

## ECSEL Research and Innovation actions (RIA)



# AMASS

## Architecture-driven, Multi-concern and Seamless Assurance and Certification of Cyber-Physical Systems

### AMASS demonstrators (a) D1.4

<b>Work Package:</b>	WP1 Case Studies and Benchmarking
<b>Dissemination level:</b>	PU = Public
<b>Status:</b>	Final
<b>Date:</b>	30 April 2017
<b>Responsible partner:</b>	Miguel Gomez (Thales Alenia Space - Spain) Juan Castillo (Thales Alenia Space - Spain)
<b>Contact information:</b>	Miguel.GomezIrusta@thalesaleniaspace.com Juan.CastilloRevuelta@thalesaleniaspace.com
<b>Document reference:</b>	AMASS_D1.4_WP1_TAS_V1.0

#### PROPRIETARY RIGHTS STATEMENT

This document contains information, which is proprietary to the AMASS Consortium. Neither this document nor the information contained herein shall be used, duplicated or communicated by any means to any third party, in whole or in parts, except with prior written consent of the AMASS consortium.

---

## Contributors

Names	Organisation
Miguel Gomez, Juan Castillo	Thales Alenia Space – Spain
Alejandra Ruiz, Huáscar Espinoza, Garazi Juez, Estibaliz Amparan	TECNALIA Research & Innovation
Javier Herrero Martín, Elena Alaña Salazar	GMV Aerospace and Defence, S.A.U
Fredrik Warg, Martin Skoglund	RISE Research Institutes of Sweden
Benito Caracuel, David Pampliega	Schneider Electric
Anna Carlsson	OHB Sweden
Barbara Gallina, Inmaculada Ayala, Irfan Sljivo	MDH
Zhendong Ma, Dorottya Papp, Thomas Gruber	Austrian Institute of Technology
Thierry Lecomte	ClearSy
Bernhard Kaiser, Behrang Monajemi	Assystem Germany GmbH (formerly: Berner&Mattner Systemtechnik GmbH)

## Reviewers

Names	Organisation
Benito Caracuel / David Pampliega (Peer-Reviewer)	Schneider Electric España
Thierry Lecomte (Peer Reviewer)	Clearsy SAS
Cristina Martínez (Quality Manager)	Tecnalia Research & Innovation
Jose Luis de la Vara	Universidad Carlos III de Madrid

## Document History

Version	Date	Status	Author (Partner)	Remarks
V0.1	2017-01-27	ToC	Miguel Gómez (TAS)	
V0.2	2017-04-14	First draft version	Miguel Gómez (TAS)	
V0.3	2017-04-21	First full version	Miguel Gómez (TAS)	
V0.4	2017-04-24	Ready for WP review	Miguel Gómez (TAS)	
V0.5	2017-04-24	Ready for TC review	Miguel Gómez (TAS)	
V1.0	2017-04-30	Approved by TC	Huáscar Espinoza (TEC)	Final version

# TABLE OF CONTENTS

<b>Executive Summary.....</b>	<b>7</b>
<b>1. Introduction.....</b>	<b>8</b>
1.1. Scope and Purpose .....	9
1.2. Structure of the Document .....	9
<b>2. Background.....</b>	<b>10</b>
2.1. AMASS Prototyping Roadmap.....	10
2.2. AMASS Usage Scenarios per Case Study .....	10
2.3. AMASS Tool Approach Basics.....	11
2.4. Challenges in implementing AMASS Case Studies .....	12
2.4.1. Comparison of AMASS Scenarios with Real Projects.....	12
2.4.2. Timing for the First Prototype Setting .....	12
<b>3. Case Study Realization.....</b>	<b>14</b>
3.1. Case Study 1: Industrial Automation domain: Industrial and Automation Control Systems (IACS) .....	14
3.1.1. Case Study Specification .....	14
3.1.2. Usage Scenario 1: Managing compliance with IEC 61508, IEC 62443 and IEC 62351 .....	14
3.1.3. Usage Scenario 2: Perform safety and security co-assessment .....	20
3.2. Case Study 3: Automotive domain: Collaborative automated fleet of vehicles. ....	29
3.2.1. Case Study Specification .....	29
3.2.2. Usage Scenario 1: Safety Assessment of collaborative automated vehicle functions by model-based safety analysis and fault injection simulations .....	29
3.3. Case Study 4: Space domain: Design and safety assessment of on-board software applications in Space Systems.....	36
3.3.1. Case Study Specification .....	36
3.3.2. Usage Scenario 1: Baseline (Common to all the Usage Scenarios) .....	36
3.4. Case Study 5: Railway domain: Platform Screen Doors Controller .....	43
3.4.1. Case Study Specification .....	43
3.4.2. Usage Scenario 1: Generation of Frama-C asserted C code from B models .....	43
3.4.3. Usage Scenario 2: Support for system-level model, including safety and security aspects .....	44
3.5. Case Study 10: Space domain: Certification basis to boost the usage of MPSoC architectures in the Space Market .....	47
3.5.1. Case Study Specification .....	47
3.5.2. Usage Scenarios US1 and US3 .....	47
3.6. Case Study 11: Space domain: Design and efficiency assessment of model based Attitude and Orbit Control software .....	58
3.6.1. Case Study Specification .....	58
3.6.2. Usage Scenario 1 - Managing compliance with ECSS-E-ST-40C .....	58
<b>4. Conclusions.....</b>	<b>62</b>
<b>5. Appendix A: Ongoing Activities on other Case Studies .....</b>	<b>63</b>
5.1. Case Study 2: Automotive domain: Advanced driver assistance function with electric vehicle sub-system.....	63
5.1.1. Case Study Specification .....	63
5.1.2. Usage Scenario 1 .....	63
5.2. Case Study 7: Avionics domain: Safety assessment of multi-modal interactions in cockpits.....	66
5.2.1. Case Study Specification .....	66
5.2.2. Usage Scenario 1: Safety Assessment.....	66

5.2.3. Usage Scenario 2: Automated Verification .....	68
5.3. Case Study 8: Automotive domain: Telematics function.....	70
5.3.1. Case Study Specification .....	70
5.3.2. Usage Scenarios (Common to all the Scenarios) .....	70
5.4. Case Study 9: Air Traffic Management domain: Safety-Critical SW Lifecycle of a Monitoring Syst. for NavAid .....	72
5.4.1. Case Study Specification .....	72
5.4.2. Usage Scenario 1: SWD (Software Design) .....	72
5.4.3. Usage Scenario 2: SWV (Software Verification & Validation) .....	73
<b>Abbreviations and Definitions.....</b>	<b>74</b>
<b>References .....</b>	<b>76</b>

## List of Figures

Figure 1.	AMASS Building blocks .....	8
Figure 2.	IEC61508 – Part 3 extract and its counterpart model view using the Standards Editor from OpenCert.....	16
Figure 3.	Excerpt of 62443-4-2 standard model using the Standards Editor from OpenCert.....	17
Figure 4.	Structure of the two assurance projects created for CS 1.....	18
Figure 5.	Evidence model for the CS1-RTU assurance project, referring to the company own files naming conventions .....	19
Figure 6.	Web application to follow the compliance progress of the assurance project .....	20
Figure 7.	View to follow the compliance progress of the assurance project in the OpenCert client.....	20
Figure 8.	ISASecure EDSA conformance scheme (source: ISASecure) .....	21
Figure 9.	Baseline software platform architecture in Papyrus.....	22
Figure 10.	Baseline software platform architecture in MORETO .....	23
Figure 11.	Example security requirements for Devices Configuration Files in Papyrus (left) and MORETO (right) .....	23
Figure 12.	Example security requirements of Web Server in Papyrus (left) and MORETO (right).....	24
Figure 13.	IEC 62443-4-2 requirement hierarchy in Papyrus (left) and MORETO (right) .....	24
Figure 14.	Relationship matrix of the RTU components and its requirements in MORETO .....	25
Figure 15.	Overview of IEC 62443-4-2 and RTU specification.....	26
Figure 16.	Snippet of relevant security requirements from IEC 62443-4-2 .....	27
Figure 17.	OpenCert argumentation module.....	28
Figure 18.	CS3 Main Scenario .....	29
Figure 19.	Simplified System Architecture in the model car.....	31
Figure 20.	Simplified state machine .....	34
Figure 21.	CACC argument-fragment created in OpenCert .....	35
Figure 22.	CHESS Model Structure .....	37
Figure 23.	Data Types Diagram .....	38
Figure 24.	Interfaces Diagram .....	39
Figure 25.	Components Diagram.....	40
Figure 26.	VAM_Controller_CImp Composite Diagram.....	40
Figure 27.	First Diagram of the OLCI OPSW Architecture .....	41
Figure 28.	Second Diagram of the OLCI OPSW Architecture.....	41
Figure 29.	Copilot system with its different subsystems: laser scanner, steel wheel sensor and the door control command.....	43
Figure 30.	High level functional decomposition .....	45
Figure 31.	System Environment Architecture .....	45
Figure 32.	ECSS-E-40 Standard Structure in OpenCert for CS10 .....	49
Figure 33.	ECSS-E-40 Standard Life Cycle in OpenCert for CS10 .....	50
Figure 34.	ECSS-E-40 Standard Requirements and Outputs in OpenCert for CS10.....	50
Figure 35.	Assurance Project Process in OpenCert for CS10.....	52
Figure 36.	System Design & Dependability Assessment in CS10.....	54
Figure 37.	Evidence Management / Metrication CS10.....	56
Figure 38.	Parsed requirements .....	59
Figure 39.	Excerpt of the extended EARM used for the semi-automatic import of ECSS requirements .....	60
Figure 40.	Method composer requirements ECSS-E-ST-40.....	60
Figure 41.	Method composer SW validation and mapped requirements .....	61
Figure 42.	Quality management System OHB.....	61

## List of Tables

<b>Table 1.</b>	Proposed functional groups: summary .....	11
<b>Table 2.</b>	CS1 – Standards Models Creation .....	14
<b>Table 3.</b>	CS1 – Assurance Project Creation .....	17
<b>Table 4.</b>	CS1 – Evidence Management.....	18
<b>Table 5.</b>	CS1 – Compliance Management .....	19
<b>Table 6.</b>	CS1 – Model based safety&security product requirement management.....	21
<b>Table 7.</b>	CS1 – Safety & Security co-analysis.....	25
<b>Table 8.</b>	CS1 – Safety & Security Assurance Case .....	27
<b>Table 9.</b>	CS3 – Assurance Project Creation .....	29
<b>Table 10.</b>	CS3 – System Component Specification .....	30
<b>Table 11.</b>	CS3 – Evidence Management.....	33
<b>Table 12.</b>	CS4 – Assurance Project Creation .....	36
<b>Table 13.</b>	CS4 – System Design and Dependability Assessment .....	36
<b>Table 14.</b>	CS4 – Evidence Management.....	42
<b>Table 15.</b>	CS4 – Compliance Management .....	42
<b>Table 16.</b>	CS5 – US1: Safety Assessment .....	43
<b>Table 17.</b>	CS5 – US2: Model-Based System Component Specification.....	44
<b>Table 18.</b>	CS5 – US2: Safety Assessment .....	45
<b>Table 19.</b>	CS10 – Standard Model Creation .....	48
<b>Table 20.</b>	CS10 – Assurance Project Creation .....	51
<b>Table 21.</b>	CS10 – System Design and Dependability Assessment .....	53
<b>Table 22.</b>	CS10 – Evidence Management.....	55
<b>Table 23.</b>	CS10 – Compliance Management .....	56
<b>Table 24.</b>	CS11 – Standards Model Creation.....	58
<b>Table 25.</b>	CS11 – Compliance Management .....	60

## Executive Summary

The AMASS project is developing the first European-wide open certification/qualification platform for assurance and certification of Cyber-Physical Systems (CPS). Task T1.4 provides feedback and an active proof of the performance of the AMASS tools in the industry.

In order to evaluate such a disruptive tool, WP1 “Case Studies and Benchmarking” will provide support and advice to the tool’s developers (WP3-WP6) for future iterations based on case studies. Those case studies represent meaningful segments of the different application domains addressed in AMASS. WP1 in future iterations will provide benchmarking for AMASS tools more widely, when task 1.3 “Benchmarking Framework” is finished.

This task (T1.4) also will be input for WP2 “Reference Architecture and Integration” in validation of AMASS platform and user guidance methodological framework.

This document (deliverable D1.4) describes the result of evaluating the tools by industrial partners and its first introduction to industry domains. Partners have focused on modelling standards depending on its domain (industrial automation, automotive, railway, avionics, space and air traffic), establishing an assurance project, and using the tools of the different main building blocks that the tools are created for.

Data required to develop this task has been taken from the deliverable D1.2 [3], which is related to data collection usage scenarios for each case study described in D1.1 [2].

Deliverable D1.4 focuses on validating the Prototype Core tools. The following case studies have been selected to use AMASS tools in this iteration:

- CS1: Industrial Automation case study: Industrial and Automation Control Systems (IACS)
- CS3: Automotive case study: Collaborative automated fleet of vehicles
- CS4: Space case study: Design and safety assessment of on-board software applications in Space Systems
- CS5: Railways case study: Platform screen-doors controller
- CS10: Space case study: Certification basis to boost the usage of Multiprocessor System-on-Chip (MPSoC) architectures in the Space Market
- CS11: Space case study: Design and efficiency assessment of model based Attitude and Orbit Control software development.

The following case studies developed conceptual work. Their activities are presented in Appendix A.

- CS2: Automotive case study: Advanced driver assistance function with electric vehicle sub-system
- CS7: Avionics case study (and cross-domain plus Automotive): Safety assessment of multi-modal interactions in cockpits
- CS8: Automotive case study: Telematics function
- CS9: Air Traffic Management domain: Safety-Critical SW Lifecycle of a Monitoring Syst. for NavAid

CS6 has changed from the initial proposal, due to the fact that one contributor left the consortium and Alstom took its place, with the consequent delay. Anyway, the contributions will be complete for next iterations of this deliverable planned throughout the project lifetime.

Finally, this document provides input for the implementation tasks in the technical work packages, in the form of feedback about aspects that could be improved or addressed in the future.

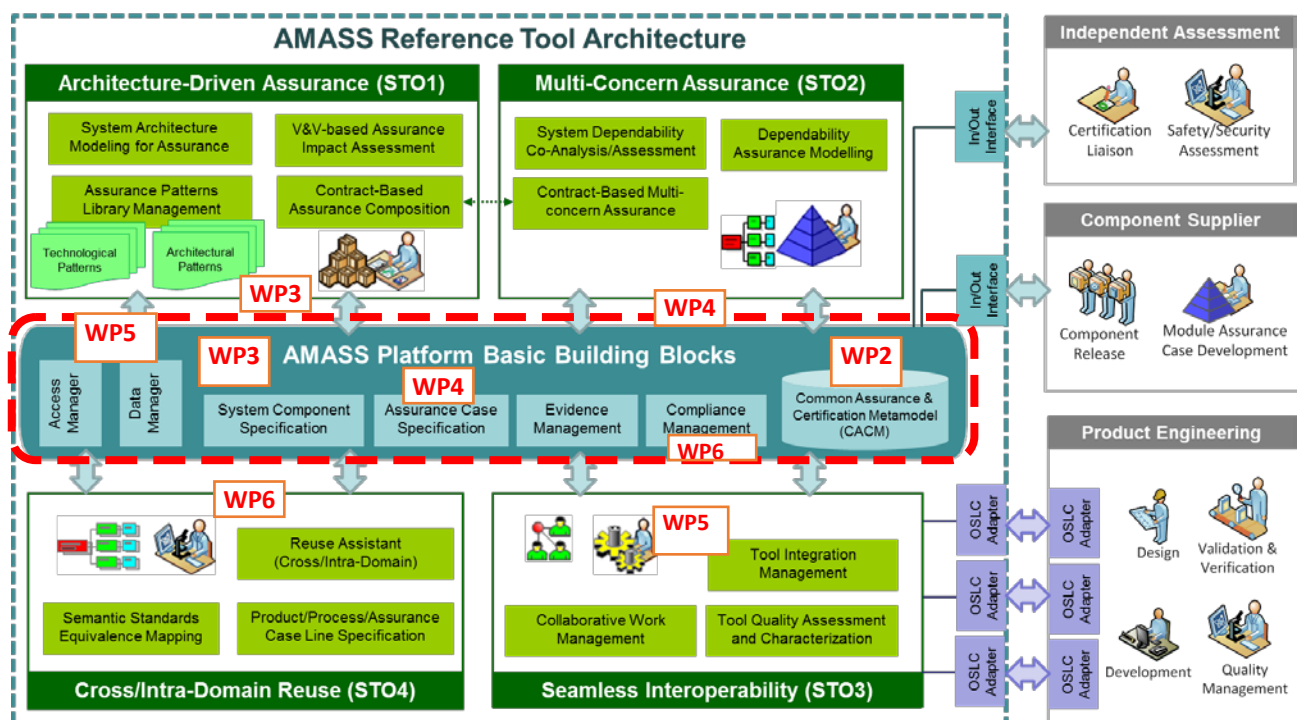
# 1. Introduction

The AMASS approach focuses on the development and consolidation of an open and holistic assurance and certification framework for CPS, which constitutes the evolution of the approaches proposed by the EU projects OPENCROSS [11] and SafeCer [13] towards an architecture-driven, multi-concern assurance, reuse-oriented, and seamlessly interoperable tool platform.

The expected tangible AMASS results are:

- The **AMASS Reference Tool Architecture**, which will extend the OPENCROSS and SafeCer conceptual, modelling and methodological frameworks for architecture-driven and multi-concern assurance, as well as for further cross-domain and intra-domain reuse capabilities and seamless interoperability mechanisms.
- The **AMASS Open Tool Platform**, which will correspond to a collaborative tool environment supporting CPS assurance and certification. This platform represents a concrete implementation of the AMASS Reference Tool Architecture, with a capability for evolution and adaptation, which will be released as an open technological solution by the AMASS project.
- The **Open AMASS Community**, which will manage the project outcomes, for maintenance, evolution and industrialization. The Open Community will be supported by a governance board, and by rules, policies, and quality models. This includes support for AMASS base tools (tool infrastructure for database and access management, among others) and extension tools enriching AMASS platform functionalities.

To achieve the AMASS results, as depicted in Figure 1, the multiple challenges and corresponding scientific and technical project objectives are addressed by different work-packages.



**Figure 1.** AMASS Building blocks

The scope of this deliverable (D1.4) is the Prototype “Core”, which covers the AMASS Platform Basic Building Blocks in the middle of Figure 1.



## **1.1. Scope and Purpose**

The objective of this deliverable is to validate the first prototype of the AMASS solution and to provide feedback for future prototypes. This first deliverable related to the task T1.4 “Case Study Implementation and Benchmarking” is based in the case study specifications from task T1.1, as well as from the data collection usage scenarios presented in deliverable D1.2 [3]. Deliverable D1.4 (and its next iterations) are key-task to evolve the work done since now. Task 1.4 provides the user validation for the developing work packages and is in charge of benchmarking in real projects the capability of the AMASS solution.

For this first iteration, only the implementation of the AMASS Platform Basic Building Blocks will be covered. Benchmarking will be addressed in the next deliverable (D1.5), when task T1.3 “Benchmarking Framework” will be completed with a validated and stable benchmarking framework.

Besides, given the importance of the usability when dealing with software tools, stakeholders from AMASS industrial partners will dedicate time and effort to give feedback about the targeted features and usability to AMASS tool providers (solution developers). The results of the industrial participation will be matched with the AMASS technical requirements and test cases (WP3-WP6) and the achievement of the goals, from the end-user perspective.

## **1.2. Structure of the Document**

The rest of the deliverable is structured as follows: Section 2 offers an overview of the schedule of the project, the basics for understanding the tools that have been used and the main issues and challenges encountered while proving these tools. In Section 3 each case study will present an assessment of the platform, and some preliminary results. Section 4 provides conclusions.

## 2. Background

### 2.1. AMASS Prototyping Roadmap

Since AMASS targets high-risk objectives, the AMASS Consortium decided to follow an incremental approach by developing rapid and early prototypes. The benefits of following a prototyping approach are:

- Better assessment of ideas by initially focusing on a few aspects of the solution.
- Ability to change critical decisions based on practical and industrial feedback (case studies).

The AMASS project has three milestones (M2 to M4) to demonstrate this incremental evolution:

1. During the **first prototyping** iteration (Prototype Core), the AMASS Platform Basic Building Blocks (see Figure 1), will be aligned, merged and consolidated. This iteration covers the basic functionality as specified by the project backend needs. Since the beginning of the project, every technical work package (WP3-WP6) has contributed to complete the first prototype until milestone M2.
2. During the **second prototyping** iteration (Prototype P1), the AMASS-specific Building Blocks will be developed and benchmarked at TRL4; this comprises the blue basic building blocks as well as the green building blocks in Figure 1. By milestone M3, the second prototype must be available with the improvements and new features included. For this second prototype, test benches will be done based in the patterns developed by other task that are not currently finished ([4]).
3. Finally, at the **third prototyping** iteration (Prototype P2), all AMASS building blocks shall be integrated in a comprehensive toolset operating at TRL5. The third and last prototype will conclude the project with all the features and functionalities (by milestone M4).

Each of these iterations has the following three prototyping dimensions:

- **Conceptual/research development:** development of solutions from a conceptual perspective.
- **Tool development:** development of tools implementing conceptual solutions.
- **Case study development:** development of industrial case studies using the tool-supported solutions.

This project deliverable (D1.4) summarises the results of the Case study development dimension for the first prototype (Prototype Core).

### 2.2. AMASS Usage Scenarios per Case Study

Case Studies represent different potential applications within the targeted industrial domains by the AMASS project. AMASS Usage Scenarios offer a general overview on how the AMASS solutions are intended to be used in the proposed case studies.

The approach to specify usage scenarios is based on the following principles:

- (a) **Description of usage scenarios** are centred on the AMASS platform “user” perspective (how users will interact with the AMASS tools), in the context of typical business cases. **Deliverable D1.2** [3] provides a description of usage scenarios per case study.
- (b) **Realization of usage scenarios** reports the results of the application of usage scenarios in each of the AMASS prototyping iterations. This **deliverable (D1.4)** summarizes the main results of the realization of usage scenarios by using the Prototype Core tools.
- (c) **Benchmarking of usage scenarios** will use a number of research/industrial questions and metrics to measure the effectiveness of AMASS tools regarding the proposed business goals. This will be reported in **deliverable D1.7**.

For the evaluation of the First Prototype tools, the following case studies have been selected:

- CS1: Industrial Automation case study: Industrial and Automation Control Systems (IACS).
- CS3: Automotive case study: Collaborative automated fleet of vehicles.
- CS4: Space case study: Design and safety assessment of on-board software applications in Space Systems.
- CS5: Railways case study: Platform screen-doors controller.
- CS10: Space case study: Certification basis to boost the usage of Multiprocessor System-on-Chip (MPSoC) architectures in the Space Market.
- CS11: Space case study: Design and efficiency assessment of model based Attitude and Orbit Control software development.

The following case studies developed conceptual work. Their activities are presented in Appendix A.

- CS2: Automotive case study: Advanced driver assistance function with electric vehicle sub-system.
- CS7: Avionics case study (and cross-domain plus Automotive): Safety assessment of multi-modal interactions in cockpits.
- CS8: Automotive case study: Telematics function.
- CS9: Air Traffic Management domain: Safety-Critical SW Lifecycle of a Monitoring Syst. for NavAid.

CS6 has changed from the initial proposal, due to the fact that one contributor left the consortium and Alstom took its place, with the consequent delay.

## 2.3. AMASS Tool Approach Basics

The AMASS platform is composed of a set of tools providing the functionalities described in AMASS deliverable D2.2 (AMASS Reference Architecture, Prototype Core). This first prototype has been built upon three pre-existing toolsets:

1. Tools from the pre-existing OpenCert project [9].
2. Tools from the CHES Project (Polarsys Platform) [8].
3. Tools from the EPF (Eclipse Process Framework) Project [10].

Table 1 summarizes the implemented functionalities.

**Table 1.** Proposed functional groups: summary

Functionality Group	Description
<b>System Component Specification (Papyrus+CHES)</b>	This group manages System architecture specification by decomposing a system into components. It also includes mechanisms to support compositional assurance, contract based approaches, and architectural patterns management.
<b>Assurance Case Specification (OpenCert)</b>	This group manages argumentation information in a modular fashion. It also includes mechanisms to support compositional assurance and assurance patterns management.
<b>Evidence management (OpenCert)</b>	This module manages the full lifecycle of evidence artefacts and evidence chains. This includes evidence traceability management and impact analysis.
<b>Compliance Management (OpenCert and EPF)</b>	Functionality related to the management (edition, search, transfer, etc.) of process and standards' information as well as of any other information derived from them, such as interpretations about intents and mapping between processes and standards. This functional group maintains a knowledge database about "standards & processes", which can be consulted by other AMASS functionalities.
<b>Assurance Project Lifecycle</b>	This functionality factorizes aspects such as the creation of safety assurance projects locally in AMASS and any project baseline information that may be

<b>Management (OpenCert)</b>	shared by the different functional modules. A project baseline is a subset of reference framework (e.g., subset of a standard) that will be applied to a given assurance project.
<b>Access Management (Not implemented)</b>	This is an infrastructure functional module. It includes generic functionality for security, permissions, and profiles.
<b>Data Management (Eclipse CDO)</b>	This is an infrastructure functional module. It includes generic functionality for data storage, visualization, and reporting.

From a user interface perspective, the AMASS tool platform has been realised in the form of:

- **Eclipse-based editors** are used for creating and defining process and standard models, assurance projects, assurance case argumentation, evidence and system component models.
- **Web application**, which synthesizes and summarises compliance information by means of different reports (e.g., gap analysis report), and can also be used for consulting the evidence, compliance justification, and argumentation information of an assurance project.

## 2.4. Challenges in implementing AMASS Case Studies

This section discusses the main challenges that we have found for implementing the case studies.

The wide spectrum in AMASS case studies implies a high complexity on developing a tool which satisfies all the necessities for each domain.

### 2.4.1. Comparison of AMASS Scenarios with Real Projects

WP1 in general focuses on the evaluation framework and benchmarking of AMASS tools. In particular, it aims at demonstrating the benefits of using AMASS tools with regard to current practice on safety/security assurance and certification.

One issue to work in real industrial projects was that a complete data set is not available for confidentiality and competitive pressure reasons. As mitigation measures, the following action lines were agreed upon:

1. The industrial partners sanitise the case study data for approval.
2. The scope of AMASS evaluation was initially narrowed to specific parts of the product life-cycle, still meaningful to validate AMASS benefits.

Another challenge is the comparison of AMASS results regarding the current practice in industrial companies. In practice, the only way to really compare the situation before and after the availability of the AMASS platform, would be to execute the same project twice. This is most often not economical and has methodological issues as well. For example the same team cannot be used as it would bias the second execution of the project. Hence, the most obvious method would be for a given organisation that has sufficient historical metrics, to compare how subsequent projects are executed and deliver after the AMASS is introduced and used. Another aspect that compounds the comparison is that the reuse of components and assurance artefacts is only measureable over successive projects. The first project is likely not to have much benefit as the work has to be done once, but subsequent projects can benefit from it.

As a general conclusion, we can say that the evaluation of the benefits of using the AMASS platform will mainly happen in the future in the assumption that metrics have been collected before its introduction (Task 1.3).

### 2.4.2. Timing for the First Prototype Setting

Data collection required as an input for this deliverable was finished only one month before this deliverable. In addition the first prototypes of the tools were launched by the end of January (two months before the deadline of this report). Task related to Data Collection [3] was delayed three months because

the leader of the task left the consortium, so all the partners had to work harder not to delay the whole project three months, doing all work with months less; paralleling in those situation it is impossible with the same resources. Main challenges founded are:

- First prototypes always require a first sprint of proving and detecting the problems and the understanding of how to install and run properly the applications.
- Industry standards differ widely between domains. Certification processes involve different amount of requirements and some are laxer, and some aspects neither have sense in other domains.
- Practices are very diverse, ranging from paper driven top-down processes to incremental software engineering. Hence comparison is difficult.

In the ideal case, the same project should be executed with and without AMASS support and the resource and time consumption compared. In line with the second observation, this is not a reachable goal and can be prohibitive in cost.

Benchmarking will add consistency and extra information about the needs of the future potential markets in different domains. Some more trials are going to be done with the new improvements to make every partner capable of using the tools in a perfect way.

### 3. Case Study Realization

#### 3.1. Case Study 1: Industrial Automation domain: Industrial and Automation Control Systems (IACS)

##### 3.1.1. Case Study Specification

The case study 1 is based on an IACS (Industrial and Automation Control System). These systems are in charge of the control and monitoring of the electrical infrastructures, such as the primary and secondary substations. In particular, the case study 1 focuses on the RTU (Remote Terminal Unit) devices. The RTU is one of the main elements in the control system due to they execute the commands received by the control centre, acting directly over the devices placed in the field site. Therefore, they are the last link in the control system.

In this sense, security and safety aspects are one of the primary concerns for manufacturers and end users. Standards such as: IEC 61508, IEC 62443 and IEC 62351 are the reference in the Smart Grid domain. The aim of this case study is to integrate the new AMASS tool in the lifecycle of the RTU development process in order to provide assistance for assurance and certification and taking into account these standards.

The case study is described more in depth in D1.1 “Case studies description and business impact” [2].

##### 3.1.2. Usage Scenario 1: Managing compliance with IEC 61508, IEC 62443 and IEC 62351

This usage scenario is related to process assurance, i.e. to ensure that the RTU development process follows a given set of recommendations from the targeted standards. It is composed by four general processes:

- Standards Models Creation
- Assurance Project Creation
- Evidence Management
- Compliance Management

The following subsections describe the activities realised for each one.

###### 3.1.2.1. Standards Models Creation

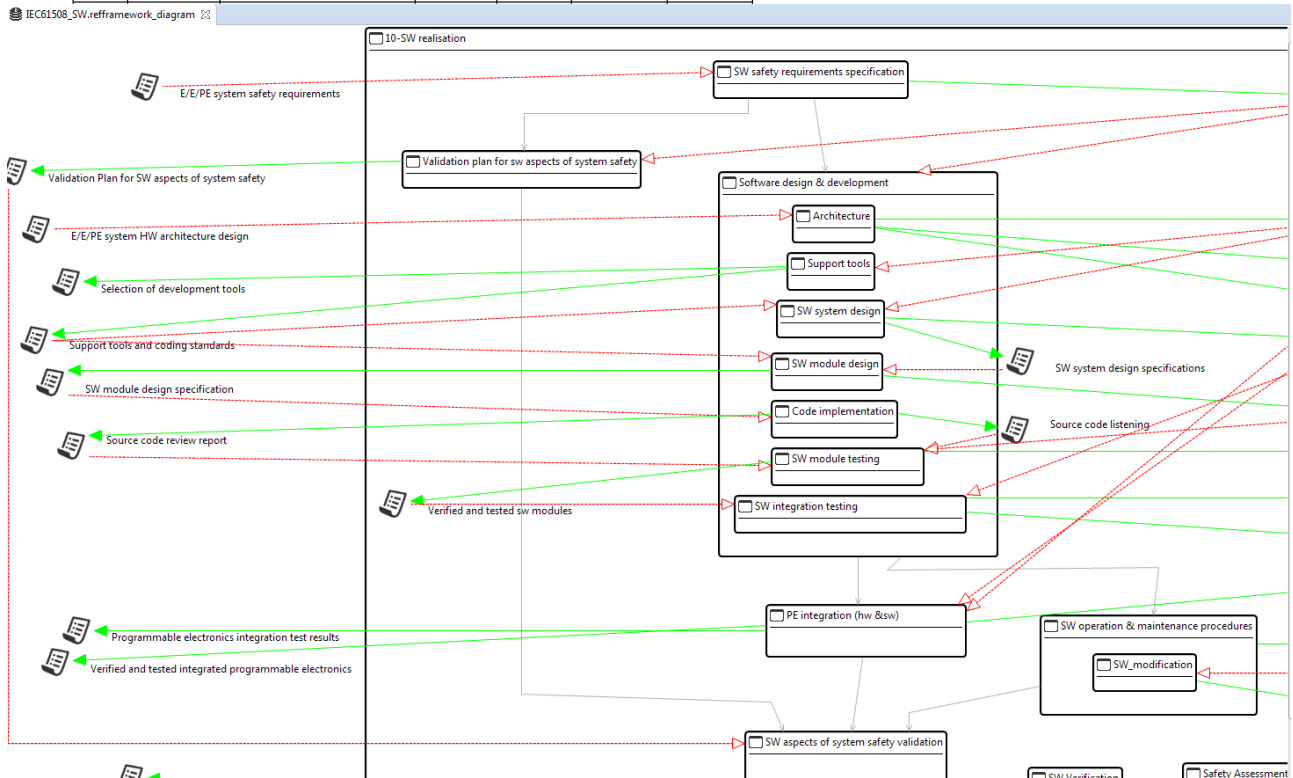
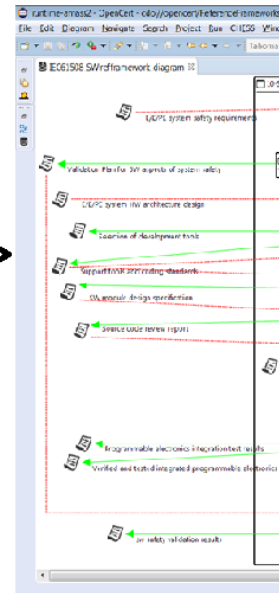
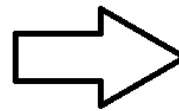
**Table 2.** CS1 – Standards Models Creation

Realisation Scenario	Standards Models Creation
Scope	Modelling of some meaningful excerpts of IEC 61508, IEC 62443 and IEC 62351. For the first prototype, we will target the following sections from these standards. <ul style="list-style-type: none"> <li>• IEC 61508 Part 3.</li> <li>• IEC 62443 Part 4.2</li> </ul> Respect to the standard IEC 62351, it will be addressed in the second prototype.
Tool Settings	OpenCert Tools: Standards Editor
Participants	<ul style="list-style-type: none"> <li>• Data Analysis: TLV</li> <li>• Tool User: TLV, TEC</li> <li>• Results Analysis: TLV, AIT</li> </ul>

<b>Activities realised</b>	<ol style="list-style-type: none"> <li>1. Conceptually analyse the structure of IEC61508, IEC 62443 as well as the core concepts such as phases, activities, artefacts, requirements and criticality levels. The goal is to map these concepts to Reference Framework concepts in OpenCert.</li> <li>2. Create a Reference Framework diagram for each of the targeted standards and populate the information related to the document sections focused on this prototype (only done for IEC 61508 and IEC 62443 at this stage).</li> <li>3. Validate the interpretations by sharing the reference framework models with other safety and security experts.</li> <li>4. At concept level, analyse the concepts from the Standards metamodel needed to be filtered by the level of capable SIL the activities, techniques and evidences to be presented for compliance</li> </ol>
<b>Usage Decisions</b>	We will not use EPF for modelling the targeted Standards (reference frameworks). Hence, the Standards models are created from scratch.
<b>Expected Results</b>	<ul style="list-style-type: none"> <li>• Reference Framework model for IEC 61508-3</li> <li>• Reference Framework model for IEC 62443-4.2</li> <li>• Conceptual knowledge to propose some support to filter by the SIL.</li> </ul>

The first step was to model the standard 61508 Part 3, with the support of OpenCert tools. In Figure 2, we can see an extract of the standard with a summary table. This table has been used as an input to the model. The different phases have been modelled as Reference Activities, the objectives column was modelled as reference objectives and the requirements column as reference requirements for that specific reference activity. Inputs and outputs columns were modelled as reference activities.

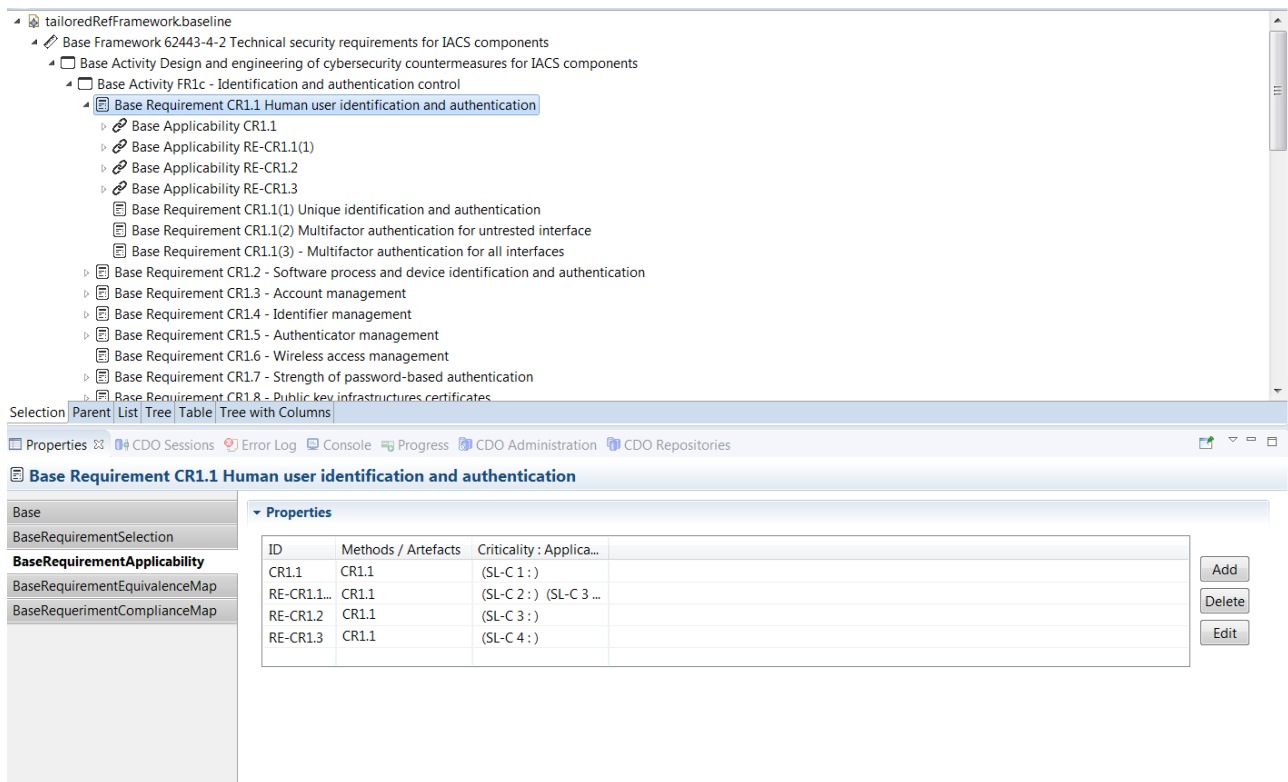
Safety lifecycle phase		Objectives	Scope	Requirements subclause	Inputs (information required)	Outputs (information produced)
Figure 4 box number	Title					
10.1	Software safety requirements specification	To specify the requirements for safety-related software in terms of the requirements for software safety functions and the requirements for software systematic capability;  To specify the requirements for the software safety functions for each E/E/PE safety-related system necessary to implement the required safety functions;  To specify the requirements for software systematic capability for each E/E/PE safety-related system necessary to achieve the safety integrity level specified for each safety function allocated to that E/E/PE safety-related system	PE system; software system	7.2.2	E/E/PE safety requirements specification as developed during allocation (see IEC 61508-1)  E/E/PE system safety requirements specification (from IEC 61508-2)	software safety requirements specification
10.2	Validation plan for software aspects of system safety	To develop a plan for validating the software aspects of system safety	PE system; software system	7.3.2	software safety requirements specification	validation plan for software aspects of system safety
10.3	Software design and development	Architecture: To create a software architecture that fulfils the specified requirements for safety-related software with respect to the required safety integrity level;  To evaluate the requirements placed on the software by the hardware architecture of the E/E/PE safety-related system, including the significance of E/E/PE hardware/software interactions for safety of the equipment under control	PE system; software system	7.4.3	software safety requirements specification; E/E/PE system hardware architecture design (from IEC 61508-2)	software architecture design; software architecture integration test specification (also required by IEC 61508-2)
10.3	Software design and development	Support tools and programming languages: To select a suitable set of tools, including languages and compilers, run-time system interfaces, user interfaces, and data formats and representations for the required safety integrity level, over the whole safety lifecycle of the software which assists verification, validation, assessment and modification	PE system; software system; support tools; programming language	7.4.4	software safety requirements specification; software architecture design	support tools and coding standards; selection of development tools



**Figure 2.** IEC61508 – Part 3 extract and its counterparty model view using the Standards Editor from OpenCert



After the safety standard, we have focused on the security standard, IEC 62443. For this case study, we have focused on part 4, as the subject of our case study, the RTU is considered as a product of the system and consequently, it is the part 4 the one that applies.



**Figure 3.** Excerpt of 62443-4-2 standard model using the Standards Editor from OpenCert

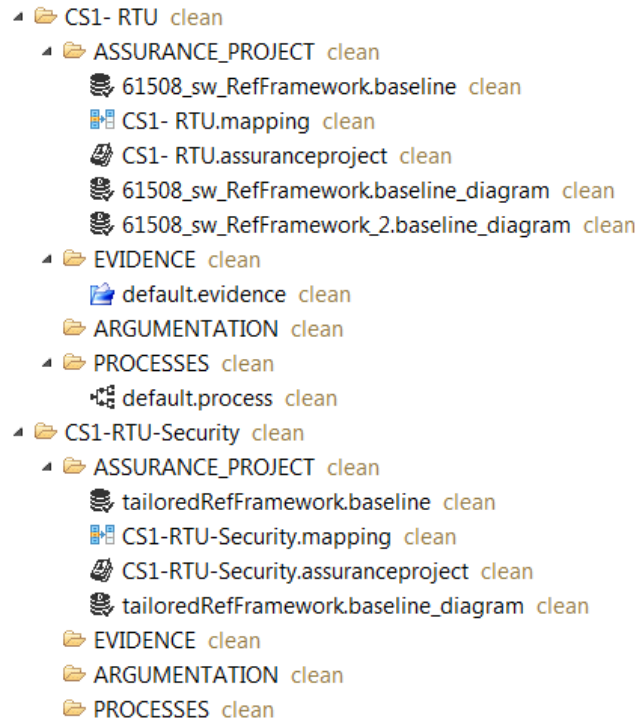
### 3.1.2.2. Assurance Project Creation

**Table 3.** CS1 – Assurance Project Creation

Realisation Scenario	Assurance Project Creation
Scope	Creation of two Assurance Projects, one for RTU Safety assurance and the other for RTU Security assurance. The scope for the first prototype is on the creation of Baseline models (instances of Reference Frameworks for specific assurance projects). Other activities related to the specification of team member roles and access permissions will be not covered in this prototype iteration, since these functionalities have not been implemented yet in AMASS tools.
Tool Settings	OpenCert Tools: Assurance Project Management Editor
Participants	<ul style="list-style-type: none"> <li>Data Analysis: TLV</li> <li>Tool User: TLV, TEC</li> </ul>
Activities realised	<ol style="list-style-type: none"> <li>Create assurance projects for RTU Safety and for RTU Security.</li> <li>When creating the Baseline models, specify the activities we will focus on for the first prototype benchmarking.</li> </ol>
Usage Decisions	None
Expected Results	<ul style="list-style-type: none"> <li>Assurance Project structure and Baseline model for RTU Safety</li> <li>Assurance Project structure and Baseline model for RTU Security</li> </ul>

During the creation of the assurance project, we have selected those parts of the standards which apply to the specific project. We have created one assurance project called CS1-RTU related to the IEC 61508

standard compliance, where only the activities of part 3 apply. A second assurance project called CS1-RTU-Security was created to reference to the IEC 62443 standard Part 4 section2 which are the technical security requirements for IACS components. In Figure 4 we can see the structure of the two assurance projects.



**Figure 4.** Structure of the two assurance projects created for CS 1.

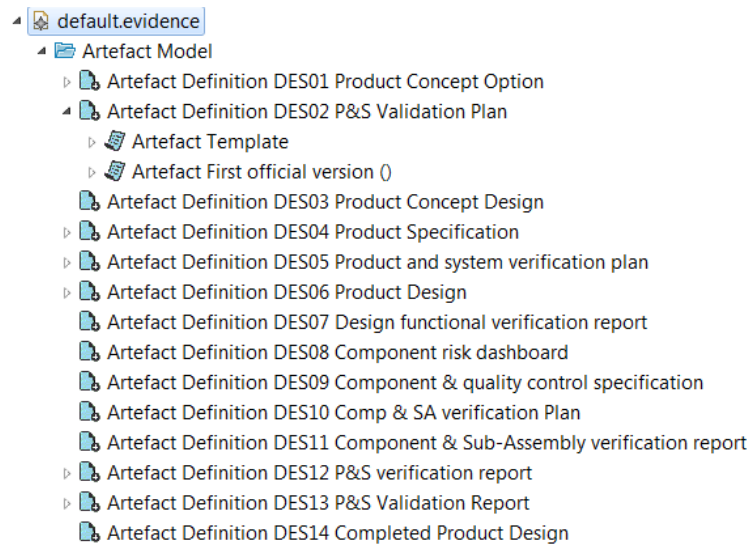
### 3.1.2.3. Evidence Management

**Table 4.** CS1 – Evidence Management

Realisation Scenario	Evidence Management
<b>Scope</b>	Characterising a subset of evidence documents for the RTU Safety and Security assurance projects in terms of authors, versions and evaluations. Some evidential documents from the RTU Security assurance project could be reused or referred in the RTU Safety assurance project. The reuse opportunities will be analysed and executed by using the Cross-Domain Assistant. A SVN repository of actual evidence documents will be created. The evidence model must have links to this repository by means of the Artefact and Resource concepts in OpenCert.
<b>Tool Settings</b>	OpenCert Tools: Evidence Management Editor SVN repository to store actual evidence documents
<b>Participants</b>	<ul style="list-style-type: none"> <li>• Data Analysis: TLV</li> <li>• Tool User: TLV, TEC</li> <li>• Results Analysis: TLV</li> </ul>
<b>Activities realised</b>	<ol style="list-style-type: none"> <li>1. Create artefact model for RTU Security and RTU Safety.</li> <li>2. Create SVN repository for RTU Security and RTU Safety.</li> <li>3. Collect evidence documents into the SVN repository for RTU Security.</li> <li>4. Specify characteristics of RTU Security artefacts.</li> <li>5. Collect evidence documents into the SVN repository for RTU Safety.</li> <li>6. Use cross-domain functionality to reuse Artefact models from RTU Security project in RTU Safety project.</li> <li>7. Complete any evaluation of the artefact elements in both assurance projects.</li> </ol>

<b>Usage Decisions</b>	Reuse of some artefacts from the RTU Security assurance project into the RTU Safety one.
<b>Expected Results</b>	<ul style="list-style-type: none"> <li>• Evidence model and artefact repository for RTU Safety</li> <li>• Evidence model and artefact repository for RTU Security</li> </ul>

We have created two evidence models respectively for each of the assurance project. The company has a well-defined process with specific documents naming that are referenced in the evidence models. The different versions used by the configuration management system are also included.



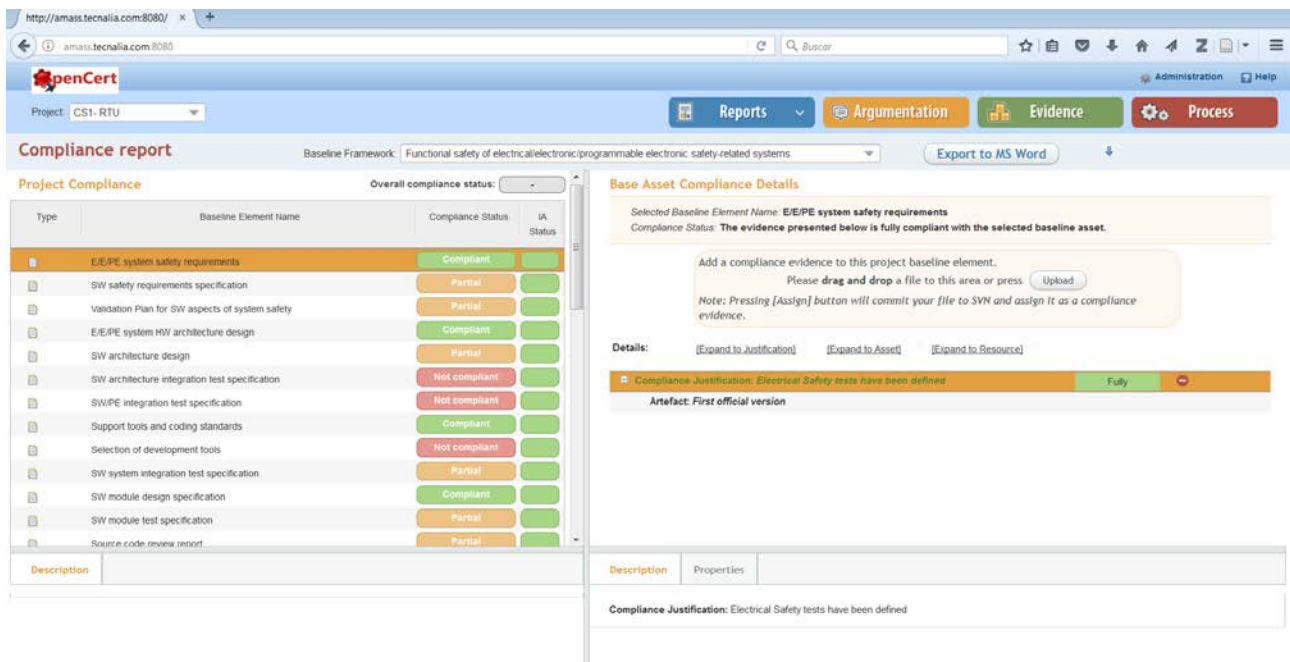
**Figure 5.** Evidence model for the CS1-RTU assurance project, referring to the company own files naming conventions

#### 3.1.2.4. Compliance Management

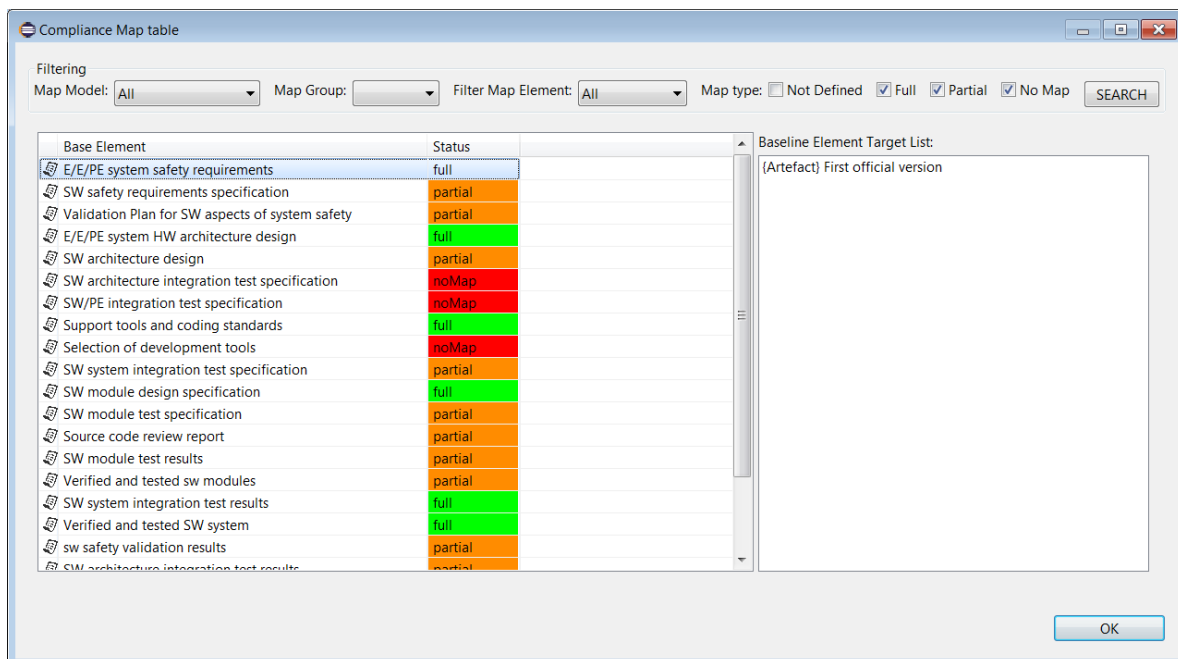
**Table 5.** CS1 – Compliance Management

Realisation Scenario	Compliance Management
<b>Scope</b>	During the first prototype, the focus will be on measuring compliance of artefacts regarding the artefacts prescribed by industry standards and their associated ones.
<b>Tool Settings</b>	OpenCert Tools: Assurance Project Management and Compliance Reporter Web Client
<b>Participants</b>	<ul style="list-style-type: none"> <li>• Data Analysis: TLV</li> <li>• Tool User: TLV, TEC</li> <li>• Results Analysis: TLV</li> </ul>
<b>Activities realised</b>	<ol style="list-style-type: none"> <li>1. Specify compliance maps for Requirements and Artefacts in both Baseline models: RTU Safety and RTU Security.</li> <li>2. Analyse compliance accomplishment and gaps for both assurance projects.</li> <li>3. Generate the compliance report for both assurance projects.</li> </ol>
<b>Usage Decisions</b>	None
<b>Expected Results</b>	<ul style="list-style-type: none"> <li>• Compliance report for IEC 61508</li> <li>• Compliance report for IEC 62443</li> </ul>

During the assurance project development, the different compliance maps have been created. A number of users were involved in the creation and reviewing of the compliance status, using the OpenCert clients and the web application.



**Figure 6.** Web application to follow the compliance progress of the assurance project



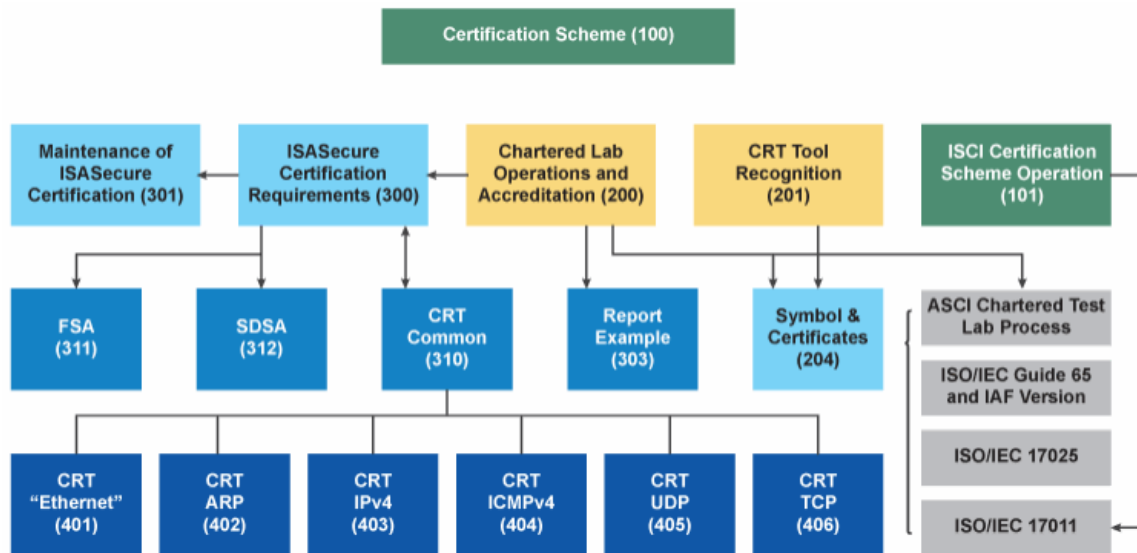
**Figure 7.** View to follow the compliance progress of the assurance project in the OpenCert client

### 3.1.3. Usage Scenario 2: Perform safety and security co-assessment

The initial objective of the US2 was to provide safety and security co-analysis and co-assessment support for the RTU design and development based on IEC 61508 for safety and IEC 62351, IEEE 1686, and IEC 62443 for cybersecurity. After a RTU analysis, we decided for this first iteration to focus mainly on IEC 62443 (the other standards will be considered depending on the need). In this sense, we will use the ISASecure EDSA certification as a benchmark to showcase the safety & security co-assessment method and the supporting tools for safety & security assurance (as an example of multi-concern assurance). It also intends to explore the AMASS approach for reducing certification time and cost leveraging reusable artefacts (e.g. evidence).

Therefore, this usage scenario is based on the scenario of preparing the Saitel RTU for the ISASecure EDSA (Embedded Device Security Assurance) certification<sup>1</sup>. The ISASecure EDSA certification focuses on the security of embedded devices. Two standards are used for the certification, i.e. IEC 62443-4-1 Product Development Requirements and IEC 62443-4-2 Technical Security Requirements for IACS Components. In US2, we will focus on product security requirements specified in IEC 62443-4-2 as well as the additional security requirements from ISASecure EDSA specification.

The ISASecure EDSA conformance scheme comprises of five categories. Our focus is on the technical requirements.



**Figure 8.** ISASecure EDSA conformance scheme (source: ISASecure)

### 3.1.3.1. Model-based safety & Security product requirement management

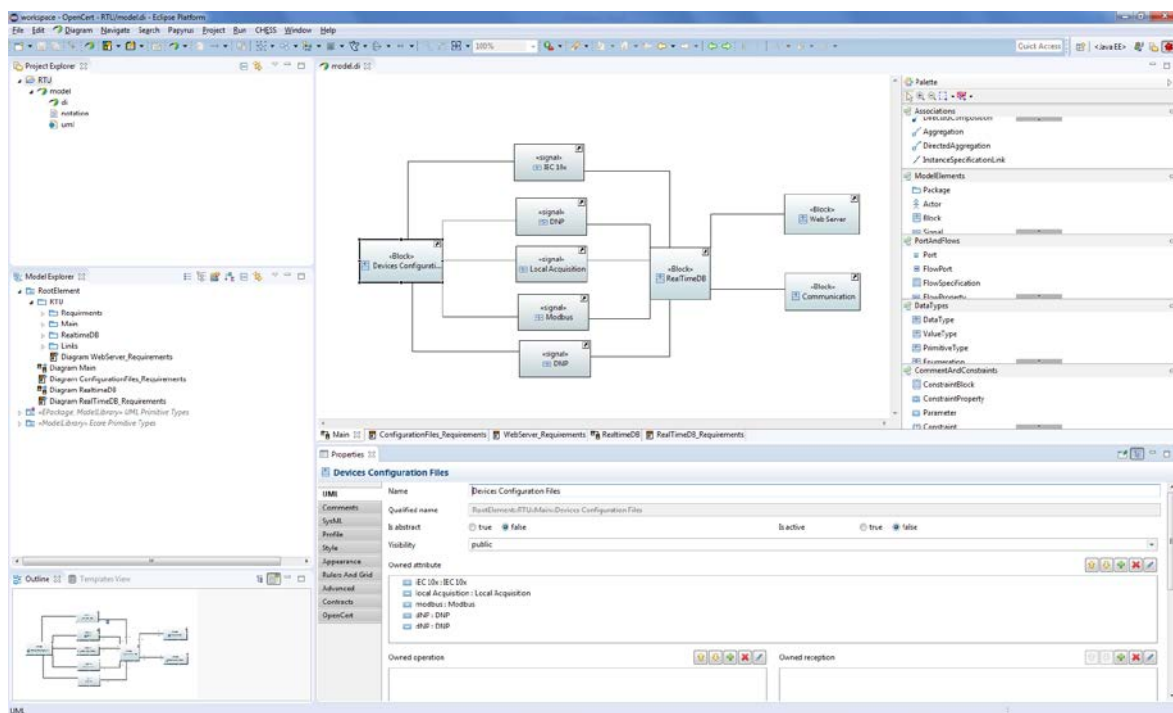
**Table 6.** CS1 – Model based safety&security product requirement management

Realisation Scenario	Model-based safety & security product requirement management
<b>Scope</b>	<p>We will use a subset of the following standards as a basis for safety and security product-related requirements. Related requirements will be selected, additional requirements might be derived based on system specifics.</p> <ul style="list-style-type: none"> <li>IEC 61508</li> <li>IEC 62443-4-2</li> </ul>
<b>Tool Settings</b>	<ul style="list-style-type: none"> <li>Model-based Requirement Management Tool (MORETO), which is an extension of Sparx Systems Enterprise Architect, developed by AIT</li> <li>Eclipse Papyrus</li> </ul>
<b>Participants</b>	<ul style="list-style-type: none"> <li>System specification: TLV</li> <li>Tool user: AIT</li> <li>Result evaluation: TLV</li> </ul>

<sup>1</sup> ISASecure, <http://www.isasecure.org/en-US/Certification/IEC-62443-EDSA-Certification>

<b>Activities realised</b>	<ol style="list-style-type: none"> <li>1. Model the RTU system in SysML diagram</li> <li>2. Import related requirements as SysML requirement diagram</li> <li>3. Identify and link requirements related to specific parts of the system model from (1)</li> <li>4. Specify additional safety and security requirements based on co-analysis</li> <li>5. Evaluate the results of the correctness of the requirements and soundness of the approach with relation to multi-concern assurance</li> </ol>
<b>Usage Decisions</b>	<p>MORETO is an external tool to the AMASS platform. It might be possible to import the artefacts into AMASS platform using third-party adapter. However, in the meantime, the same models will be duplicated in Papyrus, as an internal tool for the AMASS platform.</p> <p>As a starting point, we will focus on IEC 62443-4-2 for cybersecurity. Other standards such as IEC 61508, IEC 62351 and IEEE 1686 will be considered depending on the need.</p>
<b>Expected Results</b>	<ul style="list-style-type: none"> <li>• Product-related system and component safety and security requirement specification in the form of a list and in package of SysML diagrams.</li> </ul>

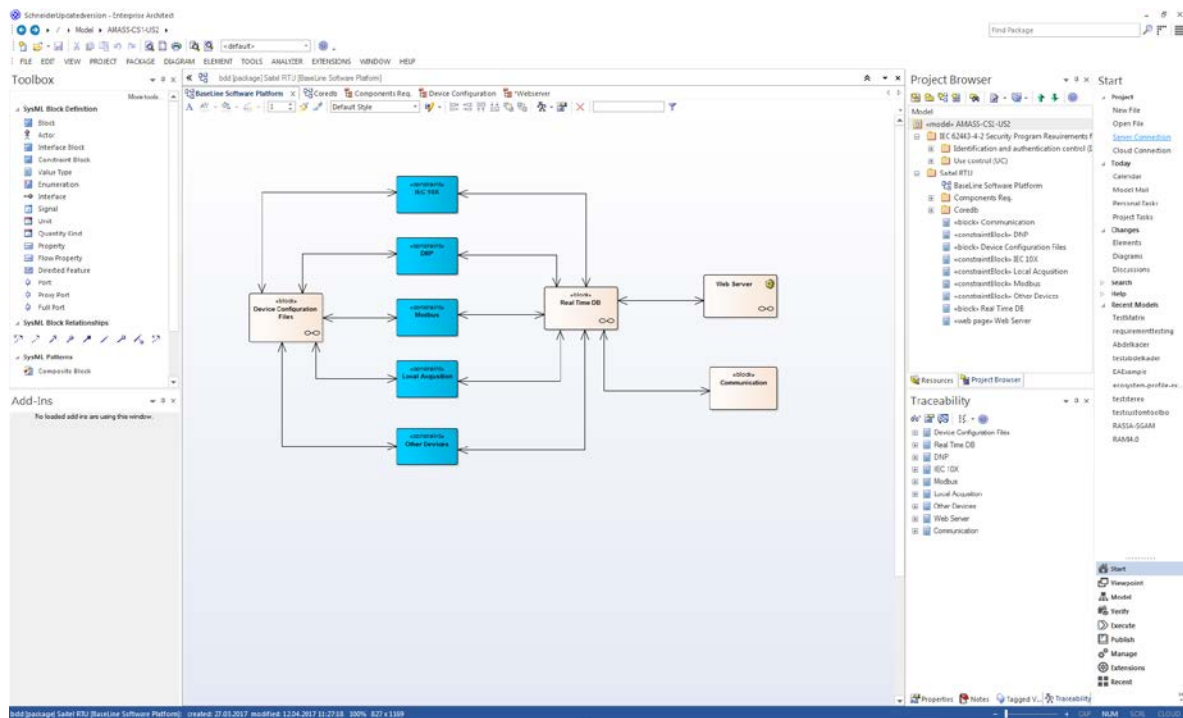
The next two figures show the baseline software platform architecture using two modelling tools. Figure 9 implements in Papyrus modelling environment, where Figure 10 is designed by MORETO tool.



**Figure 9.** Baseline software platform architecture in Papyrus

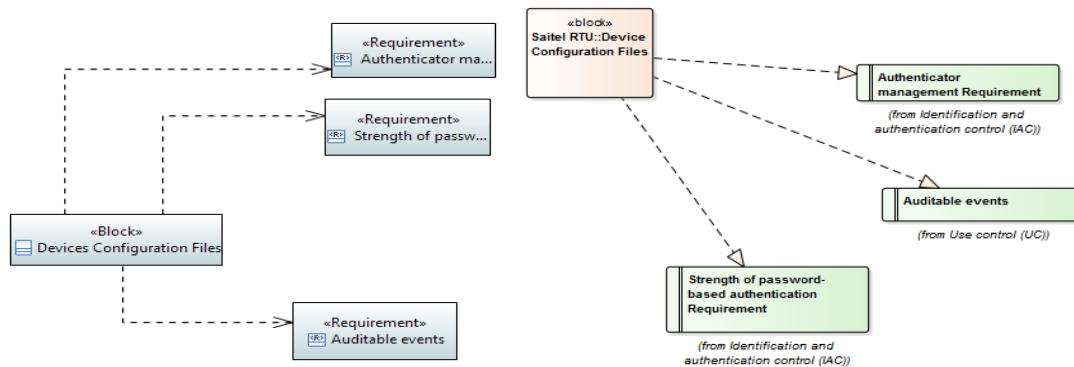
The Block diagram is the primary kind of diagrams which is used to describe the structural information about a system. Consequently, these two diagrams were implemented in SysML Block diagram.





**Figure 10.** Baseline software platform architecture in MORETO

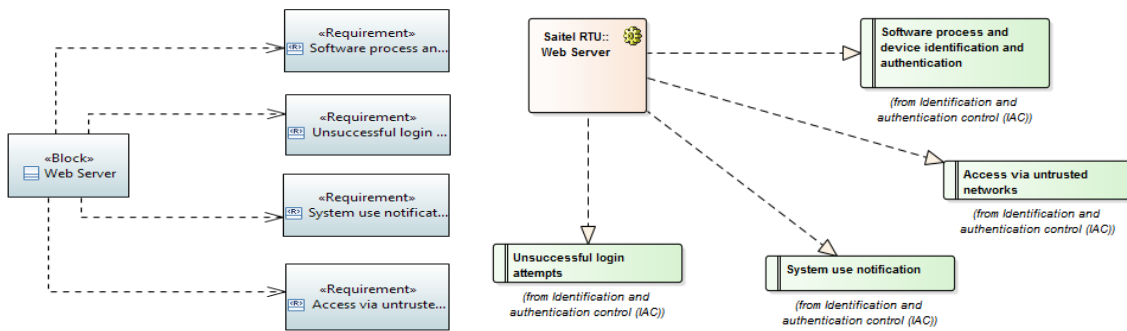
**Device Configuration Files Unit.** The device configuration file contains information that is required to read backup data and restore the database. Related security requirements according to IEC 62443-4-2 are modelled in SysML requirement diagram.



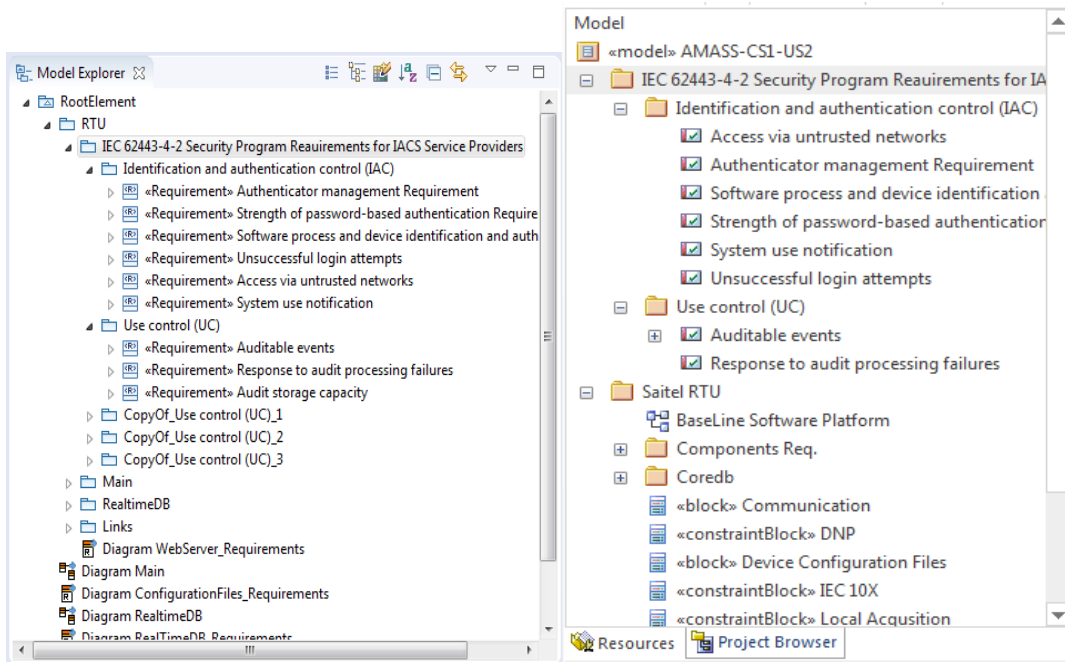
**Figure 11.** Example security requirements for Devices Configuration Files in Papyrus (left) and MORETO (right)

The above figure shows some of selected security requirements according to **Device configuration Files Unit** in Papyrus and MORETO.

**Web Server Unit.** It provides simple Web-based management functions.



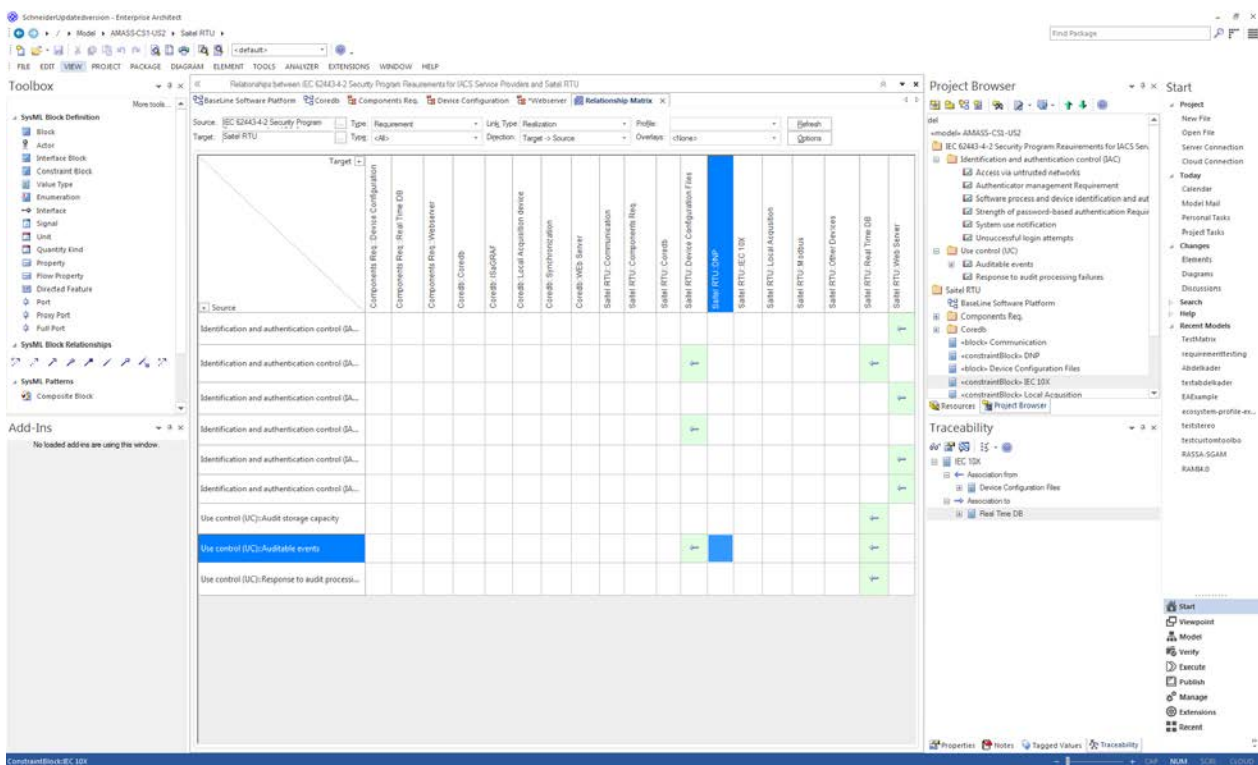
**Figure 12.** Example security requirements of Web Server in Papyrus (left) and MORETO (right)



**Figure 13.** IEC 62443-4-2 requirement hierarchy in Papyrus (left) and MORETO (right)

**Requirement management in MORETO.** MORETO is an extension of Enterprise Architect. Enterprise Architect integrates requirements management with other software development disciplines, by creating requirements directly in the model. Requirements management is built into the core product, solving many of the issues of traceability, interdisciplinary team divisions, integration with change and configuration management systems. In addition, Enterprise Architect provides the relationship matrix to allow create and view relationships between components and its requirements.





**Figure 14.** Relationship matrix of the RTU components and its requirements in MORETO

MORETO has a very useful feature to manage a huge number of requirements in a matrix representation which let user to create and view relationships among requirements as is shows in the above figure.

### 3.1.3.2. Safety & Security Co-analysis

**Table 7.** CS1 – Safety & Security co-analysis

Realisation Scenario	Safety & Security Co-analysis
<b>Scope</b>	<ul style="list-style-type: none"> <li>We conduct analysis of the product security requirements from IEC 62443-4-2 and ISASecure EDSA specification in order to 1) interpret technical requirements with respect to concrete product and usage context, 2) conduct requirement allocation, and 3) decide testing method and tool chain for assurance proof.</li> <li>We use FMVEA analysis to identify safety hazards and security threats, and their interrelations.</li> </ul>
<b>Tool Settings</b>	Excel sheet, optionally Microsoft Threat Modelling Tool for additional threat identification.
<b>Participants</b>	<ul style="list-style-type: none"> <li>System specification: TLV</li> <li>Data analysis: AIT</li> <li>Result evaluator: TLV, AIT</li> </ul>
<b>Activities realised</b>	<ol style="list-style-type: none"> <li>TLV provides system description.</li> <li>AIT conduct FMVEA analysis that cover both safety and security aspects, related standards and additional information (e.g. NVD database, NESCOR Pentest guide) will be taken into consideration.</li> <li>Evaluate the results.</li> </ol>
<b>Usage Decisions</b>	The approach in this activity should be repeatable to be applied to other IACS components for the same assurance scheme.

**Expected Results**

- List of Relevant product security requirements
- Method and specification of tools for security testing
- List of safety hazards and security threats

IEEE 62443-4-2 is the basis for ISASecure EDSA certification scheme. The requirements from IEEE 62443-4-2 are analysed to identify the relevant requirements to Saitel RTU. The results of the analysis are kept in a excel sheet. The following screen shot is an overview of the analysis of 62443-4-2 standards: what are the functional requirements and what security levels the device can achieve. For example, there are 7 foundational requirements (FRs), IEC 62443-4-2 refers to other documents, 4 security levels are considered, and the product to be “certified”.

Overview	
7 foundational requirements (FRs)	
Identification and authentication control (IAC)	
Use control (UC)	
System integrity (SI)	
Data confidentiality (DC)	
Restricted data flow (RDF)	
Timely response to events (TRE)	
Resource availability (RA)	
Referenced documents of IEC 62443-4-2	
IEC 62443-1-1	
IEC 62443-3-3	
ISO/IEC 19790:2012	
Security levels: measure of confidence	
SL1 Prevent the unauthorized disclosure of information via eavesdropping or casual exposure	
SL2 Prevent the unauthorized disclosure of information to an entity actively searching for it using simple means with low resources, generic skills and low motivation	
SL3 Prevent the unauthorized disclosure of information to an entity actively searching for it using sophisticated means with moderate resources, IACS specific skills and moderate motivation	
SL4 Prevent the unauthorized disclosure of information to an entity actively searching for it using sophisticated means with extended resources, IACS specific skills and high motivation	
IACS is composed of	
applications	
host device	
network component	
embedded device	
Schneider product to certify	
SM_COU866e	
purpose: RTU	
has a dual core processor	
uses the 32-bit Power Architecture	
memory...	
connectivity: USB, SD slot, RS-232, Ethernet	
synchronization: GPS, IRIG-B, SNTP, Protocol (PTP1588)	
OS: Linux	
cyber security: SEC 3.3.2 security engine (XOR acceleration), crypto algorithms, single pass encryption/message auth, IPsec, SSL, SRTP, CyberSecurity Brick	
under IEC 62443-4-2, the product qualifies as EMBEDDED DEVICE	

**Figure 15.** Overview of IEC 62443-4-2 and RTU specification

The following screen shot is the excel sheet containing the requirements for embedded devices. In the excel sheet, we include the requirements categorized into the (high-level) functional requirements. We also include the criteria needed for each security assurance level.

Security Requirements of Embedded Devices	
embedded device	special purpose device running embedded software designed to directly monitor, control or actuate an industrial process Typical attributes: no rotating media, limited number of exposed services, programmed through an external interface, embedded OS or firmware equivalent, real-time
Identification and authentication control (IAC)	
goal	Identify and authenticate all users (humans, software processes and devices) by mechanisms which protect against
for SL1	casual or coincidental access by unauthenticated entities
for SL2	intentional unauthenticated access by entities using simple means with low resources, generic skills and low motivation.
for SL3	intentional unauthenticated access by entities using sophisticated means with moderate resources, IACS specific skills and moderate motivation.
for SL4	intentional unauthenticated access by entities using sophisticated means with extended resources, IACS specific skills and high motivation.
Human user, process and device identification and authentication	
	capability to identify and authenticate all human users according to SR 1.1 [IEC 62443-3-3]
	<i>enforce such identification and authentication on all interfaces which provide human user access</i>
	<i>support segregation of duties and least privilege</i>
	<i>should be accomplished by using methods such as passwords, tokens, biometrics or, in the case of multifactor authentication, some combination thereof</i>
	<i>geographic location of human users can also be used as part of the authentication process.</i>
	<i>should be applied to both local and remote access</i>
	<i>comes on top of the requirement of having such an authentication and identification at the system level</i>
	<i>user identification and authentication may be role-based or group-based</i>
	<i>should not hamper fast local emergency actions</i>
	<i>the device verifies the identity of all human users as a first step. In a second step, the permissions assigned to the identified human user are enforced</i>
for SAL-C1	the above mentioned criteria
for SAL-C2	SL1 + unique identification and authentication
for SAL-C3	SL2 + multifactor authentication for untrusted interface (interface to untrusted network)
for SAL-C4	SL1 + multifactor authentication on all interfaces
Device authentication	
goal	provide authentication methods for device identification prior to establishing a connection
failures in cryptography services	
	embedded device shall not be dependent on outside cryptography services that could result in denial of service for the embedded device
	required on all ISASecure levels
basic device authentication	
	embedded device shall provide at least basic measures for authentication of device identification
	required above ISASecure level 1
cryptographic device authentication	
	embedded device shall provide cryptographic measures for authentication of device identification
	required above ISASecure level 2

**Figure 16.** Snippet of relevant security requirements from IEC 62443-4-2

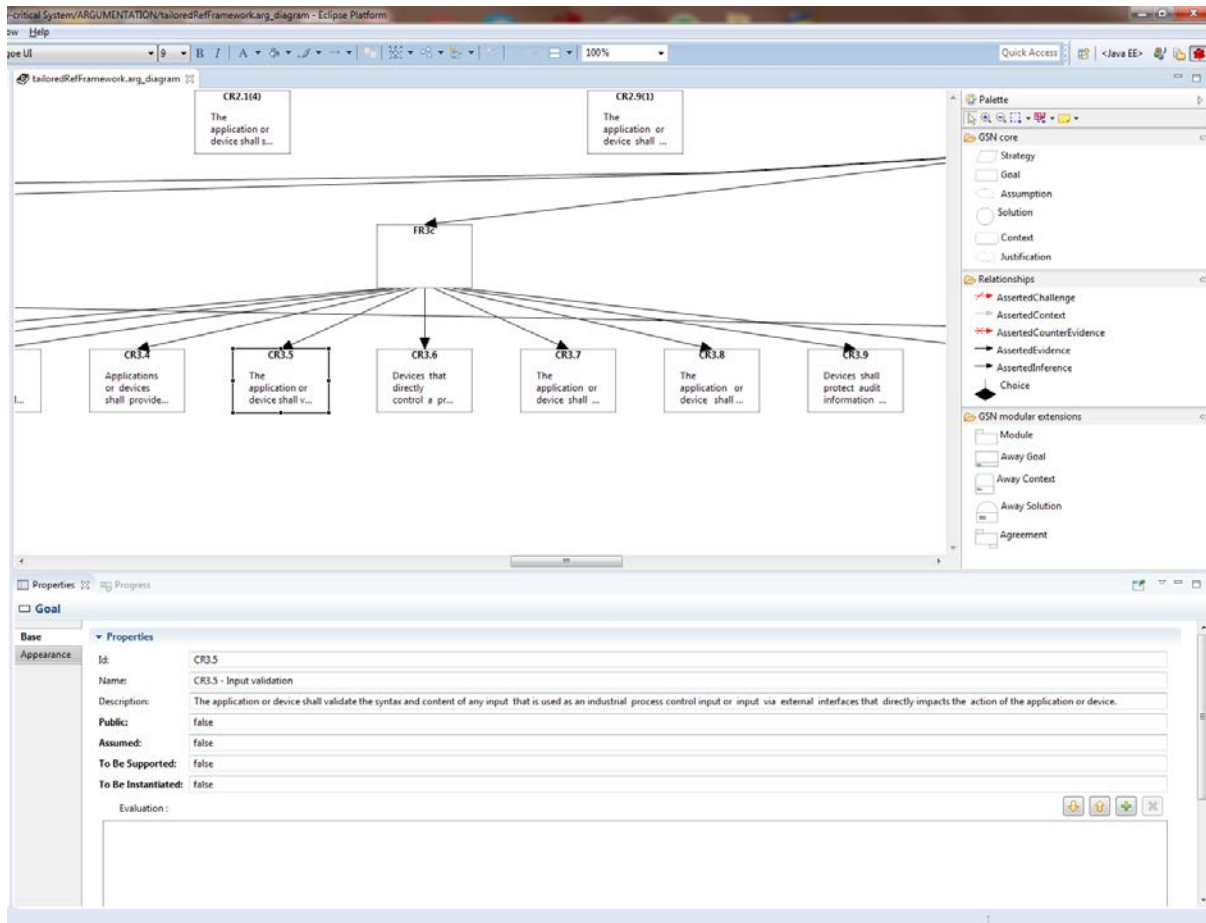
Requirements in Excel sheet can be directly imported into MORETO as SysML requirement diagram elements.

### 3.1.3.3. Safety & Security Assurance Case

**Table 8.** CS1 – Safety & Security Assurance Case

Realisation Scenario	Safety & Security Assurance Case
Scope	In the first iteration, we will choose and cover only a specific safety and security assurance goal to reduce the complexity. The assurance goal is related to standards IEC 61508, IEC 62443, or optionally IEC 62351 and IEEE 1686.
Tool Settings	MS Visio or tools developed in AMASS
Participants	<ul style="list-style-type: none"> <li>Assurance goal specification: TLV</li> <li>Assurance case development: AIT</li> <li>Evaluator: TLV, TEC</li> </ul>
Activities realised	<ol style="list-style-type: none"> <li>Agree and specify on a concrete safety &amp; security multi-concern assurance goal</li> <li>Develop assurance case according to the methodology specified in WP4</li> <li>Evaluate the results for improvement for next iteration</li> </ol>
Usage Decisions	MS Visio might be replaced by the assurance editor developed in AMASS, the result will be specified in next deliverable.
Expected Results	<ul style="list-style-type: none"> <li>A proof-of-concept multi-concern assurance case as the result of applying the multi-concern assurance concept from WP4</li> </ul>

The safety & security assurance case reflects the concept developed in WP4 on how to achieve safety & security assurance as a part of the multi-concern assurance approach. Since the concept is still under development, we show an example of the argument module of the OpenCert tool, which can be used to specify the safety & security assurance case according to the concept. In next deliverable, the detailed assurance case will be provided.

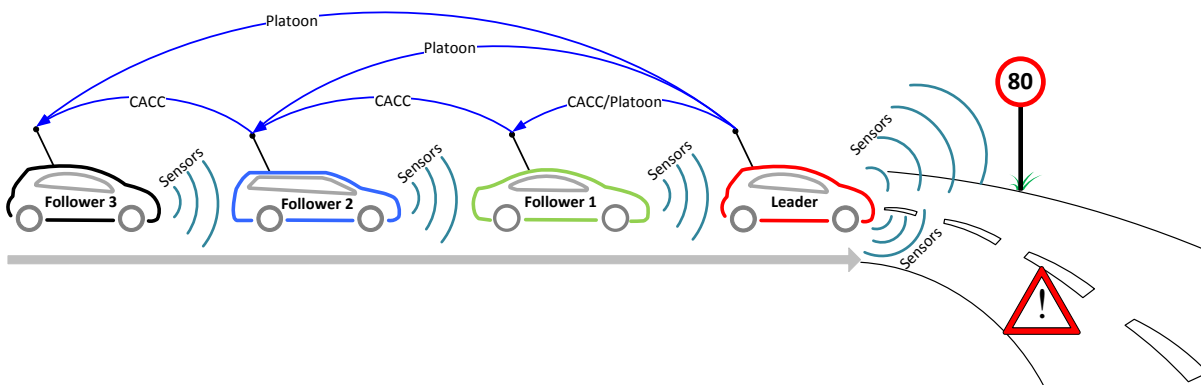


**Figure 17.** OpenCert argumentation module

## 3.2. Case Study 3: Automotive domain: Collaborative automated fleet of vehicles.

### 3.2.1. Case Study Specification

This Case Study handles with a typical example of a collaborative safety-critical system: a platoon of several vehicles. A fleet of autonomous model cars in the scale 1:8 (at the state four of them are physically available) drives and communicate together at runtime via Car2Car communication (based on peer-to-peer WIFI) to form a system-of-systems (SoS) in a controllable environment. Figure 18 shows the case study setting.



**Figure 18.** CS3 Main Scenario

Based on the initial definition of CS3 usage scenarios provided in Deliverable D1.1 [2], we selected some primary research topics and created derived usage scenarios that cover the different AMASS evaluation areas:

- US1: Safety Assessment of collaborative automated vehicle functions by model-based safety analysis and fault injection simulations.
- US2: Model-based safety and systems engineering based on contracts for a distributed system-of-systems.
- US3: Systematic creation of functional and technical safety concepts based on contracts for cooperative vehicle automation.

This first iteration covers US1.

### 3.2.2. Usage Scenario 1: Safety Assessment of collaborative automated vehicle functions by model-based safety analysis and fault injection simulations

#### 3.2.2.1. Assurance Project Creation

**Table 9.** CS3 – Assurance Project Creation

Usage Scenario	Assurance Project Creation
<b>Description</b>	The main purpose of this Usage Scenario is to create a safety assurance project valid for Automotive Domain that guarantees compliance with ISO 26262.  The following standards must be followed: <ul style="list-style-type: none"> <li>• ISO 26262</li> </ul>
<b>User</b>	B&M (owner), MDH, UC3, TRC, AIT, KMT, VIF
<b>Goal</b>	To create an assurance project that will be used to manage the project in terms of safety and compliance to ISO 26262.

<b>Activity #1</b>	<p>The main purpose of this Activity is to create a safety assurance project valid for Automotive Domain that guarantees compliance with ISO 26262.</p> <p>The following standards must be followed:</p> <ul style="list-style-type: none"> <li>• ISO 26262</li> </ul> <p><b>Create an assurance project</b></p> <ul style="list-style-type: none"> <li>• Inputs <ul style="list-style-type: none"> <li>○ ISO 26262</li> </ul> </li> <li>• Tools used <ul style="list-style-type: none"> <li>○ medini analyze</li> <li>○ SVN</li> </ul> </li> <li>• Outputs <ul style="list-style-type: none"> <li>○ Functional Safety Project Plan</li> </ul> </li> </ul>
--------------------	--

At this state, several use cases have been modelled and implemented in the controllable environment using the model cars. The main use cases are:

- Create platoon
- Join platoon
- Leave platoon
- Dissolve Platoon

Beside the normal manual driving (which means a remote controlling of the model cars), the following operation modes (resp. operative states) are implemented for the case study:

- Cruise Control Mode (CC):  
This State includes the speed-controlled driving of a model car with only odometrical sensor systems.
- Adaptive Cruise Control Mode (ACC):  
This State includes the time-gap-controlled driving of a model car behind another model car with odometrical and optical sensor systems.
- Collaborative Adaptive Cruise Control (CACC):  
This State includes the time-gap-controlled driving of a model car in front of or behind another model car with ACC-Sensor-configuration and additional WiFi-Connection between these cars.
- Platoon Mode (Platoon):  
This State includes the system-wide, time-gap-controlled driving of several model cars (more than two) in a platoon with the CACC-sensor-configuration of every car in the platoon.

### 3.2.2.2. System Component Specification

**Table 10.** CS3 – System Component Specification

Usage Scenario	System Component Specification
<b>Description</b>	<p>The main purpose of this Usage Scenario is to define the safety requirements for the item, the safety architecture. These specifications must be taken into account:</p> <ul style="list-style-type: none"> <li>• Item definition</li> <li>• Functional safety requirements</li> <li>• Technical safety requirements</li> </ul>
<b>User</b>	B&M (owner), MDH, UC3, TRC, AIT, KMT, VIF
<b>Goal</b>	To create an assurance project that will be used to manage the project in terms of safety and compliance to ISO 26262.
<b>Activity #1</b>	The main purpose of this Usage Scenario is to define the safety requirements for

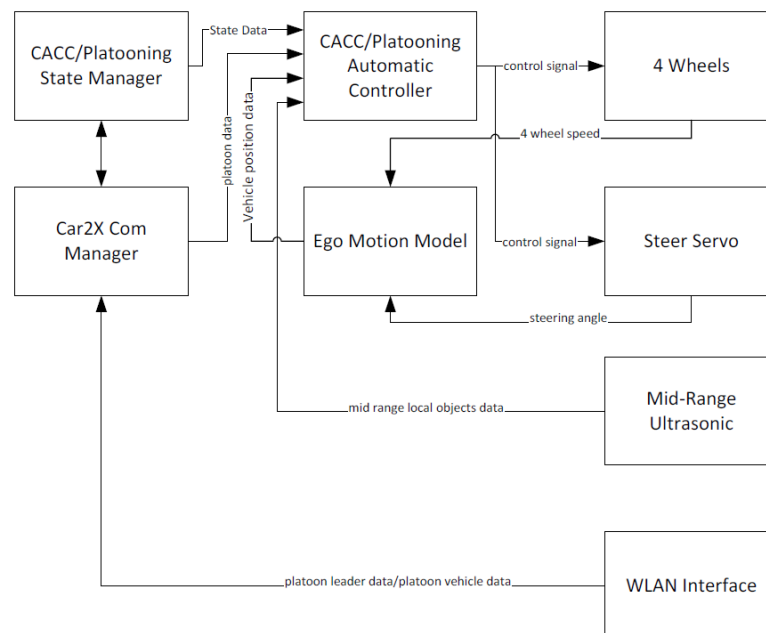
the item, the safety architecture. These specifications must taken into account:

- Item definition
- Functional safety requirements
- Technical safety requirements
- Hardware safety requirements
- Software safety requirements

#### Define safety requirements and system design

- Inputs
  - ISO 26262
  - Functional Safety Concept ACC by B&M
- Tools used
  - medini analyze
  - SVN
- Outputs
  - Part 1 of Safety Concept ACC by KMT:
    - Item definition
    - functional safety requirements
    - technical safety requirements
    - hardware safety requirements
    - software safety requirements
    - System Design
    - Hardware Architecture
    - Software Architecture

To handle the states mentioned before and the transitions between them, each of the cars implements a state-machine. The state machine is a major function block of the functional architecture, along with components like speed/distance controller, remote vehicle handler and many more. A simplified version of that architecture is shown in Figure 19.



**Figure 19.** Simplified System Architecture in the model car

This simplified Architecture includes the following modules:



- the CACC/Platooning State Manager, who keeps the state machine of the CC-, ACC-, CACC-, Platoon-States,
- the CACC/Platooning Controller with the closed-loop control for speed and time-gap,
- the Car2X Communication Manager, who is responsible for controlling the WLAN stack, keeping the list of peer vehicles and the state machine for Car2X communication establishment, as well as the communication failure diagnostics (e.g. timeout, CRC failure) and the signal evaluation of the platoon-vehicles (leader or predecessor),
- the Ego Motion Model that computes the own car trajectory from sensor data,
- the needed actuators and sensors of the model car and the WiFi-module.

When taken as a representative for actual cars on a public highway, the SoS obviously bares some safety hazard. The chosen running example is a front-rear-collision. To avoid the risk of this case, a safety concept is necessary and a final argument must be given that the system is safe. Safety Standards like the ISO 26262 cannot be chosen for the SoS, because this standard sees the overall safety responsibility with the system manufacturer – a platoon has no manufacturer, as the participating cars come from different carmakers.

The global safety goals can be set from the engineer in the same way as in traditional safety engineering, such as “Any two cars shall always maintain a front-rear-distance of at least 2 meters to each other” (scaled to the model cars). It has to be proven that the safety condition holds for each mode of operation, each use cases and for each expectable environmental situation (e.g. sudden strong braking of the leading vehicle, which can be constrained by an assumption about physically reasonable deceleration values), even in presence of failures.

For the development and safety assurance process, we follow a proceeding initially outlined in [5], which mainly comprises the following steps:

1. Develop System for the nominal (fault-free) case and show that it complies to its requirements, using SysML architectural modelling and step-by-step refinement of requirements using contracts.
2. Perform safety analysis (e.g. interface-centered FMEA, Component Fault Tree Analysis) to identify failure modes and their propagation along the architecture (where failure modes can be defined in terms of violation of the nominal contracts).
3. Enrich the system with safety mechanisms that are able to detect and mitigate all relevant failure modes and describe the improved system by (safety) contracts.

This is repeated iteratively, until all identified failure modes are appropriately covered and the residual risk is acceptable. Then, the safety assurance case can be directly derived from the decomposition of the final contracts. This can be depicted, for instance, as a GSN diagram, showing how the top-level safety requirements are step-by-step refined down to the individual safety tasks of each function block of the system (and, by allocation onto technical blocks, down to each actual technical component of the implementation, such as hardware and software components).

Example: A function block is responsible for measuring the distance to the proceeding vehicle to a specified accuracy of +/- 10 cm (to the scale of the model cars). This can be specified by a contract:

*“The reported distance equals the real distance +/- 10 cm.”*

Obviously, reporting an object more than 10 cm further away than it is actually located defines the failure mode “too high” for the sensor output “distance”. This failure can entail bad consequences, reaching to a rear crash, which is a safety goal violation (hazard). So, if this failure mode can reasonably occur (e.g. by a hardware failure in the actual sensor to which this function is allocated), a safety mechanism must be introduced to detect the failure and mitigate it. This mechanism could be an additional function block that checks the distance with for plausibility, e.g. by comparing it to a kinetic model of the other car, or to information about its position received via the radio connection. If a deviation is detected, the sensor output could be marked as invalid, and subsequent blocks would take appropriate action to cope with this situation (e.g. requesting the driver to take over control). The new (safety) contract would read as



*“The reported distance equals the real distance +/- 10 cm, or the sensor valid flag is set to FALSE.”*

In traditional embedded systems, failures are restricted to component failures inside of the system. In loosely coupled systems-of-systems, this has to be extended towards three major classes of failures:

1. Technical Failures in the local car (e.g. sensor failures, actuator failures).
2. Technical Failures in another participating car, on which some other car currently relies.
3. Failures in the communication between any cars (e.g. message loss, message delay, message corruption).

### 3.2.2.3. Evidence Management

**Table 11.** CS3 – Evidence Management

Usage Scenario	Evidence Management
<b>Description</b>	Allow the user to define the set of evidences to be provided to meet the requirements defined by ISO 26262 and customer requirements. These evidences must be then mapped to the actual artefacts collected from the different tools as evidence.
<b>User</b>	B&M (owner), MDH, UC3, TRC, AIT, KMT, VIF
<b>Goal</b>	<ul style="list-style-type: none"> <li>• Visualization of the evidences to be provided in compliance with what required by ISO 26262 and any additional reference document (e.g. company processes, regulation requirements, etc.).</li> <li>• Definition and visualization of the mappings between the evidences to be provided and the corresponding artefacts used as evidence.</li> </ul>
<b>Activity #1</b>	<p><b>Specify evidence and safety case model</b></p> <p>Allow the user to define the set of evidences to be provided to meet the requirements defined by ISO 26262 and customer requirements. These evidences must be then mapped to the actual artefacts collected from the different tools as evidence.</p> <p><b>Define safety requirements and system design</b></p> <ul style="list-style-type: none"> <li>• Inputs <ul style="list-style-type: none"> <li>○ ISO 26262</li> <li>○ Functional Safety Concept ACC by B&amp;M</li> <li>○ Part 1 of Safety Concept ACC by KMT:</li> </ul> </li> <li>• Tools used <ul style="list-style-type: none"> <li>○ medini analyze</li> <li>○ SVN</li> </ul> </li> <li>• Outputs <ul style="list-style-type: none"> <li>○ Part 2 of Safety Concept ACC by KMT: <ul style="list-style-type: none"> <li>▪ GSN Requirements Trees (including safety requirements)</li> <li>▪ Fault Tree Analysis (FTA)</li> <li>▪ Allocation between preliminary-, system-, hardware- and software architecture</li> <li>▪ Allocation of requirements</li> <li>▪ Safety Concept Report</li> </ul> </li> </ul> </li> </ul>

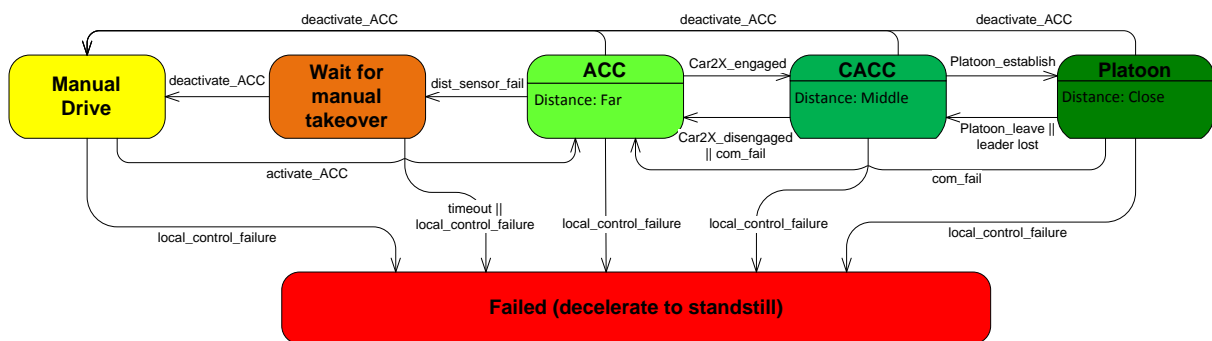
Evidence data used: The VeloxCar – a demonstrator for automated driving. Technical Report [6]

The safety case will have to show that for any combination of platoon configurations, environmental situations (listed and classified before) and failure modes (neglecting multiple failures, at first) the safety goal is not violated.

Traditional safety approaches, e.g. according ISO 26262, just consider safety goal violations due to systematic faults and random hardware failures at runtime. However, upcoming standards such as PAS 214481 recognize the fact that in automated driving systems, it has first to be shown that all safety goals are met by the system if no failure is present. This is referred to as “Safety of the intended function” (SOTIF). Our approach works for both aspects of safety, and we depict both in separate branches of the GSN graph (Figure 21).

In traditional safety engineering, the default reaction to any kind of failure is to shut down the system (or at least the affected function) completely. This is not acceptable in case of highly automated driving, not only because it significantly decreases the system availability if even minor failures lead to a standstill, but also because an uncontrolled shutdown can even constitute an hazard in its own right (imagine a steering system shutting down when the driver is unable to take over within a short time, because he is allowed to perform side tasks). We have to introduce mechanisms of smart degradation and iteratively improve our safety assurance case until it is shown that even in the degraded cases the risk of violation of any of the safety goals is sufficiently low. As the combinatory of situations tends to explode, systematic approaches are needed to check that these conditions always hold. First approaches have been published in [7]. The result of this chain of activity leads to a set of failure detection blocks (monitors) and a central governor that selects the appropriate degradation mode. This is realized as a state machine, actually as an extension of the initially presented application state machine of the CACC function (see above).

Figure 20 shows a simplified version of this state machine which is part of each particular model car. The degradation chain is color coded from green as "safe and preferred state" to yellow as "safe but not preferred state" and the critical "Wait for manual takeover" State in orange plus the Failure State in red. External events contain trigger events for operation mode transitions and failure transitions. An instance for operation mode transition is the transition from manual drive to ACC, triggered by a user intervention (activation button pressed). An Instance for failure transitions is the breakdown of a sensor. While operation mode transitions lead to a defined state, a failure leads to a decision for degradation to a lower state or the transition into the failed mode and the deceleration to standstill. As a further Instance, the breakdown of the Wifi-Connection in the CACC-mode could lead to the state transition in the ACC-mode, a Breakdown of the distance controller forces a state transition into the failed mode.



**Figure 20.** Simplified state machine



**WLANConf**

WLAN interface implements the  
allocated nominal requirements  
with sufficient confidence

**Figure 21.** CACC argument-fragment created in OpenCert

### 3.3. Case Study 4: Space domain: Design and safety assessment of on-board software applications in Space Systems

#### 3.3.1. Case Study Specification

This case study is based on the Ocean & Land Colour Instrument (OLCI) used in the satellite Sentinel-3. In particular, the study focuses on:

- The algorithm for controlling the Video Acquisition Module (VAM).
- The Focal Plane Assembly (FPA), that provides the Science Video Frames for creating the Science Report that is part of the satellite telemetry.

For a detailed description on the case study, see the Deliverable “D1.1.Case studies description and business impact” [2]. As described in the Deliverable D1.1 [2], the usage scenarios in this case study will be performed over a Model Based Design of the OEU (OLCI Electronic Unit) ICM (Instrument Control Module) SW that implements the algorithm for controlling the Video Acquisition Module (VAM) and the Focal Plane Assembly (FPA).

#### 3.3.2. Usage Scenario 1: Baseline (Common to all the Usage Scenarios)

Taking into account the aforementioned, the case study starting point is the creation of the Model Based design taking into account the space standards (ECSS) using the tools integrated in the AMASS Platform.

The efforts realized in this activity will be measured and compared with the approach followed in the development of the OLCI application software.

##### 3.3.2.1. Assurance Project Creation

**Table 12.** CS4 – Assurance Project Creation

Realisation Scenario	Assurance Project Creation	
Scope	Perform a Model Based Design of the OLCI Application Software using the AMASS Platform.	
Tool Settings	AMASS Platform	
Participants	<ul style="list-style-type: none"> <li>• System and Software Design: GMV</li> <li>• Results analysis: TASE, TEC</li> <li>• Tool support: FBK, INT, TEC, RPT</li> </ul>	
Activities realised	02/2017	AMASS First Prototype Tools installation and configuration
Usage Decisions	02/2017	Papyrus and CHESS
Expected Results	System and Software design	

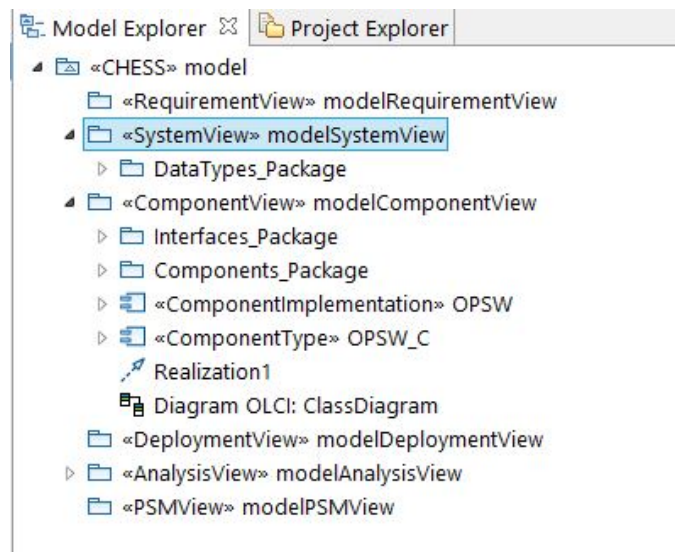
##### 3.3.2.2. Design and Dependability Assessment

**Table 13.** CS4 – System Design and Dependability Assessment

Realisation Scenario	System Design and Dependability Assessment	
Scope	Model based design of the following parts of the OLCI equipment: <ul style="list-style-type: none"> <li>• The algorithm for controlling the Video Acquisition Module (VAM).</li> <li>• The Focal Plane Assembly (FPA), which provides the Science Video Frames for creating the Science Report, which is part of the satellite telemetry.</li> </ul>	
Tool Settings	AMASS Platform	

<b>Participants</b>	<ul style="list-style-type: none"> <li>• RAMS Analysis: GMV</li> <li>• Results analysis: TASE, TEC</li> <li>• Tool support: FBK, INT, TEC, RPT</li> </ul>	
<b>Activities realised</b>	03/2017	First System Design has been created
	03/2017	The System Design has been documented in this deliverable
<b>Usage Decisions</b>	03/2017	The System Design is developed using the Papyrus/CHESS tool
<b>Expected Results</b>	OLCI model with RAMS information	

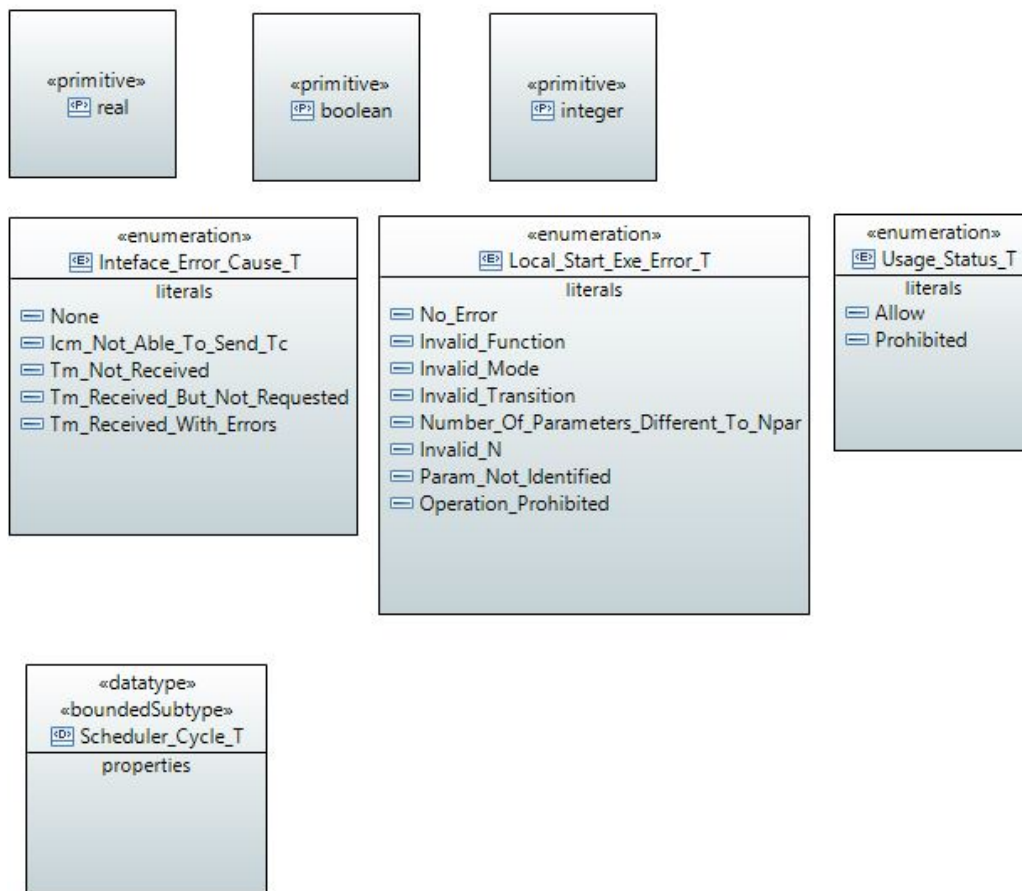
The CHESS model has been created and organized in different Packages in order to facilitate the development. Figure 22 depicts the model organization.



**Figure 22.** CHESS Model Structure

The first step while creating the model is defining the types to be used. They can be seen in Figure 23. The original source code was written using Ada language, which is a strongly typed language. It is possible to define all the characteristics the Ada types have within a CHESS model as CHESS is able to create a high range of data types (primitive, enumeration, compound, arrays, etc.).

Figure 23 shows the definition of primitive types, enumeration types and bounded type. Some of the properties are shown in the 'Properties' view and not in the diagram.

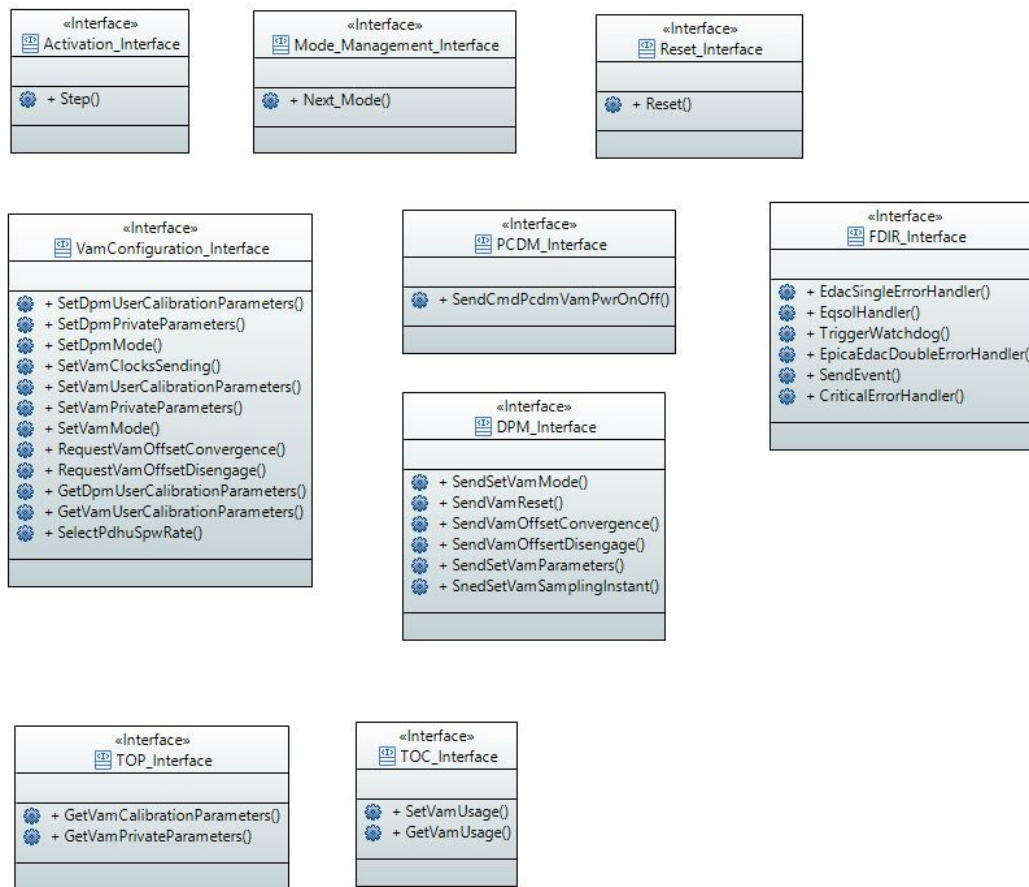


**Figure 23.** Data Types Diagram

A diagram for defining the interfaces among Components has been created. A diagram for defining the interfaces among Components has been created. It is depicted in Figure 24.

In order to complete the Components definition, the Interfaces are needed. CHESS provides the Interface Diagram for creating and designing the Interfaces, their attributes and their methods.

In a Model Based Design the more generic the interfaces are created, the better they will be completed with non-functional properties when used in Components, Component Implementation or Component Instances. It is depicted in Figure 24.

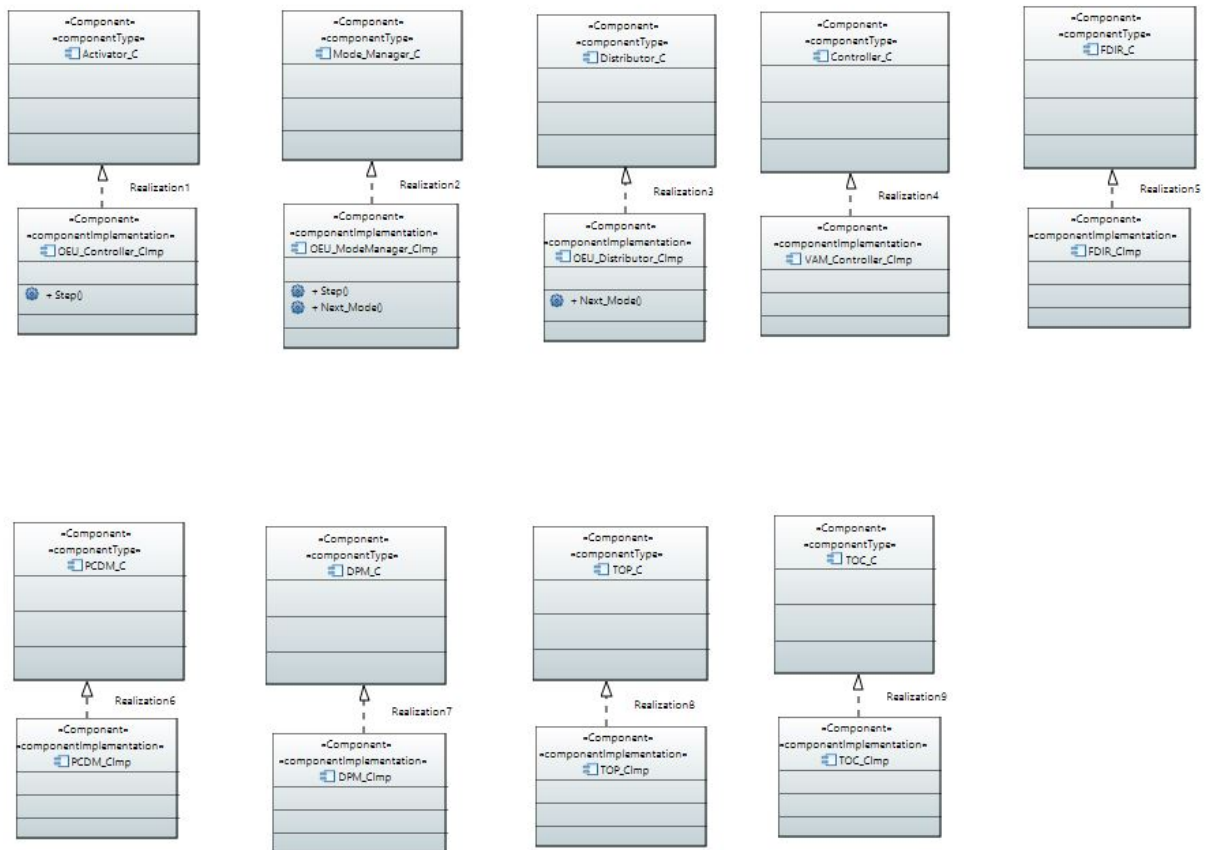


**Figure 24.** Interfaces Diagram

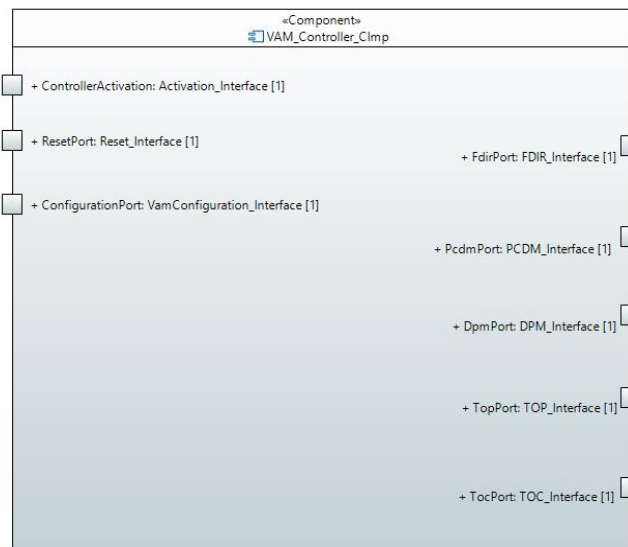
The Component Types and their Component Implementations are defined in a dedicated diagram (see Figure 25). The Interfaces, provided and required, bound to the Components and Component implementations are defined using Composite Diagrams. As an example Figure 26 depicts one of these diagrams. The Component Types and their Component Implementations are defined in a dedicated diagram (see Figure 25).

In this model a set of Components have been defined. In this case there is only one Component Implementation per Component as the system has already been developed and in use.

The Components created represent the software elements that implement the Video Acquisition Module (VAM) in the Ocean & Land Colour Instrument (OLCI), and the software elements that communicates with this element (mode management, FDIR management, telecommanding and telemetry, etc.).



**Figure 25.** Components Diagram

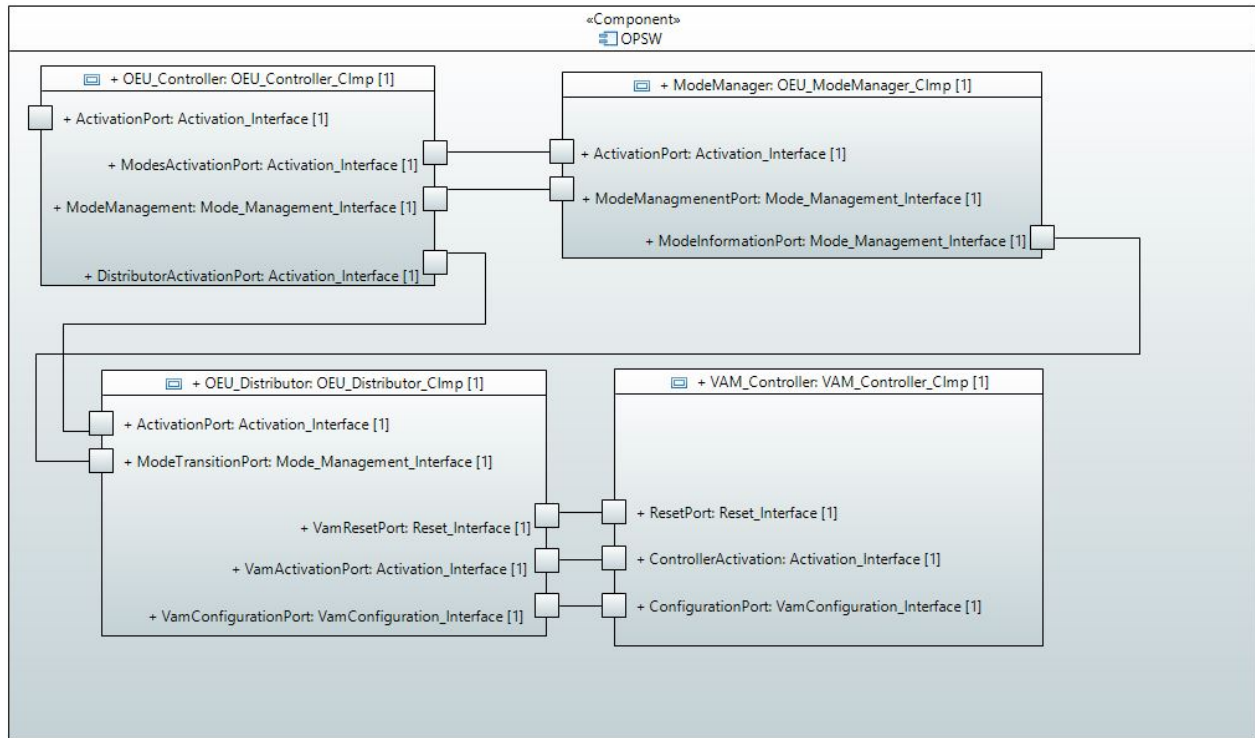


**Figure 26.** VAM\_Controller\_CImp Composite Diagram

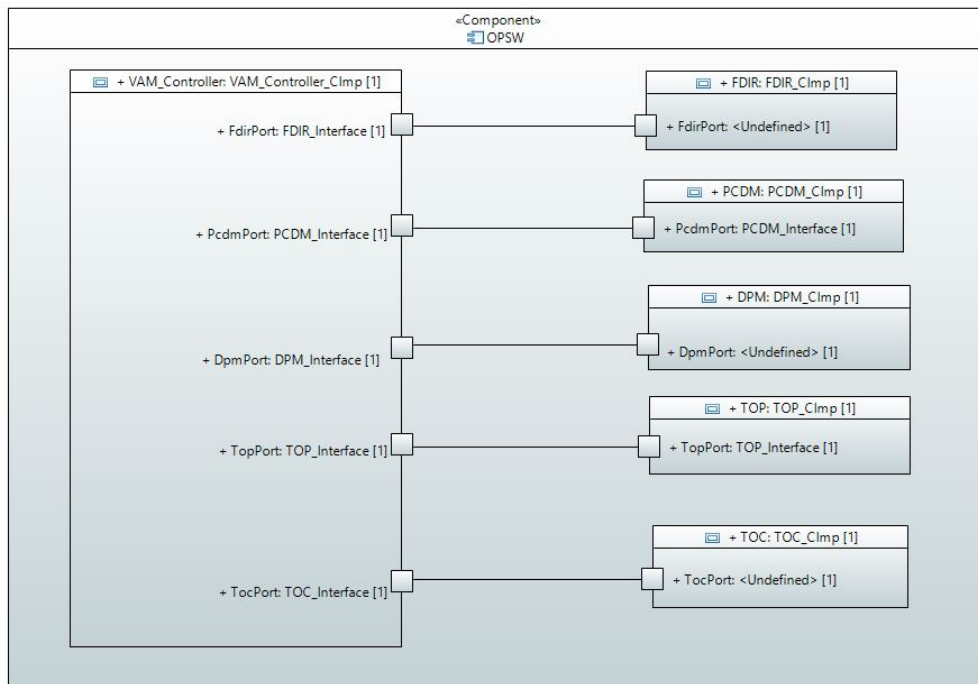
The Interfaces, provided and required, bound to the Components and Component implementations are defined using Composite Diagrams. As an example Figure 26 depicts one of these diagrams.



For creating the complete architecture of the OLCI OPSW, a new Composite Diagram is defined. In the diagram the different Instances are created and connected through the Component Interfaces. Finally, for better reading, the diagram has been divided into two diagrams (see Figure 27 and Figure 28).



**Figure 27.** First Diagram of the OLCI OPSW Architecture



**Figure 28.** Second Diagram of the OLCI OPSW Architecture

### 3.3.2.3. Evidence Management

**Table 14.** CS4 – Evidence Management

Data Usage Scenario	Evidence Management
Scope	<p>Compilation and analysis of the evidence artefacts for the following purposes:</p> <ul style="list-style-type: none"> <li>• Verification and validation (V&amp;V) processes</li> <li>• Qualification process</li> </ul> <p>The evidence management is part of the entire project life cycle, from the baseline creation to the usage scenarios</p>
Tool Settings	AMASS Platform
Participants	<ul style="list-style-type: none"> <li>• Evidence management: GMV</li> <li>• Results analysis: TASE, TEC</li> <li>• Tool support: FBK, INT, TEC, RPT</li> </ul>
Activities realized	
Usage decisions	
Expected results	<ul style="list-style-type: none"> <li>• Verification and validation report</li> <li>• Qualification process report</li> </ul>

### 3.3.2.4. Compliance Management

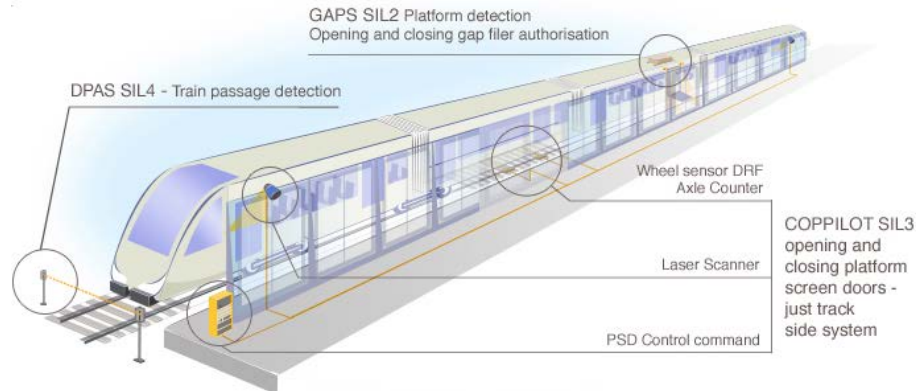
**Table 15.** CS4 – Compliance Management

Realisation Scenario	Compliance Management
Scope	Analysis of the compliance of the AMASS process to the ECSS standards applicable in the space domain
Tool Settings	AMASS Platform
Participants	<p>Results analysis: GMV, TASE, TEC</p> <p>Tool support: FBK, INT, TEC, RPT</p>
Activities realised	
Usage Decisions	
Expected Results	<ul style="list-style-type: none"> <li>• Compliance report</li> </ul>

### 3.4. Case Study 5: Railway domain: Platform Screen Doors Controller

#### 3.4.1. Case Study Specification

Automatic trains have to stop at predefined positions on metro platforms in front of so-called platform screen doors, ensuring optimal passengers transfer between train and platform while avoiding passengers to fall on tracks at peak hours.



**Figure 29.** Copilot system with its different subsystems: laser scanner, steel wheel sensor and the door control command

Such safety critical systems are often specified with a very concise requirement: “*ensure a function at a safety level of {SIL2, SIL3 or SIL4} in less than xx milliseconds*”. The systems engineering phase consists in refining this requirement into a set of functions that are distributed over an architecture that includes sensors, computers and actuators. Then the design phase and safety demonstration are performed in parallel in order to iteratively obtain a working, reliable and safe-enough system. System engineering is mainly based on human experience and expertise, Microsoft tools and sometimes on formal methods when some advanced aspects need to be managed or when trustworthy software is required. The combination of formal models of both discrete controllers and continuous physical environment helps to better analyse (some dimensions of) the system that could be animated/checked earlier.

Both hardware and software of these systems have to be in conformance with EN50126, 8 & 9 standards, including devices for fine-tuning sensors and supervision facilities. These systems have to provide safety functions that require cross-domain skills and knowledge, and dedicated/diverse engineering tooling.

#### 3.4.2. Usage Scenario 1: Generation of Frama-C asserted C code from B models

This usage scenario should demonstrate that, for the second generation of COPPILOT systems, the level of confidence of the code generation process (as well as the reuse of third party libraries) has improved, as it allows avoiding source code peer reviews, thus easing the writing of the safety case.

##### 3.4.2.1. Safety Assessment

**Table 16.** CS5 – US1: Safety Assessment

Realisation Scenario	Safety Assessment
Scope	Conformance of the generated safety critical C code with formal B models
Tool Settings	Atelier B formal IDE, Frama-C
Participants	<ul style="list-style-type: none"> <li>Leader: CLS</li> <li>CEA</li> </ul>
Activities realised	<ul style="list-style-type: none"> <li>Definition of the requirements that define the formal verification framework</li> <li>Writing of a specification document that defines the BXML file format</li> </ul>

	<p>(persisting format for B models)</p> <ul style="list-style-type: none"> <li>• Definition of the process [generation of assertions from B models] <ul style="list-style-type: none"> <li>a. Exploitation of B0 models to produce ACSL (C formal specification language) formulas</li> <li>b. Modular treatment of operations calls</li> <li>c. Exploration of solutions to deal with B0 loops</li> </ul> </li> <li>• Proof of concept on significant, non-trivial properties and source code from Stockholm PSD project.</li> </ul>
<b>Usage Decisions</b>	<p>Use of the BXML format as starting point for the ACSL formulas generation.</p> <p>Definition of target code: generation of assertions for generated code but also for third party code (extension to).</p>
<b>Expected Results</b>	<p>Conformance established automatically by formal proof in the Frama-C framework.</p>

### 3.4.3. Usage Scenario 2: Support for system-level model, including safety and security aspects

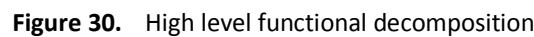
This usage scenario should demonstrate that AMASS approach is able to take into account the way our safety case is built in order to exhibit the safety of our systems. The scenario is to be played twice:

- First, on a system developed in 2009 for the Sao Paulo metro, and
- Second, for a metro in Stockholm (with a different design) for which a certificate (SIL3, Bureau Veritas) was obtained at the beginning of 2017.

#### 3.4.3.1. System Component Specification

**Table 17.** CS5 – US2: Model-Based System Component Specification

<b>Realisation Scenario</b>	<b>Model_based System component specification</b>
<b>Scope</b>	The system components will be specified including system requirements.
<b>Tool Settings</b>	AMASS Platform (Papyrus)
<b>Participants</b>	<ul style="list-style-type: none"> <li>• Leader: CLS</li> <li>• CEA</li> </ul>
<b>Activities realised</b>	<ol style="list-style-type: none"> <li>1. Analysis of the COPPILOT specifications.</li> <li>2. System modelling: context/environment, requirements, system architecture, high level functional decomposition, traceability.</li> <li>3. Refinement of architecture models: include different configurations for the architecture.</li> </ol>
<b>Usage Decisions</b>	
<b>Expected Results</b>	Architecture analysis



**Table 18. CS5 – US2: Safety Assessment**

Page 45 of 76

	4. Definition of a security analysis approach based on EBIOS
<b>Usage Decisions</b>	Definition of an <i>a priori</i> vulnerabilities list How to conduct a preliminary security analysis?
<b>Expected Results</b>	Contribution to the safety case Directions for integrating security analysis to safety case

### **3.5. Case Study 10: Space domain: Certification basis to boost the usage of MPSoC architectures in the Space Market**

#### **3.5.1. Case Study Specification**

The objective of this Case of Study is to validate the different architectures and related tools developed in AMASS. For this first iteration, feedback will be provided. For the second iteration, a benchmarking for those tools will be provided.

The case study of TAS is mainly focalized in including multicore architectures capable of in-flight reconfiguration in actual payload data processing equipment. The target is to replace legacy designs in actual flight missions using multicore improved performances to overcome the limitations imposed by classic ASIC designs. This implies two Usage Scenarios:

- US1: Scalable Sensor Data Processor Breadboard (SSDP)
- US2: Reconfigurable FPGAs

Usage Scenarios one and two follows the same Data Collection structure described in [3]. US1 is more focused in multicore systems and US2 is mainly focused in the validation of reconfigurable in-flight changes is the FPGA design. The main target is keep as compliant with the standards the parts of the project that have not been modified, when other individual block or classes that do not interact with the modified ones. As summary: validate and certificate the whole solution by validating only the parts that have changed.

A technical description of this case study can be found in the deliverable “Case studies description and business impact” [2].

Based on the definition of usage scenarios provided in D1.1 [2] and the data stored in D1.2 [3], this case study provides feedback to AMASS tools by testing the tools and giving advices to tool developers.

#### **3.5.2. Usage Scenarios US1 and US3**

TAS-E usage scenarios follow the same standards and the same high level design. The differences between them are inside the requirements and specifications. For this first iteration a valid model is done that applies to both USs.

##### **Usage Scenario 1: Scalable Sensor Data Processor Breadboard (SSDP)**

SSDP is an architecture developed to satisfy the needs of the applications that request the fast processing of a high amount of data for smart sensors, to be used in space exploration missions. This architecture combines fixed point DSP IP with a LEON controller. The inherent scalability of the Network-on-chip (NoC) architecture, as well as the efficient combination of GPP and DSP processor cores are very interesting for future large and ultra-powerful processor ASICs, however, a strict validation and certification strategy will be key to allow the widespread usage of such a powerful device in different scenarios with very different criticality constraints.

Multicore programming is still not approved for in-flight missions due the hard requirements to validate and certificate in actual payload data processing equipment. For SSDP one LEON core and also contains 2 programmable processing cores based on Xentium® DSP cores.

##### **Usage Scenario 2: Reconfigurable FPGAs**

The telecommunication broadband regenerative payloads and its associated platforms and the Earth Observation payloads, need to be adaptable while in-flight missions. Every piece of software must be completely proven and validated before taking off, after that moment no change is allowed to continue the mission. Reconfigurable FPGAs with self-healing capabilities are not allowed to operate completely in space



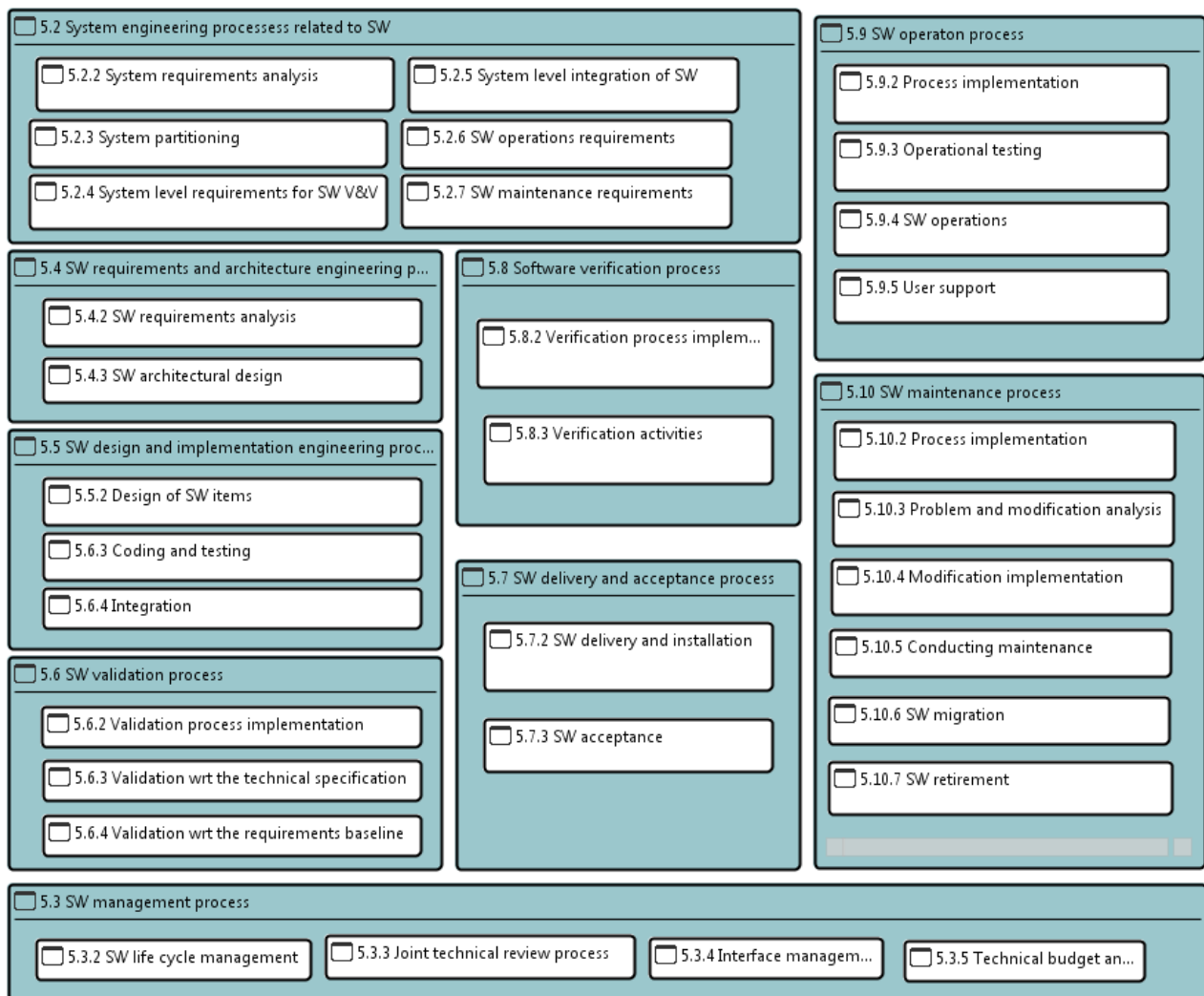
missions. AMASS will be the platform which should guarantee that every change will be compliant with the standards and all the rigid rules imposed by the ESA or other international agencies.

### 3.5.2.1. Standard Model Creation

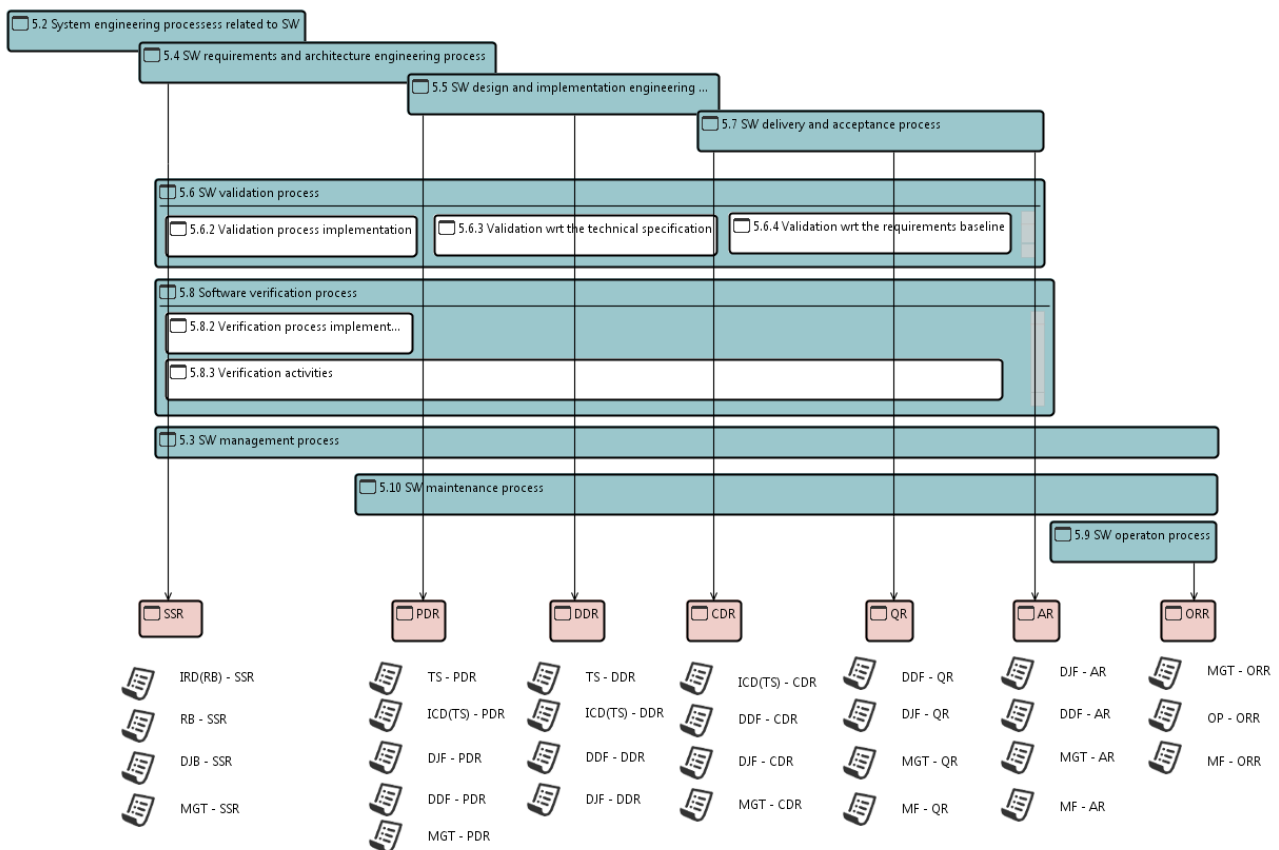
This conjunct of standards modelling creation applies to both usage scenarios depicted in this epigraph. Both Usage Scenarios must be compliant with both standards, and the same modelling process is valid for them.

**Table 19.** CS10 – Standard Model Creation

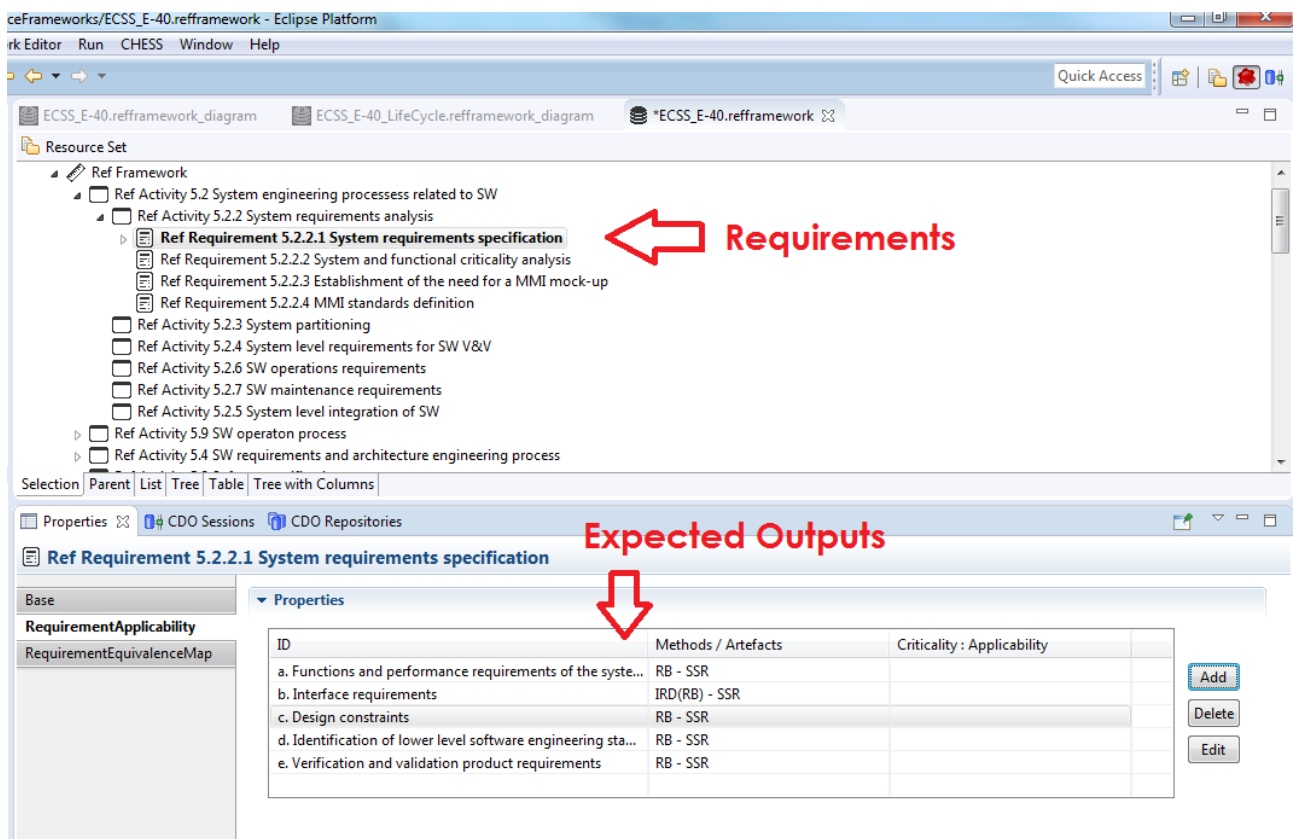
Realisation Scenario	Standards Models Creation
<b>Scope</b>	Modelling of some meaningful excerpts of ECSS_E .We will target the following sections from these standards. <ul style="list-style-type: none"> <li>• ECSS_E-40</li> </ul>
<b>Tool Settings</b>	OpenCert Tools: Standards Editor
<b>Participants</b>	<ul style="list-style-type: none"> <li>• Data Analysis: TAS</li> <li>• Tool User: TAS, TEC</li> <li>• Results Analysis: TAS, TEC</li> </ul>
<b>Activities realised</b>	<ol style="list-style-type: none"> <li>1. Conceptually analyse the structure of ECSS-E-40 and ECSS-Q-80 as well as development phases, activities, artefacts, requirements, and criticality regions. The objective is to materialize these concepts to a Reference Framework in OpenCert.</li> <li>2. Create a Reference Framework diagram, focused on this prototype. ECSS-E-40 standard modelled in OpenCert is shown in Figure 32, Figure 33 and Figure 34.</li> <li>3. Have a valid standard model by the evaluation of framework models with contributors whose field of expertise is the space domain.</li> <li>4. Identify equivalence relationships between assets (artefacts, activities, requirements) from both standards.</li> </ol>
<b>Usage Decisions</b>	We will not use EPF for modelling the targeted Standards (reference frameworks). Hence, the Standards models are created from scratch in OpenCert.
<b>Expected Results</b>	<ul style="list-style-type: none"> <li>• Reference Framework model for ECSS-E-40</li> <li>• Equivalence Mapping model ECSS-E-40</li> </ul>



**Figure 32.** ECSS-E-40 Standard Structure in OpenCert for CS10



**Figure 33.** ECSS-E-40 Standard Life Cycle in OpenCert for CS10



**Requirements**

**Expected Outputs**

ID	Methods / Artefacts	Criticality : Applicability
a. Functions and performance requirements of the syste...	RB - SSR	
b. Interface requirements	IRD(RB) - SSR	
c. Design constraints	RB - SSR	
d. Identification of lower level software engineering sta...	RB - SSR	
e. Verification and validation product requirements	RB - SSR	

Buttons: Add, Delete, Edit

**Figure 34.** ECSS-E-40 Standard Requirements and Outputs in OpenCert for CS10

### 3.5.2.2. Assurance Project Creation

**Table 20.** CS10 – Assurance Project Creation

Realisation Scenario	Assurance Project Creation
<b>Scope</b>	Creation of an assurance [define phases of interest] project providing an independent oversight of projects that will incorporate multicore processors. The neediness of providing confidence for project or programme stakeholders, high capital or high risk projects. Within ECSS-Q-ST-10 (Product assurance management) section standards.
<b>Tool Settings</b>	OpenCert as the tool for compliance management
<b>Participants</b>	<ul style="list-style-type: none"> <li>• System and Software Design: TAS-E</li> <li>• Results analysis: TASE, GMV</li> <li>• Tool Support: TEC</li> </ul>
<b>Activities realised</b>	<p>The following list of activities will be executed (Figure 35):</p> <ul style="list-style-type: none"> <li>• Establish assurance process development for SW, covering the following functional blocks: <ul style="list-style-type: none"> <li>○ Capture Information from ECSS-Q- ST-80C</li> <li>○ Create Space Project for Specific Profile</li> <li>○ Design Reference Framework Model for the project</li> <li>○ Establish Team-Member Roles for Space Project</li> <li>○ Configure the different levels of access for the created assurance project subsections</li> </ul> </li> </ul>
<b>Usage Decisions</b>	Equivalence mappings for overlapping/equivalences of requirements
<b>Expected Results</b>	<p>Control of the project status along its development.</p> <p>Guarantee the product development follows ECSS standards.</p> <p>The possibility of replacing several tools for only one, which will make the team work with more synergy between members. The efficiency increases in the developing software process since the kick-off, reducing time and costs.</p>



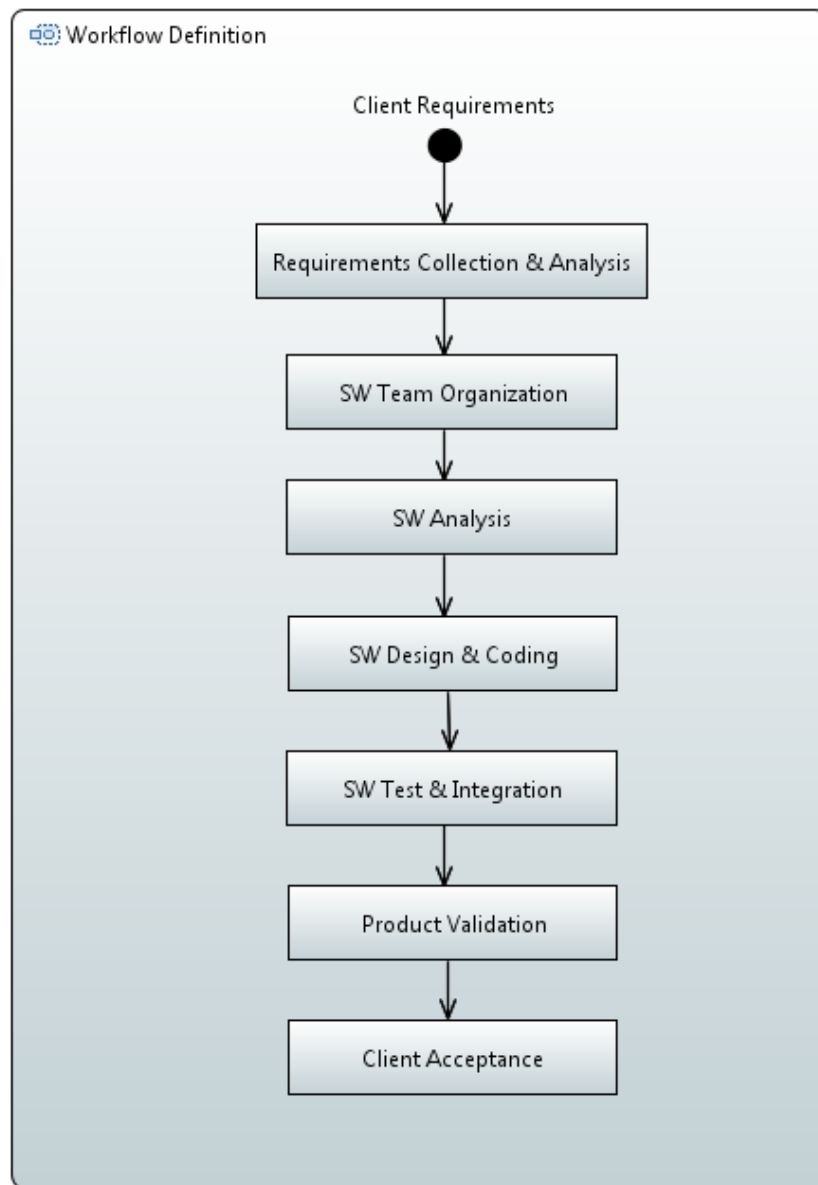
### 3.5.2.3. System Design and Dependability Assessment

As part of the Assurance Project creation, dependability Assessment requires long-term vision from dependencies derived from inside the project and from external agents.

Those dependencies can vary the design steps, the architecture, the requirement, etc. The client must define clearly the requirements, but it is mandatory to inform if the dependencies changes the deliverable periods, the economic terms or the designing patterns.

**Table 21.** CS10 – System Design and Dependability Assessment

Realisation Scenario	System Design and Dependability Assessment
Scope	Configure a whole system description which accomplishes the different needs and subsections/sprints accorded with the client. Evidence the risks and design plans for solving in case of occurrence. Configure proceeding methods or debug responsibilities in case of system dependability changes: change of requisites by the clients, study if non predictable changes affect and how much them affect to the architecture, if the time schedule changes or if extension is required.
Tool Settings	Papyrus / OperCert from scratch
Participants	<ul style="list-style-type: none"> <li>• System and Software Design: TAS-E</li> <li>• Results analysis: TASE, GMV</li> <li>• Tool Support: TEC</li> </ul>
Activities realised	<ol style="list-style-type: none"> <li>1. Client Requirements specification document</li> <li>2. Requirements collection &amp; analysis</li> <li>3. SW team organization</li> <li>4. SW analysis</li> <li>5. SW design &amp; coding</li> <li>6. SW test &amp; integration</li> <li>7. Product validation</li> <li>8. Client acceptance</li> </ol>
Usage Decisions	None
Expected Results	<ul style="list-style-type: none"> <li>• Activity flows and work flow relationship between the elements of the tool chain. Obtain, in case of potential risk, debugged activities and roles to lead the actions to be done.</li> <li>• Definition and visualization of the mappings between the evidences to be provided and the corresponding artefacts used as evidence.</li> <li>• Visualization of the evidences to be provided in compliance with what required by the applicable standards and any additional reference document (e.g. company processes, regulation requirements, etc.).</li> </ul>



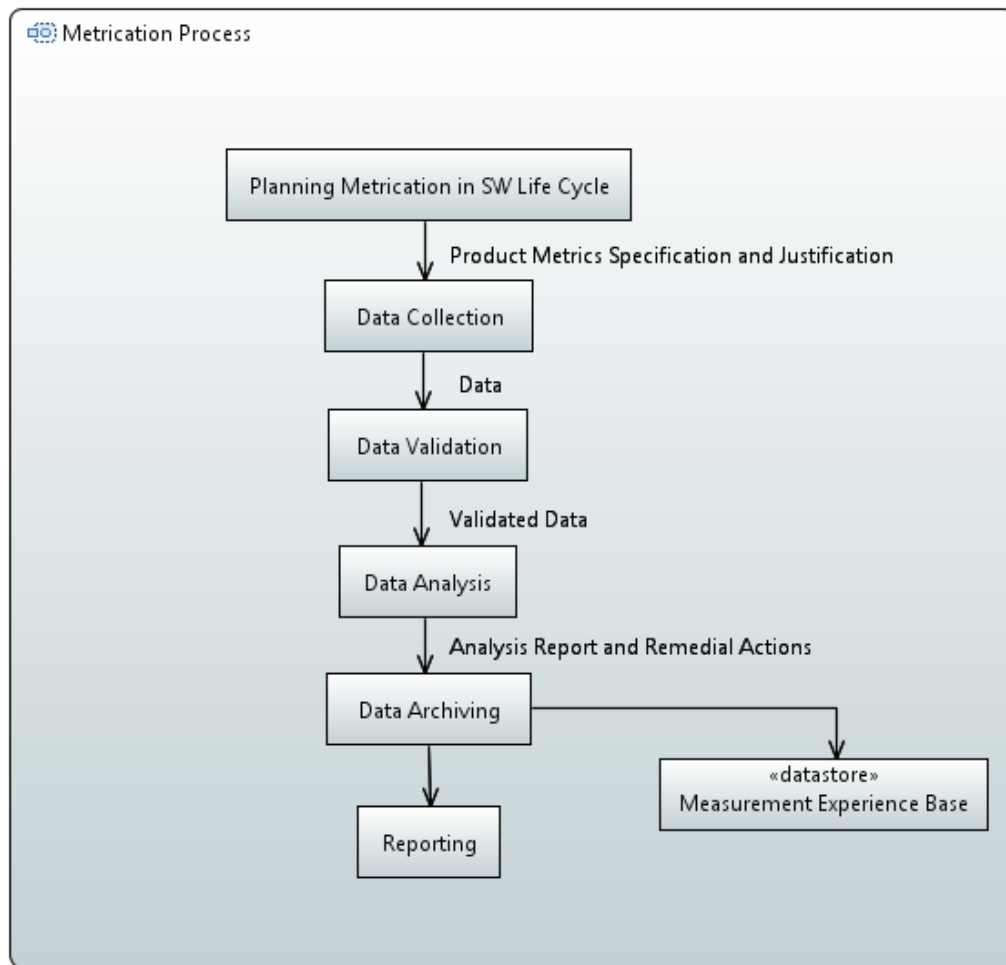
**Figure 36.** System Design & Dependability Assessment in CS10



### 3.5.2.4. Evidence Management

**Table 22.** CS10 – Evidence Management

Realisation Scenario	Evidence Management
<b>Scope</b>	The AMASS platform will allow the user to define the set of evidences to be provided to meet the requirements defined by the applicable standards and customer requirements. These evidences must be then mapped and be presented to be corroborated or give a prove to the client that everything works as expected.
<b>Tool Settings</b>	Papyrus
<b>Participants</b>	<ul style="list-style-type: none"> <li>• System and Software Design: TAS-E</li> <li>• Results analysis: TASE, GMV</li> <li>• Tool Support: TEC</li> </ul>
<b>Activities realised</b>	<ol style="list-style-type: none"> <li>1. Create artefact model for Space Domain</li> <li>2. Capture evidence documents into the SVN repository</li> <li>3. Specify characteristics of Space Domain artefacts</li> <li>4. Use cross-domain functionality to reuse Artefact models from other Space Domain projects</li> <li>5. Data collection</li> <li>6. Data analysis</li> <li>7. Data storage</li> <li>8. Prepare a set of evidences of different subsections and for the complete project</li> <li>9. Reporting</li> </ol>
<b>Usage Decisions</b>	None
<b>Expected Results</b>	<ul style="list-style-type: none"> <li>• Evidence model and artefact repository for Space Domain</li> <li>• Evidence reporting model</li> </ul>



**Figure 37.** Evidence Management / Metrication CS10

### 3.5.2.5. Compliance Management

**Table 23.** CS10 – Compliance Management

Realisation Scenario	Compliance Management
Scope	<p>Obtain a document where the requisites are shaped, indicating which ones are accomplished completely, and which ones only partially or directly non-compliant. It is also explained the reason if there are non-conformities, and why some requisites are not fully achieved.</p> <p>It is recommendable to establish two different modes of working:</p> <ul style="list-style-type: none"> <li>• Nominal Mode: the way it works in normal conditions. In terms of safety the code should be marked with CE certification, and also must comply the following standards: <ul style="list-style-type: none"> <li>◦ If it will be used in the USA, it must be market with UL certification as well.</li> </ul> </li> <li>• Failure Mode (FMEA): Used to test elements that might be potentially dangerous, including hardware hazards caused by malfunction of the software. The following standards must be followed: <ul style="list-style-type: none"> <li>◦ ECSS-Q-ST: Failure modes, effects (and criticality) analysis (FMEA/FMECA).</li> </ul> </li> </ul>
Tool Settings	Papyrus

<b>Participants</b>	<ul style="list-style-type: none"> <li>• System and Software Design: TAS-E</li> <li>• Results analysis: TASE, GMV</li> <li>• Tool Support: TEC</li> </ul>
<b>Activities realised</b>	<ol style="list-style-type: none"> <li>1. Establish matrix of compliance in nominal mode</li> <li>2. Establish matrix of compliance in failure mode</li> <li>3. Evaluate the results</li> <li>4. Create a report for compliance management for ECSS-E</li> </ol>
<b>Usage Decisions</b>	None
<b>Expected Results</b>	Cross-reference table that indicates proposal evaluators where they can find responses to specific Request for Proposal (RFP) requirements.

### 3.6. Case Study 11: Space domain: Design and efficiency assessment of model based Attitude and Orbit Control software

#### 3.6.1. Case Study Specification

The attitude and orbit control subsystem (AOCS) is used for a number of different telecommunication satellite platforms. Attitude control is controlling the orientation of the satellite with respect to an inertial frame of reference or other entity. Orbit control is controlling the positioning of the satellite in orbit. Controlling the attitude and orbit requires sensors to measure the satellite orientation, actuators to apply the torques needed to re-orient the satellite to desired attitude and/or orbit and algorithms to command the actuators based on sensor measurements and specification of desired attitude and/or orbit.

The development of critical on-board software applications such as AOCS is continuously becoming more complex as space missions become more autonomous. At the same time, it is expected that the pressure on budget and schedule will continue to increase such that the demand for efficient software development still ensuring dependability will increase.

In European space projects, the development of any SW must be fully compliant to at least the following ECSS standards:

- ECSS-E-ST-40C Software general requirements
- ECSS-Q-ST-80C Software product assurance

There are a number of additional standards for management processes:

- ECSS-M-ST-10C\_Rev.1 Project planning and implementation (6March2009)
- ECSS-M-ST-40C\_Rev.1 Configuration and information management (6March2009)
- ECSS-M-ST-60C Cost & schedule management (31July2008)
- ECSS-M-ST-80C Risk management (31July2008)

The ECSS also addresses dependability and safety processes on system and software level:

- ECSS-Q-ST-30C Dependability (6March2009)
- ECSS-Q-ST-40C Safety (6March2009)

This case study has multiple goals. In this document, efficient compliance management is in focus. More specifically, this case study considers a limited portion of ECSS-E-ST-40C in order to perform an initial evaluation.

#### 3.6.2. Usage Scenario 1 - Managing compliance with ECSS-E-ST-40C

##### 3.6.2.1. Standards model creation

This usage scenario is related to process assurance, i.e. to ensure that the AOCS-related SW development process follows a given set of recommendations from the targeted standard.

**Table 24.** CS11 – Standards Model Creation

Realisation Scenario	Standards Models Creation
<b>Scope</b>	Modelling of some meaningful excerpts of ECSS-E-ST-40C. We target the following sections from these standards. <ul style="list-style-type: none"> <li>• ECSS-E-ST-40C, Section 5</li> </ul>
<b>Tool Settings</b>	EPF Composer +: OHB-transformer for importing requirements automatically
<b>Participants</b>	<ul style="list-style-type: none"> <li>• Data Analysis: OHB</li> <li>• Tool User: OHB, MDH</li> <li>• Results Analysis: OHB, MDH</li> </ul>
<b>Activities realised</b>	Conceptually analyse the structure of ECSS-E-ST-40C as well as the core concepts

such as: phases, activities, artefacts, requirements and criticality levels. The goal is to map these concepts described above to EPF-Composer method content, as well as mapping custom practice concepts.

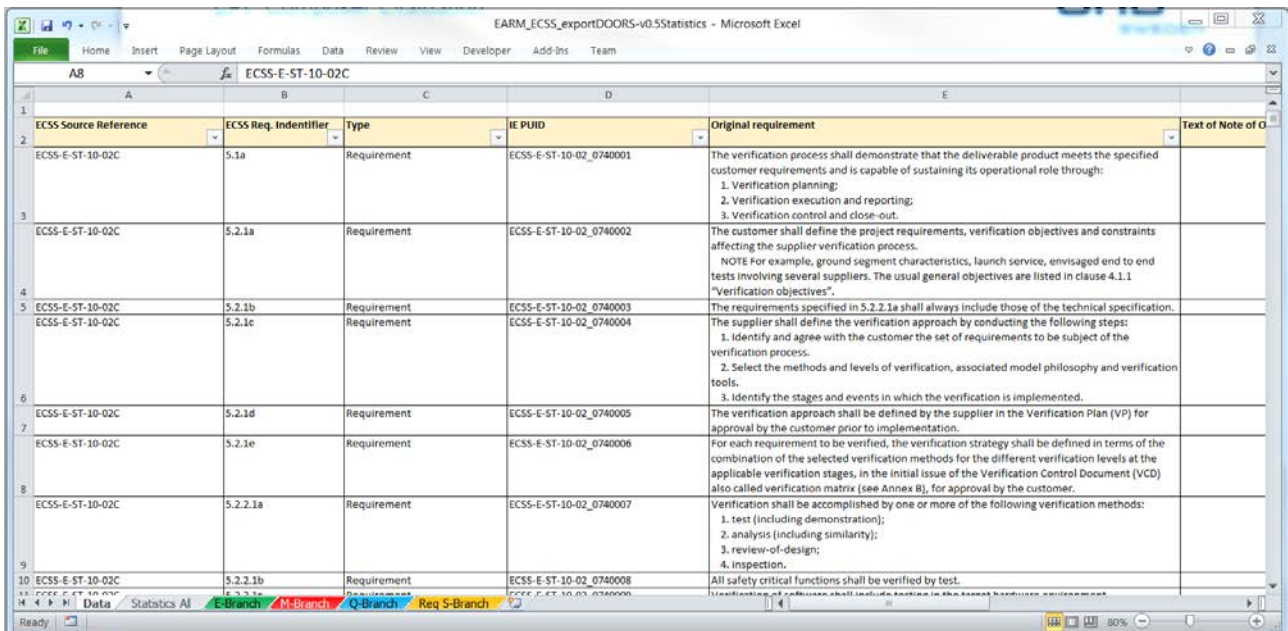
To automatically import the standard requirements into the tool (EPF Composer), the following steps have been taken:

- From EARM\_ECSS\_exportDOORS-v0.5Statistics.xlsx export all requirements related to ECSS-E-ST-40C.
- Add columns 'Processes' and 'Notes'.
- Fill in the column 'Processes' with the name of the process that will ensure compliance with respective requirement. (Fill in the column 'Notes' if extra information is required).
- Export the complete table from excel to xml. (Follow instructions on <https://support.office.com/en-us/article/Create-an-XML-data-file-and-XML-schema-file-from-worksheet-data-e35400d4-0e10-4669-9a50-59a8c57d677e>)
- Create xslt-schema (XSLInputFile-ECSS-E-ST-40C.xsl).
- Transform the exported xml-format to a format compatible with EPF Composer using Free online XSL Transformer (<http://www.freeformatter.com/xsl-transformer.html>) and XSLInputFile-ECSS-E-ST-40C.xsl
- Import generated xml-file into EPF Composer.

#### Usage Decisions

#### Expected Results

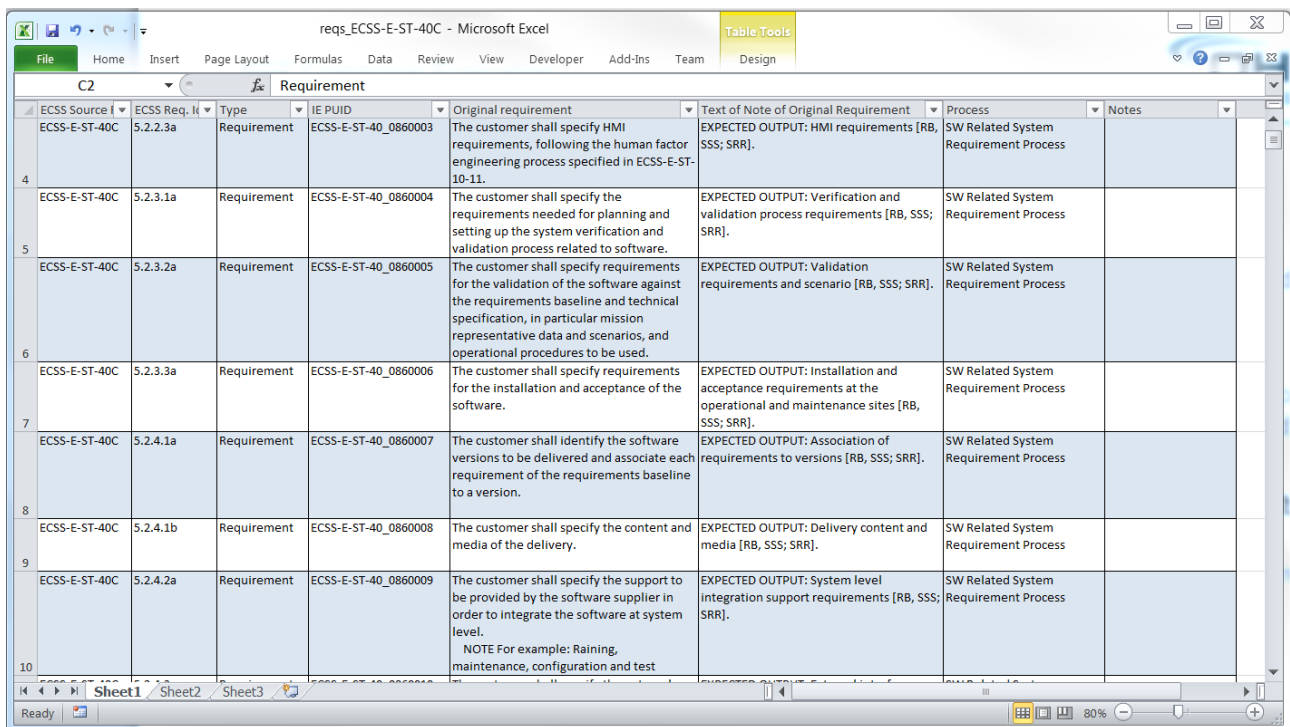
- EPF-Composer –based model of standards



ECSS Source Reference	ECSS Req. Identifier	Type	IE PUID	Original requirement	Text of Note of O
ECSS-E-ST-10-02C	5.1a	Requirement	ECSS-E-ST-10-02_0740001	The verification process shall demonstrate that the deliverable product meets the specified customer requirements and is capable of sustaining its operational role through: 1. Verification planning; 2. Verification execution and reporting; 3. Verification control and close-out.	
ECSS-E-ST-10-02C	5.2.1a	Requirement	ECSS-E-ST-10-02_0740002	The customer shall define the project requirements, verification objectives and constraints affecting the supplier verification process. NOTE For example, ground segment characteristics, launch service, envisaged end to end tests involving several suppliers. The usual general objectives are listed in clause 4.1.1 "Verification objectives".	
ECSS-E-ST-10-02C	5.2.1b	Requirement	ECSS-E-ST-10-02_0740003	The requirements specified in 5.2.2.1a shall always include those of the technical specification.	
ECSS-E-ST-10-02C	5.2.1c	Requirement	ECSS-E-ST-10-02_0740004	The supplier shall define the verification approach by conducting the following steps: 1. Identify and agree with the customer the set of requirements to be subject of the verification process. 2. Select the methods and levels of verification, associated model philosophy and verification tools. 3. Identify the stages and events in which the verification is implemented.	
ECSS-E-ST-10-02C	5.2.1d	Requirement	ECSS-E-ST-10-02_0740005	The verification approach shall be defined by the supplier in the Verification Plan (VP) for approval by the customer prior to implementation.	
ECSS-E-ST-10-02C	5.2.1e	Requirement	ECSS-E-ST-10-02_0740006	For each requirement to be verified, the verification strategy shall be defined in terms of the combination of the selected verification methods for the different verification levels at the applicable verification stages, in the initial issue of the Verification Control Document (VCD) also called verification matrix (see Annex B), for approval by the customer.	
ECSS-E-ST-10-02C	5.2.2.1a	Requirement	ECSS-E-ST-10-02_0740007	Verification shall be accomplished by one or more of the following verification methods: 1. test (including demonstration); 2. analysis (including similarity); 3. review-of-design; 4. inspection.	
ECSS-E-ST-10-02C	5.2.2.1b	Requirement	ECSS-E-ST-10-02_0740008	All safety critical functions shall be verified by test.	

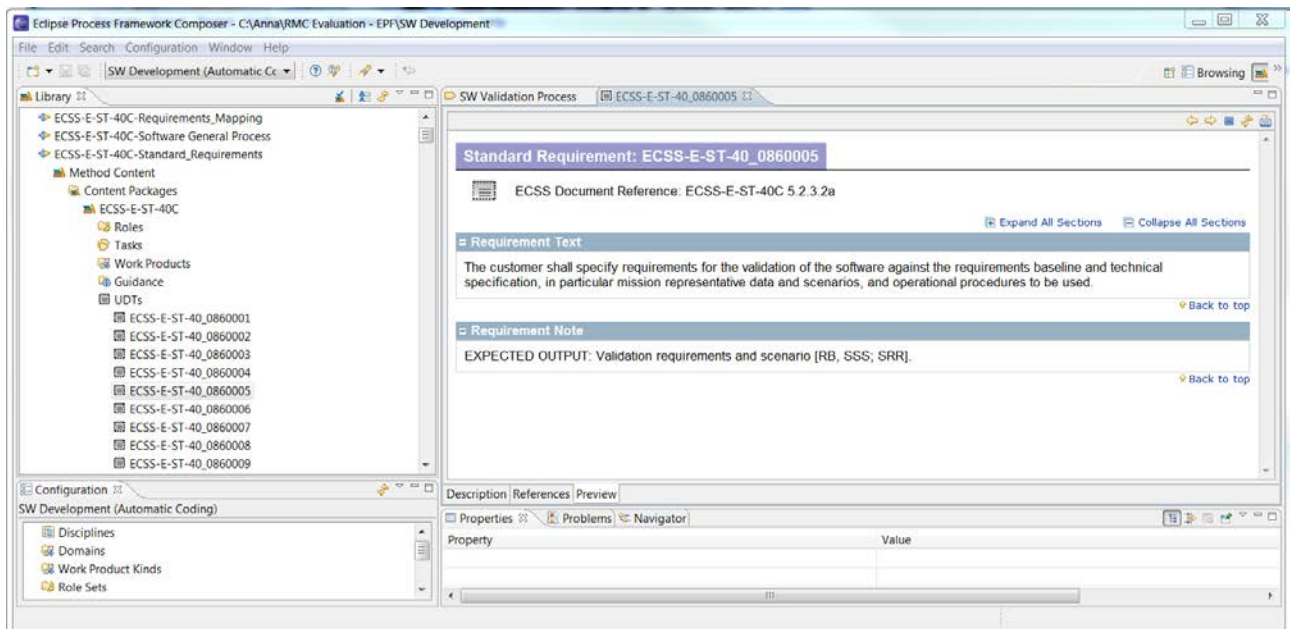
**Figure 38.** Parsed requirements

EARM\_ECSS\_exportDOORS-v0.5Statistics.xlsx has been parsed and the set of requirements has been filtered.



ECSS Source	ECSS Req. ID	Type	IE PUID	Original requirement	Text of Note of Original Requirement	Process	Notes
ECSS-E-ST-40C	5.2.2.3a	Requirement	ECSS-E-ST-40_0860003	The customer shall specify HMI requirements, following the human factor engineering process specified in ECSS-E-ST-10-11.	EXPECTED OUTPUT: HMI requirements [RB, SSS; SRR].	SW Related System Requirement Process	
ECSS-E-ST-40C	5.2.3.1a	Requirement	ECSS-E-ST-40_0860004	The customer shall specify the requirements needed for planning and setting up the system verification and validation process related to software.	EXPECTED OUTPUT: Verification and validation process requirements [RB, SSS; SRR].	SW Related System Requirement Process	
ECSS-E-ST-40C	5.2.3.2a	Requirement	ECSS-E-ST-40_0860005	The customer shall specify requirements for the validation of the software against the requirements baseline and technical specification, in particular mission representative data and scenarios, and operational procedures to be used.	EXPECTED OUTPUT: Validation requirements and scenario [RB, SSS; SRR].	SW Related System Requirement Process	
ECSS-E-ST-40C	5.2.3.3a	Requirement	ECSS-E-ST-40_0860006	The customer shall specify requirements for the installation and acceptance of the software.	EXPECTED OUTPUT: Installation and acceptance requirements at the operational and maintenance sites [RB, SSS; SRR].	SW Related System Requirement Process	
ECSS-E-ST-40C	5.2.4.1a	Requirement	ECSS-E-ST-40_0860007	The customer shall identify the software versions to be delivered and associate each requirement of the requirements baseline to a version.	EXPECTED OUTPUT: Association of requirements to versions [RB, SSS; SRR].	SW Related System Requirement Process	
ECSS-E-ST-40C	5.2.4.1b	Requirement	ECSS-E-ST-40_0860008	The customer shall specify the content and media of the delivery.	EXPECTED OUTPUT: Delivery content and media [RB, SSS; SRR].	SW Related System Requirement Process	
ECSS-E-ST-40C	5.2.4.2a	Requirement	ECSS-E-ST-40_0860009	The customer shall specify the support to be provided by the software supplier in order to integrate the software at system level. NOTE For example: Raining, maintenance, configuration and test	EXPECTED OUTPUT: System level integration support requirements [RB, SSS; SRR].	SW Related System Requirement Process	

**Figure 39.** Excerpt of the extended EARM used for the semi-automatic import of ECSS requirements



**Eclipse Process Framework Composer - C:\Anna\RMC Evaluation - EPF\SW Development**

File Edit Search Configuration Window Help

Library

- ECSS-E-ST-40C-Requirements\_Mapping
- ECSS-E-ST-40C-Software General Process
- ECSS-E-ST-40C-Standard\_Requirements
  - Method Content
    - Content Packages
      - ECSS-E-ST-40C
        - Roles
        - Tasks
        - Work Products
        - Guidance
        - UDTs
          - ECSS-E-ST-40\_0860001
          - ECSS-E-ST-40\_0860002
          - ECSS-E-ST-40\_0860003
          - ECSS-E-ST-40\_0860004
          - ECSS-E-ST-40\_0860005
          - ECSS-E-ST-40\_0860006
          - ECSS-E-ST-40\_0860007
          - ECSS-E-ST-40\_0860008
          - ECSS-E-ST-40\_0860009

Configuration

SW Development (Automatic Coding)

- Disciplines
- Domains
- Work Product Kinds
- Role Sets

SW Validation Process

ECSS-E-ST-40\_0860005

**Standard Requirement: ECSS-E-ST-40\_0860005**

ECSS Document Reference: ECSS-E-ST-40C 5.2.3.2a

Expand All Sections Collapse All Sections

**Requirement Text**

The customer shall specify requirements for the validation of the software against the requirements baseline and technical specification, in particular mission representative data and scenarios, and operational procedures to be used.

Back to top

**Requirement Note**

EXPECTED OUTPUT: Validation requirements and scenario [RB, SSS; SRR].

Back to top

Description References Preview

Properties Problems Navigator

Property	Value

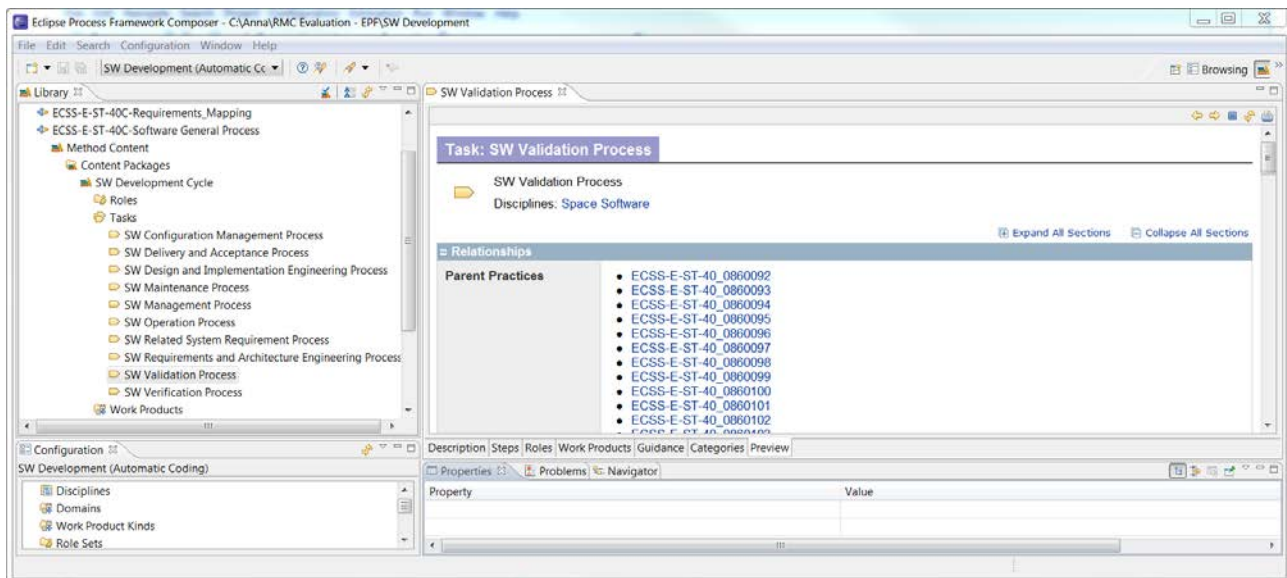
**Figure 40.** Method composer requirements ECSS-E-ST-40

### 3.6.2.2. Compliance Management

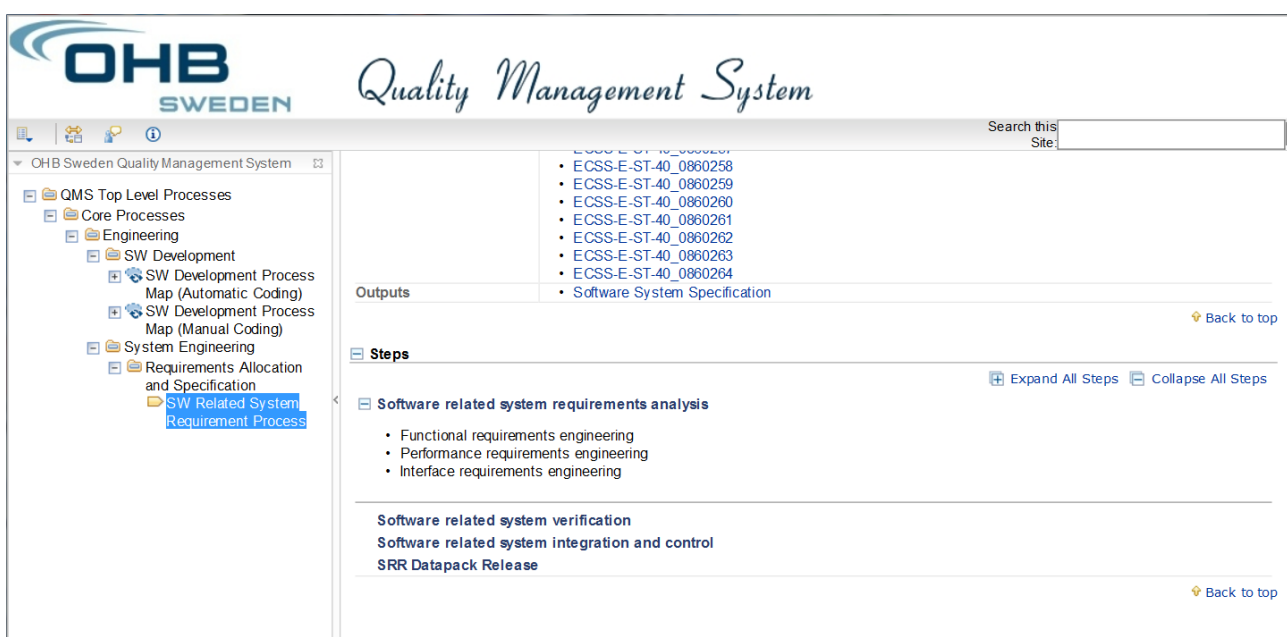
**Table 25.** CS11 – Compliance Management

Realisation Scenario	Compliance Management
Scope	During the first prototype, the focus will be on measuring compliance of artefacts regarding the artefacts prescribed by industry standards and their associated ones.
Tool Settings	EPF Composer with usage of AMASS specific guidelines

<b>Participants</b>	<ul style="list-style-type: none"> <li>• Data Analysis: OHB</li> <li>• Tool User: OHB, MDH</li> <li>• Results Analysis: OHB</li> </ul>
<b>Activities realised</b>	<ol style="list-style-type: none"> <li>1. Mapping of standard requirements to activities and methods that shall ensure fulfilment of the requirements.</li> <li>2. Published content to the web for easy access and availability.</li> </ol>
<b>Usage Decisions</b>	None
<b>Expected Results</b>	<ul style="list-style-type: none"> <li>• Compliance for ECSS-E-ST-40C, evidence provided through mapping of requirements to activities.</li> </ul>



**Figure 41.** Method composer SW validation and mapped requirements



**Figure 42.** Quality management System OHB



## 4. Conclusions

This document presents the report of AMASS case study realization in different usage scenarios proposed in D1.1 [2] and based in the data collection done in D1.2 [3]. This deliverable iteration (D1.4) focuses on the first AMASS prototype (Prototype Core).

This deliverable offers an overview of the AMASS platform tools and the different approaches of the problems solved with them, in different application domains and from an industrial perspective. It is a keystone for how the AMASS solutions can be improved, to get closer to the industrial sector, and to provide feedback to architecture and software developers for future iterations.

After using the AMASS tools, some feedback has been gained by project partners:

- The different project baseline elements to be mapped with evidences should provide more details to the user in order to better evaluate in which degree the evidences fulfil them.
- The automatically generated word document, although very detailed, should offer a summary at the beginning, evaluating the degree of compliance with respect to the analysed standard. Maybe by just adding a chart with a few categories of baseline elements with a compliance degree for each category.
- Option of attaching pictures (jpgs, bmps...) to the boxes, so that some evidences could be easily shown.
- Option of introducing hyperlinks to most of forms and the possibility of going to them with the same click.
- Option of (in case of Requirements specification) having a “check button” for the completed requirements. It would be also desirable if after clicking the accomplishment of the requirement, the user can look at the evidence.
- Integrated option of filling with colours if requirements are accomplished and validated.
- Option of managing the permissions for accessing the functionalities of the tools by the Project Manager.

Benchmarking will be one of the main aspects for the evolutions of this deliverable, i.e. the reports about the AMASS demonstrators with the future versions of the AMASS Tool Platform. D1.3 [4] will provide the necessary guidelines to implement a complete benchmarking (September 2017).

## 5. Appendix A: Ongoing Activities on other Case Studies

### 5.1. Case Study 2: Automotive domain: Advanced driver assistance function with electric vehicle sub-system.

#### 5.1.1. Case Study Specification

The focus of this case study is on building blocks for ADAS with electric vehicle sub-system. The collaboration within AMASS will support the collection of field data and system requirements.

For a detailed description on the case study see the Deliverable "D1.1. Case studies description and business impact" [2].

#### 5.1.2. Usage Scenario 1

The goal of US1 is to bring intra domain reuse forward. In the automotive industry, price sensitivity leads to a strategy to develop product families rather than single products. Product families require certification of all "family members", which would drive efforts (and price!). Therefore, the target is on re-use of certification and/or use of pre-certified components.

##### 5.1.2.1. Assurance Project Creation

Usage Scenario	Assurance Project Creation
<b>Description</b>	The main purpose of this Usage Scenario is to create a safety assurance project valid for Automotive Domain that guarantees compliance with ISO 26262. The following standards must be followed: <ul style="list-style-type: none"> <li>• ISO 26262</li> </ul>
<b>User</b>	IFX (owner), B&M, KMT
<b>Goal</b>	To create an assurance project that will be used to manage the project in terms of safety and compliance to ISO 26262.
<b>Activity #1</b>	KMT: The main purpose of this Activity is to create a safety assurance project valid for Automotive Domain that guarantees compliance with ISO 26262. The following standards must be followed: <ul style="list-style-type: none"> <li>• ISO 26262</li> </ul> <b>Create an assurance project</b> <ul style="list-style-type: none"> <li>• Inputs <ul style="list-style-type: none"> <li>○ ISO 26262</li> </ul> </li> <li>• Tools used <ul style="list-style-type: none"> <li>○ medini analyze</li> <li>○ SVN</li> </ul> </li> <li>• Outputs <ul style="list-style-type: none"> <li>○ Functional Safety Project Plan</li> </ul> </li> </ul>

##### 5.1.2.2. System Component Specification

Usage Scenario	Assurance Project Creation
<b>Description</b>	The main purpose of this Usage Scenario is to define the safety requirements for the item, the safety architecture. These specifications must take into account: <ul style="list-style-type: none"> <li>• Item definition</li> </ul>

	<ul style="list-style-type: none"> <li>functional safety requirements</li> <li>technical safety requirements</li> </ul>
<b>User</b>	IFX (owner), B&M, KMT
<b>Goal</b>	To create an assurance project that will be used to manage the project in terms of safety and compliance to ISO 26262.
<b>Activity #1</b>	<p><b>KMT:</b></p> <p>The main purpose of this Usage Scenario is to define the safety requirements for the item, the safety architecture. These specifications must be taken into account:</p> <ul style="list-style-type: none"> <li>Item definition</li> <li>Functional safety requirements</li> <li>Technical safety requirements</li> <li>Hardware safety requirements</li> <li>Software safety requirements</li> </ul> <p><b>Define safety requirements and system design</b></p> <ul style="list-style-type: none"> <li>Inputs <ul style="list-style-type: none"> <li>ISO 26262</li> <li>Functional Safety Concept ACC by B&amp;M</li> </ul> </li> <li>Tools used <ul style="list-style-type: none"> <li>medini analyze</li> <li>SVN</li> </ul> </li> <li>Outputs <ul style="list-style-type: none"> <li>Part 1 of Safety Concept ACC by KMT: <ul style="list-style-type: none"> <li>Item definition</li> <li>Functional safety requirements</li> <li>Technical safety requirements</li> <li>Hardware safety requirements</li> <li>Software safety requirements</li> <li>System Design</li> <li>Hardware Architecture</li> <li>Software Architecture</li> </ul> </li> </ul> </li> </ul>

### 5.1.2.3. Evidence Management

<b>Usage Scenario</b>	<b>Evidence Management</b>
<b>Description</b>	Allow the user to define the set of evidences to be provided to meet the requirements defined by ISO 26262 and customer requirements. These evidences must be then mapped to the actual artefacts collected from the different tools as evidence.
<b>User</b>	IFX (owner), B&M, KMT
<b>Goal</b>	<ul style="list-style-type: none"> <li>Visualization of the evidences to be provided in compliance with what is required by ISO 26262 and any additional reference document (e.g. company processes, regulation requirements, etc.).</li> <li>Definition and visualization of the mappings between the evidences to be provided and the corresponding artefacts used as evidence.</li> </ul>
<b>Activity #1</b>	<p><b>Specify evidence model</b></p> <p><b>KMT:</b></p> <p>Allow the user to define the set of evidences to be provided to meet the requirements defined by ISO 26262 and customer requirements. These</p>

evidences must be then mapped to the actual artefacts collected from the different tools as evidence.

**Define safety requirements and system design**

- Inputs
  - ISO 26262
  - Functional Safety Concept ACC by B&M
  - Part 1 of Safety Concept ACC by KMT:
- Tools used
  - medini analyze
  - SVN
- Outputs
  - Part 2 of Safety Concept ACC by KMT:
    - GSN Requirements Trees (including safety requirements)
    - Fault Tree Analysis (FTA)
    - Allocation between preliminary-, system-, hardware- and software architecture
    - Allocation of requirements
    - Safety Concept Report

## 5.2. Case Study 7: Avionics domain: Safety assessment of multi-modal interactions in cockpits

### 5.2.1. Case Study Specification

This case study focuses on the following aspects:

1. Application of aerospace industrial standards for safety assessments,
2. automation of the verification objectives, and
3. reuse of assurance artefacts from automotive technology into the avionics domains.

For a detailed description on the case study see the Deliverable “D1.1. Case studies description and business impact” [2].

### 5.2.2. Usage Scenario 1: Safety Assessment

#### 5.2.2.1. System Component Specification

Realisation Scenario	Model-based System Component Specification
<b>Scope</b>	<p>The system components will be specified including system requirements. The following standards will be applied:</p> <ul style="list-style-type: none"> <li>• <i>Safety</i>: SAE ARP 4761 – EUROCAE ED-135 – Guidelines and methods for conducting the safety assessment process on civil airborne systems and equipment.</li> <li>• <i>System</i>: SAE ARP 4754A – EUROCAE ED-79A – Guidelines for development of civil aircraft and systems.</li> </ul>
<b>Tool Settings</b>	<ul style="list-style-type: none"> <li>• Mathworks Matlab Simulink – system architecture only</li> <li>• Sparx Systems Enterprise Architect – 3 View Systems Engineering</li> <li>• Internal tools to specify system requirements and contracts</li> </ul>
<b>Participants</b>	<ul style="list-style-type: none"> <li>• Leader: HON</li> </ul>
<b>Activities realised</b>	<ol style="list-style-type: none"> <li>1. Author safety requirements</li> <li>2. Formalize safety requirements</li> <li>3. Create system architecture – 3 views of the system</li> <li>4. Allocate requirements to system.</li> </ol>
<b>Usage Decisions</b>	<p>Decide which part of the system will be used in Simulink and which in Enterprise Architect.</p> <p>Decide how to allocate the requirements in order to enable automated verification.</p>
<b>Expected Results</b>	<ul style="list-style-type: none"> <li>• System architecture, formal safety requirement specification.</li> </ul>

#### 5.2.2.2. Safety Assessment

Realisation Scenario	Safety Assessment
<b>Scope</b>	<p>Automated safety assessment for:</p> <ul style="list-style-type: none"> <li>• Complete MMI system</li> <li>• Touch recognition</li> </ul>
<b>Tool Settings</b>	<ul style="list-style-type: none"> <li>• Mathworks Matlab Simulink – system architecture only</li> <li>• Sparx Systems Enterprise Architect – 3 View Systems Engineering</li> <li>• Tools for automated safety assessment.</li> </ul>

<b>Participants</b>	<ul style="list-style-type: none"> <li>• Leader: Hon</li> <li>• Tool providers: UOM, FBK</li> </ul>
<b>Activities realised</b>	<ol style="list-style-type: none"> <li>1. Identify system failures</li> <li>2. Select appropriate system architecture pattern to reach design assurance level.</li> <li>3. Apply fault injection mechanism</li> <li>4. Compute fault propagation automatically using for example FBK and UOM tools</li> <li>5. Perform safety assessment, for example generate Fault Tree Analysis (FTA)</li> <li>6. Evaluate the results</li> </ol>
<b>Usage Decisions</b>	<p>Which parts of safety assessment should be automated.</p> <p>Which safety assessment tools used.</p>
<b>Expected Results</b>	A list of safety hazards and security threats

#### 5.2.2.3. Evidence Management

<b>Realisation Scenario</b>	<b>Evidence Assessment</b>
<b>Scope</b>	Collect and manage evidence artefacts required to fulfil the selected standards
<b>Tool Settings</b>	OpenCert Tools: Evidence Management
<b>Participants</b>	<ul style="list-style-type: none"> <li>• Leader: Hon</li> <li>• TEC</li> </ul>
<b>Activities realised</b>	<ol style="list-style-type: none"> <li>1. Create artefact model for safety.</li> <li>2. Collect evidence documents</li> <li>3. Initial verification and judgment of the quality of the evidence</li> <li>4. Perform evaluation of results based on D1.3 [4]</li> <li>5. Report the result to the customer.</li> </ol>
<b>Usage Decisions</b>	How to measure baseline process – a process performed by separate team?
<b>Expected Results</b>	Evidence model and artefact repository.

#### 5.2.2.4. Compliance Management

<b>Realisation Scenario</b>	<b>Compliance Management</b>
<b>Scope</b>	Evaluate compliance of artefacts as per the avionics standards.
<b>Tool Settings</b>	OpenCert Tools: Assurance Project Management
<b>Participants</b>	<ul style="list-style-type: none"> <li>• Leader: HON</li> <li>• TEC</li> </ul>
<b>Activities realised</b>	<ol style="list-style-type: none"> <li>1. Definition of the development plan</li> <li>2. Definition of tasks and tools to be integrated</li> <li>3. Evaluation of the development lifecycle based on AMASS evaluation framework</li> <li>4. Reporting of compliance results to the customer</li> <li>5. Analyse compliance accomplishment</li> </ol>
<b>Usage Decisions</b>	None
<b>Expected Results</b>	<p>Compliance report complying with standards:</p> <ul style="list-style-type: none"> <li>• SAE ARP 4761 – EUROCAE ED-135</li> <li>• SAE ARP 4754A – EUROCAE ED-79A</li> <li>• RTCA DO-178C – EUROCAE ED-12C</li> </ul>

### 5.2.3. Usage Scenario 2: Automated Verification

#### 5.2.3.1. System Component Specification

Realisation Scenario	Model-based System Component Specification
<b>Scope</b>	<p>The system components will be specified including system requirements. The following standards will be applied:</p> <ul style="list-style-type: none"> <li>• <i>System</i>: SAE ARP 4754A – EUROCAE ED-79A – Guidelines for development of civil aircraft and systems</li> </ul>
<b>Tool Settings</b>	<ul style="list-style-type: none"> <li>• Mathworks Matlab Simulink – system architecture only</li> <li>• Sparx Systems Enterprise Architect – 3 View Systems Engineering</li> <li>• Internal tools to specify system requirements and contracts</li> </ul>
<b>Participants</b>	<ul style="list-style-type: none"> <li>• Leader: HON</li> </ul>
<b>Activities realised</b>	<ol style="list-style-type: none"> <li>1. Author system requirements</li> <li>2. Formalize behavioural system requirements</li> <li>3. Create system architecture – 3 views of the system</li> <li>4. Allocate requirements to the system.</li> </ol>
<b>Usage Decisions</b>	<p>Decide which part of the system will be used in Simulink and which in Enterprise Architect.</p> <p>Decide how to allocate the requirements in order to enable automated verification.</p>
<b>Expected Results</b>	<ul style="list-style-type: none"> <li>• System architecture, formal system requirement specification.</li> </ul>

#### 5.2.3.2. Automated Verification

Realisation Scenario	Automated Verification
<b>Scope</b>	<p>Automated formal semantic verification and validation of requirements. Contribution to the following objectives:</p> <ul style="list-style-type: none"> <li>• Enforced verifiability (DO-178C A-3.4)</li> <li>• Ensured conformance to requirement standards (DO-178C A-3.5)</li> <li>• Decrease the number of defects (ARP 4761 objective, DO-178C A-3.2)</li> </ul> <p>Automated formal verification that requirements comply with system. Contribution to the objectives: DO178C A-4.1. and when possible to DO178C A-3.1</p>
<b>Tool Settings</b>	<ul style="list-style-type: none"> <li>• Mathworks Matlab Simulink – system architecture only</li> <li>• Sparx Systems Enterprise Architect – 3 View Systems Engineering</li> <li>• Internal tool for formalization, analysis and brokerage of verification tasks: ForReq</li> <li>• Tools for automated formal verification: DIVINE, NuSMV, nuXmv, Looney, Acacia+</li> </ul>
<b>Participants</b>	<ul style="list-style-type: none"> <li>• Leader: HON</li> <li>• TEC</li> <li>• Tool providers: UOM, FBK</li> </ul>
<b>Activities realised</b>	<ol style="list-style-type: none"> <li>1. Automated semantic requirement analysis</li> <li>2. Create system design</li> <li>3. Automated formal verification of requirement compliance with system design</li> <li>4. Automated generation of test cases if requested</li> </ol>



	5. Evaluate the results
<b>Usage Decisions</b>	Which parts of safety assessment should be automated. Which safety assessment tools used.
<b>Expected Results</b>	Verification results, measurements of the injected, detected and remove defects in all development phases.

#### 5.2.3.3. Evidence Management

<b>Realisation Scenario</b>	<b>Evidence Assessment</b>
<b>Scope</b>	Collect and manage evidence artefacts required to fulfil the selected standards.
<b>Tool Settings</b>	OpenCert Tools: Evidence Management
<b>Participants</b>	<ul style="list-style-type: none"> <li>• Leader: HON</li> <li>• TEC</li> </ul>
<b>Activities realised</b>	<ol style="list-style-type: none"> <li>1. Create artefact model for the system.</li> <li>2. Collect evidence documents</li> <li>3. Initial verification and judgment of the quality of the evidence</li> <li>4. Perform evaluation of results based on D1.3 [4]</li> <li>5. Report the result to the customer.</li> </ol>
<b>Usage Decisions</b>	How to measure baseline process – a process performed by separate team?
<b>Expected Results</b>	Evidence model and artefact repository.

#### 5.2.3.4. Compliance Management

<b>Realisation Scenario</b>	<b>Compliance Management</b>
<b>Scope</b>	Evaluate compliance of artefacts as per the avionics standards.
<b>Tool Settings</b>	OpenCert Tools: Assurance Project Management
<b>Participants</b>	<ul style="list-style-type: none"> <li>• Leader: HON</li> <li>• TEC</li> </ul>
<b>Activities realised</b>	<ol style="list-style-type: none"> <li>1. Definition of the development plan</li> <li>2. Definition of tasks and tools to be integrated.</li> <li>3. Evaluation of the development lifecycle based on AMASS evaluation framework</li> <li>4. Reporting of compliance results to the customer</li> <li>5. Analyse compliance accomplishment</li> </ol>
<b>Usage Decisions</b>	None
<b>Expected Results</b>	Compliance report complying with standards: <ul style="list-style-type: none"> <li>• SAE ARP 4761 – EUROCAE ED-135</li> <li>• SAE ARP 4754A – EUROCAE ED-79A</li> <li>• RTCA DO-178C – EUROCAE ED-12C</li> </ul>

## 5.3. Case Study 8: Automotive domain: Telematics function

### 5.3.1. Case Study Specification

This case study focuses on component-based (element-out-of-context) multi-concern assurance, analysis and assessment for a positioning component where safety, security and performance are critical quality attributes.

For a detailed description of the case study, see the Deliverable “D1.1 Case studies description and business impact” [2].

### 5.3.2. Usage Scenarios (Common to all the Scenarios)

#### Usage Scenario 1 Multi-concern assurance case for safety/security (US1 MCAC)

This scenario is about creation of an assurance case and processes for multiple standards (safety: ISO 26262 and cybersecurity: SAE J3061).

This scenario is mainly related to assurance case specification, evidence management and compliance management.

#### Usage Scenario 2 Multi-concern assessment (US2 MCASS)

This scenario deals with assessment of multiple quality attributes (i.e. against multiple standards) based on the multi-concern assurance case in scenario 1. However, this scenario has not yet been started and will be addressed in the next iteration of this deliverable. The AMASS tools (e.g. OpenCert) will be used.

This scenario is mainly related to evidence management and compliance management.

#### Usage Scenario 3 Multi-concern specification, analysis, assurance (US3 SAASSA)

This scenario is about specification (co-engineering of function, safety, and security), analysis and collection of assurance evidence for multiple concerns. The same assurance case as scenarios 1 and 2 is used.

This scenario is mainly related to system component specification, evidence management and compliance management.

#### 5.3.2.1. System Component Specification

Realisation Scenario	System Component Specification
Scope	Model and specify a safety and security-related element-out-of-context extended with relations/traceability to the assurance case.
Tool Settings	Papyrus/CHESS and OpenCert (for traceability)
Participants	SPS, COM
Activities realised	<ul style="list-style-type: none"> <li>Item definition</li> <li>Tool installation/planning</li> </ul>
Usage Decisions	
Expected Results	System and software design for the prototype in case study 8.

#### 5.3.2.2. Assurance Case Specification

Realisation Scenario	Assurance Case Specification
Scope	Creation of an assurance case for multiple standards (safety: ISO 26262 and cybersecurity: SAE J3061).
Tool Settings	Planned use of OpenCert in AMASS iteration 2.

<b>Participants</b>	SPS, COM
<b>Activities realised</b>	
<b>Usage Decisions</b>	
<b>Expected Results</b>	Assurance case applicable for each concern, and later also with interdependences.

#### 5.3.2.3. Evidence Management

<b>Realisation Scenario</b>	<b>Evidence Management</b>
<b>Scope</b>	Storage and management of assurance artefacts.
<b>Tool Settings</b>	Planned use of OpenCert in AMASS iteration 2.
<b>Participants</b>	SPS, COM
<b>Activities realised</b>	
<b>Usage Decisions</b>	
<b>Expected Results</b>	Enable automatic impact analysis on changes.

#### 5.3.2.4. Compliance Management

<b>Realisation Scenario</b>	<b>Compliance Management</b>
<b>Scope</b>	Safeguard compliant process and product
<b>Tool Settings</b>	Planned use of EPF and OpenCert in AMASS iteration 2.
<b>Participants</b>	SPS, COM, MDH
<b>Activities realised</b>	
<b>Usage Decisions</b>	
<b>Expected Results</b>	Continuous compliance support for ISO26262 and SAE J3061

## 5.4. Case Study 9: Air Traffic Management domain: Safety-Critical SW Lifecycle of a Monitoring Syst. for NavAid

### 5.4.1. Case Study Specification

CS9 is aimed to re-engineer, through the usage of tools and methods provided by the AMASS project:

- The SW of the Monitoring subsystem of a safety-critical CPS such as the DME (DME: Distance Measuring Equipment; it is a radio-navigation system which provides the distance information between the aircraft and the location of the DME ground equipment).
- More in general, the processes of the whole SW development lifecycle for such a CPS, applying the CNS/ATM safety certification standards ('EUROCAE ED-109', 'RTCA DO-278', 'EUROCAE ED-153').

### 5.4.2. Usage Scenario 1: SWD (Software Design)

During this phase of the SW development process:

- the AMASS SW modelling tools will be used for the SW Architecture definition, verification and validation;
- the AMASS platform will be used to define, verify and validate the SW module interaction (through contracts and formal methods), in order to help the qualification/certification process at the architecture level;
- the AMASS platform will be used to conduct safety analyses (e.g., FMEA, FTA, Component FTA) taking as input the model-based design and evaluating the contracts;

#### 5.4.2.1. Assurance Project Creation

Realisation Scenario	Assurance Project Creation
Scope	<p>US1 will assess the possibility to use the AMASS-platform tools to improve the design and the efficiency of the development cycle for CNS/ATM Software.</p> <p>The main activities will consist in defining the System and Software Architecture, including module interactions and Safety Analyses (e.g. FMEA, FTA, Component FTA).</p> <p>AMASS tools will also be used for evidence and compliance management to reduce certification and re-certification effort.</p>
Tool Settings	AMASS Tools
Participants	System Engineer Software Engineer Safety Engineer
Activities realised	<ul style="list-style-type: none"> <li>Definition of the software architecture (including safety properties prescribed by a Safety Plan), through the SW modeling tools provided by AMASS.</li> <li>Use of the AMASS platform to define, verify and validate the SW module interaction (through contracts and formal methods).</li> </ul>
Usage Decisions	-
Expected Results	Early detection of architectural issues and reduction of integration effort.

#### 5.4.2.2. System Design and Dependability Assessment

Realisation Scenario	System Design and Dependability Assessment
Scope	US1 will assess the possibility to use the AMASS-platform methods/tools (formal methods for module interface definition) for early software-design validation and

	safety assessment, with the parallel conduction of typical safety analyses (e.g. FMEA, FTA, Component FTA).
<b>Tool Settings</b>	AMASS Tools
<b>Participants</b>	System Engineer Software Engineer Safety Engineer
<b>Activities realised</b>	Use of AMASS platform to conduct safety analyses (e.g., FMEA, FTA, Component FTA) taking as input the model-based design and evaluating the contracts.
<b>Usage Decisions</b>	-
<b>Expected Results</b>	Improve design validation and safety, reducing the effort.

### 5.4.3. Usage Scenario 2: SWV (Software Verification & Validation)

During this phase of the SW development process the AMASS platform will help to guarantee that the SW Development Process follows the correct procedures according to ED-109 standards: the complete traceability should be assured, from the ED-109 objectives to the source code, through all the artefacts.

#### 5.4.3.1. Evidence Management

Realisation Scenario	Evidence Management
<b>Scope</b>	Within US2, the AMASS platform will allow the collection of the evidences to be provided to meet the requirements defined by the applicable standards. Such evidences shall be then mapped to the actual artefacts.
<b>Tool Settings</b>	AMASS Tools
<b>Participants</b>	System Engineer Software Engineer Safety Engineer
<b>Activities realised</b>	1. Definition of the evidences required according to assurance case specification 2. Collection of artefacts through the AMASS tools 3. Data verification, validation and analysis: reporting
<b>Usage Decisions</b>	-
<b>Expected Results</b>	Simplification of the Safety Assurance process.

#### 5.4.3.2. Compliance Management

Realisation Scenario	Compliance Management
<b>Scope</b>	US2 will allow the analysis about the capability of the AMASS processes to guarantee that the software lifecycle follows the correct procedure according to the CNS/ATM standards.
<b>Tool Settings</b>	AMASS Tools
<b>Participants</b>	System Engineer Software Engineer Safety Engineer
<b>Activities realised</b>	1. Modelling of ED 109 objectives using AMASS tools and mapping of the evidences with the objectives 2. Performance evaluation and reporting
<b>Usage Decisions</b>	-
<b>Expected Results</b>	Simplification of the process aimed at certifying the alignment with the applicable standards.

## Abbreviations and Definitions

ACSL	A C Specification Language
AOCS	Attitude and Orbit Control System
API	Application Programming Interface
ARTEMIS	ARTEMIS Industry Association is the association for actors in Embedded Intelligent Systems within Europe
ASIC	Application-specific integrated circuit
BXML	XML-based format for B models
CA	Consortium Agreement
COPPILOT	System to Open and Close the Platform Screen Doors DPAS - Détecteur de Passage (Crossing Detection Equipment)
CS	Case Study
DRF	Détecteur de Roue Fer (Steel Wheel Presence Sensor)
EBIOS	Méthode pour l'Expression des Besoins et l'Identification des Objectifs de Sécurité (method to Express and Identify the Security Needs and Objectives)
ECSS	European Cooperation for Space Standardization
EDSA	Embedded Device Security Assurance
ESA	European Space Agency
FMEA	Failure Mode and Effects Analysis
FMECA	Failure Mode, Effects and Criticality Analysis
FMVEA	Failure Mode, Vulnerabilities and Effects Analysis
FPA	Focal Plane Assembly
FPGA	Field-Programmable Gate Array
FTA	Fault Tree Analysis
GAPS	A ClearSy system to measure the gap between the train and the platform and authorize the roll-out of a gap filling system
GPP	General-purpose pre-processor
IACS	Industrial and Automation Control System
ICM	Instrument Control Module
IDE	Integrated Development Editor
MMI	Multi-Modal Interactions
NoC	Network-on-chip
NVD	National Vulnerability Database
OEU	OLCI Electronic Unit
OLCI	Ocean & Land Colour Instrument
PSD	Platform Screen Door
RFP	Request for Proposal
RTU	Remote Terminal Unit
SIL	Safety Integrity Level
SoC	System-On-Chip
SoS	System of Systems
SSDP	Scalable Sensor Data Processor Breadboard
SW	Software
SWD	Software Design
SWV	Software Verification & Validation
SysML	System Modelling Language
TARA	Threat Analysis and Risk Assessment
TRL	Technology Readiness Levels
UL	Underwriters Laboratories
UML	Unified Modelling Language



US	Usage Scenario
V&V	Verification and validation
VAM	Video Acquisition Module

.

## References

- [1] Book: “Better Embedded System Software”, P. Koopman, Carnegie Melon University.
- [2] AMASS [D1.1 Case studies description and business impact](#)
- [3] AMASS D1.2 Case Study Data Collection
- [4] AMASS D1.3 Evaluation Framework and Quality Metrics
- [5] Kaiser, Bernhard and Weber, Raphael and Oertel, Markus and Monajemi Nejad, Behrang and Zander, Justyna: Contract-Based Design of Embedded Systems Integrating Nominal Behavior and Safety. IN: Complex Systems Informatics and Modeling Quarterly (CSIMQ) 04/2015
- [6] The VeloxCar – a demonstrator for automated driving. Technical Report. Assystem Germany GmbH, 2017 (available on AMASS project repository)
- [7] Kaiser, B. ; Monajemi Nejad, B. ; Kusche, D. ; Schulte, H. : Systematic Design and Validation of Degradation Cascades for Safety-Relevant Systems, S. 1 - 9, 27th European Safety and Reliability Conference ESREL 2017, Portoroz, Slovenia, 2017
- [8] PolarSys: CHES project. <https://www.polarsys.org/projects/polarsys.chess>
- [9] PolarSys: OpenCert project. <https://www.polarsys.org/projects/polarsys.opencert>
- [10] Eclipse Process Framework Project (EPF) <https://eclipse.org/epf/>
- [11] OPENCROSS project <http://www.opencross-project.eu/>
- [12] CHES project <http://www.chess-project.org/>
- [13] SafeCer project <http://www.safecer.eu/>