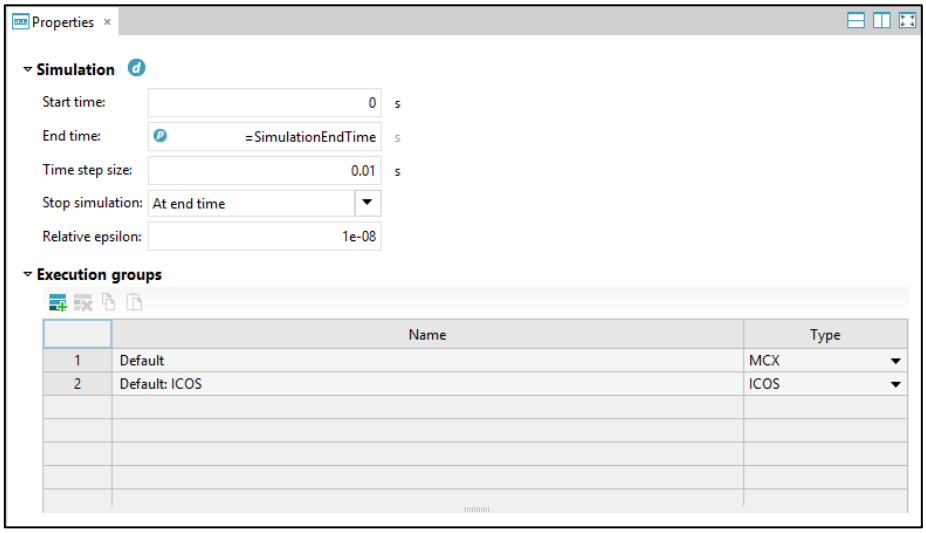


**Figure 27: Simulation Setting in Model.CONNECT**

### Model.CONNECT execution properties

The 'Step time Size' is the synchronization time of the all the components in the simulation. In this case it is defined as 10ms. The evaluation order of the components is defined as 'parallel' since the simulation intends to execute all the model parallelly. There are two possibilities how the input is updated between the component one with values at the end of the coupling step or with values at the beginning of the coupling step. By default, is defined at 'Beginning of the coupling step'.

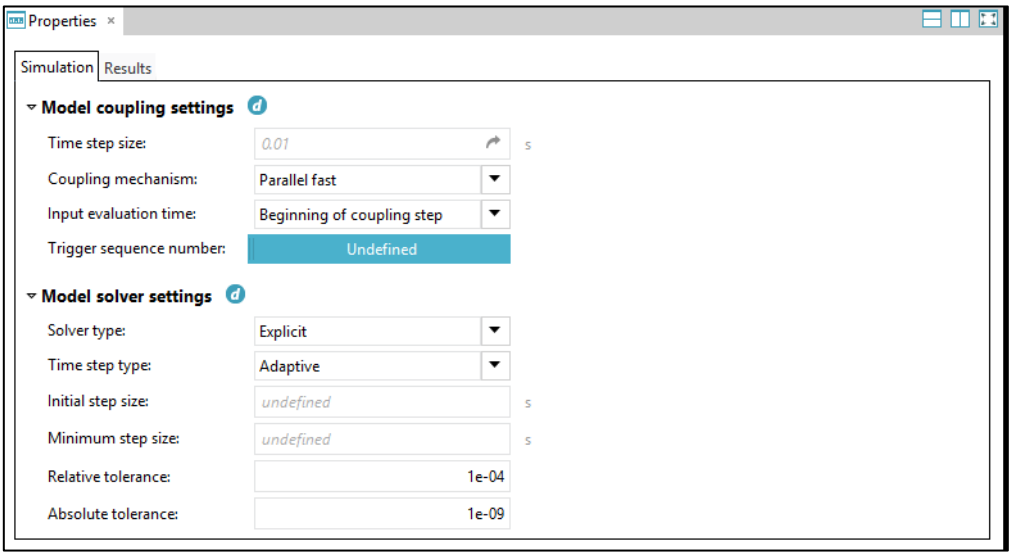


**Figure 28: Simulation and execution settings**



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.

The default solver setting is shown in Figure 28. The expect solver type uses the explicit integration method for numerical solver. Time step type specifies if an adaptive or a fixed step-size method is used for the numerical integration. The solver time-step-size is the coupling time-step-size. The default option 'Adaptive' used for the calculations are explicit adaptive Runge Kutta methods or implicit BDF methods. Based on the kind of problem the solver chooses an appropriate method. For the Adaptive time step type, the integration error can be controlled by setting the relative tolerance and absolute tolerance. The default value of relative tolerance is 1e-4 and absolute tolerance is 1e-9.

## 3.2 Ford Otosan Use Case

In this section, the final co-simulation setup used for the simulation of FO use cases will be introduced. In this setup, components with different functionality and from different platforms are connected to simulate the L3AD system that will be later implemented on the FO mule vehicle, i.e. demonstrator vehicle.

In Figure 29, the overall co-simulation block diagram of FO use cases is shown. The blocks in green are MiL vehicle model components from IPG TruckMaker high-fidelity software and they are used to simulate the mule vehicle. The orange blocks are interface blocks between the IPG TruckMaker and Matlab/Simulink platforms. The Simulink components are in grey.

Sensors are modelled in the MiL vehicle model and their data are given by the environment model and the vehicle interface. The environment interface processes the sensor data and gives as output signals about the surrounding road and obstacles to the trajectory planner interface. The latter in its turn make calculations on the signals at its inputs to provide the trajectory planner with the required signals to generate a safe and trustworthy reference trajectory. This reference, represented by reference velocities position and angles of a ghost vehicle, is given to the longitudinal and lateral trajectory controller to be followed.

The trajectory controller component provides the control signals to the IPG vehicle model through the Simulink2IPG interface component. The signals given to the Simulink2IPG interface are terminated in the MiL vehicle model, as seen in Figure 29, and are fed instead from Simulink. The handover switch components are placed in a cyan block.



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.

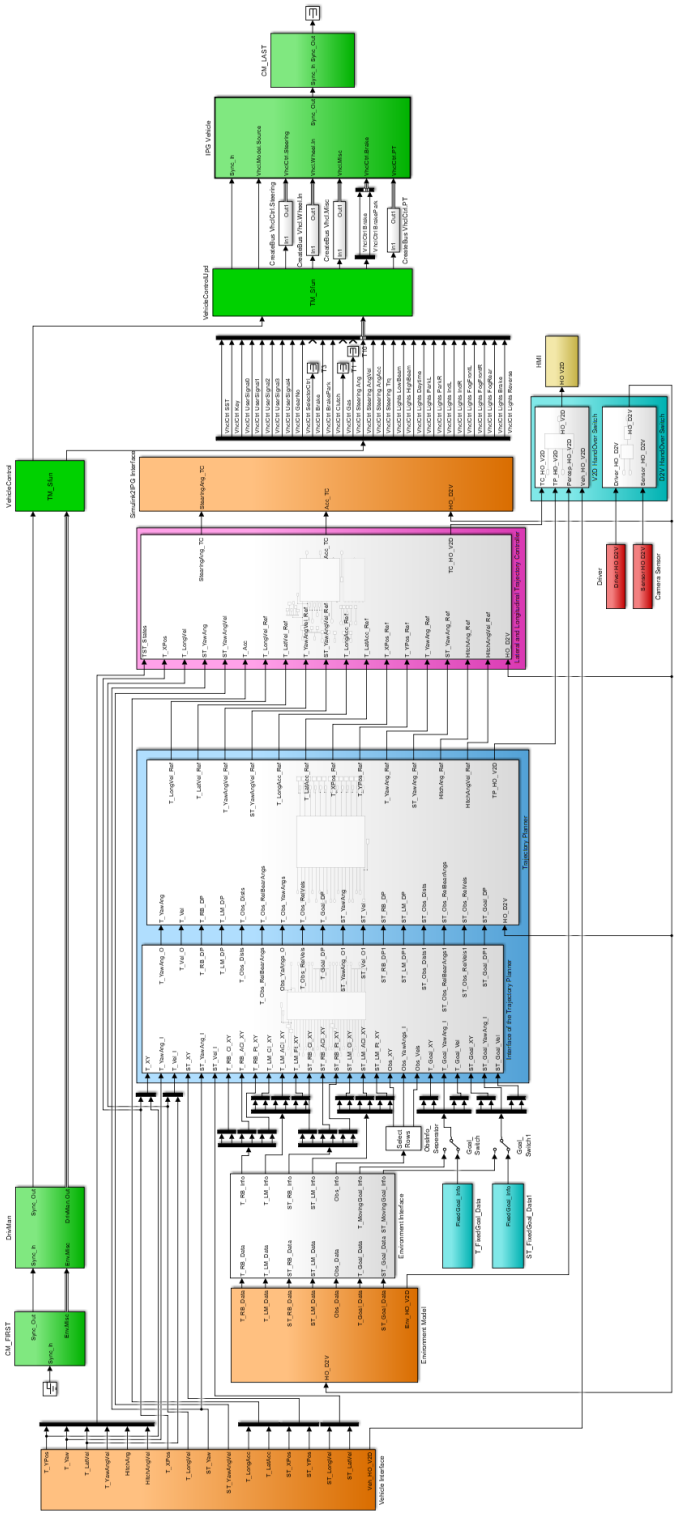

**Figure 29: Overall co-simulation block diagram for FO UC**



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 Components

As mentioned in the previous section the co-simulation block diagram consists of several components. Those components are:

- Vehicle interface component
- Environment model component
- Environment interface component
- Trajectory planner interface
- Trajectory planner
- Lateral and longitudinal trajectory controller component
- Simulink2IPG interface component

The signals setup for each of the components is shown in the following sections.

#### 3.2.1.1 Vehicle interface component

The vehicle interface component is responsible for reading signals data from the IPG vehicle model. This signals data is given by the sensors modelled in IPG and it contains information about the truck and semitrailer positions, velocities, yaw and hitch angle, ... etc. This data is passed to the trajectory planner interface and to the longitudinal and lateral trajectory controller components.

The list of signals that are given as an output of the vehicle interface component are listed in the following table with related descriptions.

**Table 1: Output signals of the vehicle parameters interface component**

| Sub-system | Signals Name | Signal description | Symbol | Type | Size | Unit | I/O |
|---|---|---|---|---|---|---|---|
| Vehicle interface | T_YPos | Coordinates of the CoG of the truck on the Y axis of the inertia (global) frame | $y_t$ | Scalar | 1x1 | m | O |
| | T_Yaw | Yaw angle of the truck | $\psi_t$ | Scalar | 1x1 | rad | O |
| | T_LatVel | Lateral velocity of the truck | $v_{y,t}$ | Scalar | 1x1 | m/s | O |
| | T_YawAngVel | Yaw angle change rate of the truck | $\omega_{z,t}$ | Scalar | 1x1 | rad/s | O |
| | HitchAng | Hitch angle of the HV | $\phi$ | Scalar | 1x1 | rad | O |



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.

| Sub-system | Signals Name | Signal description | Symbol | Type | Size | Unit | I/O |
|---|---|---|---|---|---|---|---|
| | HitchAngVel | Hitch angle change rate of the HV | $\dot{\phi}$ | Scalar | 1x1 | rad/s | O |
| | T_XPos | Coordinates of the CoG of the truck on the X axis of the inertia (global) frame | $x_t$ | Scalar | 1x1 | m | O |
| | T_LongVel | Longitudinal velocity of the truck | $v_{x,t}$ | Scalar | 1x1 | m/s | O |
| | ST_YawAngVel | Yaw angle change rate of the semitrailer | $\omega_{z,st}$ | Scalar | 1x1 | rad/s | O |
| | ST_Yaw | Yaw angle of the semitrailer | $\psi_{st}$ | Scalar | 1x1 | rad | O |
| | T_LongAcc | Longitudinal acceleration of the truck | $\dot{v}_{x,t}$, | Scalar | 1x1 | m/s | O |
| | T_LatAcc | lateral acceleration of the truck | $\dot{v}_{y,t}$ | Scalar | 1x1 | m/s | O |
| | ST_XPos | Coordinates of the CoG of the semitrailer on the X axis of the inertia (global) frame | $x_{st}$ | Scalar | 1x1 | m | O |
| | ST_YPos | Coordinates of the CoG of the semitrailer on the Y axis of the inertia (global) frame | $y_{st}$ | Scalar | 1x1 | m | O |
| | ST_LongVel | Longitudinal velocity of the semitrailer | $v_{x,st}$ | Scalar | 1x1 | m/s | O |
| | ST_LatVel | Lateral velocity of the semitrailer | $v_{x,st}$ | Scalar | 1x1 | m/s | O |
| | Veh_HO_V2D | Handover flag of the vehicle to give control from the vehicle to driver | $\text{Flag}_{\text{Tc}}$ | Boolean | 1x1 | - | O |

#### 3.2.1.2 Environment model component

As in the case of the vehicle interface component, the environment component is dedicated to obtain signals data from IPG-modelled sensors. The data has information about the environment surrounding the truck and semitrailer, namely Road Border (RB), Lane Markings (LM) and obstacles. This data is passed to the environment interface component.

The input and output signals of this component are listed in the following table.



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 2: Input and output signals of the environment component**

| Sub-system | Signals Name | Signal description | Symbol | Type | Size | Unit | I/O |
|---|---|---|---|---|---|---|---|
| Environment model | HO_D2V | Handover flag to give control from the drive to the vehicle | $Flag_{HO}$ | Boolean | 1x1 | - | I |
| | T_RB_Data | Trucks distance to the Road border, same as lane marking data, according to IPG's Road Sensor | NA | Scalar | 1x1 | m | O |
| | T_LM_Data | Trucks distance to the Lane Marking, according to IPG's Road Sensor | NA | Scalar | 1x1 | m | O |
| | ST_RB_Data | NA – Road Sensor on Trailer not Available | NA | - | - | - | O |
| | ST_LM_Data | NA – Road Sensor on Trailer not Available | NA | - | - | - | O |
| | Obs_Data | Coordinates of the CoG of n obstacles on the X and Y axis of the inertia frame, Yaw angle of n obstacles and Longitudinal and lateral velocities of n obstacles | $\begin{pmatrix} x_1 & \cdots & x_n \\ y_1 & \cdots & y_n \\ \psi_1 & \cdots & \psi_n \\ v_{x,1} & & v_{x,n} \\ v_{y,1} & \cdots & v_{y,n} \end{pmatrix}$ | Matrix | 5xn | m<br>rad<br>m/s | O |
| | Env_HO_V2D | Handover flag of the sensors to give control from the vehicle to driver | $Flag_{Env}$ | Boolean | 1x1 | - | O |

### 3.2.1.3 Environment interface component

The environment interface component extract from the data given by the environment component the required information by the trajectory planner interface.

The outputs of the environment block are all given as input signals of the environment interface component. Therefore, to avoid repetition only the list of output signals of the environment interface component is given in Table 3.



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: Output signals of the environment interface component**

| Sub-system | Signals Name | Signal description | Symbol | Type | Size | Unit | I/O |
|---|---|---|---|---|---|---|---|
| Environment interface | T_RB_Info | Coordinates of the Cl, ACl and Ft points to the truck on the Cl RB on X and Y axis of the inertia frame | $\begin{pmatrix} x_{t,cl,rb} \\ y_{t,cl,rb} \\ x_{t,acl,rb} \\ y_{t,acl,rb} \\ x_{t,ft,rb} \\ y_{t,ft,rb} \end{pmatrix}$ | Vector | 6x1 | m | O |
| | T_LM_Info | Coordinates of the Cl, ACl and Ft points to the truck on the Cl LM on X and Y axis of the inertia frame | $\begin{pmatrix} x_{t,cl,lm} \\ y_{t,cl,lm} \\ x_{t,acl,lm} \\ y_{t,acl,lm} \\ x_{t,ft,lm} \\ y_{t,ft,lm} \end{pmatrix}$ | Vector | 6x1 | m | O |
| | ST_RB_Info | Coordinates of the Cl, ACl and Ft points to the semitrailer on the Cl RB on X and Y axis of the inertia frame | $\begin{pmatrix} x_{st,cl,rb} \\ y_{st,cl,rb} \\ x_{st,acl,rb} \\ y_{st,acl,rb} \\ x_{st,ft,rb} \\ y_{st,ft,rb} \end{pmatrix}$ | Vector | 6x1 | m | O |
| | ST_LM_Info | Coordinates of the Cl, ACl and Ft points to the semitrailer on the Cl LM on X and Y axis of the inertia frame | $\begin{pmatrix} x_{st,cl,lm} \\ y_{st,cl,lm} \\ x_{st,acl,lm} \\ y_{st,acl,lm} \\ x_{st,ft,lm} \\ y_{st,ft,lm} \end{pmatrix}$ | Vector | 6x1 | m | O |
| | Obs_Info | Coordinates of the CoG of n obstacles on the X and Y axis of the inertia frame, Yaw angle of n obstacles and Longitudinal and lateral velocities of n obstacles | $\begin{pmatrix} x_1 & \cdots & x_n \\ y_1 & \cdots & y_n \\ \psi_1 & \cdots & \psi_n \\ v_{x,1} & & v_{x,n} \\ v_{y,1} & \cdots & v_{y,n} \end{pmatrix}$ | Matrix | 5xn | m<br>rad<br>m/s | O |

### 3.2.1.4 Trajectory planner interface component

Since the trajectory planner requires certain information from sensors about the surrounding environment to produce the reference trajectory. An interface component for the trajectory planner was developed to provide the right signals based on the signals data given by the sensors. For details on the calculations carried by this component please refer to D3.3.



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 input signals that are used by this component are listed in the following table.

**Table 4: Input signals of the interface of the trajectory planner component**

| Sub-system | Signals Name | Signal description | Symbol | Type | Size | Unit | I/O |
|---|---|---|---|---|---|---|---|
| Trajectory planner Interface | T _XY | Coordinates of the CoG of the truck on the X and Y axis of the inertia (global) frame | $\begin{pmatrix} x_t \\ y_t \end{pmatrix}$ | Vector | 2x1 | m | I |
| | T_YawAng_I | Yaw angle of the truck | $\psi_t$ | Scalar | 1x1 | rad | I |
| | T_Vel_I | Longitudinal and Lateral velocities of the truck | $\begin{pmatrix} v_{x,t} \\ v_{y,t} \end{pmatrix}$ | Vector | 2x1 | m/s | I |
| | ST _XY | Coordinates of the CoG of the semitrailer on the X and Y axis of the inertia (global) frame | $\begin{pmatrix} x_{st} \\ y_{st} \end{pmatrix}$ | Vector | 2x1 | m | I |
| | ST_YawAng_I | Yaw angle of the semitrailer | $\psi_{st}$ | Scalar | 1x1 | rad | I |
| | ST_Vel_I | Longitudinal and Lateral velocities of the semitrailer | $\begin{pmatrix} v_{x,st} \\ v_{y,st} \end{pmatrix}$ | Vector | 2x1 | m/s | I |
| | T_RB_Cl_XY | Coordinates of the Cl point to the truck on the Cl RB on X and Y axis of the inertia frame | $\begin{pmatrix} x_{t,cl,rb} \\ y_{t,cl,rb} \end{pmatrix}$ | Vector | 2x1 | m | I |
| | T_RB_ACl_XY | Coordinates of the ACl point to the truck on the CL RB on X and Y axis of the inertia frame | $\begin{pmatrix} x_{t,acl,rb} \\ y_{t,acl,rb} \end{pmatrix}$ | Vector | 2x1 | m | I |
| | T_RB_Ft_XY | Coordinates of the Ft point to the truck on the Cl RB on X and Y axis of the inertia frame | $\begin{pmatrix} x_{t,ft,rb} \\ y_{t,ft,rb} \end{pmatrix}$ | Vector | 2x1 | m | I |
| | T_LM_Cl_XY | Coordinates of the Cl point to the truck on the Cl LM on X and Y axis of the inertia frame | $\begin{pmatrix} x_{t,cl,lm} \\ y_{t,cl,lm} \end{pmatrix}$ | Vector | 2x1 | m | I |
| | T_LM_ACl_XY | Coordinates of ACl point to the truck on the Cl LM on X and Y axis of the inertia frame | $\begin{pmatrix} x_{t,acl,lm} \\ y_{t,acl,lm} \end{pmatrix}$ | Vector | 2x1 | m | I |
| | T_LM_Ft_XY | Coordinates of the Fl point to the truck on the Cl LM on X and Y axis of the inertia frame | $\begin{pmatrix} x_{t,ft,lm} \\ y_{t,ft,lm} \end{pmatrix}$ | Vector | 2x1 | m | I |
| | ST_RB_Cl_XY | Coordinates of the Cl point to the semitrailer on the Cl RB on X and Y inertia frame axis | $\begin{pmatrix} x_{st,cl,rb} \\ y_{st,cl,rb} \end{pmatrix}$ | Vector | 2x1 | m | I |
| | ST_RB_ACl_XY | Coordinates of the ACl point to the semitrailer on the CL RB on X and Y inertia frame axis | $\begin{pmatrix} x_{st,acl,rb} \\ y_{st,acl,rb} \end{pmatrix}$ | Vector | 2x1 | m | I |
| | ST_RB_Ft_XY | Coordinates of the Ft point to the semitrailer on the Cl RB on X and Y inertia frame axis | $\begin{pmatrix} x_{st,ft,rb} \\ y_{st,ft,rb} \end{pmatrix}$ | Vector | 2x1 | m | I |



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.

| Sub-system | Signals Name | Signal description | Symbol | Type | Size | Unit | I/O |
|---|---|---|---|---|---|---|---|
| | ST_LM_Cl_XY | Coordinates of the Cl point to the semitrailer on the Cl LM on X and Y inertia frame axis | $\begin{pmatrix} x_{st,cl,lm} \\ y_{st,cl,lm} \end{pmatrix}$ | Vector | 2x1 | m | I |
| | ST_LM_ACl_XY | Coordinates of ACl point to the semitrailer on the Cl LM on X and Y axis of the inertia frame | $\begin{pmatrix} x_{st,acl,lm} \\ y_{st,acl,lm} \end{pmatrix}$ | Vector | 2x1 | m | I |
| | ST_LM_Ft_XY | Coordinates of the Fl point to the semitrailer on the Cl LM on X and Y axis of the inertia frame | $\begin{pmatrix} x_{st,ft,lm} \\ y_{st,ft,lm} \end{pmatrix}$ | Vector | 2x1 | m | I |
| | Obs_XY | Coordinates of the CoG of n obstacles on the X and Y axis of the inertia frame | $\begin{pmatrix} x_1 & \dots & x_n \\ y_1 & & y_n \end{pmatrix}$ | Matrix | 2xn | m | I |
| | Obs_YawAngs | Yaw angle of n obstacles | $(\psi_1 \quad \dots \quad \psi_n)$ | Vector | 1xn | rad | I |
| | Obs_Vels | Longitudinal and lateral velocities of n obstacles | $\begin{pmatrix} v_{x,1} & \dots & v_{x,n} \\ v_{y,1} & & v_{y,n} \end{pmatrix}$ | Matrix | 2xn | m/s | I |
| | T_Goal_XY | Coordinates of the CoG of the truck goal on the X and Y axis of the inertia frame | $\begin{pmatrix} x_{t,g} \\ y_{t,g} \end{pmatrix}$ | Vector | 2x1 | m | I |
| | T_Goal_YawAng_I | Yaw angle of the truck goal | $\psi_{t,g}$ | Scalar | 1x1 | rad | I |
| | T_Goal_Vel | Longitudinal and lateral velocities of the truck goal | $\begin{pmatrix} v_{t,x,g} \\ v_{t,y,g} \end{pmatrix}$ | Vector | 2x1 | m/s | I |
| | ST_Goal_XY | Coordinates of the CoG of the semitrailer goal on the X and Y axis of the inertia frame | $\begin{pmatrix} x_{st,g} \\ y_{st,g} \end{pmatrix}$ | Vector | 2x1 | m | I |
| | ST_Goal_YawAng_I | Yaw angle of the semitrailer goal | $\psi_{st,g}$ | Scalar | 1x1 | rad | I |
| | ST_Goal_Vel | Longitudinal and lateral velocities of the semitrailer goal | $\begin{pmatrix} v_{st,x,g} \\ v_{st,y,g} \end{pmatrix}$ | Vector | 2x1 | m/s | I |

For demonstration on the some of the signals mentioned in the table above, please refer to section 3.4.1.4 later in this document.

#### 3.2.1.5 Trajectory planner component

The trajectory planner component generated the reference trajectories depending on the information given about the environment and the current state of the T-ST. It uses the concept of reference ghost vehicle; thus, its output signals are the current state of this vehicle. The input signals together with the output signals of the trajectory planner component are listed in Table 5.



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 5: Input signals of the trajectory planner component**

| Sub-system | Signals Name | Signal description | Symbol | Type | Size | Unit | I/O |
|---|---|---|---|---|---|---|---|
| Trajectory planner | T_YawAng | Yaw angle of the truck | $\psi_t$ | Scalar | 1x1 | rad | I |
| | T_Vel | Longitudinal and Lateral velocities of the truck | $\begin{pmatrix} v_{x,t} \\ v_{y,t} \end{pmatrix}$ | Vector | 2x1 | m/s | I |
| | T_RB_DP | Data regarding the RB for the truck | $(\rho_{t,cl,rb}, \theta_{t,cl,rb}, \Delta\theta_{t,rb}, \psi_{t,cl,rb})^T$ | Vector | 4x1 | m<br>rad<br>rad<br>rad | I |
| | T_LM_DP | Data regarding the LM for the truck | $(\rho_{t,cl,lm}, \theta_{t,cl,lm}, \Delta\theta_{t,lm}, \psi_{t,cl,lm})^T$ | Vector | 4x1 | m<br>rad<br>rad<br>rad | I |
| | T_Obs_Dists | The distances between the truck and n obstacles | $(\rho_{1,t} \quad \dots \quad \rho_{n,t})$ | Vector | 1xn | m | I |
| | T_Obs_RelBearAngs | The relative bearing angles of n obstacles to the truck | $(\theta_{1,t} \quad \dots \quad \theta_{n,t})$ | Vector | 1xn | rad | I |
| | Obs_YawAngs | Yaw angle of the n obstacle | $(\psi_1 \quad \dots \quad \psi_n)$ | Vector | 1xn | rad | I |
| | T_Obs_RelVels | Relative velocity between the truck and n obstacles | $(v_{rel,t,1} \dots v_{rel,t,n})$ | Vector | 1xn | m/s | I |
| | T_Goal_DP | Data regarding the truck goal | $(\rho_{t,g}, \theta_{g,t}, \psi_{g,t}, v_{x,rel,t,g}, v_{y,rel,t,g})^T$ | Vector | 5x1 | m<br>rad<br>rad<br>m/s<br>m/s | I |
| | ST_YawAng | Yaw angle of the semitrailer | $\psi_{st}$ | Scalar | 1x1 | rad | I |
| | ST_Vel | Longitudinal and Lateral velocities of the semitrailer | $\begin{pmatrix} v_{x,st} \\ v_{y,st} \end{pmatrix}$ | Vector | 2x1 | m/s | I |
| | ST_RB_DP | Data regarding the RB for the semitrailer | $(\rho_{st,cl,rb}, \theta_{st,cl,rb}, \Delta\theta_{st,rb}, \psi_{st,cl,rb})^T$ | Vector | 4x1 | m<br>rad<br>rad<br>rad | I |
| | ST_LM_DP | Data regarding the LM for the semitrailer | $(\rho_{st,cl,lm}, \theta_{st,cl,lm}, \Delta\theta_{st,lm}, \psi_{st,cl,lm})^T$ | Vector | 4x1 | m<br>rad<br>rad<br>rad | I |
| | ST_Obs_Dists | The distances between the semitrailer and n obstacles | $(\rho_{1,st} \quad \dots \quad \rho_{n,st})$ | Vector | 1xn | m | I |
| | ST_Obs_RelBearAngs | The relative bearing angles of n obstacles to the semitrailer | $(\theta_{1,st} \quad \dots \quad \theta_{n,st})$ | Vector | 1xn | rad | I |
| | ST_Obs_RelVels | Relative velocity between the semitrailer and n obstacles | $(v_{rel,st,1} \dots v_{rel,st,n})$ | Vector | 1xn | m/s | I |
| | ST_Goal_DP | Data regarding the semitrailer goal | $(\rho_{st,g}, \theta_{g,st}, \psi_{g,st}, v_{x,rel,st,g}, v_{y,rel,st,g})^T$ | Vector | 5x1 | m<br>rad<br>rad<br>m/s<br>m/s | I |



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.

| Sub-system | Signals Name | Signal description | Symbol | Type | Size | Unit | I/O |
|---|---|---|---|---|---|---|---|
| | HO_D2V | Handover flag to give control from the drive to the vehicle | $Flag_{HO}$ | Boolean | 1x1 | - | I |
| | T_LongVel_Ref | Longitudinal velocity of the truck GV | $v_{x,t,ref}$ | Scalar | 1x1 | m/s | O |
| | T_LatVel_Ref | Lateral velocity of the truck GV | $v_{y,t,ref}$ | Scalar | 1x1 | m/s | O |
| | T_YawAngVel_Ref | Angular velocity of the truck GV | $\omega_{z,t,ref}$ | Scalar | 1x1 | rad/s | O |
| | ST_YawAngVel_Ref | Angular velocity of the semitrailer GV | $\omega_{z,st,ref}$ | Scalar | 1x1 | rad/s | O |
| | T_LongAcc_Ref | Longitudinal acceleration of the truck GV | $\dot{v}_{x,t,ref}$ | Scalar | 1x1 | $m/s^2$ | O |
| | T_LatAcc_Ref | Lateral acceleration of the truck GV | $\dot{v}_{y,t,ref}$ | Scalar | 1x1 | $m/s^2$ | O |
| | T_YawAngAcc_Ref | Yaw angular acceleration of the truck GV | $\dot{\omega}_{z,t,ref}$ | Scalar | 1x1 | $rad/s^2$ | O |
| | T_XPos_Ref | Coordinates of the CoG of the truck GV on the X-axis of the inertia frame | $x_{t,ref}$ | Scalar | 1x1 | m | O |
| | T_YPos_Ref | Coordinates of the CoG of the truck GV on the Y-axis of the inertia frame | $y_{t,ref}$ | Scalar | 1x1 | m | O |
| | T_YawAng_Ref | Yaw angle of the truck GV | $\psi_{t,ref}$ | Scalar | 1x1 | rad | O |
| | ST_YawAng_Ref | Yaw angle of the semitrailer GV | $\psi_{st,ref}$ | Scalar | 1x1 | rad | O |
| | HitchAng_Ref | Hitch angle | $\phi_{ref}$ | Scalar | 1x1 | rad | O |
| | HitchAngVel_Ref | Hitch angle change rate | $\dot{\phi}_{ref}$ | Scalar | 1x1 | Rad/s | O |
| | TP_HO_V2D | Handover flag of the trajectory planner to give control from the vehicle to driver | $Flag_{Tp}$ | Boolean | 1x1 | - | O |

For simplification, it is assumed that all obstacles are aligned with the road boarder/lane marking to which they are close. Therefore, the yaw angle of the obstacle could be assumed equal to the road boarder/lane marking tangential angle. In addition, the goal of the T-ST is set to be constant and no information will be provided by the environment interface component about a moving T-ST goal.

#### 3.2.1.6 Lateral and longitudinal trajectory controller component

The trajectory controller component receives the reference trajectory and current state of the T-ST as listed in Table 6. To track the reference, the trajectory controller provides the control signal needed as output signals listed in the same table.



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 6: Input and output signals of the lateral and longitudinal trajectory controller component**

| Sub-system | Signals Name | Signal description | Symbol | Type | Size | Unit | I/O |
|---|---|---|---|---|---|---|---|
| Lateral and longitudinal controller | TST_States | States of the truck-semitrailer | $(y_t,$ $\psi_t,$ $v_{y,t},$ $\omega_{z,t},$ $\phi,$ $\dot{\phi})$ | Vector | 6x1 | m rad m/s rad/s rad rad/s | I |
| | T_XPos | Coordinates of the CoG of the truck on the X axis of the inertia (global) frame | $x_t$ | Scalar | 1x1 | m | I |
| | T_LongVel | Longitudinal velocities of the truck | $v_{x,t}$ | Scalar | 1x1 | m/s | I |
| | ST_YawAng | Yaw angle of the semitrailer | $\psi_{st}$ | Scalar | 1x1 | rad | I |
| | ST_YawAngVel | Yaw angle rate of the semitrailer | $\omega_{z,st}$ | Scalar | 1x1 | rad/s | I |
| | T_Acc | Longitudinal and lateral of the truck | $(\dot{v}_{x,t}, \dot{v}_{y,t})$ | Vector | 2x1 | $m/s^2$ $m/s^2$ $rad/s^2$ | I |
| | T_LongVel_Ref | Longitudinal velocity of the truck GV | $v_{x,t,ref}$ | Scalar | 1x1 | m/s | I |
| | T_LatVel_Ref | Lateral velocity of the truck GV | $v_{y,t,ref}$ | Scalar | 1x1 | m/s | I |
| | T_YawAngVel_Ref | Angular velocity of the truck GV | $\omega_{z,t,ref}$ | Scalar | 1x1 | rad/s | I |
| | ST_YawAngVel_Ref | Angular velocity of the semitrailer GV | $\omega_{z,st,ref}$ | Scalar | 1x1 | rad/s | I |
| | T_LongAcc_Ref | Longitudinal acceleration of the truck GV | $\dot{v}_{x,t,ref}$ | Scalar | 1x1 | $m/s^2$ | I |
| | T_LatAcc_Ref | Lateral acceleration of the truck GV | $\dot{v}_{y,t,ref}$ | Scalar | 1x1 | $m/s^2$ | I |
| | T_YawAngAcc_Ref | Yaw angular acceleration of the truck GV | $\dot{\omega}_{z,t,ref}$ | Scalar | 1x1 | $rad/s^2$ | I |
| | T_XPos_Ref | Coordinates of the CoG of the truck GV on the X-axis of the inertia frame | $x_{t,ref}$ | Scalar | 1x1 | m | I |
| | T_YPos_Ref | Coordinates of the CoG of the truck GV on the Y-axis of the inertia frame | $y_{t,ref}$ | Scalar | 1x1 | m | I |



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.

| Sub-system | Signals Name | Signal description | Symbol | Type | Size | Unit | I/O |
|---|---|---|---|---|---|---|---|
| | T_YawAng_Ref | Yaw angle of the truck GV | $\psi_{t,ref}$ | Scalar | 1x1 | rad | I |
| | ST_YawAng_Ref | Yaw angle of the semitrailer GV | $\psi_{st,ref}$ | Scalar | 1x1 | rad | I |
| | HitchAng_Ref | Hitch angle of the GV | $\phi_{ref}$ | Scalar | 1x1 | rad | I |
| | HitchAngVel_Ref | Hitch angle change rate of the GV | $\dot{\phi}_{ref}$ | Scalar | 1x1 | Rad/s | I |
| | HO_D2V | Handover flag to give control from the drive to the vehicle | $\text{Flag}_{\text{HO}}$ | Boolean | 1x1 | - | I |
| | Acc_TC | Accelerator from the trajectory controller | $a_{x,t,Tc}$ | Scalar | 1x1 | $m/s^2$ | O |
| | SteeringAng_TC | Steering angle from the trajectory controller | $\delta_{Tc}$ | Scalar | 1x1 | rad | O |
| | TC_HO_V2D | Handover flag of the trajectory controller to give control from the vehicle to driver | $\text{Flag}_{\text{Tc}}$ | Boolean | 1x1 | - | O |

#### 3.2.1.7 Simulink2IPG interface component

All the output signals of the lateral and longitudinal trajectory controller block are connected to the input signals of the Simulink2IPG interface component. However, the Simulink2IPG block has an addition input signal from the Handover switch block which is HO_D2V, as in the case of the trajectory controller and the trajectory planner. This input is a Handover flag to give control from the drive to the vehicle.



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 Simulink and IPG signal naming

Since the signal naming of Simulink interface components and for IPG TruckMaker is different, Table 7 shows the naming of signals for both software.

**Table 7: Signals naming between Simulink and IPG**

| Sub-system | Simulink Signals Name | IPG Signals Name |
|---|---|---|
| Simulink2IPG Interface | VCGas | VC.Gas |
| | VCBrake | VC.Brake |
| | SteeringAng | VC.Steer.Ang |
| | HO_D2V | - |
| Vehicle Interface | T_YPos | Vhcl.PoI.y |
| | T_Yaw | Vhcl.Yaw |
| | T_LatVel | Vhcl.PoI.vy_1 |
| | T_YawAngVel | Vhcl.YawRate |
| | HitchAng | Tr.Hitch.dr_0.z |
| | HitchAngVel | Tr.Hitch.vz |
| | T_XPos | Vhcl.PoI.x |
| | T_LongVel | Vhcl.PoI.vx_1 |
| | ST_YawAngVel | Tr.YawVel |
| | ST_Yaw | Tr.Yaw |
| | T_LongAcc | Vhcl.PoI.ax_1 |
| | T_LatAcc | Vhcl.PoI.ay_1 |



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.

| Sub-system | Simulink Signals Name | IPG Signals Name |
|---|---|---|
| | T_YawAngAcc | Vhcl.YawAcc |
| | GearNo | Vhcl.GearNo |
| | ST_XPos | Tr.Con.tx |
| | ST_YPos | Tr.Con.ty |
| | ST_LongVel | Tr.Con.vx_1 |
| | ST_LatVel | Tr.Con.vy_1 |

### 3.2.3 Usage

#### 3.2.3.1 System limitations

In this section the limitation imposed on the system and on its input and output signals are defined. Some of the limitations where already listed in D 3.1 for earlier stages of the project, however the final setup is given here. The following table lists the limitations for some of the system components.



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: System components and signals limitations**

| Subsystem model specific information | | | | | | |
|---|---|---|---|---|---|---|
| **Name of subsystem** | Truck Trailer and Environment | | | | | |
| **From Partner** | FO | | | | | |
| **Use Case affiliation** | Use Case 1 (FO) | | | | | |
| **Short description** | environmental modelling of trailer backing and construction site backing problems with static and dynamic obstacles vehicle dynamic and kinematic modelling and correlation of demo vehicle | | | | | |
| **Known limitations** | for specific simulations only - valid in specific operating conditions, … | | | | | |
| **Inputs of the subsystem** | | | | | | |
| **Name / Description** | **Unit** | **Lower Limit** | **Upper Limit** | **Data Type** | **Dimension** | **Init Value** |
| Vehicle characterization data | various | NA | NA | Simulink + IPG Data | NA | NA |
| Obstacle ID | ID | 0 | 3 | Integer | 3 | NA |
| **Outputs of the subsystem** | | | | | | |
| **Name / Description** | **Unit** | **Lower Limit** | **Upper Limit** | **Data Type** | **Dimension** | **Init Value** |
| Obstacle matrix for each scenario | m<br>rad<br>m/s | 0<br>0<br>0 | 80<br>6.28<br>6 | Matrix | 5x3 | 3 |

| Subsystem model specific information | |
|---|---|
| **Name of subsystem** | Trajectory Planners and Controllers |
| **From Partner** | US |
| **Use Case affiliation** | Use Case 1 (FO) |
| **Short description** | Trajectory Planner Development for the Truck-Trailer Reverse Manoeuvring for i.) Static Obstacles ii.) Dynamic Obstacles |
| **Known limitations** | Computational time of the algorithm |
| **Disturbances** | Sensor noise, or Model mismatch |



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.

| Inputs of the subsystem | | | | | | |
|---|---|---|---|---|---|---|
| **Name / Description** | **Unit** | **Lower Limit** | **Upper Limit** | **Data Type** | **Dimension** | **Init Value** |
| Vehicle States | m | N/A | N/A | Float | 6 | |
| Emergency Deceleration | $m/s^2$ | -1 | -5 | Float | 1 | -1 |
| Desired Deceleration | $m/s^2$ | 0 | -1 | Float | 1 | 0 |
| Desired Acceleration | $m/s^2$ | 0 | 0.5 | Float | 1 | 0 |
| Desired Yaw Rate | rad/s | 0 | 0.22 | Float | 1 | **0** |
| **Outputs of the subsystem** | | | | | | |
| **Name / Description** | **Unit** | **Lower Limit** | **Upper Limit** | **Data Type** | **Dimension** | **Init Value** |
| Steering Angle | deg | -900 | 900 | Float | 1 | 0 |
| Braking Torque | Nm | 0 | -5 | int | 1 | 0 |
| Acceleration Torque | Nm | 0 | 5 | int | 1 | 0 |



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.

| Subsystem model specific information | | | | | | |
|---|---|---|---|---|---|---|
| **Name of subsystem** | Trajectory Planners and Controllers | | | | | |
| **From Partner** | US | | | | | |
| **Use Case affiliation** | Use Case 1 (FO) | | | | | |
| **Short description** | Trajectory Controller Development for the Truck-Trailer Reverse Manoeuvring for i.) Static Obstacles ii.) Dynamic Obstacles | | | | | |
| **Known limitations** | Computational time of the algorithm, Road Boundaries, Limitation due to the vehicle | | | | | |
| **Disturbances** | Sensor noise, or Model mismatch | | | | | |
| **Inputs of the subsystem** | | | | | | |
| **Name / Description** | **Unit** | **Lower Limit** | **Upper Limit** | **Data Type** | **Dimension** | **Init Value** |
| Boundaries (Road) | m | N/A | N/A | Float | 2 | N/A |
| Vehicle States | m | N/A | N/A | Float | 6 | - |
| Position of the obstacle | m | N/A | N/A | Float | 2 | - |
| **Outputs of the subsystem (at least the relevant ones)** | | | | | | |
| **Name / Description** | **Unit** | **Lower Limit** | **Upper Limit** | **Data Type** | **Dimension** | **Init Value** |
| Desired Deceleration | $m/s^2$ | 0 | -1 | Float | 1 | 0 |
| Desired Acceleration | $m/s^2$ | 0 | 0.5 | Float | 1 | 0 |
| Desired Yaw Rate | rad/s | 0 | 0.20 | Float | 1 | 0 |

#### 3.2.3.2 Predefined settings

Some settings were predefined in previous deliverables of the project for the simulation of FO use cases. Those setting include the parameters of the T-ST vehicle and the use case environment. In addition, the settings of the hardware and simulation software used for the implementation of the FO use case.

##### 3.2.3.2.1 Parameters

After the correlation activities which will be reported in D3.5, the parameters defined for the MiL T-ST vehicle model, i.e. IPG TruckMaker vehicle model, are listed in Table 9.



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 9: MiL T-ST vehicle model parameters**

| **Truck Vehicle Parameters** | | | | |
|---|---|---|---|---|
| **Truck Vehicle Mass** | | | | |
| Curb Truck Weight ($m_t$) **before** hitching | | | 8,000 | kg |
| | Total Mass on | Front Wheel | 6,000 | kg |
| | | Rear Wheel | 2,000 | kg |
| Truck Weight **after** hitching (Coupled T-ST) with unloaded trailer | | | 10,000 | kg |
| | Total Mass on | Front Wheel | 6,000 | kg |
| | | Rear Wheel | 4,000 | kg |
| Truck Weight **after** hitching (Coupled T-ST) with loaded trailer | | | 18,000 | kg |
| | Total Mass on | Front Wheel | 8,000 | kg |
| | | Rear Wheel | 10,000 | kg |
| **Truck Vehicle Moment of Inertia** | | | | |
| Yaw Inertia ($I_{z,t}$) | No trailer | | 8000 | $kgm^2$ |
| | loaded trailer | | | $kgm^2$ |
| | Unloaded trailer | | | $kgm^2$ |
| **Truck Vehicle Geometry** | | | | |
| Distance from front axle to CG (Center of Gravity) ($l_{f,t}$) | | | 1.054 | m |
| Distance from Rear axle to CG ($l_{r,t}$) | | | 2.546 | m |
| Wheelbase ($l_t = l_{f,t} + l_{r,t}$) | | | 3.60 | m |
| Distance from front axle to Hitch Point ($l_{h,t}$) | | | 2.09 | m |
| Truck vehicle width ($w_t$) | | | 2.50 | m |
| Truck vehicle length ($L_t = l_{ro,t} + w_t + l_{fo,t}$) | | | 5.925 | m |
| **Steering Parameters** | | | | |
| Gear ratio between steering wheel and steering angle | | | 17-19 | - |
| **Truck Tier Lateral Parameters** | | | | |
| Cornering stiffness | Front Wheel | | 62500 | N/rad |
| | Rear Wheel | | 45000 | N/rad |
| **Trailer Parameters** | | | | |
| **Trailer Mass** | | | | |
| Total T-ST Curb Weight | 14,500,00 | | | kg |
| Total T-ST Gross Weight | 39,000,00 | | | kg |



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.

| | | | |
|---|---|---|---|
| Curb Trailer Weight ($m_{st,c}$) | 6,450 | | kg |
| | Mass on Each Wheel | 1,500 | kg |
| Gross Trailer Weight ($m_{st,g}$) | 31,000 | | kg |
| | Mass on Each Wheel | 7,000 | kg |
| **Trailer Moment of Inertia** | | | |
| Yaw Inertia ($I_{z,st}$) | Loaded trailer | | $kgm^2$ |
| | Unloaded trailer | | $kgm^2$ |
| **Trailer Geometry** | | | |
| Distance between axles | ($l_{li,st}$) | 1.315 | m |
| Distance from front axle to CG | ($l_{f,st}$) | 0.25 | m |
| Distance from middle axle to CG | ($l_{m,st}$) | 1.565 | m |
| Distance from rear axle to CG | ($l_{r,st}$) | 2.88 | m |
| Wheelbase | ($l_{st}$) | 4.9 | m |
| Distance from CG to hitch point | ($l_{h,st}$) | 3.445 | m |
| Trailer vehicle width | ($w_{st}$) | 2.55 | m |
| Trailer vehicle length | ($L_{st} = l_{ro,st} + w_{st} + l_{fo,st}$) | 9 | m |
| **Trailer Tier Lateral Parameters** | | | |
| Cornering stiffness | Each Wheel | 17000 | N/rad |

The parameters environment of the Use Case 1 of FO is listed in following tables and figures.

For the parameters, Use Case 1 (Docking station) is displayed as below:

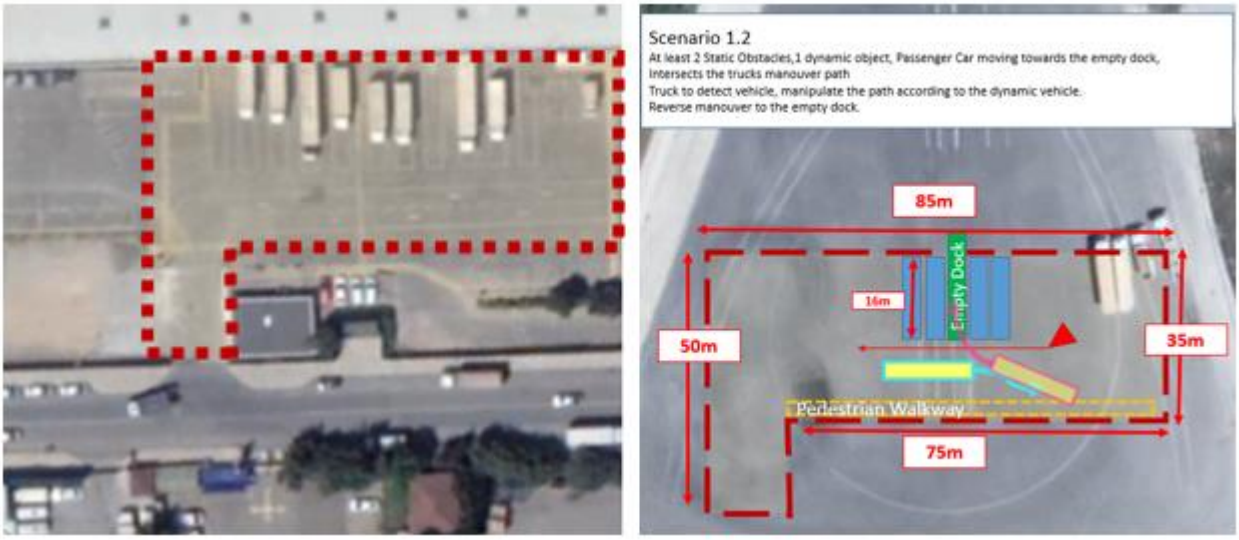


**Figure 30: Docking Station Environment will be demonstrated, Dimensions are defined.**



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.

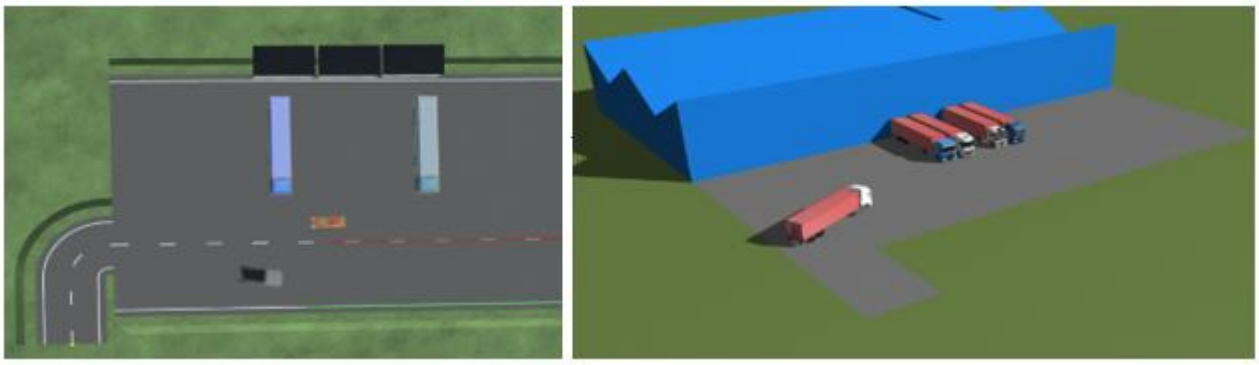

**Figure 31: Docking Station Environment is built in both IPG TruckMaker and PreScan**

For the construction site parametrization, same methods will be applied and environmental setup and perception will be accomplished via co-simulation framework. Current blocks of the perception are created within IPG-TruckMaker with object sensor models built in. Current perception outputs are designed as following:

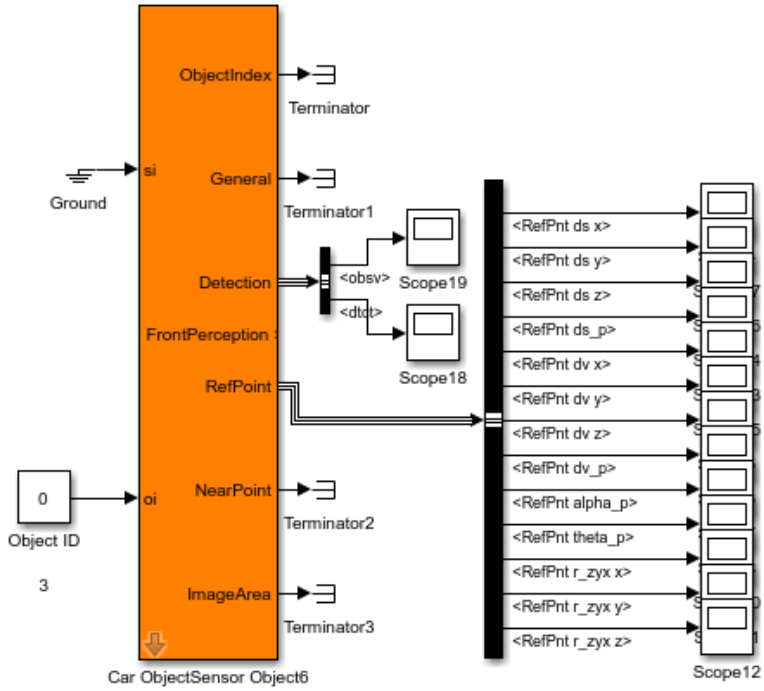


**Figure 32: IPG TruckMaker Object Sensor outputs**



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.

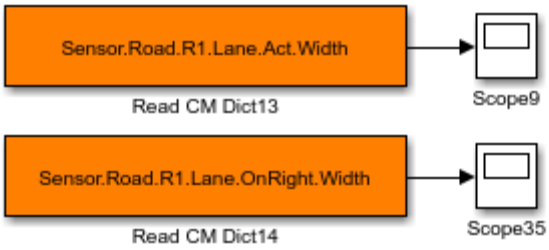


**Figure 33: IPG Truckmaker Road Sensor Configuration – Simulink**

With Truckmaker's Read Dict. output boxes, road sensor provides the truck frame's (Fr1) distance to the lane markings.

**Table 10: IPG Truckmaker Object Sensor Utilized Signals**

| | Data Type | Explanation | I/O - Block |
|---|---|---|---|
| | Data Type | Explanation | I/O - Block |
| **Object ID** | Integer, Constant | Integer that determines the object according to the truckmaker traffic reference | Input: Car Object Sensor |
| **RefPnt ds x** | Float, Variable [m] | Fr1 + Sensor Position - Traffic Object Frame X Distance | Output: Car Object Sensor |
| **RefPnt ds y** | Float, Variable [m] | Fr1 + Sensor Position - Traffic Object Frame Y Distance | Output: Car Object Sensor |
| **RefPnt r_zyx z** | Float, Variable [rad] | Fr1 to Traffic Object Frame relative angle, heading | Output: Car Object Sensor |
| **<pre>.Road.*SensorName*.Lane.Act.Width** | Float, Variable [m] | Actual Width gives the lave width | Output: Road Sensor |

For FO Docking use case simulations, three traffic objects are defined according to the scenario. Each object's information can be concatenated into time-series or separate arrays or can be connected to trajectory planner as the environmental outputs.



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.

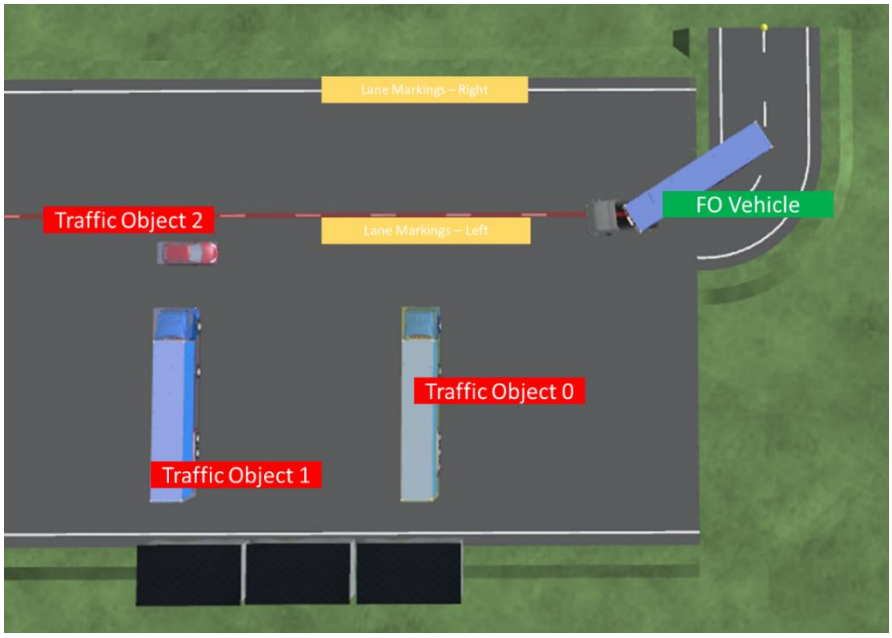


**Figure 34: Current Simulation Environment with Object ID's and Designated Lane Markings**

#### 3.2.3.2.2 Hardware and simulation software settings

The settings for the hardware and simulation software are given in this section. Table 11 list the settings of the essential components of the system. For better explanations of the co-simulation framework, abstract visuals are provided below:

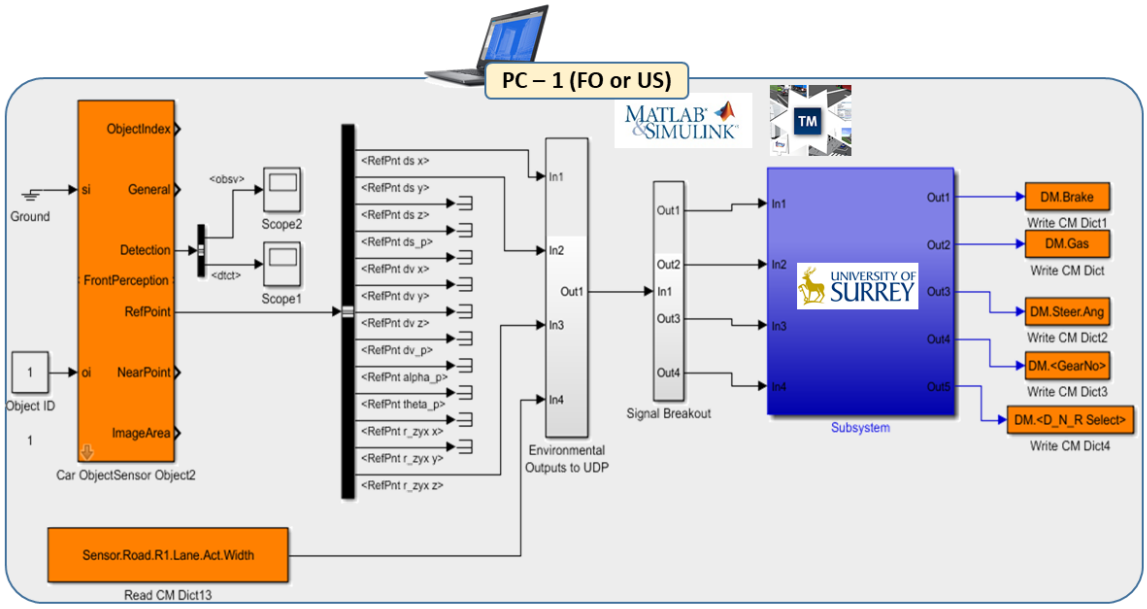


**Figure 35: I/O Visualization of the US-FO Blocks running simultaneously on single PC/Simulink/IPG – TM**



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.

There needs to be no message conversion or communication interface, since both software are running on a single PC, it is essential to combine the IPG project data created for sensor simulation and implement US blocks to the Vehicle Control subsystem of the IPG Simulink model, which FO sensor items from CM library will be located.

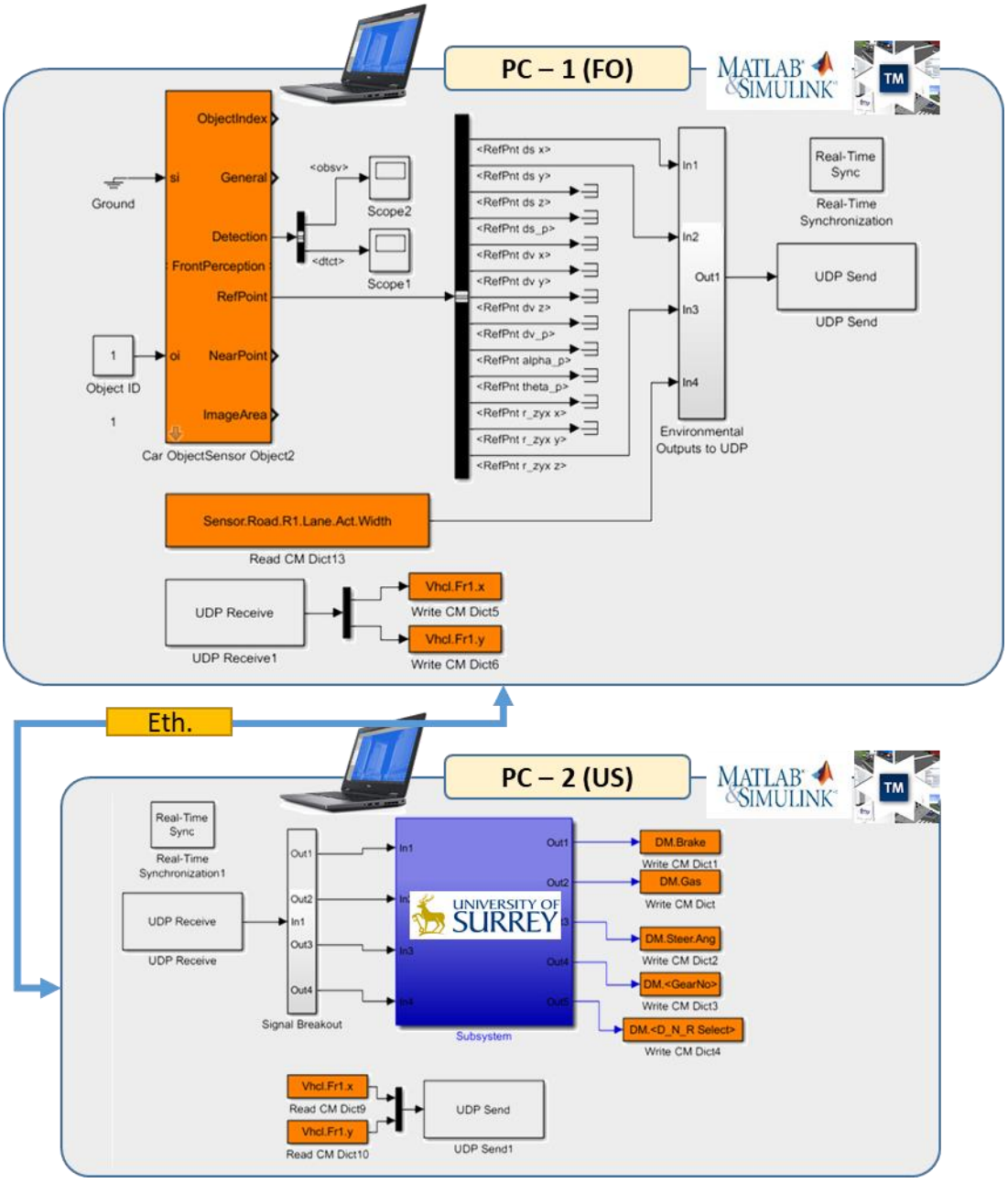


**Figure 36: Environmental Blocks running on FO PC, US Trajectory Block running on another PC.**

For separate hardware setup, UDP blocks and real time synchronization capabilities of Simulink to be combined with the IPG blocks of the FO's environment model and US' trajectory controller. PC-1 IPG is basically sourced with the vehicle position and providing sensor outputs, simultaneously PC-2 US



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.

controller is providing the vehicle controller signals to PC-2 IPG and sourcing the position to vehicle to the PC-1 IPG. For this setup, there needs to be two physical PC's and two separate IPG TM 6.0 licenses within the same physical environment. It may be beneficial to create data replays and ensure the UDP communication and I/O's are valid then connect the FO and US IPG –TMs.

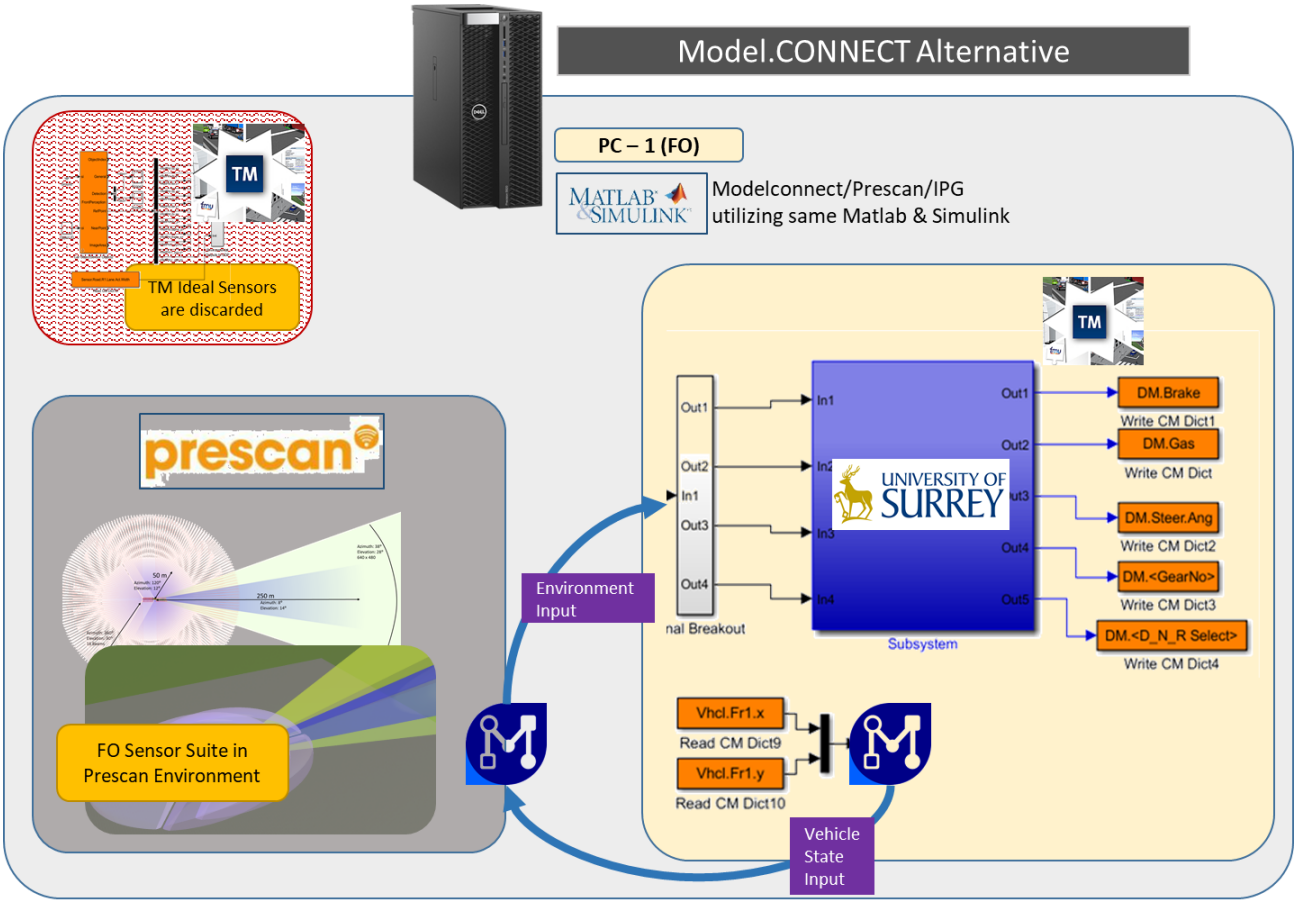


**Figure 37: Alternative to IPG – Simulink based co-simulation platforms, AVL's Modelconnect can be utilized as a middleware to communicate IPG/Simulink (US) and Prescan (FO) environments.**

If satisfactory results can be obtained from IPG-Simulink based co-simulation studies, which are the main route for FO-US co-simulation framework. AVL's Model.CONNECT can also be considered as a middleware between the environmental components and the vehicle trajectory and planning component. With abilities of Model.CONNECT, connecting Prescan's sensor outputs to the Truckmaker is being studied with FO and AVL teams. For Model.CONNECT alternative, it is needed to have a consolidated decision between US, FO and AVL teams before partner's efforts of Deliverable 3.6 start.



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 11: Hardware and simulation software settings**

| Truck-Semitrailer and environment subsystem | |
|---|---|
| **Simulation specific information** | |
| **Simulation tool & version** | Matlab/Simulink 2016a, IPG TruckMaker 6.0 |
| **Numerical solver** | ode3 |
| **Kind of numerical step-sizes** | fixed |
| **Numerical step-size** | fixed step-size - max step-size in case of variable step-size solver: 0.01 |
| **Real-time capable** | Yes |
| **Project licence available** | Yes |
| **Hardware specific information** | |
| **Hardware description** | Na |
| **Communication step-size** | Na |
| **Communication medium** | Na |

| Trajectory planner subsystem | |
|---|---|
| **Simulation specific information** | |
| **Simulation tool & version** | Matlab/Simulink 2016a, IPG TruckMaker 6.0 |
| **Numerical solver** | Ode45/Ode 3 |
| **Kind of numerical step-sizes** | fixed |
| **Numerical step-size** | fixed step-size: 0.01 |
| **Real-time capable** | Yes |
| **Project licence available** | Yes |
| **Hardware specific information** | |
| **Hardware description** | ECU(dSPACE MicroAutoBox) |
| **Communication step-size** | fixed step-size |
| **Communication medium** | CAN |

| Trajectory controller subsystem | |
|---|---|
| **Simulation specific information** | |
| **Simulation tool & version** | Matlab/Simulink 2016a, IPG TruckMaker 6.0 |
| **Numerical solver** | Ode45/Ode 3 |
| **Kind of numerical step-sizes** | fixed |
| **Numerical step-size** | fixed step-size: 0.01 |
| **Real-time capable** | yes |
| **Project licence available** | 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.

| Hardware specific information | |
|---|---|
| **Hardware description** | ECU(dSPACE MicroAutoBox) |
| **Communication step-size** | fixed step-size |
| **Communication medium** | CAN |

## 3.3 Tofaş Use Case

The final co-simulation setup used for the simulation of the TF use cases is introduced in this section. Components with different functionality and from different platforms are connected in this setup to simulate the L3AD system.

The overall co-simulation block diagram of the TF use cases is shown in Figure 38. The blocks in green are MiL vehicle model components from ASM dSpace high-fidelity software and they are used to simulate the mule vehicle. The orange blocks are interface blocks between the ASM dSpace and Matlab/Simulink platforms. The Simulink components are in grey.

Sensors are modelled in the MiL vehicle model and their data are given by the perception and the vehicle model. The perception interface processes the sensor data and gives as output signals about the surrounding road and obstacles to the trajectory planner interface. The latter in its turn make calculations on the signals at its inputs to provide the trajectory planner with the required signals to generate a safe and trustworthy reference trajectory. This reference, represented by reference velocities position and angles of a ghost vehicle, is given to the longitudinal and lateral trajectory controller to be followed.

The trajectory controller component provides the control signals to the ASM vehicle model. The handover switch components are placed in a cyan block.



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.

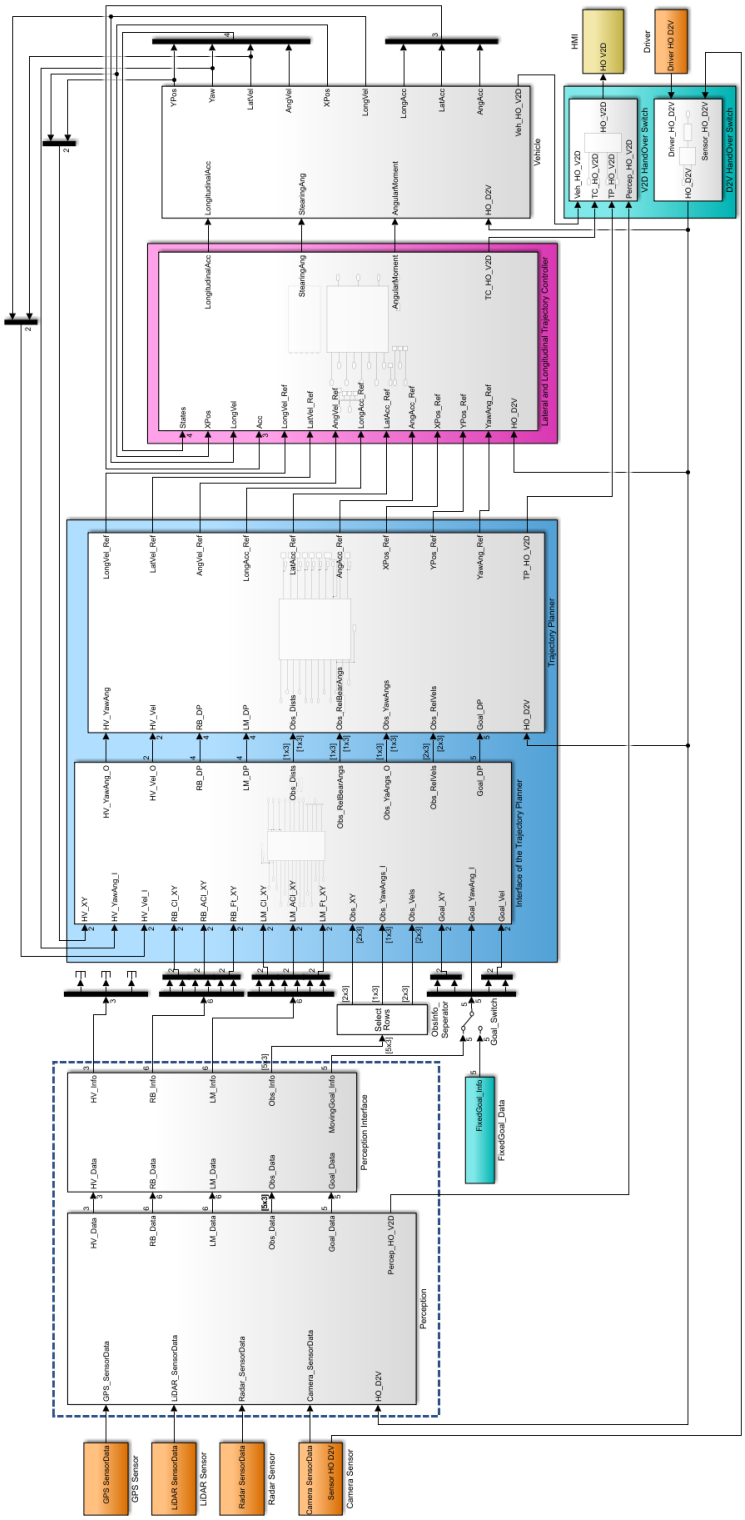


**Figure 38: Overall co-simulation block diagram for TF UC**



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 Components

As mentioned in the previous section the co-simulation block diagram consists of several components. Those components are

- Vehicle model component
- Perception model component
- Perception interface component
- Trajectory planner interface
- Trajectory planner
- Lateral and longitudinal trajectory controller component

The signals setup for each of the components is described in the following sections.

#### 3.3.1.1 Vehicle model component

The vehicle model component is responsible for reading signals data from the ASM vehicle model. This signals data is given by the sensors modelled in ASM and it contains information about the eLCV positions, velocities, yaw and hitch angle, ... etc. This data is passed to the trajectory planner interface and to the longitudinal and lateral trajectory controller components.

The list of signals that are given as an output of the vehicle model component are listed in the following table with related descriptions.

**Table 12: Input and output signals of the vehicle model component**

| Sub-system | Signals Name | Signal description | Symbol | Type | Size | Unit | I/O |
|---|---|---|---|---|---|---|---|
| Vehicle | VehicleYawAngle | Yaw angle of Vehicle | $\psi$ | Scalar | 1x1 | deg | O |
| | VehicleSpeed | Speed of Vehicle | $V$ | Scalar | 1x1 | m/s | O |
| | ACCPedal | Position of acceleration pedal of vehicle | $ACC_p$ | Scalar | 1x1 | % | I |
| | BrakePedal | Position of brake pedal of vehicle | $BRK_p$ | Scalar | 1x1 | % | I |
| | Steering Angle | Steering Angle | $\delta$ | Scalar | 1x1 | deg | I |



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.

#### 3.3.1.2 Perception component

As in the case of the vehicle model component, the perception component is dedicated to obtain signals data from dSpace ASM sensors. The data has information about the environment surrounding the eLCV, namely Road Boarder (RB), Lane Markings (LM) and obstacles. This data is passed to the perception interface component.

The input and output signals of this component are listed in the following table.

**Table 13: Input and output signals of the perception component**

| Sub-system | Signals Name | Signal description | Symbol | Type | Size | Unit | I/O |
|---|---|---|---|---|---|---|---|
| Perception | Line1DistanceToVehY | First lane border distance to vehicle on Y-axis | N/A | Scalar | 1x1 | m | O |
| | Line2DistanceToVehY | Second lane border distance to vehicle on Y-axis | N/A | Scalar | 1x1 | m | O |
| | LaneCenterDistanceToVehY | Lane center distance to vehicle on Y- axis | N/A | Scalar | 1x1 | m | O |
| | Line1DistanceToVehX | First lane border distance to vehicle on X-axis | N/A | Scalar | 1x1 | m | O |
| | Line2DistanceToVehX | Second lane border distance to vehicle on Y-axis | N/A | Scalar | 1x1 | m | O |
| | LaneCenterDistanceToVehX | Lane center distance to vehicle on X - axis | N/A | Scalar | 1x1 | m | O |
| | SensorDistanceInfoInM | Obstacle Distances | N/A | Vector | Can be Adjusted | m | O |
| | SensorBearingAnglesinRad | Obstacle Bearings | N/A | Vector | Can be Adjusted | rad | O |
| | SensorRelativeSpeedInMs | Obstacle Relative Speeds in m/s | N/A | Vector | Can be Adjusted | m/s | O |

#### 3.3.1.3 Perception interface component

The perception interface component extract from the data given by the perception component the required information by the trajectory planner interface.



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 outputs of the perception block are all given as input signals of the perception interface component. Therefore, to avoid repetition only the list of output signals of the perception interface component is given in Table 16. Furthermore, these parameters will be adjusted according to trajectory planner. This is why they are not given in here yet. Those will be presented along with trajectory planning integration. Since input signals are directly using as input of trajectory planner, related signals are below table 16.

**Table 14: Output signals of the perception interface component**

| Sub-system | Signals Name | Signal description | Symbol | Type | Size | Unit | I/O |
|---|---|---|---|---|---|---|---|
| Perception | Yaw Angle of the HW | Host vehicles Yaw angle | $\psi_e$ | Scalar | 1x1 | Rad | I |
| | Velocity of the HW | Host Vehicles Velocity | $\begin{pmatrix} v_{x,e} \\ v_{y,e} \end{pmatrix}$ | Vector | 2x1 | m/s | I |
| | Vector Road Boarder Info | Vectoral information road borders | $(\rho_{e,cl,rb}, \theta_{cl,e,rb}, \Delta\theta_{e,rb}, \psi_{cl,rb})^T$ | Vector | 4x1 | (m,rad rad,rad) | I |
| | Vector of Lane Marking Info | Information of lanes on the road | $(\rho_{e,cl,lm}, \theta_{cl,e,lm}, \Delta\theta_{e,lm}, \psi_{cl,lm})^T$ | Vector | 4x1 | (mrad radrad) | I |
| | Vector of distances to Obstacles | Distance between objects and vehicle | $(\rho_1 \quad \dots \quad \rho_n)$ | Vector | 1xn | m | I |
| | Vector of obstacles RBA to HV | The relative bearing angles of n obstacles to the host HV | $(\theta_1 \quad \dots \quad \theta_n)$ | Vector | 1xn | rad | I |
| | Vector of Relative Velocities | Relative velocity between the host HV and n obstacles | $(v_{rel,e,1} \cdots v_{rel,e,n}$ | Vector | 1xn | m/s | I |

### 3.3.1.4 Trajectory planner interface component

Since the trajectory planner requires certain information from sensors about the surrounding environment to produce the reference trajectory. An interface component for the trajectory planner was developed to provide the right signals based on the signals data given by the sensors. For details on the calculations carried by this component please refer to D3.3 (“Report on model repository” chapter 3.2).

The input signals that are used by this component are listed in the following table.



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 15: Input signals of the interface of the trajectory planner component**

| Sub-system | Signals Name | Signal description | Symbol | Type | Size | Unit | I/O |
|---|---|---|---|---|---|---|---|
| Interface of the trajectory planner | HV _XY | Coordinates of the CoG of the HV HV on the X and Y axis of the inertia (global) frame | $\begin{pmatrix} x_e \\ y_e \end{pmatrix}$ | Vector | 2x1 | m | I |
| | HV_YawAng_I | Yaw angle of the HV | $\psi_e$ | Scalar | 1x1 | rad | I |
| | HV_Vel_I | Longitudinal and Lateral velocities of the *HV* | $\begin{pmatrix} v_{x,e} \\ v_{y,e} \end{pmatrix}$ | Vector | 2x1 | m/s | I |
| | RB_Cl_XY | Coordinates of the Cl point on the closest RB on X and Y axis of the inertia frame | $\begin{pmatrix} x_{cl,rb} \\ y_{cl,rb} \end{pmatrix}$ | Vector | 2x1 | m | I |
| | RB_ACl_XY | Coordinates of the ACl point on the closest RB on X and Y axis of the inertia frame | $\begin{pmatrix} x_{acl,rb} \\ y_{acl,rb} \end{pmatrix}$ | Vector | 2x1 | m | I |
| | RB_Ft_XY | Coordinates of the Ft point on the closest RB on X and Y axis of the inertia frame | $\begin{pmatrix} x_{ft,rb} \\ y_{ft,rb} \end{pmatrix}$ | Vector | 2x1 | m | I |
| | LM_Cl_XY | Coordinates of the Cl point on the closest LM on X and Y axis of the inertia frame | $\begin{pmatrix} x_{cl,lm} \\ y_{cl,lm} \end{pmatrix}$ | Vector | 2x1 | m | I |
| | LM_ACl_XY | Coordinates of ACl point on the closest LM on X and Y axis of the inertia frame | $\begin{pmatrix} x_{acl,lm} \\ y_{acl,lm} \end{pmatrix}$ | Vector | 2x1 | m | I |
| | LM_Ft_XY | Coordinates of the Fl point on the closest LM on X and Y axis of the inertia frame | $\begin{pmatrix} x_{ft,lm} \\ y_{ft,lm} \end{pmatrix}$ | Vector | 2x1 | m | I |
| | Obs_XY | Coordinates of the CoG of n obstacles on the X and Y axis of the inertia frame | $\begin{pmatrix} x_1 & \dots & x_n \\ y_1 & \dots & y_n \end{pmatrix}$ | Matrix | 2xn | m | I |
| | Obs_YawAngs | Yaw angle of n obstacles | $(\psi_1 \quad \dots \quad \psi_n)$ | Vector | 1xn | rad | I |
| | Obs_Vels | Longitudinal and lateral velocities of n obstacles | $\begin{pmatrix} v_{x,1} & \dots & v_{x,n} \\ v_{y,1} & \dots & v_{y,n} \end{pmatrix}$ | Matrix | 2xn | m/s | I |
| | Goal_XY | Coordinates of the CoG of the goal on the X and Y axis of the inertia frame | $\begin{pmatrix} x_g \\ y_g \end{pmatrix}$ | Vector | 2x1 | m | I |
| | Goal_YawAng_I | Yaw angle of the goal | $\psi_g$ | Scalar | 1x1 | rad | I |
| | Goal_Vel | Longitudinal and lateral velocities of the goal | $\begin{pmatrix} v_{x,g} \\ v_{y,g} \end{pmatrix}$ | Vector | 2x1 | m/s | I |

The outputs of the interface of the trajectory planner block are the same as the inputs of the trajectory planner block. However, the trajectory planner block has an addition input from the Handover switch block. The inputs of the trajectory planner block are shown in the following table.



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.

### 3.3.1.5 Demonstration of the interface

Figure 39 demonstrates the input signals of the trajectory planner interface on a scenario. The point S is the place of the sensor used to detect the road boarder and lane marking. In addition, the points $P_{Cl}$ and $P_{Ft}$ are the closest point and the furthest point to the vehicle on the closet road boarder (i.e. right boarder). The point $P_{ACl}$ is the adjacent point to the $P_{Cl}$. The position of those mentioned points on the X and Y axis of the inertia frame are inputs to the trajectory planner interface. As a suggestion, the RB and LM sensor max range could be taken as 8m and field of view azimuth as 45°.

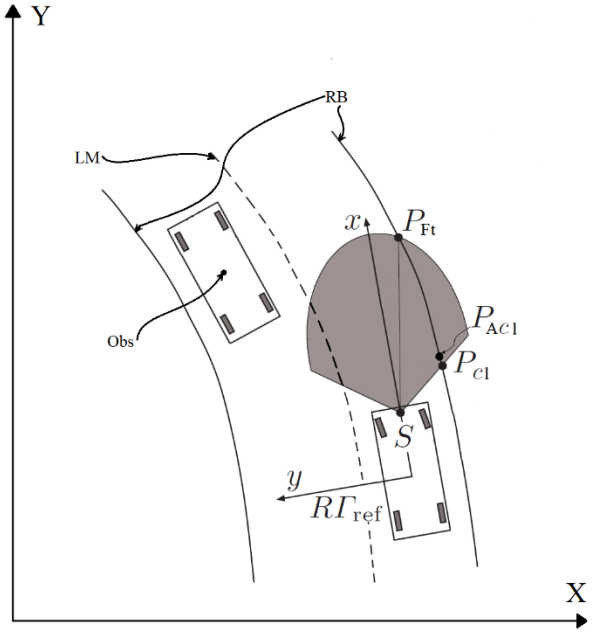


**Figure 39: Demonstration of the inputs of the trajectory planner interface on a scenario**

The output signals of the interface of the trajectory planner block are directly connected to the inputs of the trajectory planner block. However, the trajectory planner block has an additional input from the Handover switch block. Therefore, to avoid repetition, only the input signals of the trajectory planner block will be listed in the following section.

### 3.3.1.6 Trajectory planner component

The trajectory planner component generated the reference trajectories depending on the information given about the environment and the current state of the T-ST. It uses the concept of reference ghost vehicle; thus, its output signals are the current state of this vehicle. The input signals together with the output signals of the trajectory planner component are listed in Table 16.



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 16: Input signals of the trajectory planner component**

| Sub-system | Signals Name | Signal description | Symbol | Type | Size | Unit | I/O |
|---|---|---|---|---|---|---|---|
| Trajectory planner | HV_YawAng | Yaw angle of the HV | $\psi_e$ | Scalar | 1x1 | rad | I |
| | HV_Vel | Longitudinal and Lateral velocities of the HV | $\begin{pmatrix} v_{x,e} \\ v_{y,e} \end{pmatrix}$ | Vector | 2x1 | m/s | I |
| | RB_DP | Data regarding the RB | $(\rho_{e,cl,rb}, \theta_{cl,e,rb}, \Delta\theta_{e,rb}, \psi_{cl,rb})^T$ | Vector | 4x1 | m, rad, rad, rad | I |
| | LM_DP | Data regarding the LM | $(\rho_{e,cl,lm}, \theta_{cl,e,lm}, \Delta\theta_{e,lm}, \psi_{cl,lm})^T$ | Vector | 4x1 | m rad rad rad | I |
| | Obs_Dists | The distances between the HV and n obstacles | $(\rho_1 \quad \dots \quad \rho_n)$ | Vector | 1xn | m | I |
| | Obs_RelBearAngs | The relative bearing angles of n obstacles to the host HV | $(\theta_1 \quad \dots \quad \theta_n)$ | Vector | 1xn | rad | I |
| | Obs_YawAngs | Yaw angle of the n obstacle | $(\psi_1 \quad \dots \quad \psi_n)$ | Vector | 1xn | rad | I |
| | Obs_RelVels | Relative velocity between the host HV and n obstacles | $(v_{rel,e,1} \dots v_{rel,e,n})$ | Vector | 1xn | m/s | I |
| | Goal_DP | Data regarding the goal | $(\rho_{e,g}, \theta_{g,e}, \psi_g, v_{x,rel,e,g}, v_{y,rel,e,g})^T$ | Vector | 5x1 | m rad, rad, m/s, m/s | I |
| | HO_D2V | Handover flag to give control from the drive to the vehicle | $\text{Flag}_{HO}$ | Boolean | 1x1 | - | I |
| | LongVel_Ref | Longitudinal velocity of the GV | $v_{x,e,ref}$ | Scalar | 1x1 | m/s | O |
| | LatVel_Ref | Lateral velocity of the GV | $v_{y,e,ref}$ | Scalar | 1x1 | m/s | O |
| | AngVel_Ref | Angular velocity of the GV | $\omega_{z,e,ref}$ | Scalar | 1x1 | rad/s | O |
| | LongAcc_Ref | Longitudinal acceleration of the GV | $\dot{v}_{x,e,ref}$ | Scalar | 1x1 | $m/s^2$ | O |
| | LatAcc_Ref | Lateral acceleration of the GV | $\dot{v}_{y,e,ref}$ | Scalar | 1x1 | $m/s^2$ | O |
| | AngAcc_Ref | Angular acceleration of the GV | $\dot{\omega}_{z,e,ref}$ | Scalar | 1x1 | $rad/s^2$ | O |



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.

| Sub-system | Signals Name | Signal description | Symbol | Type | Size | Unit | I/O |
|---|---|---|---|---|---|---|---|
| | XPos_Ref | Coordinates of the CoG of the GV on the X-axis of the inertia frame | $x_{e,ref}$ | Scalar | 1x1 | m | O |
| | YPos_Ref | Coordinates of the CoG of the GV on the Y-axis of the inertia frame | $y_{e,ref}$ | Scalar | 1x1 | m | O |
| | YawAng_Ref | Yaw angle of the GV | $\psi_{e,ref}$ | Scalar | 1x1 | rad | O |
| | TP_HO_V2D | Handover flag of the trajectory planner to give control from the vehicle to driver | $Flag_{Tp}$ | Boolean | 1x1 | - | O |

For simplification, it is assumed that all obstacles are aligned with the road boarder/lane marking to which they are close. Therefore, the yaw angle of the obstacle could be assumed equal to the road boarder/lane marking tangential angle. In addition, the goal of the eLCV is set to be constant and no information will be provided by the environment interface component about a moving eLCV goal.

#### 3.3.1.7 Lateral and longitudinal controller component

The trajectory controller component receives the reference trajectory and current state of the T-ST as listed in Table 17. To track the reference, the trajectory controller provides the control signal needed as output signals listed in the same table.

**Table 17: Input and output signals of the lateral and longitudinal trajectory controller component**

| Sub-system | Signals Name | Signal description | Symbol | Type | Size | Unit | I/O |
|---|---|---|---|---|---|---|---|
| Lateral and longitudinal controller | RefSpeed | Desired speed of vehicle | $V_{REF}$ | Scalar | 1x1 | $m/s$ | I |
| | VehicleSpeed | Speed of vehicle | $V_{VEH}$ | Scalar | 1x1 | $m/s$ | I |
| | DeltaX | Difference between vehicle and target point on global X axis | $\Delta X$ | Scalar | 1x1 | $m$ | I |
| | DeltaY | Difference between vehicle and target point on global Y axis | $\Delta Y$ | Scalar | 1x1 | $m$ | I |
| | YawAngle | Yaw angle of vehicle | $\psi\_veh$ | Scalar | 1x1 | $rad$ | I |
| Lateral and | ACCPedal | Position of acceleration pedal of vehicle | $ACC_p$ | Scalar | 1x1 | % | O |



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.

| Sub-system | Signals Name | Signal description | Symbol | Type | Size | Unit | I/O |
|---|---|---|---|---|---|---|---|
| | BrakePedal | Position of brake pedal of vehicle | $BRK_p$ | Scalar | 1x1 | % | O |
| | Steering Angle | Desired Steering Angle | $\delta_{des}$ | Scalar | 1x1 | $rad$ | I |

### 3.3.2 Usage

#### 3.3.2.1 System limitations

In this section the limitation imposed on the system and on its input and output signals are defined. Some of the limitations where already listed in D3.1 ("Requirements for the co-simulationframework" chapter 2.2) for earlier stages of the project, however the final setup is given here. The following table lists the limitations for some of the system components.

**Table 18: System components and signals limitations**

| **Subsystem model specific information** | | | | | | |
|---|---|---|---|---|---|---|
| **Name of subsystem** | LCV Dynamics | | | | | |
| **From Partner** | Tofaş | | | | | |
| **Use Case affilation** | Use Case 2 (TF) | | | | | |
| **Short description** | Subsystem will be used for modelling vehicle dynamics on simulation environment. Baseline reference vehicle will be modelled and 2 dedicated scenarios will run according to this model. | | | | | |
| **Known limitations** | Vehicle model with base parameters. High fidelity simulation will be modelled additionally. | | | | | |
| **Inputs of the subsystem** | | | | | | |
| **Name / Description** | **Unit** | **Lower Limit** | **Upper Limit** | **Data Type** | **Dimension** | **Init Value** |
| Shifter lever | - | 0 | 3 | Double | 1 | Neutral (0) |
| upshift request | - | - | - | Bool | 1 | FALSE |
| downshift request | - | - | - | Bool | 1 | FALSE |
| Throttle | **%** | 0 | 100 | Double | 1 | 0 |
| Brake Force | Nm | 0 | tbd | Double | 1 | 0 |
| Ground height level | m | 10 | 100 | Double | 1 | 0 |
| Grip level at wheel | % | 10 | 100 | Double | 1 | 100 |
| Steering Wheel Angle | Degree | -489 | 489 | Double | 1 | 0 |
| **Outputs of the subsystem** | | | | | | |
| **Name / Description** | **Unit** | **Lower Limit** | **Upper Limit** | **Data Type** | **Dimension** | |
| Longitudinal acceleration | m/s2 | -2.2 | 2 | Double | 1 | |
| Vehicle Speed | m/s | 0 | 145 | Double | 1 | |



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.

| | | | | | | |
|---|---|---|---|---|---|---|
| Vehicle position | m | 0 | tbd | Double | 1-by-n | |
| Vehicle orientation | deg | 0 | 360 | Double | 1 | |
| wheel travel | mm | 0 | 120 | Double | 1 | |
| wheel speed | rpm | 0 | 1200 | Double | 1 | |
| e-motor speed | rpm | 0 | 12000 | Double | 1 | |

| **Subsystem model specific information** | | | | | | |
|---|---|---|---|---|---|---|
| **Name of subsystem** | Trajectory Planning | | | | | |
| **From Partner** | Tofaş | | | | | |
| **Use Case affilation** | Use Case 2 (TF) | | | | | |
| **Short description** | Subsystem will be used for for controlling visual models. Trajectory planners and control alghorithms will be developed for 2 different scenarios. | | | | | |
| **Known limitations** | Trajectory planners and controlling alghorithms will be developed according to L3AD SW requirements and when a sensor fail or an uncoverable condition occurs, the system will hand over to driver. No limitation has been foreseen additionally. | | | | | |
| **Inputs of the subsystem** | | | | | | |
| **Name / Description** | **Unit** | **Lower Limit** | **Upper Limit** | **Data Type** | **Dimension** | **Init Value** |
| Object Lists from sensors | - | - | - | Double | n-by-m | - |
| **Outputs of the subsystem** | | | | | | |
| **Name / Description** | **Unit** | **Lower Limit** | **Upper Limit** | **Data Type** | **Dimension** | **Inıt Value** |
| Automated system status | - | **0** | **1** | Bool | 1 | FALSE |
| Lateral offset request | M | -0,30 | 0,30 | Double | **1** | 0 |
| Vehicle speed | m/s | 0 | 10 | Double | 1 | 0 |
| Obstacle detected | - | **0** | **1** | Bool | **1** | **FALSE** |
| distance to target vehicle | M | 0 | 255 | double | 1 | 0 |
| look-ahead distance (LAD) | M | 0 | 80 | double | 1-by-n | 0 |
| lateral offset at LAD | M | -0,3 | 0,3 | double | 1-by-n | - |
| angular deviation at LAD | rad | - | - | double | 1-by-n | - |
| road curvature at LAD | 1/m | - | - | double | 1-by-n | - |
| Target vs Ego vehicle velocity difference | m/s | -100 | 50 | double | 1 | 0 |
| Max acc @longitudinal | $m/s^2$ | >0 | - | double | 1 | 2 |
| Min acc @longitudinal | $m/s^2$ | - | <0 | double | 1 | -2,2 |



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 | S | 1 | 2,2 | double | 1 | 1 |
| desired ego velocity | m/s | 0 | 10 | double | 1 | 16 |
| max. ego velocity | m/s | 0 | 10 | double | 1 | 36 |
| actual ego velocity | m/s | 0 | 10 | double | 1 | - |
| minimum cleareance to target | M | 0 | - | double | 1 | 2 |
| Steering Wheel Angle | Degree | -489 | 489 | Double | | 0 |

| **Subsystem model specific information** | | | | | | |
|---|---|---|---|---|---|---|
| **Name of subsystem** | Controlling Alghorithms | | | | | |
| **From Partner** | Tofaş | | | | | |
| **Use Case affilation** | Use Case 2 (TF) | | | | | |
| **Short description** | Subsystem will be used for for controlling visual models. Trajectory planners and control alghorithms will be developed for 2 different scenarios. | | | | | |
| **Known limitations** | Trajectory planners and controlling alghorithms will be developed according to L3AD SW requirements and when a sensor fail or an uncoverable condition occurs, the system will hand over to driver. No limitation has been foreseen additionally. | | | | | |
| **Inputs of the subsystem** | | | | | | |
| **Name / Description** | **Unit** | **Lower Limit** | **Upper Limit** | **Data Type** | **Dimension** | **Init Value** |
| Enable signal | - | | | Bool | 1 | FALSE |
| Lateral offset request | m | tbd | tbd | Double | 1 | 0 |
| Vehicle speed | m/s | 0 | 37 | Double | 1 | 0 |
| Obstacle detected | - | - | - | Bool | 1 | FALSE |
| distance to target vehicle | m | 0 | - | double | 1 | 0 |
| look-ahead distance (LAD) | m | 0 | 80 | double | 1-by-n | - |
| lateral offset at LAD | m | - | - | double | 1-by-n | - |
| angular deviation at LAD | rad | - | - | double | 1-by-n | - |
| road curvature at LAD | 1/m | - | - | double | 1-by-n | - |
| Ego vs Target velocity difference | m/s | - | - | double | 1 | 0 |
| Maximum acc @longitudinal | $m/s^2$ | >0 | - | double | 1 | 2 |
| Minimum acc @longitudinal | $m/s^2$ | - | <0 | double | 1 | -2,5 |
| desired time gap to target vehicle | s | 1 | 2,2 | double | 1 | 1 |



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 ego velocity | m/s | 0 | 10 | double | 1 | 16 |
| max. ego velocity | m/s | 0 | 10 | double | 1 | 36 |
| actual ego velocity | m/s | 0 | 10 | double | 1 | - |
| minimum clearance to target | m | >0 | - | double | 1 | 2 |
| Steering Wheel Angle | Degree | -489 | 489 | Double | | 0 |
| **Outputs of the subsystem** | | | | | | |
| **Name / Description** | **Unit** | **Lower Limit** | **Upper Limit** | **Data Type** | **Dimension** | |
| actual look-ahead distance | m | 0 | 80 | double | 1 | |
| actual lateral offset | m | - | - | double | 1 | |
| desired longitudinal acceleration | m/s² | -2,2 | 2 | double | 1 | 0 |
| actual angular deviation | rad | - | - | double | 1 | |

| **Subsystem model specific information** | | | | | | |
|---|---|---|---|---|---|---|
| **Name of subsystem** | Environment | | | | | |
| **From Partner** | Tofaş | | | | | |
| **Use Case affilation** | Use Case 2 (TF) | | | | | |
| **Short description** | Subsystem is the simulation environment itself for virtual use case validations. | | | | | |
| **Known limitations** | Simulation environment a number of moving and static obstacles in its library to create realistic environmental conditions but might not cover all the real-life conditions. | | | | | |
| **Inputs of the subsystem** | | | | | | |
| **Name / Description** | **Unit** | **Lower Limit** | **Upper Limit** | **Data Type** | **Dimension** | **Init Value** |
| Sensor specific model inputs | - | - | - | - | - | - |
| **Outputs of the subsystem (at least the relevant ones)** | | | | | | |
| **Name / Description** | **Unit** | **Lower Limit** | **Upper Limit** | **Data Type** | **Dimension** | |
| Sensor specific model outputs | - | - | - | - | - | |

### 3.3.2.2 Predefined settings

Some settings were predefined in previous deliverables of the project for the simulation of TF use cases. Those setting include the parameters of the eLCV vehicle and the use case environment. In addition, the settings of the hardware and simulation software used for the implementation of the TF use case.



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.

#### 3.3.2.2.1 Parameters

After the corelation activities which are reported in D3.5, the parameters defined for the MiL eLCV vehicle model, i.e. dSpace ASM vehicle model, are listed in Table 19.

**Table 19: MiL eLCV vehicle model parameters**

| **eLCV Vehicle Parameters** | | | | |
|---|---|---|---|---|
| **eLCV Vehicle Mass** | | | | |
| Curb eLCV Weight ($m_e$) | wheelbase | | | kg |
| | Total Mass on | Front Left Wheel | 456,5 | kg |
| | | Front Right Wheel | 452 | kg |
| | | Rear Left Wheel | 401 | kg |
| | | Rear Right Wheel | 398 | kg |
| | Sprung Mass | Front Left Wheel | 405,5 | kg |
| | | Front Right Wheel | 401 | kg |
| | | Rear Left Wheel | 353 | kg |
| | | Rear Right Wheel | 350 | kg |
| | Unsprung Mass | Front Left Wheel | 51 | kg |
| | | Front Right Wheel | 51 | kg |
| | | Rear Left Wheel | 48 | kg |
| | | Rear Right Wheel | 48 | kg |
| **Driver Mass** | | | 75 | kg |
| **eLCV Vehicle Moment of Inertia** | | | | |
| Yaw Inertia | | ($I_{z,e}$) | 3013 | $kgm^2$ |
| **eLCV Vehicle Geometry** | | | | |
| Distance from front axle to CG (Center of Gravity) | | ($l_{f,e}$) | 1453 | mm |
| Distance from Rear axle to CG | | ($l_{r,e}$) | 1652 | mm |
| Wheelbase | | ($l_e = l_{f,e} + l_{r,e}$) | 3105 | mm |
| eLCV vehicle width | | ($w_e$) | Wheel track Front:1510 Rear: 1530 | mm |
| Rear Overhang | | ($l_{ro,e}$) | 740 | mm |
| Front Overhang | | ($l_{fo,e}$) | 911 | mm |
| eLCV vehicle length | | ($L_e = l_{ro,e} + l_e + l_{fo,e}$) | 4756 | mm |
| CG height | | ($h_{cg,e}$) | 580 | mm |



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.

#### 3.3.2.2.2 Hardware and simulation software settings

The settings for the hardware and simulation software are given in section. Table 20 list the settings of the essential components of the system.

**Table 20: Hardware and simulation software settings**

| **Vehicle model** | |
|---|---|
| **Simulation specific information** | |
| **Simulation tool & version** | dSpace |
| **Numerical solver** | Simulink |
| **Kind of numerical step-sizes** | Fixed |
| **Numerical step-size** | 1ms |
| **Real-time capable** | Yes |
| **Project licence available** | |
| **Hardware specific information** | |
| **Hardware description** | NA |

| **Trajectory planner** | |
|---|---|
| **Simulation specific information** | |
| **Simulation tool & version** | Matlab/Simulink |
| **Numerical solver** | Ode |
| **Kind of numerical step-sizes** | Fixed |
| **Numerical step-size** | 1ms |
| **Real-time capable** | Yes |
| **Project licence available** | |
| **Hardware specific information** | |
| **Hardware description** | NA |
| **Communication step-size** | NA |
| **Communication medium** | NA |

| **Trajectory controller** | |
|---|---|
| **Simulation specific information** | |
| **Simulation tool & version** | Matlab/Simulink 2017a |
| **Numerical solver** | Ode |
| **Kind of numerical step-sizes** | Fixed |
| **Numerical step-size** | 1ms |
| **Real-time capable** | Yes |
| **Project licence available** | |
| **Hardware specific information** | |



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.

| | |
|---|---|
| **Hardware description** | NA |
| **Communication step-size** | NA |
| **Communication medium** | NA |

| **Environment** | |
|---|---|
| **Simulation specific information** | |
| **Simulation tool & version** | dSpace |
| **Numerical solver** | - |
| **Kind of numerical step-sizes** | Fixed |
| **Numerical step-size** | 1ms |
| **Real-time capable** | Yes |
| **Project licence available** | |
| **Hardware specific information** | |
| **Hardware description** | Mule Vehicle for sensor data collection/correlation activities will be built. |

| **Simulation specific information** | |
|---|---|
| **Simulation tool & version** | Matlab/Simulink, C/C++ |
| **Numerical solver** | - |
| **Kind of numerical step-sizes** | - |
| **Numerical step-size** | - |
| **Real-time capable** | ? |
| **Project licence available** | - |

| **Hardware specific information** | |
|---|---|
| **Hardware description** | ECU |
| **Communication step-size** | - |
| **Communication medium** | UDP, CAN, Ethernet |



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.

## 3.4 Volvo Use Case

In this use case the increase in automated driving function availability is studied. For this a sensor monitoring module is used. Its purpose is to observe and classify sensor faults. The information given by the sensor monitoring module regarding sensor availability (either permanently or temporarily if for example a sensor cleaning system is integrated) is used in the driving function setup, beginning in the sensor fusion and hence environment modeling and further in the selection of the respective functionalities that are available given the status of the environment model.

### 3.4.1 Driving Function

The automated driving function for the VCC use case has to be a L3AD function and it has to be able to cope with sensor failures appropriately. Since this function was descripted in D3.3 "Report on model repository" chapter 3.4, this section will give only a short overview of the functionalities that were already implemented at this time and instead focuses on the changes that were made.

The driving function consist of three main blocks (see Figure 40):

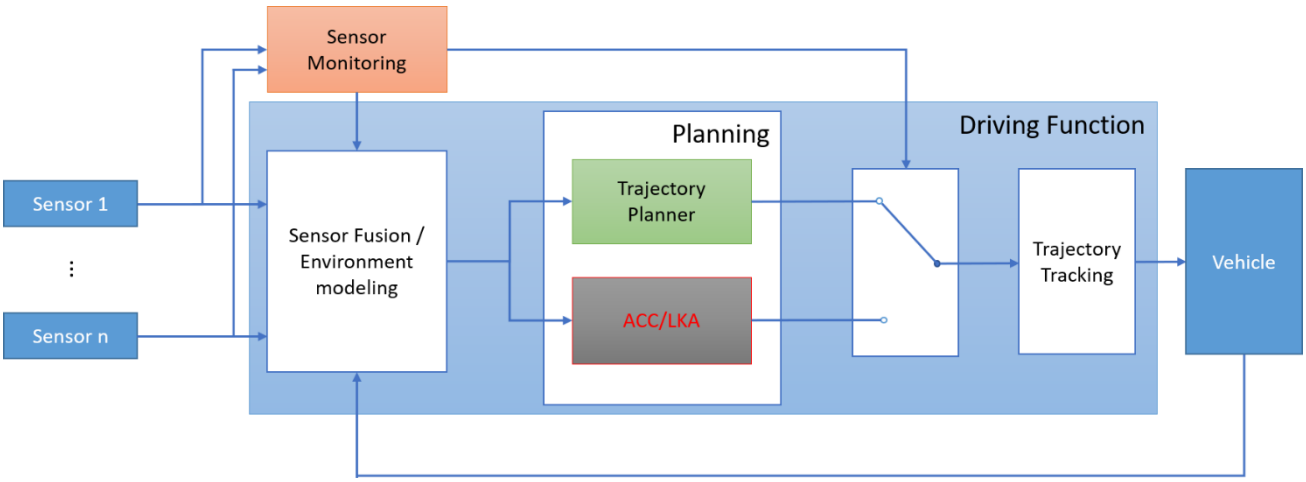


**Figure 40: Fail-operational architecture using VIF's driving function**

Sensor Fusion:

For the VCC use case on the driving simulator, a multi-sensor multi-object tracking system from VIF, based on radar, lidar and camera sensors will be used. The underlying assumption is that each sensor provides sensor data on object level, meaning that multiple measurements that refer to the same object are clustered together and represented as single measurement. This assumption is reasonable as many standard automotive grade sensors provide information in this format. Furthermore, each sensor must explicitly provide longitudinal and lateral distance information of the objects w.r.t. the ego-vehicle, but also additional geometric information, velocity and semantic information can be provided as input.



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 output of the system is a collection of tracks, where each track is represented by the variables given in Table 21.

**Table 21: Track output format**

| Variable | Unit |
|---|---|
| **Longitudinal and lateral position ($x$ and $y$)** | m |
| **Longitudinal velocity ($v$)** | m |
| **Orientation ($\varphi$)** | rad |
| **Radial velocity ($\omega$)** | rad/s |
| **Width and length ($w$ and $l$)** | m |

In the Sensor Level Tracking block of each sensor a measurement covariance matrix must be specified that describes the uncertainty of the measurement. This information should (in general) be deducible from the datasheet of the sensor. Furthermore, a motion model uncertainty matrix must be tuned before system exploitation.

Planning:

The Automated Driving Function developed at VIF is a SAE Level 3 driving function, relieving the driver of longitudinal as well as lateral control responsibilities including adaptive cruise control (ACC), lane keeping (LK) and lane change (LC). In order to achieve a fail-operational architecture, two sets of functionalities are executed in parallel as can be seen in Figure 40. The primary driving function, including full longitudinal and lateral control in the ego vehicle's own lane as well as the adjacent lanes (lane change functionality), is the Trajectory Planner. A combination of ACC and LKA serves as the Hot Standby. In case the sensor monitoring system detects a failure in the sensor system that affects the availability of specific functionalities (in this case lane change), the switch position changes and signals from ACC/LKA instead of from the Trajectory Planner are sent to the Trajectory Tracking.

Trajectory Tracking:

The Trajectory Tracking component implements the feedback control of the vehicle's lateral and longitudinal motion with respect to the reference trajectories provided by the Trajectory Planning subsystem. Its implementation uses a separate control strategy for the vehicle's lateral and longitudinal motion. Since the vehicle's motion is planned within the Trajectory Planning, the resulting reference values for lateral and longitudinal control are always consistent. Depending on the rate of execution, repetitive re-planning also ensures consistency in the presence of unknown disturbances.



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.

### 3.4.2 Model.CONNECT Setup

The driving function model is only one part of the Model.CONNECT setup. In Figure 41 all components of the Simulation setup for the VCC use case is shown. The component structure is the same as in the first driving simulator study. The VTD block represents the connection to the environment simulation tool Vires Vitual Test Drive (VTD). This tool is working on a separate computer. The IP address and port number are stored in the block to enable the communication. The user can also select the scenario in this block. When the Model.CONNECT simulation is started, it starts also the scenario in VTD.

The green block with the white car on it is the VSM block. It simulates the vehicle dynamics in a detailed way. Since it uses another coordinate system as VTD a transformation block is connected between them. VSM is also connected to a small block that controls the function of the driving simulator motion platform.

The third major block is the blue one. It represents the MATLAB/Simulink model of the Driving Function as mentioned in the chapter above. Among other signals the data of the LookAhead sensor from VTD is needed for the driving function. But before these data can be used, they have to be prepared in another small block that is called LookAhead_Data.

The other blocks are for the functionality of the Time-of-Flight camera for driver monitoring (the top left subsystem block), for collecting data for AVL Drive (bottom) and some constant blocks that provide some constant signals.



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.

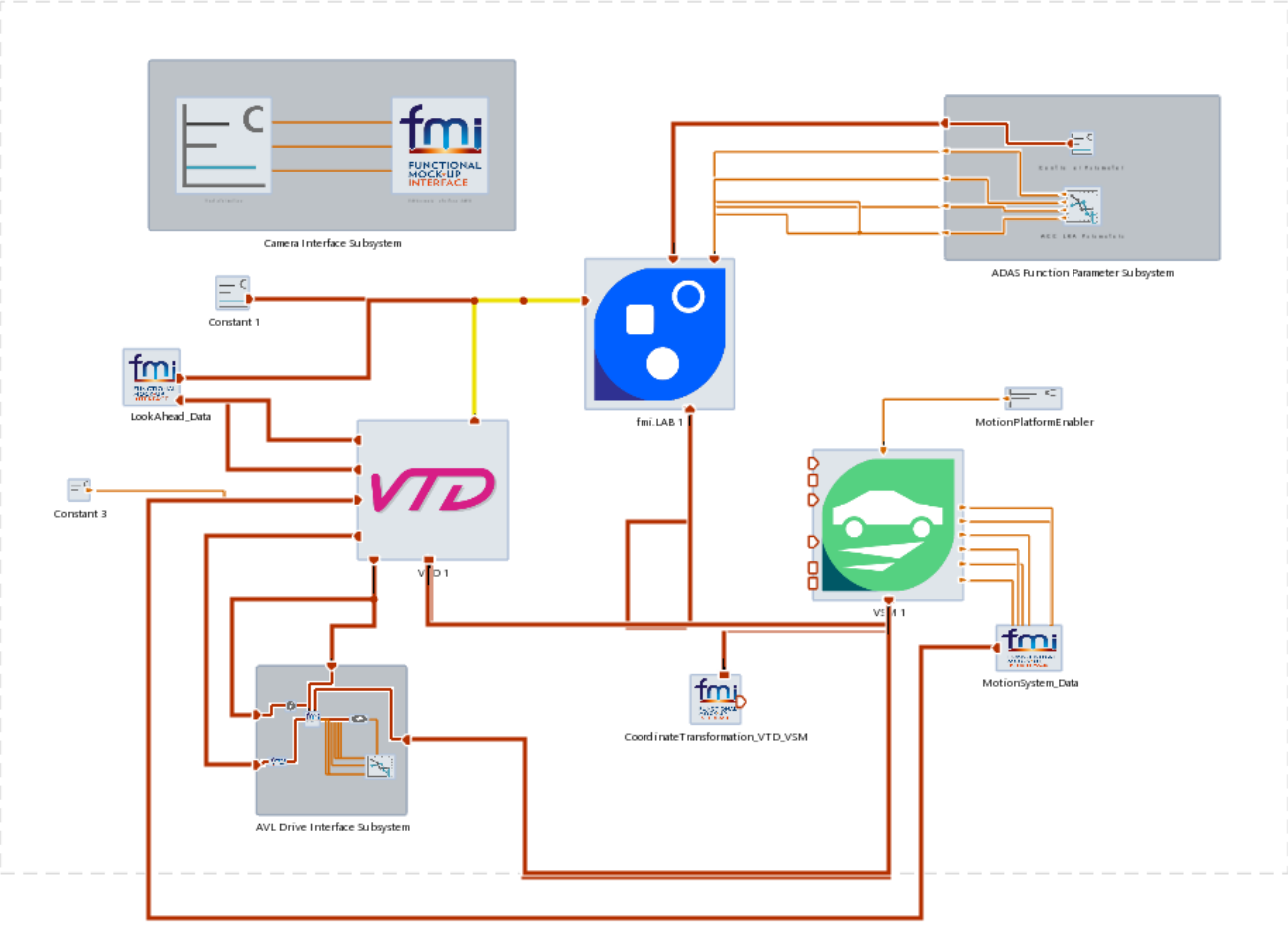


**Figure 41: Model.CONNECT setup for the next driving simulator study**

The simulation structure is a parallel one. It means that all models are executed at the same time and provide the output signals for the other blocks at every time step simultaneously. To prevent unwanted behavior at the beginning of the simulation initial values are set.



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

This deliverable gives a further overview of co-simulation in general and especially of the co-simulation platforms that are provided by the TrustVehicle partners AVL and CISC. The provided manuals give the user an idea how the concepts of these tools work and how to get started with the simulation work. Additional information to this topic can also be found in the previous deliverables of this work package (D3.2 “Modular co-simulation architecture plan” and D3.3 “Report on model repository”).

This report also shows how the co-simulation platform Model.CONNECT was already used in the first driving simulator study. It shows the setup of the Volvo use case as it was at that time. It explains also how Model.CONNECT works in combination with the driving simulator setup.

For the Ford Otosan and Tofaş use cases, the components like trajectory planner as well as the vehicle dynamics for truck/trailer and light commercial vehicles are given. In the Volvo use case, the focus is on sensor monitoring and fail-operability in this deliverable again with a strong focus on the upcoming second driving simulator study in August/September 2019.



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] Accellera. Universal Verification Methodology (UVM) 1.1 User's Guide. Technical report, Accellera, May 2011.

[2] Accellera. Standard SystemC AMS extensions 2.0 Language Reference Manual. Technical report, Accellera, March 2013.

[3] OMG. OMG Systems Modeling Language (OMG SysML), Version 1.3. Technical report, Object Management Group, April 2012.

[4] OMG. OMG Unified Modeling Language (OMG UML), Superstructure, Version 2.4.1. Technical report, Object Management Group, August 2011.

[5] Laurent Balmelli. An overview of the systems modeling language for products and systems development. Journal of Object Technology, 6(6), August 2007.

[6] IEEE. IEEE Standard for Standard SystemC Language Reference Manual. Technical report, The Institute of Electrical and Electronics Engineers, January 2012.

[7] Thorsten Grötker, Stan Liao, Grant Martin, and Stuart Martin. System Design with SystemCTM. Kluwer Academic Publishers, 2002.

[8] Adam Sherer. Accellera's UVM in SystemC Standardization: Going Universal for ESL. http://accellera.org/resources/articles/ accelleras-uvm-in-systemc-standardization-going-universal-for-esl, Mai 2015. [Online; accessed 20-August-2015].

[9] Ralph Weissnegger, Markus Pistauer, Christian Kreiner, Kay Roemer, and Christian Steger. A Novel Design Method for Automotive Safety-Critical Systems based on UML/MARTE. In Proceedings of the Forum on Specification & Design Languages (FDL 2015), pages 35 – 42, Barcelona, Spain, September 2015.



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.