



Contract number: 2004 - 026976

Advancing Traffic Efficiency and Safety through Software Technology (ATESST)

Report type	Deliverable D6.3.1
Report name	Year 1 evaluation results & Demonstrator step 1
Status	Final
Version number	1.0
Date of preparation	2006-02-28

Authors**Editor**

Philippe Cuenot

E-mail

SV (Philippe.cuenot@siemens.com)

Authors

Henrik Lönn

Rolf Johansson

E-mail

Volvo Technology Corporation (henrik.lonn@volvo.com)

MGH (rolf_johansson@mentor.com)

The Consortium

CEA (F)

VCT/MGH (Hu)

Carmaq (G)

The Royal Institute of Technology, KTH (S)

Technische Universität Berlin, TUB (D)

DaimlerChrysler (D)

Siemens (F)

Mecel (S)

ETAS (D)

Volvo(S)

Volvo Car (S)

Revision chart and history log

Version	Date	Reason
0.1	2006-09-25	Initial document
0.2	2006-11-27	Document update
0.3	2007-02-05	Scenario refinement
0.4	2007-02-16	Document review
1.0	2007-02-28	Status change to Final

Table of contents

Advancing Traffic Efficiency and Safety through Software Technology (ATESST) 1

Authors2

Revision chart and history log3

Table of contents4

List of abbreviations.....5

List of Figures5

1 Introduction.....6

2 Input from previous ATESST work7

3 Main Chapters8

 3.1 EAST-ADL Requirements.....8

 3.2 Case study description17

 3.3 Scenario.....22

 3.3.1 Scenario selection criteria22

 3.3.2 Scenario selected23

 3.4 Evaluation45

4 Contribution to overall ATESST objectives47

5 References48

List of abbreviations

AR: AUTOSAR
 HW: Hardware
 SIL: Software in the Loop
 RPU: Rapid Prototyping Unit
 SW: Software
 SWC: Software Component
 V&V: Verification and Validation

List of Figures

Figure 1. A list of the requirements and sub-requirements and the source scenarios 8
 Figure 2 Overview of the demonstrator in Simulink..... 18
 Figure 3 Overview of the application part of demonstrator in Simulink. 18
 Figure 4 View of the demonstrator in Papyrus environment. 19
 Figure 5 View of the VFM models. 19
 Figure 6 View of the ENV models. 20
 Figure 7 View of the FAA models..... 20
 Figure 8. View of the FDA models..... 21
 Figure 9 View of the REQ models 21

1 Introduction

The objective of this WP6 document is to validate proposed concepts of EAST-ADL 2.0 by ensuring that the identified language needs are met in the context of the ATESSST case study and assessed on scenarios set up from the initial scenario collection.

This is a way to capitalize on the experiences and expectations toward EAST-ADL extensions during case study development.

This document describes the engineers' needs towards EAST-ADL, i.e. requirements addressed during the ATESSST project.

Also it documents the automotive example selected for the case study, representing concepts developed in work packages 2 to 5. Concepts such as safety requirement from WP2, architecture description and behavioral definition from WP3 and variability definition and reuse functionality from WP4 will be tailored over the application example in order to provide material for evaluation.

This document describes also the detailed scenario set up for assessment of EAST-ADL language and methodology. These scenarios are the refinement of initial scenario recorded in the deliverable D6.1.1 to align with the demonstrator that is defined in work task 6.2.

The report will document the results of evaluation of the concepts developed in the project as identified during scenario execution. The degree of matching with the initial requirement is documented in order to assess natural fit to industrial context of automotive system product development. Practical experience, potential improvement, and best practice are also recorded.

This evaluation covers the first period of the project, and will be extended for the second period. It is the base for the final document on concept validation which is a public output document of the ATESSST project.

2 Input from previous ATESST work

This document is based on the initial deliverable from task 6.1 and 6.2. The initial description of scenarios is refined based on industrial experience and state of the art of model based development. The demonstrator definition and actual demonstrator is the basis of evaluation.

It also considers main step completed in the different work packages regarding the new semantic of the EAST-ADL description and progress, relying on the Domain Model and its UML2 profile defined at M2.5 milestone (Year 1 mid term assessment).

3 Main Chapters

3.1 EAST-ADL Requirements

A set of generic requirements and sub-requirements on ADL features and tool support were derived, and are elaborated in this section. The requirements are extracted from the previously identified engineering scenarios. The requirements are listed in Table 1.

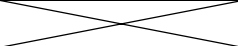
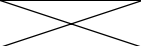
These requirements have been matched to refined selected scenarios described later in the document. From these scenarios the ADL features and tool requirements will be evaluated as pass or fail for the ones which are relevant. The Status after the Year 1 evaluation is documented. Note, however, that some of the requirements are out of the scope of the project, partly covered or not relevant for any specific scenario. This is the case for "generic" requirements or qualitative requirements, such as the requirement for UML alignment.

Figure 1. A list of the requirements and sub-requirements and the source scenarios

ID	Requirement	Scenario Link	Status (Year 1)
R1	The ADL should support the description of product requirements.	X	X
R1.4	The ADL shall support analysis of requirements.	Activity_PD_F_1 Activity_PD_F_3 Activity_PD_A_2 Activity_PD_D_1 Activity_PD_I_1 Activity_FD_A_2 Activity_FD_D_1 Activity_FD_SI_1	OK
R1.5	The ADL shall support breakdown of requirements with traceability links	Activity_PD_F_3 Activity_PD_A_4 Activity_PD_D_5 Activity_PD_I_4 Activity_FD_A_3 Activity_FD_D_5 Activity_FD_I_2 Activity_FD_SI_3	OK
R2	The ADL should support the description of overall product architecture.	Activity_PD_A_1	OK
R2.1	The ADL shall allow interface matching checks (consistency)	Activity_PD_A_3 Activity_PD_D_2 Activity_PD_D_3 Activity_PD_D_4 Activity_FD_A_2 Activity_FD_D_2 Activity_FD_D_3 Activity_FD_D_4 Activity_SI_D_1	OK

R.2.2	The ADL shall allow interface packing via communication protocol & electrical interface (extension of interface checks capability)	Activity_PD_D_4 Activity_PD_I_3 Activity_FD_D_4 Activity_FD_I_1 Activity_SI_D_1 Activity_SI_I_2	Partly (AR?)
R3	The ADL should support system dynamics.		
R3.1	The ADL shall support description of functions, including the definitions, the interfaces, and the composition.	Activity_PD_A_1 Activity_PD_D_3 Activity_FD_A_2 Activity_FD_D_3	OK
R3.2	The ADL shall support subsystem behaviors in various levels of detail.	Activity_PD_A_1 Activity_PD_D_3 Activity_PD_I_2 Activity_FD_A_2 Activity_FD_D_3 Activity_FD_I_1 Activity_SI_D_1	Partly (Native?)
R3.3	The ADL shall support executable behavioral definitions.	Activity_PD_A_5 Activity_PD_D_6 Activity_FD_A_4 Activity_FD_D_7	NOK
R.3.4	The ADL shall have behavior descriptions (including the behavioral definition and semantics, composition rules, description of requirements, etc.) that are in conformance with formal verification needs	Activity_PD_A_1 Activity_PD_A_2 Activity_PD_D_1 Activity_PD_D_3 Activity_PD_I_1 Activity_FD_A_1 Activity_FD_A_2 Activity_FD_D_1 Activity_FD_D_3	NOK
R3.5	The ADL must provide interface definitions and semantics that are clear enough to validate semantics, which may include value and time domain and other attributes such as safety	Activity_PD_A_6 Activity_PD_D_3 Activity_PD_D_4 Activity_PD_D_7 Activity_FD_A_5 Activity_FD_D_3 Activity_FD_D_4 Activity_FD_D_6 Activity_SI_D_2	Partly
R3.6	The ADL shall support integration of behavioral definitions in representative external tools (Simulink, Ascet, Statecharts, ...)	Activity_PD_A_5 Activity_PD_D_6 Activity_FD_A_4 Activity_FD_D_7	NOK
R4.0	The ADL shall support environment models in different levels of details.		

R4.1	The ADL shall provide HIL compatible representations of the relationships between environment model and sensors & actuators.	Activity_PD_A_3 Activity_PD_D_2 Activity_PD_D_4 Activity_PD_I_3 Activity_FD_A_2 Activity_FD_D_2 Activity_FD_D_4 Activity_FD_I_1 Activity_SI_D_2 Activity_SI_I_2	NOK
R5	The ADL shall provide explicit timing information.		
R5.1	The ADL shall support end-to-end timing specification.	Activity_PD_A_2 Activity_PD_D_1 Activity_FD_A_1 Activity_FD_D_1	NOK
R5.2	The ADL shall support timing information for task scheduling with a separation between requirements and required parameters for task scheduling.	Activity_PD_A_2 Activity_PD_D_1 Activity_FD_A_1 Activity_FD_D_1	NOK
R5.3	The ADL shall support timing information for message scheduling with a separation between requirements and required parameters	Activity_PD_A_2 Activity_PD_D_1 Activity_FD_A_1 Activity_FD_D_1	NOK
R6	The ADL needs to be able to distinguish between multiple entities for base functionality and for redundancy.	Activity_PD_D_8 Activity_FD_D_7 Activity_SI_D_3	NOK
R7	The ADL and tools shall provide (dedicated view for) Version & variants handling support.		
R7.1	The ADL and tools shall support hierarchical organization and representation of variability (and variation points)	Activity_PD_F_1 Activity_PD_F_3	NOK
R7.2	The ADL shall allow to control variants and associated models elements beyond abstraction levels (variations points cross-cutting abstraction levels)	Activity_PD_F_1 Activity_PD_A_7 Activity_PD_D_9 Activity_FD_A_6 Activity_FD_D_8	NOK
R7.3	The ADL variability concept shall allow PortInterface variation for SW component or composition (data flow e.g. signal and control flow e.g. runnable)	Activity_PD_A_7 Activity_PD_D_9 Activity_FD_A_6 Activity_FD_D_8	NOK
R7.4	The ADL concept shall support upward and backward traceability (between different abstraction levels) in order to reuse eFeatures, Analysis functions, Design functions, SW elements starting from requirements (and all requirements attached to the artifact), linked with V&V artifact	Activity_PD_F_3 Activity_PD_A_7 Activity_PD_D_9 Activity_FD_A_6 Activity_FD_D_8	NOK
R7.5	The ADL concept shall support links (simple or complex relation, such as selection constraints, ...) on variability of eFeatures on VFM level (OEMs Perceptive)	Activity_PD_F_1 Activity_PD_F_3	NOK

R7.6	The ADL concept shall support links (simple or complex relation, such as selection constraints ...) between variability of functions on design level and variability on Analysis Level (Supplier Perceptive, i.e. no link to VFM architecture and driver needs e.g. different controller for an actuator.	Activity_PD_A_7 Activity_FD_A_6	NOK
R7.7	The ADL concept shall support link (simple or complex relation, such as selection constraints, ...) between variability of functions on implementation level and variability on Design Level (Supplier Perceptive, i.e. no link to VFM architecture and driver needs e.g. data_range diversity between different OEMs)	Activity_PD_D_9 Activity_FD_D_8	NOK
R7.8	The ADL variability concept shall hold relation with coding technology, e.g. switch compiler in C,C++	Activity_PD_D_9 Activity_FD_D_8	NOK
R7.9	The ADL concept shall allow link (simple or complex relation, such as selection constraints, ...) between functions implementation and allocation due to variability at Implementation Level (Supplier Perceptive, no link to VFM architecture and driver needs e.g. memory allocation)		NOK
R7.10	The ADL and tools shall allow system data flow integrity check for all variant instance (before instantiation) for data and control flow (methods) considering software composition (including missing function), that could be impacted during variant selection. This is valid for all abstraction level, and all variant possible to construct on the system.	Activity_PD_D_9 Activity_FD_D_8	NOK
R7.11	The ADL and tools shall allow verification of behavior for all variant instances (before instantiation) for timing requirements (dead line, execution time and recurrences), covering sequence execution and data availability that could be impacted during variant selection. This is valid for all abstraction levels, and all variants possible to construct on the system.	Activity_PD_D_9 Activity_FD_D_8	NOK
R8	The ADL and tools shall support V&V purpose		
R8.1	The ADL and tools shall support formal verification on model level	Activity_PD_D_7 Activity_PD_D_8 Activity_FD_D_6 Activity_FD_D_7	NOK
R8.2	The ADL and tools shall support Performance estimation (RAM, ROM, CPU) with different precision in correlation with model details	Activity_PD_I_5 Activity_FD_I_3 Activity_SI_I_4	NOK
R8.3	The ADL and tools shall support scheduling analysis to verify timing requirement and platform scheduling policy.	Activity_PD_D_3 Activity_PD_D_4 Activity_PD_D_8 Activity_FD_D_3 Activity_FD_D_4 Activity_FD_D_7 Activity_SI_D_3	NOK
R8.4	The ADL and tools shall support timing requirement verification.	Activity_PD_D_8 Activity_FD_D_7 Activity_SI_D_3	NOK
R8.5	The ADL and tools shall support model simulation	Activity_PD_A_5 Activity_PD_D_6 Activity_FD_A_4 Activity_PD_D_7	NOK

R8.6	The ADL and tools shall secure the integration of EAST-ADL system models into RCP environments.	Activity_FD_D_6 Activity_FD_D_7	NOK
R8.7	The ADL and tools should support safety analysis (FMEA, FTA)	Activity_PD_D_6 Activity_FD_D_7	Partly
R8.8	The ADL and tools shall support test case and test document generation	Activity_PD_A_4 Activity_PD_D_5 Activity_PD_I_4 Activity_FD_A_3 Activity_FD_D_5 Activity_FD_I_2	Partly
R8.9	The ADL shall support V&V info supporting several types of V&V-activities, e.g. formal review, simulation, test, formal methods for verification including mathematical analyses.	Activity_PD_A_4 Activity_PD_D_5 Activity_PD_I_4 Activity_FD_A_3 Activity_FD_D_5 Activity_FD_I_2	Partly
R8.10	The ADL shall support V&V-case representation via: V&V environment/setup, the V&V-procedure, the V&V result and the acceptance criterion.	Activity_PD_A_4 Activity_PD_D_5 Activity_PD_I_4 Activity_FD_A_3 Activity_FD_D_5 Activity_FD_I_2	Partly
R8.11	The ADL shall support V&V procedure with textual description, a reference to external document, or a pointer to a native ADL-behavior	Activity_PD_A_4 Activity_PD_D_5 Activity_PD_I_4 Activity_FD_A_3 Activity_FD_D_5 Activity_FD_I_2	NOK
R8.12	The ADL shall support V&V-activity of testing with some additional attributes: the test input, the intended test outcome, the actual test outcome	Activity_PD_A_4 Activity_PD_D_5 Activity_PD_I_4 Activity_FD_A_3 Activity_FD_D_5 Activity_FD_I_2	OK
R8.13	The ADL shall support V&V-activity of testing with initial state and other setup information, that must be recordable, either as V&V object or via test input	Activity_PD_A_4 Activity_PD_D_5 Activity_PD_I_4 Activity_FD_A_3 Activity_FD_D_5 Activity_FD_I_2	NOK
R8.14	The ADL shall support V&V case associated to V&V-objects, but should not contain them	Activity_PD_A_4 Activity_PD_D_5 Activity_PD_I_4 Activity_FD_A_3 Activity_FD_D_5 Activity_FD_I_2	Partly
R8.15	The ADL shall support relation to V&V-case associated to requirements, but should not contain them	Activity_PD_A_4 Activity_PD_D_5 Activity_PD_I_4 Activity_FD_A_3 Activity_FD_D_5 Activity_FD_I_2	OK

R8.16	The ADL shall support V&V setting including the verified requirements, entities, plant, etc.	Activity_PD_A_4 Activity_PD_D_5 Activity_PD_I_4 Activity_FD_A_3 Activity_FD_D_5 Activity_FD_I_2	Partly
R8.17	The ADL shall support for each requirements an acceptance criteria (acting like abstract requirements on the V&V cases) to ensure that the requirement is verifiable. The V&V information should be designed to satisfy these criteria.	Activity_PD_A_4 Activity_PD_D_5 Activity_PD_I_4 Activity_FD_A_3 Activity_FD_D_5 Activity_FD_I_2	Partly
R9	The ADL shall support hardware models with sufficient detail to support system modeling with allocation, driver SW configuration, bus scheduling and gatewaying, etc. has to be supported.		
	The ADL shall support modeling of sensor/actuators with sufficient detail and with appropriate structure to support integration on different levels.	Activity_PD_A_3 Activity_PD_D_4 Activity_PD_I_3 Activity_FD_A_2 Activity_FD_D_4 Activity_FD_I_1 Activity_SI_I_2	Partly (AR?)
	The ADL shall provide sufficient information about the design and configuration of communication networks (e.g., CAN, LIN, FlexRay, MOST)	Activity_PD_I_3 Activity_FD_I_1 Activity_SI_I_2	NOK
R10	The ADL shall support modeling of middleware.		
R10.1	The ADL shall have compliance or integration with AUTOSAR platform/middleware representation.	Activity_PD_I_3 Activity_FD_I_1 Activity_SI_I_2	NOK
R11	The ADL shall contain sufficient information in the SW and HW models to allow HW-SW integration in a smooth and automatic way.	Activity_PD_D_4 Activity_PD_D_7 Activity_PD_I_3 Activity_FD_D_4 Activity_FD_D_6 Activity_FD_I_1 Activity_SI_D_1 Activity_SI_D_2 Activity_SI_I_2	NOK
R12	The ADL should support cross-aspect/domain mapping and traceability.		
R12.1	The ADL shall support allocation associations.	Activity_PD_D_4 Activity_PD_D_7 Activity_PD_I_3 Activity_FD_D_4 Activity_FD_D_6 Activity_FD_I_1 Activity_SI_D_1 Activity_SI_D_2 Activity_SI_I_2	NOK

R12.2	The ADL shall have well-defined (traceability) relationships between VFM and other artifacts	Activity_PD_F_1 Activity_PD_A_7 Activity_PD_D_9 Activity_FD_A_6 Activity_FD_D_8	NOK
R13	The ADL and tools shall provide adequate management support for the product information and work organization.		
R13.1	The ADL and tools shall provide a basis for managing code generation files and configurations and for specifying the behaviors for code generation	Activity_PD_I_5 Activity_FD_I_3 Activity_SI_I_4	NOK
R13.2	The ADL and tools shall support library management concept (set of model of model composition) and project space (building of system configuration)	Activity_PD_A_1 Activity_PD_A_3	
R13.3	The ADL shall provide sufficient information for documenting responsibilities of functions or features in a Vehicle Model	Activity_PD_F_1 Activity_PD_F_3	NOK
R13.4	That ADL and tools shall support a V&V matrix pointing out requirements and the activities (tests, formal analyses) that verify them.	Tool specific oriented	NOK
R14	The ADL and tools shall have the ability to link to/with external tools and models.		
R14.1	The ADL and tools shall support the integration of external modeling tools such as MatLab/Simulink .	Activity_PD_A_5 Activity_PD_D_6 Activity_FD_A_4 Activity_FD_D_7	NOK
R14.2	The ADL and tools shall support the integration of important network design and test tools, such as VNA and CANoe,)	Activity_PD_I_3 Activity_FD_I_1 Activity_SI_I_2	NOK
R14.3	The ADL and tools shall be connected to configuration management environment	Tool specific oriented	NOK
R14.4	The ADL and tools shall be connected to requirement management tool (Doors, ...) and/or standardized requirement format such as RIF	Activity_PD_F_3 Activity_FD_A_1	Partly
R14.5	The ADL and Tools for V&V purpose shall support import (XML) of test case managed in in-house tools	Tool specific oriented	NOK
R15	The ADL and tools shall have adequate usability support.		
R15.1	The ADL and tools shall allow effective navigation in different parts of the model, such as navigation up and down system hierarchies, navigation to associated entities (requirement of an entity, a datatype of a port, allocated SW of a HW entity), navigation to connected entities (function connected to function)	Tool specific oriented	NOK
R15.2	The ADL and tools shall support graphical edition of the models		OK (Papyrus)
R16	The ADL and tools shall provide adequate documentation support.		
R16.1	The ADL and tools shall support Reverse document/engineering	Tool specific oriented	NOK
R16.2	The ADL and tools shall support open solution for document generation		OK (PDT)
R16.3	The ADL and tools shall support design document generation	Activity_PD_D_9	NOK

		Activity_FD_D_8	
--	--	-----------------	--

R16.4	The ADL and tools shall support variability concept in design document generation	Activity_PD_D_9 Activity_FD_D_8	NOK
R17	The ADL and tools shall support more formal exchange mechanisms between OEM and suppliers	Activity_PD_F_3	NOK
R18	The ADL shall be aligned with other UML approaches.	Activity_PD_D_9 Activity_FD_D_8	OK
R18.1	The ADL shall be aligned with OMG approaches, including UML2, SysML, and MARTE.	Activity_PD_D_9 Activity_FD_D_8	Partly
R18.2	The ADL shall be aligned with AUTOSAR.	Activity_PD_D_9 Activity_FD_D_8	Partly

3.2 Case study description

The case study and demonstrator plan was defined based on the early definition of the scope of the demonstrator as described in [3]. It was focused to the selection of reference application examples that will serve as case studies (different refinement, structure/behavior, analysis, variability etc...) for the validation of concepts from all work packages. It will also be used as the basis for dissemination material. This "eSafety" case study was based on an integration of subsystems and experience from the partners.

This case study is composed of:

- Brake System composed of model for Brake Pedal, Brake Reduction Coordination, Anti-lock Braking System and Brake Actuator
- Engine management composed of Accelerator Pedal, Torque moderator, Engine Management System
- Rain System : Rain sensor model
- Cruise Control System : Speed calculation and Speed Moderator
- Safety interconnected traffic System : Collision mitigation, Radar Sensor
- Environment Model : Engine Dynamics model, Driver model, Traffic model, Rain model

The case was structured according to the Integrated Safety Projects (Including e.g. EASIS, PReVENT and AIDE) architecture. This defined the vehicle architecture with following blocks;

- Vehicle HMI for interaction with the driver
- Vehicle perception for perception of the environment by the vehicle
- Vehicle Dynamic Application for actuator coordination and control model involving vehicle dynamics
- Vehicle Protection System for actuator coordination and control model involving vehicle protection
- Vehicle application

As no EAST-ADL graphical editor was available, the case representation was initially edited in Simulink.

The figure below shows an overview of the case study in the Simulink environment that was made as an initial step. The structure is compliant to Integrated Safety Project architecture, and vehicle functions are described within the vehicle application. This initial step is the basis for the final demonstrator and defines the behavioral algorithm reference for the case study during project progress.

Figure 2 Overview of the demonstrator in Simulink.

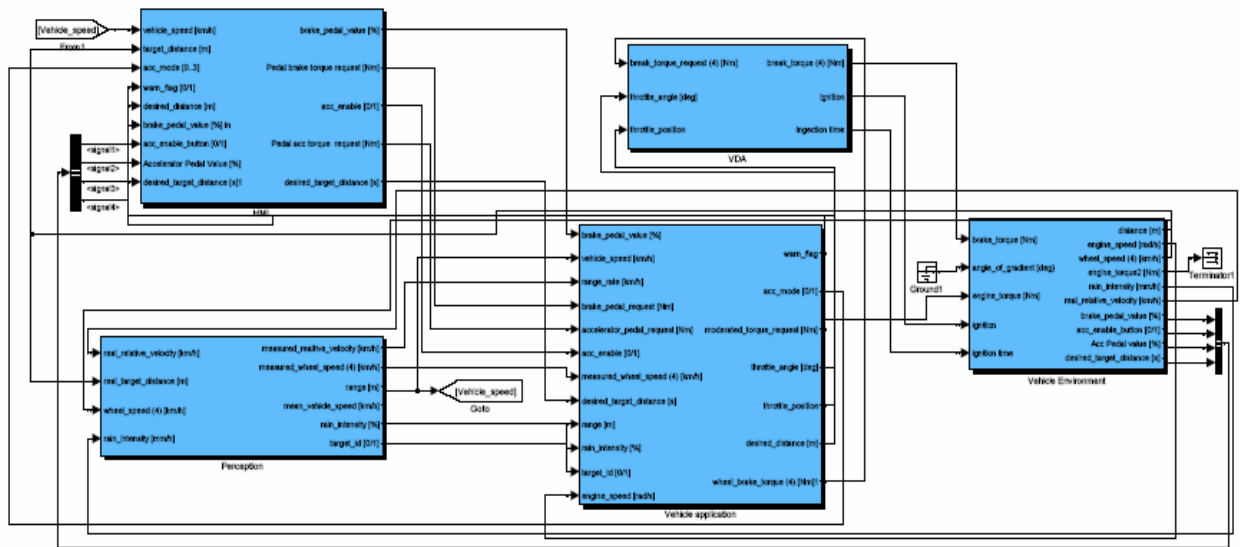
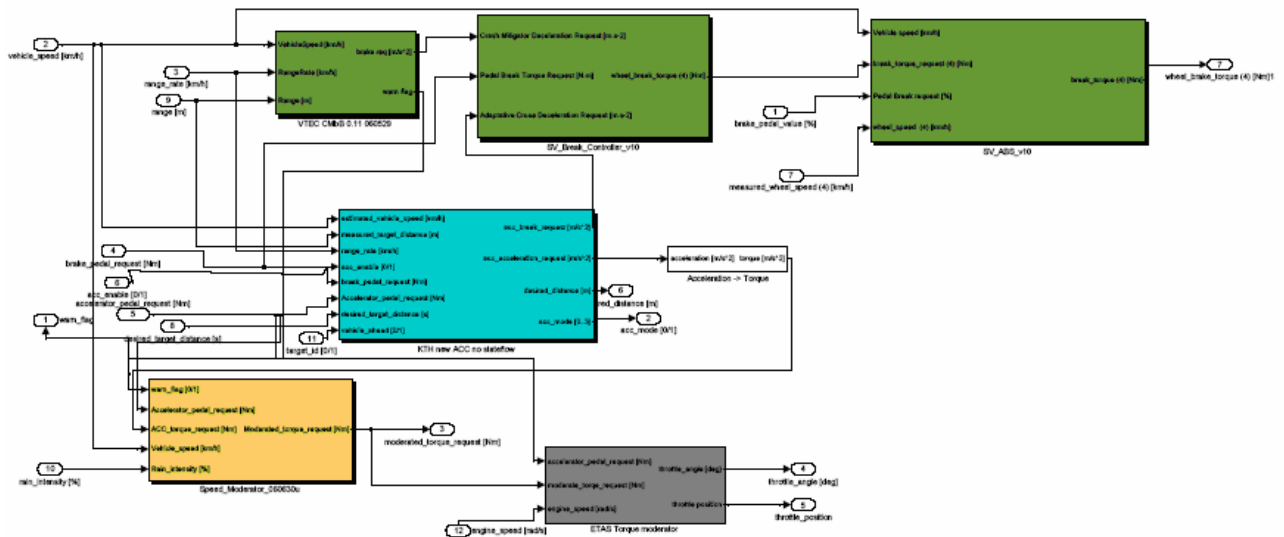


Figure 3 Overview of the application part of demonstrator in Simulink.



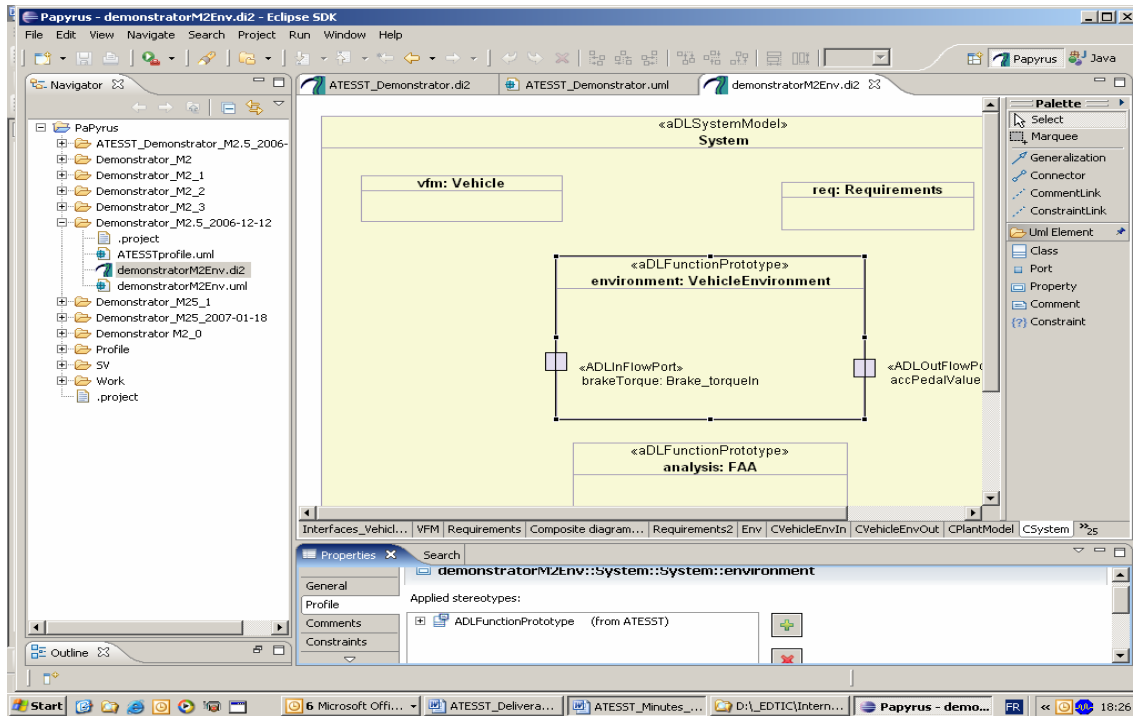
To support complete graphical editing of the EAST-ADL language, a Graphical Modeling Framework was built that allows UML2 profile generation and control, as UML2 was the technology selected for the outcome of the project. The technology selected to build this environment is Eclipse with Eclipse Modeling Framework plug-in. Associated to UML2 eclipse project plug-in, a graphical editor of UML2 profile was developed. So far only UML2 entities can be modeled. More advanced features, such as graphical palette to work with UML2 stereotypes in specific views of the editor is planned for the second period of the project.

The demonstrator or case study was defined in the EAST ADL, based on the initial Simulink model. The configurable UML tool developed in the project, Papyrus, was used as well as the UML2 profile for EAST ADL2. A top down approach was used to represent the architecture, according to the architectures, or views, of EAST-ADL.

Although the graphical representation in the tool is not fully adequate at this stage, some snapshots of the work models are included to give an idea of the representation in Papyrus.

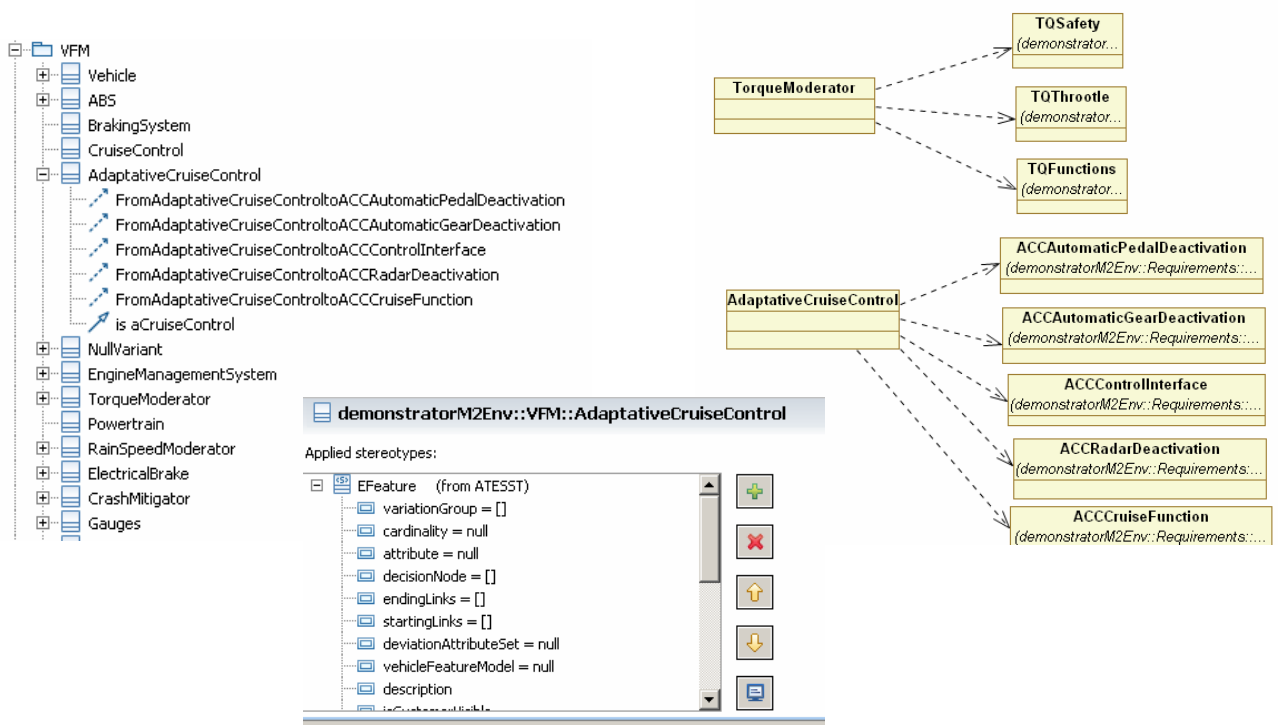
The Vehicle feature, Environment and Functional analysis architecture view were built for the case study. Then abstract of Requirement was set up and Function design architecture for Anti-lock braking system. The AUTOSAR related parts of the demonstrator are not yet included, but will start during the second period.

Figure 4 View of the demonstrator in Papyrus environment.



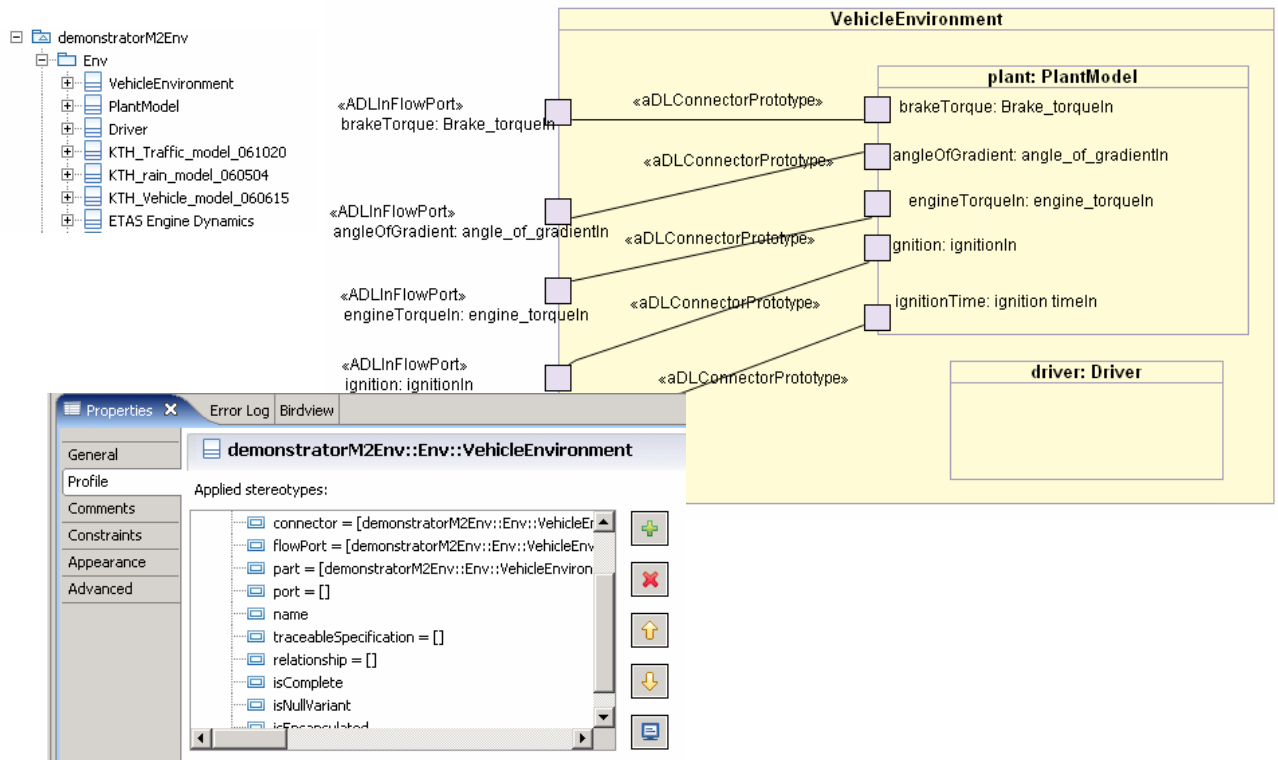
The Vehicle Feature Model captures all the feature of the case study. A snapshot is pictured below

Figure 5 View of the VFM models.



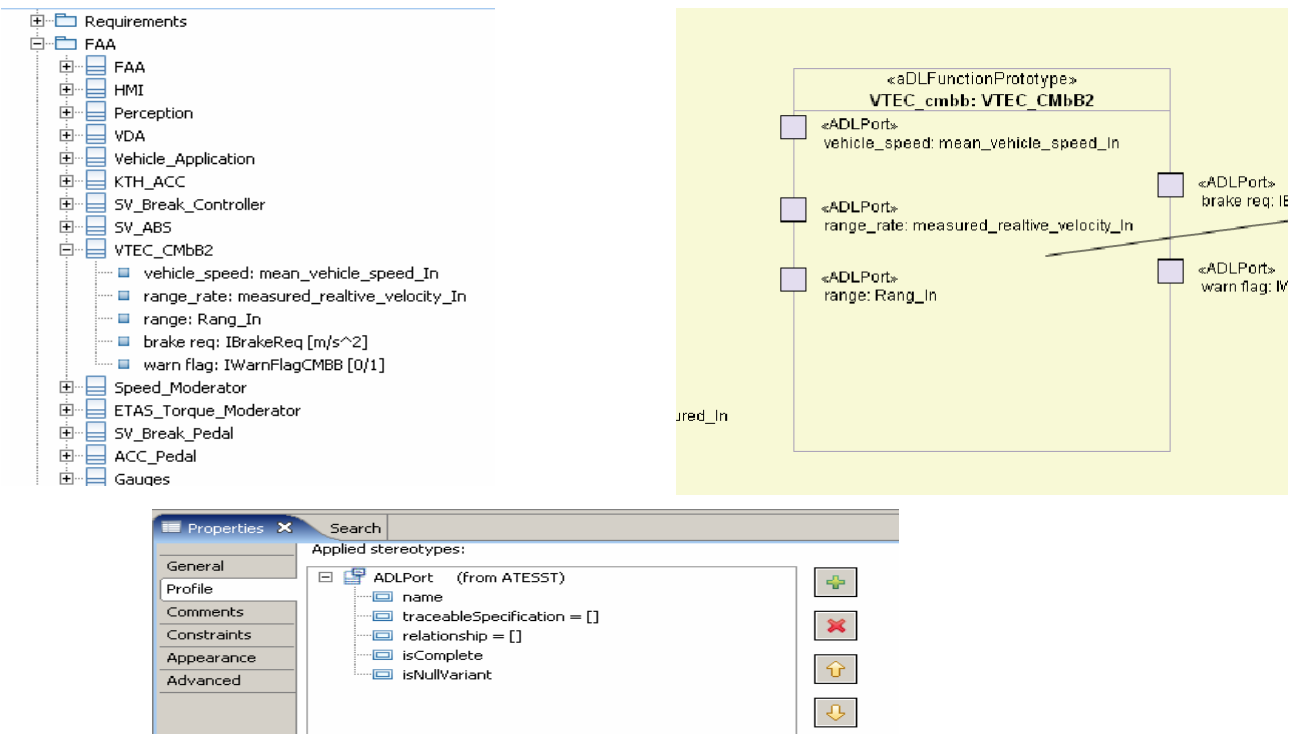
The Environment model of the case is represented as a snap shot

Figure 6 View of the ENV models.



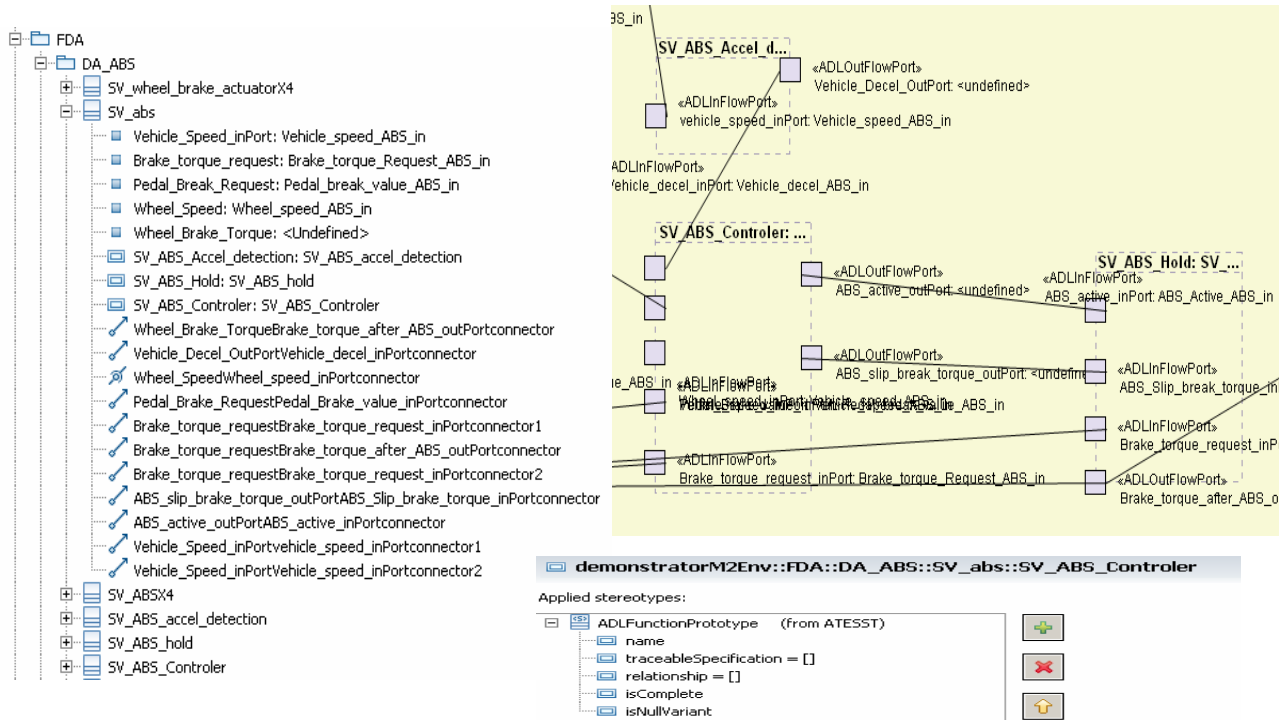
The Functional Analysis Architecture of the case study is represented below

Figure 7 View of the FAA models.



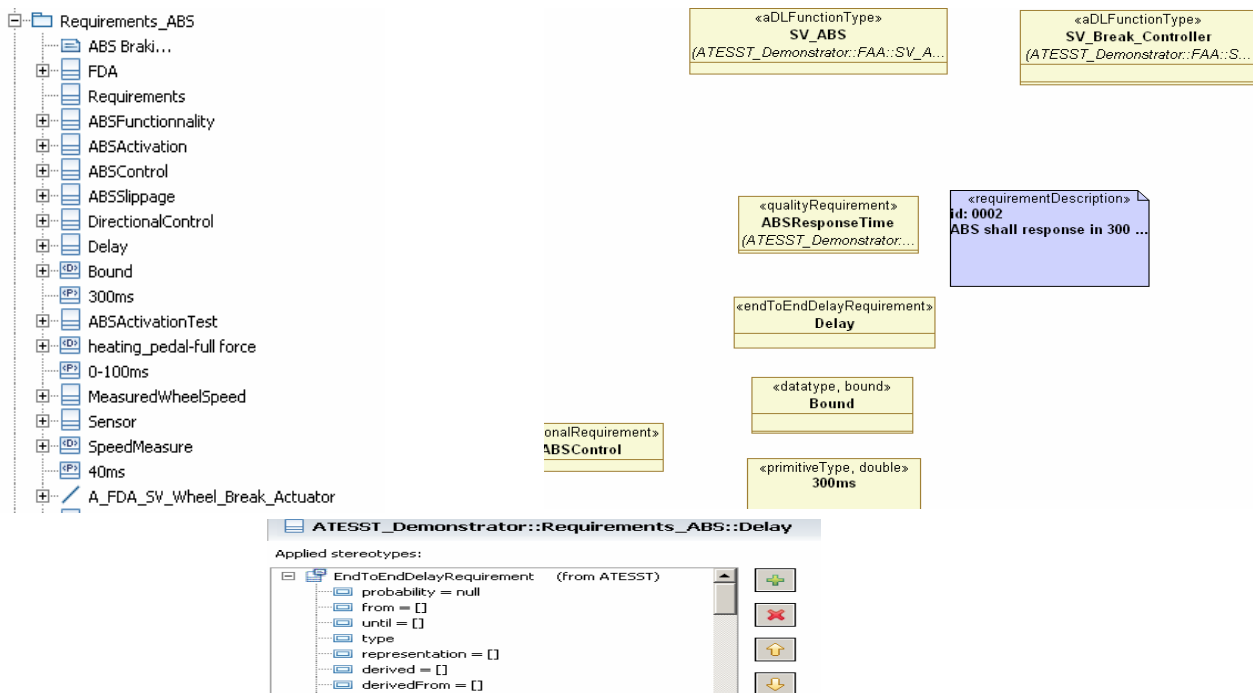
The Functional Design Architecture of the case study has been captured for the Anti-lock braking System and is represented below.

Figure 8. View of the FDA models.



Finally, an abstraction of the requirement model in the context of Anti-Blocking System is also captured.

Figure 9 View of the REQ models



The description of the case study will be refined throughout the project ending with the complete demonstrator view in the end of the project. At that stage, the graphical features will be more mature.

3.3 Scenario

As stated in the D6.1.1 deliverable [1] describing initial scenarios for ATESST, case study engineering scenarios are sequences of activities aiming to create a technical solution with respect to the information or the organization of engineering. These activities relate to the usages of the architecture description language EAST-ADL 2 with the associated methods and tools [2]. During the course of the project, these need to be reorganized and refined to build consistent and continuous engineering practice that will serve to assess the ATESST project results. Note that we describe typical scenarios, and not a fixed process. The detailed process has to be defined in the context of each company.

The purposes of the scenarios include

- Evaluation of Papyrus tools and plug-in developed in WP5
- Supporting AUTOSAR compliance and modeling regarding infrastructure
- Supporting compatibility of AUTOSAR document exchanged (DTD and template)
- Supporting Analysis, Design and Implementation engineering with model refinements
- Assessing capability for architecture description including product line definition
- Assessing solution for behavioral description including specialized tool interactions
- Evaluation of requirement attributes and adequacy to constraints
- Validation of safety requirements linked to selected methods
- Consideration of the relation between product line and project instantiation
- Consideration of reuse in engineering practice

3.3.1 Scenario selection criteria

The selection of scenarios was based from original inputs categorized into the 8 groups below.

- | | |
|--------------------------------|---|
| 1-Overall Product Design; | 2-Functional Design; |
| 3-Implementation Design; | 4-Implementation Platform Design; |
| 5-Quality Centric Design; | 6-System Integration; |
| 7-Verification and Validation; | 8-Management and Documentation Support; |

These groups have been prioritized by partners to focus only on a dedicated concern to limit the number of scenarios. Relation to requirements has been maintained in order to emphasize mandatory issue that needs to be address by scenarios, to support and solve engineering problem of today.

Scenario group selections for further refinement are:

- Overall Product Design
- Functional Design
- System Integration
- Quality Centric Design (as transversal support for others scenarios)

After prioritization, relation to ATESSST requirements has been focused on the following issues with the following categories

Engineering requirements:

- Requirement support break-down, traceability and analysis R1, R1.4, R1.5
- Architecture description of overall product R2 and traceability versus Vehicle Feature Model, VFM R12.2
- Details for hardware modeling (Actuator/Sensors, bus) R9.1, R9.2
- Compliance with AUTOSAR R10.1, R18.2

Tools requirements:

- Link to external behavioral tools R14.1, Requirement tool R14.2
- Tools features for navigation R15.1 and document generation R16

Safety and timing requirements:

- Safety analysis R8.7 and redundancy modeling R6
- Timing Modeling for model analysis R3, R3.1, R3.5, R5, R5.1, R5.2, R5.3

3.3.2 Scenario selected

Based on criteria defined in the previous chapter, a new list of scenarios will be created.

To support engineer end user's needs and a successful integration and consolidation of ATESSST results, a top down perspective is required. It would start from stakeholders' needs and reaching implementation levels, fully supported by AUTOSAR meta-model, template and authoring tools that will allow generating binary file. We will focus on the modeling view and consider the path from different automotive engineering perspectives, OEMs and supplier views. This will lead to consistent scenarios which all start form the top level for automotive engineering product development, meaning the vehicle feature view. Then, focusing on the project technical area of work and safety aspect they will be tailored with details close to engineering day to day work in order to build a sound automotive product.

These scenarios will also integrate some process view demonstrating applicability of EAST-ADL 2 language in the context of automotive product design. Main EAST-ADL view will be referred to as a major step in order to be able to reuse parts of the scenario in different engineering contexts (OEMs and supplier shared responsibility for example).

The approach to support these scenarios are to define activities with a detailed description of inputs and outputs documents, models or others; referring to WP activities and plug-in required for automation and expected results.

Depending of input document for the step activity, optional inter sub-activity are recorded.

The structure of each activity step is in the following format:

Activity code	Input		
Activity step	Activity description	WP link	Relation to tools and plug-in
	Output		

3.3.2.1 Product Design

This initial scenario describes automotive product line analysis and design from the top architectural view, referring to an initial scenario of an OEM to define the vehicle architecture. Depending on the engineering step and associated architecture view, this could also be considered as a sub scenario, considering intermediate step for engineering performed either by OEMs or supplier. Some activities could be bypassed to adjust vehicle design at the level of activity considered.

Feature view:

The interaction area of Vehicle architecture do not consider only end user driver functional needs, but also all features functions needs having an interaction with mechanical interface (of the environment).

Activity_PD_F_1		Vehicle requirement list	
Define feature for vehicle architecture	Elicit and record all features and sub-features required to be supported by the vehicle product line needed to fulfill all requirements, with no consideration of variability.	WP 3	Graphical architectural feature capture from Papyrus Environment actors capture via behavioral from Papyrus
	Identify external actors for each feature (sub system to be modeled).	WP3	
		Vehicle feature list (hierarchical organization for feature refinement) Environment actors list in relation to feature tree	
Activity_PD_F_2		Vehicle requirement list	
Define vehicle variant from the initial basic feature tree	Identify all requirements related to variability.	WP4	Variability plug-in: Build variability tree and import into ADL view of Papyrus Behavioral description for environment actors (use case, activity diagram)
	Build variability feature tree based on product line decision criteria recorded as requirements or deduced from product line strategy	WP3	
		Import into the architecture representation for linking with feature view	
		Identify impact on environment actors	
		Vehicle product line feature tree Environment actors variability tree	
Activity_PD_F_3		Vehicle requirement and constraints list Option a) from RIF format requirement document	
Allocate requirements and constraints to features, and optional environment tree including variability impact	Allocate and trace requirements and constraints to vehicle product line feature tree	WP2	Graphical requirement capture from Papyrus Option a) RIF format import plug-in for requirement
	Option a) Allocate and trace requirements and constraints from a requirement container import (RIF format document)		
		If no formal initial requirement and constraints, create and analyze them in the context of feature tree and variability	
		Traceability reference to initial document must be ensured by external link (document) or ID number (for option b)	
		Initial requirement tree associated to vehicle product line feature tree	

Architectural view:

At this analysis view, in-house partitioning of functionality could be reused to define static structure of the system and the (sub)-system.

Activity_PD_A_1	Vehicle feature list Environment actors list Option a) Existing partitioning of Functional Model		
Define abstract architectural of product (structural view)	Build architectural view of the system in relation to vehicle feature list Option a) Reuse architectural clustering based on existing partitioning of Function Model Define and structure abstract ADL Function, Environment Function (physical unit vehicle model) and Functional Device Models	WP3	Architectural graphical feature capture from Papyrus
	Composed Functional Model Environment Model Functional Device Model		
Activity_PD_A_2	Initial requirement tree Functional Model Environment Model Functional Device Model		
Refine and Derive requirement, Identify and qualify safety, and quality requirement	Reorganize and refine initial requirements in the context of different Models view Derive Requirement in the context of Models decomposition, and Refine them if necessary in relation to Models composition Analyze requirement and mark safety requirement. These requirements are typically the result of a preliminary safety analysis. Fulfill (or check) quality requirement Trace requirement to different Model in order to satisfy them	WP2 WP3	Requirement graphical feature capture from Papyrus Behavioral description for requirement (use case, activity diagram)
	Requirement tree refined		
Activity_PD_A_3	Functional Model Environment Model Functional Device Model Option a) Existing Model Requirement tree refined		
Refine architectural view with interaction and behavioral Model description	For each Model define interaction with Port and data type between Function, Environment, Functional Device element In context of Model composition, define interaction with Port and data type between the different models Define abstract behavioral description (Continuous or Discrete description for physical behavioral	WP3 WP3	Architectural graphical feature capture for Port and Interface description from Papyrus Behavioral plug-in : Export ADL representation for behavioral description, or import existing behavioral

	capture) of each model, by native representation, or external tool description tool (Simulink,...) via export from ADL representation Option a) import to ADL for behavioral description tool (Simulink, UML tools, ...)		Model for ADL representation
	Functional Model detailed Environment Model detailed Functional Device Model detailed		
Activity_PD_A_4	Functional Model detailed Environment Model detailed Functional Device Model detailed Requirement tree refined		
Review requirement tree and build V&V test strategy	Trace requirement to different Model in order to satisfy them if necessary considering new context of detailed model (behavioral and Interface) Build V&V element in relation to Model, and link V&V artifact to requirement tree	WP2 WP2	Requirement and V&V graphical feature capture from Papyrus Traceability plug-in: Check requirement, model and V&V traceability (non allocated or miss of relation)
	Verification and Validation Model Requirement tree consolidated		
Activity_PD_A_5	Functional Model detailed Environment Model detailed Functional Device Model detailed Verification and Validation Model Product line feature tree deployed at Architecture level		
Validation of abstract behavioral description	Simulate for validation purpose in the context of V&V objective of the different behavioral and architectural model (depending of the context of simulation environment)	WP3	Simulate abstract model versus environment model using physical unit vehicle model (Software in the Loop)
	Functional Model validated Environment Model validated Functional Device Model validated		
Activity_PD_A_6	Functional Model detailed Environment Model detailed Functional Device Model detailed Requirement tree consolidated		
Hazard and Functional Failure analysis	Define and capture Hazard list and relation to the model Build and capture Functional Failure Analysis of the system in relation with models	WP3	Error Model graphical feature capture from Papyrus

	Revisit and consolidated requirement tree and V&V artifact in context of safety		
	Hazard list Functional failure list Safety Requirement list Requirement tree and V&V artifact revisited at Architecture level		
Activity_PD_A_7	Requirement tree and V&V artifact revisited at Architecture level Architecture Model detailed (abstract ADL Function, Functional Device, Error model, ...) Environment Model detailed		
Update variability feature tree and generate documentation	Build product variant set on variation mechanism and Function models at architecture level Generate design documentation	WP4 WP4 WP3	Variability graphical feature capture from Papyrus Variability plug-in: Generate instance of variant instance deployed at Architecture level, verify static interface Documentation plug-in: Generate design document
	Product line feature tree deployed at Architecture level Generate design documentation at architecture level		

Design view:

The following detailed activities described sequentially, have to be repeated for all functional models abstracted at Architectural level. They are identified as exclusive steps, but in practical usage, interaction and closed loop on these activities could occur. The hardware concept representation is initially described by a single substitution of EAST-ADL 1.0 by the AUTOSAR (AR) related template.

Activity_PD_D_1	Functional Device Model validated Functional Model validated Requirement tree revisited at Architecture level		
Refine and Derive requirement	Consider Functional Device and Functional Model Derive Requirement in the context of Design decomposition, and Refine them if necessary Propagate safety requirement. Develop initial timing requirement	WP2 WP3	Requirement graphical feature capture from Papyrus Behavioral description for requirement (use case, activity diagram)
	Requirement tree refined at Design level		

Activity_PD_D_2	Environment Model detailed Hazard list		
Environment Design	Transform environmental model into vehicle model and sensor actuator model (with discrete model if necessary) in context of RPU Optimized model to be supported by Rapid Prototyping Unit (hardware and software resource context) Validated vehicle and sensor actuator environment models Define and capture Environment scenario test matching to hazard event (robustness test)	WP3 WP3 WP3	Behavioral plug-in: Transformation from continuous to discrete model Simulate environment model (electrical unit vehicle model and sensor actuator model) in context of RPU and SIL Behavioral graphical feature capture from Papyrus for test scenario (activity diagram)
	Environment Model optimized (Vehicle and Sensor Actuator model) Verification and Validation Model for safety test		
Activity_PD_D_3	Functional Model detailed Requirement tree refined at Design level		
Functional Design	Refine Functional Model in ADL Function with Composition to fit later on AR SWC. Component are build in relation to requirement, with focus on functional partitioning and configuration for better reuse For each Model (and its composition) define interaction with Port and data type for implementation in relation Functional Device element Define detailed behavioral description (Discrete description) of each model, by native representation, or external tool description tool (Simulink,...) via export from ADL representation Option a) import to ADL for behavioral description tool (Simulink, UML tools, ...) Build Software fault models from requirement elicitation and abstract behavioral representation from Architecture view	WP3 WP3	Architectural graphical feature capture for Port and Interface description from Papyrus Behavioral plug-in : Export ADL representation for behavioral description, or import existing behavioral Model for ADL representation
	ADL Function Models (with Interface) and behavioral model detailed Run able entities of SW Component Models Software fault models		
Activity_PD_D_4	Functional Device Model detailed Environment Model optimized Requirement tree refined at Design level Option a) Reuse of existing AR platform		
Hardware Design	Build ADL Function for Sensor Actuator Component to fit later on AR SWC. Components are build in relation to requirements, with focus on functional partitioning and configuration for better	WP3	Architectural graphical feature capture for Port and Interface description from Papyrus

<p>Activity_PD_D_6</p>	<p>ADL Function Models (SWC and Sensor Actuator Component)</p> <p>Environment Model optimized (Vehicle and Sensor Actuator model)</p> <p>Verification and Validation Model</p> <p>Hazard safety test scenario</p>		
<p>Validation of SW Component behavioral Model</p>	<p>Simulate for validation purpose in the context of V&V objective of the different behavioral and architectural model (depending of the context of simulation environment and variant instance)</p> <p>Build software timing models versus experiment of simulation tool</p>	<p>WP3</p>	<p>Simulate model versus environment model (vehicle and sensor Actuator model (SIL or RPU)</p>
	<p>ADL Function Models (SWC and Sensor Actuator Component) Validated</p> <p>Software timing models</p>		
<p>Activity_PD_D_7</p>	<p>ADL Function Models (SWC and Sensor Actuator Component)</p> <p>Initial AR system template (topology, OS, bus)</p> <p>Initial AR ECU resource template (Pin, Processor Unit, Bus connection)</p> <p>Software fault models</p> <p>Hardware fault models</p>		
<p>Architecture exploration and safety analysis</p>	<p>Based on initial system topology, instantiate initial relation between software and hardware error model (considering context of variant handling)</p> <p>Generate initial fault tree</p> <p>Deploy architecture into a system configuration and propagate error model into the architecture</p> <p>Evaluate different hardware and software architecture in consideration of analysis fault tree</p> <p>Identify failure node and allocation constraints</p> <p>Detect possible failure node where redundancy is required</p> <p>If hardware or software functional redundancy is required Analysis and Design phase are restarted at the adequate activity level.</p>	<p>WP3</p> <p>WP3</p> <p>WP3</p>	<p>Error model graphical feature capture from Papyrus</p> <p>Error Model plug-in: Generation of Fault tree and safety analysis</p> <p>Architectural graphical feature capture from Papyrus and interface to AR template edition</p>
	<p>Fault tree models</p> <p>Preliminary SW Component allocation (expressed from AR system template)</p> <p>Preliminary System topology for communication bus and signal (expressed from AR system template)</p> <p>Hardware and Function redundancy list</p>		
<p>Activity_PD_D_8</p>	<p>Preliminary SW Component allocation (expressed from AR system template)</p> <p>Preliminary System topology for communication bus and signal (expressed from AR system template)</p>		

	<p>Hardware timing models</p> <p>Software timing models</p> <p>Verification and Validation Model</p>		
Architecture timing analysis	<p>Define Os Task configuration list (expressed from AR system template) according to scheduling strategy and ECU considered</p> <p>Perform architecture timing analysis, end to end timing analysis</p> <p>Perform OS scheduling timing analysis</p> <p>This activity should ensure that safety analysis is not corrupted otherwise Activity_PD_D_7 must be executed</p>	<p>WP3</p> <p>WP3</p>	<p>Architectural graphical feature capture from Papyrus and interface to AR template edition</p> <p>Behavioral plug-in: Scheduling analysis, end to end analysis</p>
	<p>SW Component allocation (expressed from AR system template)</p> <p>System topology for communication bus and signal (expressed from AR system template)</p> <p>ECU resource Component configuration (expressed from AR ECU resource template)</p> <p>Os Task configuration list (expressed from AR system template)</p>		
Activity_PD_D_9	<p>Requirement tree and V&V artifact revisited at Design level</p> <p>Architecture Model detailed (ADL Function, Functional Device, Error model, ...)</p> <p>Environment Model</p> <p>SW Component allocation (expressed from AR system template)</p> <p>ECU resource Component configuration (expressed from AR ECU resource template)</p> <p>System topology for OS, communication bus and signal (expressed from AR system template)</p>		
Update variability feature tree and generate documentation	<p>Build product variant set on variation mechanism and Function models at design level</p> <p>Generate design documentation</p>	<p>WP4</p> <p>WP4</p> <p>WP3</p>	<p>Variability graphical feature capture from Papyrus</p> <p>Variability plug-in: Generate instance of variant instance deployed at Design level, verify static interface and dynamic binding time</p> <p>Documentation plug-in: Generate design document</p>
	<p>Product line feature tree deployed at Design level</p> <p>Generate design documentation at Design level</p>		

Implementation view:

As implementation level of EAST-ADL shall be aligned to AUTOSAR template representation, and Implementation level is stated as AR compliant, only the final distribution of AUTOSAR model is considering in the list of activities below.

The relation between Design model level and AR SW component is a 1:1 bi-directional relation in order to satisfy that validation performed at Design level does not need to be considered.

Furthermore in the activity below pure software and hardware verification, including integration will be described as they are not core focus of the project. Requirement activity will be described as low level of relation to implementation.

The first activity is executed on a variant instance of the system, result of Activity_PD_D_9.

Activity_PD_I_1	Requirement tree consolidated at Design level ADL Function and behavioral description for Sensor Actuator SW Component SW Component allocation (expressed from AR system template) ECU resource Component configuration (expressed from AR ECU resource template) System topology for communication bus and signal (expressed from AR system template)		
Refine and Derive requirement	Derive Requirement in the context of Implementation relation Considerer implementation constraints mainly on hardware implementation and refine requirement if necessary Propagate safety requirement. Propagate timing requirement if necessary	WP2 WP3	Requirement graphical feature capture from Papyrus Behavioral description for requirement (use case, activity diagram)
	Requirement tree refined at Implementation level		
Activity_PD_I_2	ADL Function and behavioral description for Sensor Actuator SW Component		
Code and SWC template generation	Generate C source code Satisfy AR SWC description compliance	WP3 WP3	Behavioral plug-in: ADL native code generation or external behavioral tools (Simulink, Ascet...) Behavioral plug-in: AR template compliance checks
	Source code and AR XML description SW Component AR model (expressed from AR software template)		
Activity_PD_I_3	SW Component allocation (expressed from AR system template) ECU resource Configuration (expressed from AR ECU resource template) Basic SW Component from Processor Unit (expressed from AR software template)		
Final ECU configuration	Generate configuration of basic component in relation to ECU pin and system topology configuration	WP3	Architectural graphical feature capture from Papyrus and interface to AR template edition
	Basic SW Component Configuration (expressed		

	<p>from AR ECU resource template)</p> <p>ECU resource configuration (expressed from AR ECU resource template)</p>		
Activity_PD_I_4	AR SWC description (represented in AR SWC template)		
Review requirement tree and build V&V test strategy	<p>Trace requirement to different Component in order to satisfy them</p> <p>Propagate timing requirement</p> <p>Propagate safety requirement</p> <p>Build V&V test case in relation to Component, and link V&V artifact to requirement tree</p>	<p>WP2</p> <p>WP2</p>	<p>Requirement and V&V graphical feature capture from Papyrus</p> <p>Traceability plug-in: Check requirement, model and V&V traceability (non allocated or miss of relation)</p>
	<p>Verification and Validation test case</p> <p>Requirement tree consolidated at Implementation level</p>		
Activity_PD_I_5	<p>AR SW Component (expressed from AR system template)</p> <p>Basic SW Component Configuration (expressed from AR ECU resource template)</p> <p>ECU resource configuration (expressed from AR ECU resource template)</p> <p>System topology for OS scheduling ,communication bus and signal (expressed from AR system</p>		
Binary file generation	<p>Deploy RTE and ECU configuration versus System topology</p> <p>Generate SW build for each ECU</p>	<p>Out of scope</p> <p>AR related</p>	
	<p>Binary files</p> <p>AR element and configuration</p>		

3.3.2.2 Function Design

This scenario describes Function Design analysis of system functionalities and properties at logical levels that are independent of technology details (e.g., design of software programs). Initial step for this scenario considers that the new feature to integrate is already part of the feature view and initial feature tree view. Main activities described are analysis and design for integration of the feature work to an existing product line.

A dedicated focus on quality objectives will be highlighted in order to insure a sound architecture and dependability objective of the system.

Feature view:

Note that feature view has already been set up, with initial capture of requirements, and this activity will be not described, as it could relate to previous scenario Product Design.

This means that initial requirement tree, feature and product line descriptions (initial feature tree) already exist.

Architectural view:

At this analysis view, the main objective is to design the architecture of the function and to plug it into the existing structure of the product. Means abstract ADL Function and optional Function Device exist.

Activity_FD_A_1	Initial requirement tree Vehicle feature tree Option a) RIF import of requirement		
Refine and Derive requirement, Identify and qualify safety and quality requirements	Refine initial requirement in the context of ADL Function and Functional Device Derive Requirement Option a) direct import of requirement Identify and mark safety requirement. Fulfill (or check) quality requirement Trace requirement to ADL Function or Functional Device Model in order to satisfy them	WP2 WP3	Requirement graphical feature capture from Papyrus Behavioral description for requirement (use case, activity diagram)
	Requirement tree refined		
Activity_FD_A_2	Functional Model Functional Device Model Requirement tree refined		
Integrate function into architectural view with interaction and behavioral Model description	Define interaction with Port and data type between the abstract ADL Function, Environment, and Functional Device element, by matching them into the existing environment, and adapt other abstract ADL function if required Define abstract behavioral description (Continuous or Discrete description for physical behavioral capture) and review impacted model, by native representation, or external tool description tool (Simulink,...) via export from ADL representation	WP3 WP3	Architectural graphical feature capture for Port and Interface description from Papyrus Behavioral plug-in : Export ADL representation for behavioral description
	Abstract ADL Function Model detailed		

	Environment Model detailed Functional Device Model detailed		
Activity_FD_A_3	Abstract ADL Function Model detailed Environment Model detailed Functional Device Model detailed Requirement tree refined		
Review requirement tree and build V&V test strategy	Trace requirement to ADL Function Model in order to satisfy them if necessary considering new context of detailed model (behavioral and Interface), and analyze requirement impact on ADL Function related to this feature Define abstract behavioral description of impacted function if necessary and trace new requirement Build V&V element in relation to Model, and link V&V artifact to requirement tree	WP2 WP2	Requirement and V&V graphical feature capture from Papyrus Traceability plug-in: Check requirement, model and V&V traceability (non allocated or miss of relation)
	Verification and Validation Model Requirement tree consolidated		
Activity_FD_A_4	Functional Model detailed Environment Model detailed Functional Device Model detailed Verification and Validation Model		
Validation of function behavioral description	Simulate for validation purpose in the context of V&V objective of the different behavioral and architectural model Simulate impacted function as non regression test	WP3	Simulate abstract model versus environment model using physical unit vehicle model (Software in the Loop)
	Functional Model validated Functional Device Model validated		
Activity_FD_A_5	Functional Model detailed Environment Model detailed Functional Device Model detailed Initial Requirement tree and V&V artifact at Architecture level		
Hazard and Functional Failure analysis	Review Hazard by impact analysis and capture new Hazard is necessary in relation to this feature model Revisit Functional Failure Analysis of the system Revisit and consolidated requirement tree and V&V artifact in context of safety	WP3	Error Model graphical feature capture from Papyrus
	Hazard list Functional failure list Safety Requirement list Requirement tree and V&V artifact revisited at Architecture level		

Activity_FD_A_6	Initial Product line feature tree deployed at Architecture level Requirement tree and V&V artifact revisited at Architecture level Architecture Model detailed (abstract ADL Function, Functional Device, Error model, ...) Environment Model detailed		
Update variability feature tree and generate documentation	Instantiate variant product line in the context of the new function and it possible	WP4	Variability graphical feature capture from Papyrus
		WP4	Variability plug-in: Generate instance of variant instance deployed at Architecture level, verify static interface
	Product line feature tree deployed at Architecture level		

Design view:

This analysis view is focusing on definition of behavioral description, dependability analysis and timing of the changes made on the product. A dedicated view on hardware impact analysis has to be considered because hardware allocations of the feature and communication network topology changes are limited. It could be considered as a product running after initial product volume.

Activity_FD_D_1	Functional Model validated Initial Requirement tree at Design level		
Refine and Derive requirement	Consider Functional Device and Functional Model Derive Requirement in the context of Design decomposition, and Refine them if necessary Propagate safety requirement. Develop initial timing requirement	WP2	Requirement graphical feature capture from Papyrus
		WP3	Behavioral description for requirement (use case, activity diagram)
	Requirement tree refined at Design level		
Activity_FD_D_2	Environment Model detailed Hazard list		
Environment Design	Create sensor actuator environment (with discrete model) optimized in context of RPU and validate it Merge this environment model with already existing environment model (vehicle + others sensor actuator) Revisit test case for hazard scenario	WP3	Simulate environment model (electrical unit vehicle model and sensor actuator model) in context of RPU and SIL
	New Environment Model (Vehicle and Sensor Actuator model) Verification and Validation Model for safety test		
Activity_FD_D_3	Functional Model detailed		

	Requirement tree refined at Design level		
Functional Design	<p>Refine Functional Model of the ADL Function with Composition, and define interaction with implementation data type compatible with existing interface</p> <p>Define detailed behavioral description (Discrete description), by native representation, or external tool description tool (Simulink,...) via export from ADL representation.</p> <p>Check if function still reach initial abstract performance due to interface change</p> <p>Build Software fault models of the new function</p>	<p>WP3</p> <p>WP3</p>	<p>Architectural graphical feature capture for Port and Interface description from Papyrus</p> <p>Behavioral plug-in : Export ADL representation for behavioral description</p>
	<p>ADL Function Models (with Interface) and behavioral model detailed</p> <p>Add on Runnable entities of the SW Component Models</p> <p>Software fault model</p>		
Activity_FD_D_4	<p>Functional Device Model detailed</p> <p>Environment Model optimized</p> <p>Requirement tree refined at Design level</p> <p>AR system template (topology, OS, bus)</p> <p>AR ECU resource template (Pin, Processor Unit)</p>		
Hardware Design analysis	<p>Build ADL Function and behavioral definition on the new Sensor Actuator Component to fit to 1:1 AR SWC with objective to minimize resources, and map interfaces with ADL Function of the new function.</p> <p>Simulate Sensor Actuator component model in relation to its environment model</p> <p>Build Hardware fault models of the sensor actuator and timing and error model associated.</p> <p>Identify in different ECU resource template and on the system, basic software component required, and resource space (Pin, CPU, RAM, ROM, bus context) and decide a physical allocation on these criteria</p> <p>Build initial timing model for the function</p>	<p>WP3</p> <p>WP3</p> <p>WP3</p>	<p>Architectural graphical feature capture for Port and Interface description from Papyrus</p> <p>Behavioral plug-in : Export ADL representation for behavioral description,</p> <p>Architectural graphical feature capture from Papyrus and interface to AR template edition</p>
	<p>ADL Function for Sensor Actuator Software Component Model (expressed from AR SWC template)</p> <p>AR system template (topology, OS, bus)</p> <p>AR ECU resource template (Pin, Processor Unit)</p> <p>Function software timing models</p>		
Activity_FD_D_5	<p>ADL Function Model (SWC and Sensor Actuator Component)</p> <p>Requirement tree refined at Design level</p>		
Review requirement tree and build V&V	Trace requirement to different Model in order to satisfy them	WP2	Requirement and V&V graphical feature capture

test strategy	Propagate timing requirement Propagate safety requirement Build V&V element in relation to Model, and link V&V artifact to requirement tree	WP2	from Papyrus Traceability plug-in: Check requirement, model and V&V traceability (non allocated or miss of relation)
Verification and Validation Model Requirement tree consolidated at Design level			
Activity_FD_D_6	ADL Function Models (SWC and Sensor Actuator Component) AR system template (topology, OS, bus) AR ECU resource template (Pin, Processor Unit, Bus connection) Software fault models Hardware fault models		
Architecture safety analysis	Generate initial fault tree of the new system with propagation of error model Evaluate the new architecture in consideration of analysis fault tree Detect possible failure node, and change ECU allocation if error, for finally detection of redundancy needs If hardware or software functional redundancy is required Analysis and Design phase are restarted at the adequate activity level.	WP3 WP3 WP3	Error model graphical feature capture from Papyrus Error Model plug-in: Generation of Fault tree and safety analysis Architectural graphical feature capture from Papyrus and interface to AR template edition
Fault tree models AR system template (topology, OS, bus) AR ECU resource template (Pin, Processor Unit, Bus connection) Updated Hardware and Function redundancy list			
Activity_FD_D_7	ADL Function Models (SWC and Sensor Actuator Component) Environment Model optimized (Vehicle and Sensor Actuator model) Verification and Validation Model Hazard safety test scenario		
Validation of SW Component integration Model	Perform architecture timing analysis, end to end timing analysis Perform OS scheduling timing analysis Validate with in the context of V&V objective of the different behavioral, and architectural and timing model If allocation need to reconsidered, safety analysis need to reassessed with Activity_FD_D_6	WP3 WP3	Behavioral plug-in: Scheduling analysis, end to end analysis Simulate model versus environment model (vehicle and sensor Actuator model (SIL or RPU))
System topology validation			

<p>Activity_FD_D_8</p>	<p>Initial Product line feature tree deployed at Design level</p> <p>Requirement tree and V&V artifact revisited at Design level</p> <p>Architecture Model detailed (ADL Function, Functional Device, Error model, ...)</p> <p>Environment Model</p> <p>SW Component allocation (expressed from AR system template)</p> <p>ECU resource Component configuration (expressed from AR ECU resource template)</p> <p>System topology for OS, communication bus and signal (expressed from AR system template)</p>		
<p>Update variability feature tree</p>	<p>Build product variant set on variation mechanism and Function models at design level</p>	<p>WP4</p> <p>WP4</p>	<p>Variability graphical feature capture from Papyrus</p> <p>Variability plug-in: Generate instance of variant instance deployed at Design level, verify static interface and dynamic binding time</p>
<p>Product line feature tree deployed at Design level</p>			

Implementation view:

The relation between Design model level and AR SW component is a 1:1 bi-directional relation in order to satisfy that validation performed at Design level does not need to be considered.

Furthermore in the activity below pure software and hardware verification, including integration will be described as they are not core focus of the project. Requirement activity will be described as low level of relation to implementation.

The first activity is executed on a variant instance of the system, result of Activity_FD_D_8.

<p>Activity_FD_I_1</p>	<p>ADL Function and behavioral description for Sensor Actuator SW Component for the function</p> <p>ECU resource Component configuration (expressed from AR ECU resource template)</p>		
<p>Code and SWC generation</p>	<p>Generate C source code</p> <p>Satisfy AR SWC description compliance</p> <p>Generate configuration of basic component in relation to ECU pin and system topology configuration</p>	<p>WP3</p> <p>WP3</p>	<p>Behavioral plug-in: ADL native code generation or external behavioral tools (Simulink, Ascet..)</p> <p>Behavioral plug-in: AR template compliance checks</p>
<p>Source code and AR XML description</p> <p>SW Component AR model (expressed from AR software template)</p> <p>Basic SW Component Configuration (expressed from AR ECU resource template)</p>			

<p>Activity_FD_I_2</p>	<p>Initial Requirement tree refined at Implementation level AR SWC description (represented in AR SWC template) ECU resource Component configuration (expressed from AR ECU resource template) System topology (expressed from AR system template)</p>		
<p>Refine and Derive requirement and build V&V test strategy</p>	<p>Derive Requirement in the context of the Implementation relation to the hardware selected Propagate safety requirement. Propagate timing requirement if necessary Build V&V test case in relation to Component, and link V&V artifact to requirement tree</p>	<p>WP2</p>	<p>Requirement graphical feature capture from Papyrus</p>
	<p>Verification and Validation test case Requirement tree consolidated at Implementation level</p>		
<p>Activity_FD_I_3</p>	<p>AR SW Component (expressed from AR system template) Basic SW Component Configuration (expressed from AR ECU resource template) ECU resource configuration (expressed from AR ECU resource template) System topology for OS scheduling ,communication bus and signal (expressed from AR system</p>		
<p>Binary file generation</p>	<p>Deploy RTE and ECU configuration versus System topology Generate SW build for ECU concerned</p>	<p>Out of scope AR related</p>	
	<p>Binary files AR element and configuration</p>		

3.3.2.3 System Integration

This scenario focuses on final system integration in which the application design and platform solutions are put together. It demonstrates the integration of application software and the underlying middleware/platform software for a specific hardware for a product or its subsystems

It will start at a point where the instance of product a line is selected, meaning application software, environment and initial hardware architecture is defined. The platform/middle characteristic is defined; the critical point is to validate the system requirements in the context of quality centric and analysis of dependability.

The following description of activities is only based on one instance of a variant of a product line.

Feature view:

No specific activity, already completed as described in Product Design scenario.

Architectural view:

No specific activity, already completed as described in Product Design scenario.

Design view:

Main step of similar to Product Design scenario are completed, considering design description, the following activities relates only model integration scenario. The initial step is that an instance of a product variant is set up, but no relation to hardware and middleware is defined.

<p>Activity_SI_D_1</p>	<p>Requirement tree consolidated at Design level including Verification and Validation test</p> <p>AR SWC description (represented in AR SWC template)</p> <p>Initial AR ECU resource description with basic SW Component and Processor Unit description (represented in AR ECU resource template)</p> <p>Hardware fault and timing models</p> <p>Software fault and timing models</p>		
<p>Build initial topology of the system</p>	<p>Build Initial system topology, and bus communication and OS scheduling from requirement and model analysis (represented in AR system template) fitting with allocation constraints capture in requirement tree</p>	<p>WP3</p>	<p>Architectural graphical feature capture from Papyrus and interface to AR template edition</p>
	<p>ADL Function for Sensor Actuator Software Component Model (expressed from AR SWC template)</p> <p>Initial AR system description (topology, OS, bus)</p> <p>Refined AR ECU resource description (Pin, Processor Unit, Basic software configuration)</p>		

<p>Activity_SI_D_2</p>	<p>AR SWC description and Model (represented in</p>
-------------------------------	---

	<p>AR SWC template) and behavioral models</p> <p>Initial AR system template (topology, OS, bus)</p> <p>Initial AR ECU resource template (Pin, Processor Unit, Bus connection)</p> <p>Hardware fault and timing models</p> <p>Software fault and timing models</p>		
Architecture exploration and safety analysis	<p>Based on initial system topology, instantiate initial relation between software and hardware error model (considering context of variant handling)</p> <p>Generate initial fault tree</p> <p>Deploy architecture into a system configuration and propagate error model into the architecture</p> <p>Evaluate different hardware and software architecture in consideration of analysis fault tree</p> <p>Identify failure node and allocation constraints</p> <p>Detect possible failure node where redundancy is required</p> <p>If hardware or software functional redundancy is required Analysis and Design phase are restarted at the adequate activity level.</p>	<p>WP3</p> <p>WP3</p> <p>WP3</p>	<p>Error model graphical feature capture from Papyrus</p> <p>Error Model plug-in: Generation of Fault tree and safety analysis</p> <p>Architectural graphical feature capture from Papyrus and interface to AR template edition</p>
	<p>Fault tree models</p> <p>Preliminary SW Component allocation (expressed from AR system template)</p> <p>Preliminary System topology for communication bus and signal (expressed from AR system template)</p> <p>Hardware and Function redundancy list</p>		
Activity_SI_D_3	<p>Preliminary SW Component allocation (expressed from AR system template)</p> <p>Preliminary System topology for communication bus and signal (expressed from AR system template)</p> <p>Hardware timing models</p> <p>Software timing models</p> <p>Verification and Validation Model</p>		
Architecture timing analysis	<p>Define Os Task configuration list (expressed from AR system template) in according to scheduling strategy and ECU considered</p> <p>Perform architecture timing analysis, end to end timing analysis</p> <p>Perform OS scheduling timing analysis</p> <p>This activity should ensure that safety analysis is not corrupted otherwise Activity_SI_D_2 must be executed</p>	<p>WP3</p> <p>WP3</p>	<p>Architectural graphical feature capture from Papyrus and interface to AR template edition</p> <p>Behavioral plug-in: Scheduling analysis, end to end analysis</p>
	<p>SW Component allocation (expressed from AR system template)</p> <p>System topology for communication bus and signal (expressed from AR system template)</p> <p>ECU resource Component configuration</p>		

(expressed from AR ECU resource template)
 Os Task configuration list (expressed from AR system template)

Implementation view:

Implementation level is quite similar to Product Design. Furthermore, in the activity below pure software and hardware verification, including integration, will be described as they are not core focus of the project. Requirement activity will be described as low level of relation to implementation.

Activity_SI_I_1	Requirement tree consolidated at Design level AR SWC description (represented in AR SWC template) ECU resource Component configuration (expressed from AR ECU resource template) System topology for OS, communication bus and signal (expressed from AR system template)		
Refine and Derive requirement	Derive Requirement in the context of Implementation relation Considerer implementation constraints mainly on hardware implementation and refine requirement if necessary Propagate safety requirement. Propagate timing requirement if necessary	WP2 WP3	Requirement graphical feature capture from Papyrus Behavioral description for requirement (use case, activity diagram)
	Requirement tree refined at Implementation level		
Activity_SI_I_2	AR SWC description (represented in AR SWC template) System topology for OS, communication bus and signal (expressed from AR system template) ECU resource Configuration (expressed from AR ECU resource template)		
Final ECU configuration	Generate configuration of basic component in relation to ECU pin and system topology configuration	WP3	Architectural graphical feature capture from Papyrus and interface to AR template edition
	Basic SW Component Configuration (expressed from AR ECU resource template) ECU resource configuration (expressed from AR ECU resource template)		
Activity_SI_I_3	AR SWC description (represented in AR SWC template)		
Review requirement tree and build V&V test strategy	Trace requirement to different Component in order to satisfy them Propagate timing requirement Propagate safety requirement Build V&V test case in relation to Component, and	WP2 WP2	Requirement and V&V graphical feature capture from Papyrus Traceability plug-in: Check requirement, model and V&V traceability (non

	link V&V artifact to requirement tree		allocated or miss of relation)
	Verification and Validation test case Requirement tree consolidated at Implementation level		
Activity_SI_I_4	AR SW Component (expressed from AR system template) Basic SW Component Configuration (expressed from AR ECU resource template) ECU resource configuration (expressed from AR ECU resource template) System topology for OS scheduling ,communication bus and signal (expressed from AR system		
Binary file generation	Deploy RTE and ECU configuration versus System topology Generate SW build for each ECU	Out of scope AR related	
	Binary files AR element and configuration		

3.4 Evaluation

The EAST-ADL graphical editor based on a UML2 profile, Papyrus, developed from scratch. Consequently, it took several months to build the tool prototype in order to be able to assess the EAST-ADL profile (tool expected in M1.5 milestone but available in M2).

In order to prepare the case study architecture, its connection with a behavioral model, and the final demonstrator set up, a Simulink version of the case study has been set up (completed at M2 milestone). Simulink models were restructured according to the Integrated Safety Projects (Including e.g. EASIS, PReVENT and AIDE) architecture and debugged for an official release beta1 (at M2.5). Case study is executable in Simulink, but does not include models for Engine Management System and Torque Moderator due to incompatible format (only available in ASCET).

In parallel to prepare demonstrator material for dissemination, Simulink models have been transformed to embedded C-Code and integrated with an ASCET model (Engine Management and partly torque moderator) into the Intecrio tool (ETAS). From this environment, the demonstrator is either executable on rapid prototyping system, or on virtual prototype (Intecrio VP).

Assessment of the EAST-ADL Intermediate profile starting at M2 milestone was performed based on Vehicle View, Analysis View and part of Design View, in conjunction with some requirement samples.

Nevertheless, main feedback where reported and capitalized until M2.5 milestone, on:

- profile (profile structuring, UML and SysML concept usage, links between artifact, data type concept, relation to AUTOSAR profile)
- tool (ergonomic usage, graphical features, and bug reporting)
- case study itself (missing or uncompleted parts, in particular the relation to AUTOSAR)

Mains issues revealed during this evaluation were:

- Port and data type data concept that need to be clarified, considering also relation to MARTE profile
- AUTOSAR relation to EAST-ADL that need to be consolidated due to AUTOSAR profile style relying on UML2 standard association.
- Basic software model representation that is not explicit in AUTOSAR resource template
- bugs on graphical feature of papyrus disturbing evaluation activities (composite structure diagram and relation between artifacts)

Some methodological issues were also identified during the progress of the project, and will be evaluated during the second period:

- AUTOSAR Template compliance versus function modeling approach
- Moving context of AUTOSAR ECU resource and system template
- Basic software component description in AUTOSAR
- Error model and relation to different abstraction levels
- Instantiation of error in regards to open system topology offered by AUTOSAR infrastructure
- MARTE relation into the Domain model
- Native behavior in EAST-ADL that need to be extended with scope clarification
- Relation of behavioral models (Simulink, ASCET,...) in the process view
- adjustment to AADL specially in the context of Error model

- Environment behavioral representation and relation to physical models

The capability of the tool chain has been demonstrated. The main advantage is the choice of an open solution in the Eclipse environment, where plug-in mechanisms will allow subsequent integration of verification and transformation tools.

The final scenario for demonstrator has not been formally sketched. Initial proposal is based on an architectural and design description of the case study in EAST-ADL with Papyrus. It is linked to Simulink for behavioral definitions of contained components or with native description in EAST-ADL, and then inter-connected to Intecrio (Rapid Prototyping Unit demonstrator) for allocation exploration. Extension toward EAST-ADL implementation view and connection to AUTOSAR Software Component (AUTOSAR SWC) is built on Intecrio interchange format capabilities. AUTOSAR SWC generated from Intecrio (XML scheme), are re-imported into Papyrus via AUTOSAR-OTF interchange format including AUTOSAR standard validation, while papyrus complete the overall architecture and design consistency and but also traceability. Demonstrator is executed on Intecrio Virtual Platform for Rapid Prototyping Unit.

Precise scenario and tools plug-in specification still need to be detailed, but the basic concept and potential for demonstration are defined at this stage.

4 Contribution to overall ATESST objectives

In the context of ATESST, the refined scenarios describe use cases for the ADL as well as the methods and tools. These scenarios will be used for

- Reflecting experience of industrial practice
- Validation of the WP2-5 solutions and their efficiency
- Refinement of requirements elicitation if required
- Base for further improvement and new solutions

The final demonstrator will also be based on extract from these scenarios as support for promotion of ATESST results.

These scenarios could be used as basic description of methodologies developed in the engineering work packages, and as example of possible process definition using the EAST-ADL language.

5 References

- [1] ATESST_Deliverable_D6.1.1_Part_I_Scenarios_V1.0.pdf
- [2] ATTEST_Deliverable_D6.1.1_Part_II_SOTASOP_V1.0.pdf
- [3] ATESST_Deliverable_D6.2.1_V1.0.pdf