ECSEL Research and Innovation actions (RIA)

AMASS

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

AMASS demonstrators (b) D1.5

Work Package:	WP1 Case Studies and Benchmarking
Dissemination level:	PU = Public
Status:	Final
Date:	21 April 2018
Responsible partner:	Miguel Gomez (Thales Alenia Space - Spain) Isaac Moreno (Thales Alenia Space - Spain)
Contact information:	Miguel.GomezIrusta@thalesaleniaspace.com Isaac.Moreno@thalesaleniaspace.com
Document reference:	AMASS_D1.5_WP1_TAS_V1.0

This deliverable is part of a project that has received funding from the ECSEL JU under grant agreement No 692474. This Joint Undertaking receives support from the European Union's Horizon 2020 research and innovation programme and from Spain, Czech Republic, Germany, Sweden, Italy, United Kingdom and France.

Contributors

Names	Organisation
Isaac Moreno, Miguel Gómez	Thales Alenia Space – Spain (TAS)
Garazi Juez, Estibaliz Amparan, Cristina Martínez, Angel López	TECNALIA Research & Innovation (TEC)
Elena Alaña, Javier Herrero	GMV Aerospace and Defence, S.A.U (GMV)
Fredrik Warg, Martin Skoglund	RISE Research Institutes of Sweden (SPS)
Benito Caracuel, David Pampliega	Schneider Electric (TLV)
Anna Carlsson	OHB Sweden (OHB)
Barbara Gallina	Maelardalen Hoegskola (MDH)
Thomas Gruber, Abdelkader Shabaan	Austrian Institute of Technology (AIT)
Helmut Martin, Robert Bramberger, Bernhard Winkler	Virtual Vehicle (ViF)
Thierry Lecomte	ClearSy (CLS)
Oliver Kreuzmann, Markus Grabowski	Assystem Germany GmbH (B&M)
Bernhard Kaiser, Nino Gabriel	ANSYS Medini Technologies AG (KMT)
Jose Luis de la Vara, Jose Maria Alvarez, Anabel Fraga, Roy Mendiete, Miguel Rozalen, Eugenio Parra	Universidad Carlos III de Madrid (UC3)
Luis Alonso, Borja Lopez, Elena Gallego, Roy Mendieta	The REUSE Company (TRC)
Camile Parillaud, Fabien Belmonte	Alstom Group (ALS)
Vít Koksa, Tomáš Kratochvíla	Honeywell International (HON)

Reviewers

Names	Organisation
David Pampliega, Benito Caracuel (Peer-reviewers)	Schneider Electric España (TLV)
Fredrik Warg (Peer-reviewer)	RISE Research Institutes of Sweden (SPS)
Cristina Martínez (Quality Manager)	Tecnalia Research & Innovation (TEC)
Garazi Juez (TC review)	Tecnalia Research & Innovation (TEC)
Barbara Gallina (TC review)	Maelardalen Hoegskola (MDH)

TABLE OF CONTENTS

List of Figures

List of Tables

Executive Summary

The AMASS project is developing the first European-wide open certification/qualification platform for the assurance and certification of Cyber-Physical Systems (CPS).

This deliverable, output of the Task 1.4 “Case Study Implementation and Benchmarking”, focuses on evaluating the AMASS Prototype P1 by industrial partners in several case studies. Those case studies represent meaningful segments of the different application domains addressed in AMASS. 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.

The task T1.4 provides feedback and an active proof of the performance of the AMASS platform in the industry. It provides support and advice to the tool’s developers (WP3-WP6) for future iterations based on the case studies. This task will also be an input for WP2 “Reference Architecture and Integration” to validate the AMASS platform and to create the AMASS user guidance methodological framework (D2.5). The last iteration of T1.4 will provide benchmarking for AMASS tools more widely, when the Task 1.3 “Benchmarking Framework” is finished.

The data required to develop the task T1.4 has been taken from the deliverable D1.2 [2], which is related to data collection usage scenarios for each case study described in D1.1 [1].

From the Core Prototype to the Prototype P1, several functionalities have been implemented. Besides the basic building blocks which were already available in Prototype Core, almost all the STO building blocks have been released for the Prototype P1 (see Figure 1). Apart from the new functionalities, some recommendations in terms of features and bugs found in the Core Prototype evaluation have been included/solved in the Prototype P1. For this iteration, tool providers have developed User Manuals for the tools and specifically for the different STO objectives as well, which have been an immeasurable help in the development of the Case Studies.

The deliverable D1.5 focuses on validating the Prototype P1 functionalities, having the Core Prototype tools been previously analysed for D1.4 [4].

During this second iteration, some case studies have also used the previously developed Core Prototype functionalities, such as OpenCert or EPF-Composer for compliance management. For each of the case studies, the coverage with respect to the AMASS Prototype P1 has been identified.

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, taking into account usability aspects as well.

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 OPENCOSS [10] and SafeCer [12] towards an architecture-driven, multi-concern assurance, reuse-oriented, and seamlessly interoperable tool platform.

The expected tangible AMASS results are:

a) The **AMASS Reference Tool Architecture**, which will extend the OPENCOSS 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.

b) 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.

c) 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 the AMASS base tools (tool infrastructure for database and access management, among others) and extension tools enriching the 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.

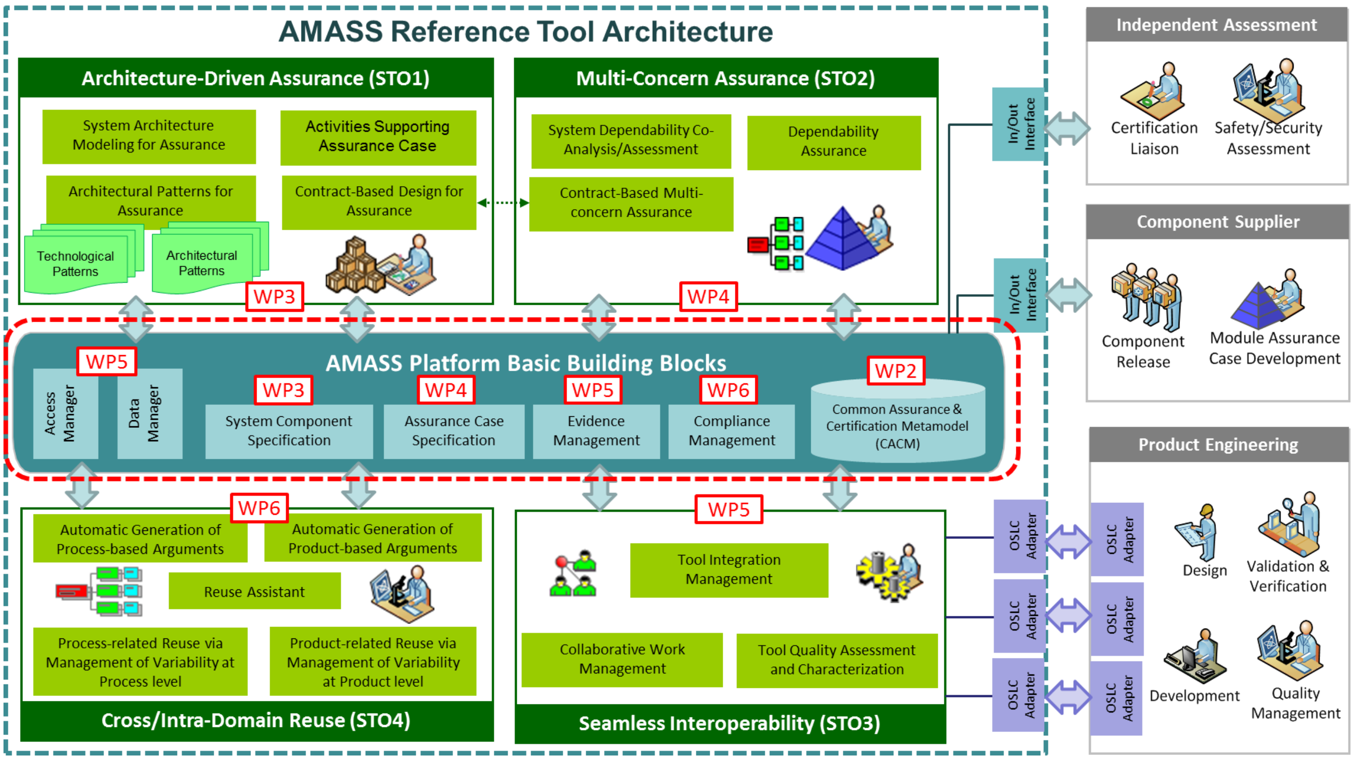

AMASS Reference Tool Architecture
Architecture-Driven Assurance (STO1)
System Architecture Modeling for Assurance
Activities Supporting Assurance Case
Architectural Patterns for Assurance
Contract-Based Design for Assurance
Technological Patterns
Architectural Patterns
WP3
Multi-Concern Assurance (STO2)
System Dependability Co-Analysis/Assessment
Dependability Assurance
Contract-Based Multi-concern Assurance
WP4
In/Out Interface
In/Out Interface
AMASS Platform Basic Building Blocks
WP5
Access Manager
Data Manager
WP3
System Component Specification
WP4
Assurance Case Specification
WP5
Evidence Management
WP6
Compliance Management
WP2
Common Assurance & Certification Metamodel (CACM)
WP6
Automatic Generation of Process-based Arguments
Automatic Generation of Product-based Arguments
Reuse Assistant
Process-related Reuse via Management of Variability at Process level
Product-related Reuse via Management of Variability at Product level
Cross/Intra-Domain Reuse (STO4)
WP5
Tool Integration Management
Collaborative Work Management
Tool Quality Assessment and Characterization
Seamless Interoperability (STO3)
OSLC Adapter
Independent Assessment
Certification Liaison
Safety/Security Assessment
Component Supplier
Component Release
Module Assurance Case Development
Product Engineering
Design
Validation & Verification
Development
Quality Management

Figure 1. AMASS Building blocks

The scope of the previous deliverable D1.4 [4] was the Core Prototype, which covers the AMASS Platform Basic Building Blocks in the middle of Figure 1.

This deliverable (D1.5) covers the Prototype P1. This second iteration addresses not only the basic building block functionalities but also evolves to tackle some of the functionalities highlighted in green in Figure 1 (not all the functionalities have been fully implemented yet, the remaining ones will be covered during the third iteration, or Prototype P2, next year).

1.1. Scope and Purpose

The objective of this deliverable is to validate the prototype P1 of the AMASS solution. This second deliverable related to the task T1.4 “Case Study Implementation and Benchmarking” is based on the case study specifications from the task T1.1, as well as from the data collection usage scenarios presented in the deliverable D1.2 [2]. The 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 the deliverable D1.4 [4], the implementation of the AMASS Platform Basic Building Blocks was covered. The deliverable D1.5 addresses the validation of more features related to the different STOs (see Figure 1). Benchmarking work will be covered once the on-going task T1.3 “Benchmarking Framework” has progressed and achieved a validated and stable benchmarking framework.

Given the importance of the industrial stakeholder’s opinion, AMASS industrial partner’s feedback has been gathered for a number of distinct aspects related to the functionality (e.g. access management) and the usability (e.g. GUI improvements) of the AMASS Prototype P1, which will be taken into consideration for further evolvements of the platform.

The results of the industrial participation will be matched with the AMASS technical requirements and test cases (WP2-WP6) and the achievement of the goals, from the end-user perspective in Space, Railway, Automotive, Industrial automation and Aeronautic domains.

1.2. Structure of the Document

The rest of the deliverable is structured as follows:

- Section 2 offers an overview of the AMASS project roadmap, the functional groups that constitute AMASS Prototype P1 and the main challenges in implementing the case studies.
- In Section 3, each case study presents an assessment of the platform, its coverage with respect to the AMASS Prototype P1, and some feedback about the main benefits and potential recommendations of the AMASS Platform functionalities.
- Section 4 provides a summary of the coverage of the AMASS Prototype P1 by the Case Studies.
- Section 5 concludes the document.

2. Background

2.1. AMASS Prototyping Roadmap

The AMASS Consortium has 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 (see Figure 2):

1. During the **first prototyping** iteration (Core Prototype), the AMASS Platform Basic Building Blocks (see Figure 1) were 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) contributed to complete the first prototype until milestone M2 (m13, April 2017).
2. During the **second prototyping** iteration (Prototype P1), the AMASS-specific Building Blocks have been 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 (m24, March 2018), the second prototype is available with the improvements and new features already included.
3. Finally, during the **third prototyping** iteration (Prototype P2), all AMASS building blocks will be integrated in a comprehensive toolset operating at TRL5. By milestone M4 (m36, March 2019) the third and last prototype will conclude the project with all the features and functionalities.

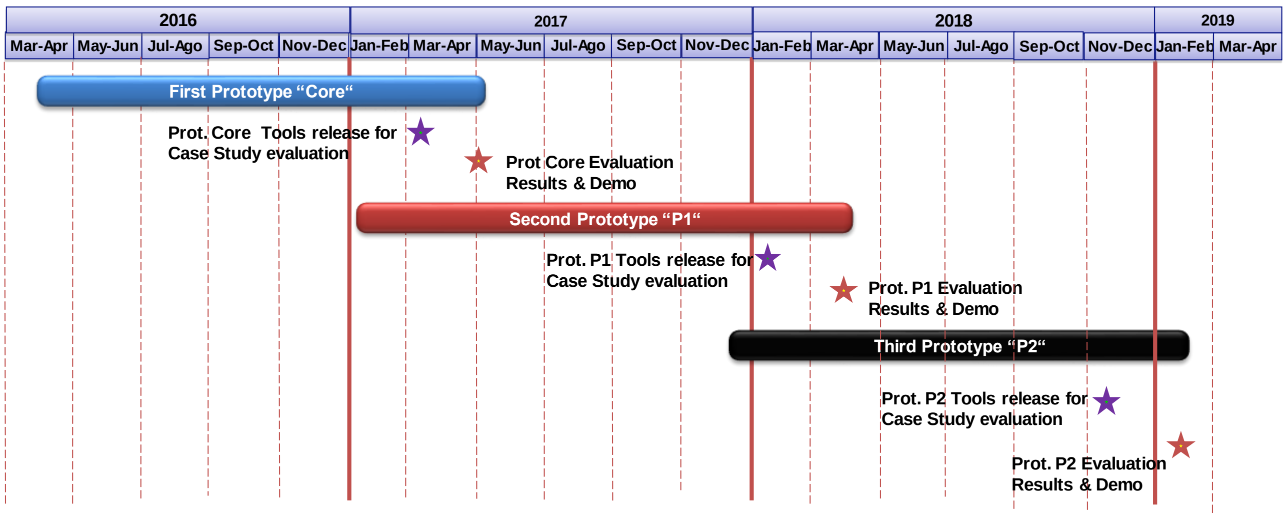


Figure 2. AMASS Prototyping roadmap

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 (D1.5) summarises the results of the "Case study development" dimension for the second AMASS prototype (Prototype P1).

2.2. 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 (i.e. how users will interact with the AMASS platform), in the context of typical business cases. The **deliverable D1.2** [2] provides a description of usage scenarios per case study.

(b) **Realisation of usage scenarios** reports the results of the application of usage scenarios in each of the AMASS prototyping iterations. This **deliverable (D1.5)** summarises the main results of the realisation of usage scenarios by using the Prototype P1.

(c) **Benchmarking of usage scenarios** will use a number of research/industrial questions and metrics to measure the effectiveness of the AMASS platform regarding the proposed business goals. This will be reported in the **deliverable D1.7** (AMASS solution benchmarking).

The AMASS Prototype P1 functionalities have been evaluated by the eleven AMASS Case Studies described in D1.1 [1] :

- CS1: Industrial and Automation Control Systems (IACS).
- CS2: Advanced driver assistance function with electric vehicle sub-system.
- CS3: Collaborative automated fleet of vehicles.
- CS4: Design and safety assessment of on-board software applications in Space Systems.
- CS5: Platform screen-doors controller.
- CS6: Automatic Train Control Formal Verification
- CS7: Safety assessment of multi-modal interactions in cockpits.
- CS8: Telematics function.
- CS9: Safety-Critical SW Lifecycle of a Monitoring Syst. for NavAid.
- CS10: Certification basis to boost the usage of Multiprocessor System-on-Chip (MPSoC) architectures in the Space Market.
- CS11: Design and efficiency assessment of model based Attitude and Orbit Control software development.

Table 1 shows the Case Studies' usage scenarios involved in the evaluation of the AMASS Prototype P1.

Table 1. Usages scenarios involved in the evaluation of the AMASS Prototype P1

CS	Owner	Short	Domain	Usage Scenarios
CS1	Schneider Electric España S.A.	TLV	Industrial Automation	US1: Managing compliance with IEC 61508, IEC 62443 and IEC 62351 US2: Perform safety and security co-assessment
CS2	Infineon	IFX	Automotive	US1: Reuse of safety artefacts within a product family (Intra-domain reuse)
CS3	Assystem Germany	B&M	Automotive	US1: Safety assessment for collaborative automated vehicle functions by model-based safety analysis and contracts US2: Process for development of collaborative automated vehicle functions, which considers functional safety, cybersecurity and reuse aspect

				US3: Collection and Analysis of Assurance Information
CS4	GMV Aerospace and Defence, S.A.U.	GMV	Space	US1: Assessment of components reuse using different execution platforms. US2: Re-qualification impact of modifying the hardware platform. US3: AMASS platform analyses to define safety, performance, reliability and availability requirements.
CS5	CLEARSY SAS	CLS	Railway	US1: Generation of Frama-C asserted C code from B models US2: Support for system-level model, including safety and security aspects
CS6	Alstom Transport SA	ALS	Railway	US1: Assurance Project Creation US2: System Design, V&V and Dependability Assessment US3: Evidence Management US4: Compliance Management
CS7	Honeywell	HON	Avionics	US1: Application of aerospace industrial standards for safety assessments US2: Automation of verification objectives
CS8	RISE Research Institutes of Sweden	SPS	Automotive	US1: Multi-concern assurance case for safety/security US2: Multi-concern assessment US3: Multi-concern specification, analysis, assurance
CS9	Thales Italia SpA	THI	Air Traffic Management	US1: System/Software Design and Safety Analysis US2: Safety Case
CS10	Thales Alenia Space	TAS-E	Space	US1: BSW modelling for SSDP US2: Reconfigurable FPGA architectures
CS11	OHB Sweden AB	OHB	Space	US1: Managing compliance with ECSS-E-ST-40C US2: V&V integration of RapiCov US3: Process-Related Reuse via Management of Process Lines US4: Product-Related Reuse via Management of Process Lines US5: Compliance Management (generation of process-based arguments)

2.3. Evaluation Scope

Table 2 lists the different AMASS functionalities grouped by STOs (cf. Figure 1).

The second iteration of the AMASS platform is built upon the basic building functionalities (blue highlighted cells) already covered during the first iteration and it is enhanced by advanced functionalities (green highlighted cells). It must be mentioned that some of the functionalities are achieved by external tools (e.g. MORETO, OCRA).

During the second iteration, besides the Prototype P1 functionalities, some case studies have also evaluated the already existing Core Prototype basic functionalities, such as System component specification (CHESS), Assurance case specification (OpenCert), Evidence Management (OpenCert) and Compliance Management (EPF-Composer/OpenCert).

The support for "Architectural Patterns for assurance" functionality provided by the AMASS platform will be addressed in the next iteration (Prototype P2).

Table 2. Summary of the AMASS Prototype P1 functionalities

STO	Functionality Group	Description	Available tools
Architecture Driven Assurance	**System Component Specification**	This group provides features to allow the modelling of the system architecture specification, in particular, to allow the definition of components as reusable entities, and then the assembly of the components themselves, at any level of the hierarchical architecture, to build/decompose the system.	CHESS SAVONA (external) Papyrus/SysML MORETO (external)
	System Architecture Modelling for Assurance	This block contains the functionalities that are focused on the modelling of the system architecture to support the system assurance, which are: • Supporting the modelling of additional aspects (not already included in the system component specification), related to the system architecture, that are needed for system assurance. • Tracing the elements of the system architecture model to the assurance case. • Generating evidence for the assurance case from the system architecture model or from the analysis thereof. • Importing the system architecture model from other tools/languages.	Papyrus/CHESS CHESS with variability SAVONA(external) Enterprise Architect (external) MORETO (Enterprise Architect plug-in) (external)
	Architectural Patterns for Assurance	Support for architectural patterns management will be provided by Prototype P2.	For now, some support with an external tool: MORETO (Enterprise Architect plug-in) (external)
	Contract-based Design for Assurance	This block introduces the functionalities that support the contract-based design of the system architecture, which provides additional arguments and evidence for system assurance. These functionalities, also include: • Contracts specification, i.e., specification of components' assumptions and guarantees. • Contract-based reuse of components, i.e., a component reuse that is supported by checks on the contracts. • Generation of assurance arguments from the contract specification and validation.	CHESS + OCRA
	Activities supporting Assurance Case	This block contains the functionalities that are focused on enriching the assurance case with advanced analysis to support the evidence of the assurance case. These functionalities include:	OCRA (external) KM (external) nuXmv (external) xSAP (external)

		• Requirements formalisation into temporal logics. • Analysis of requirements' semantics based on their formalisation into temporal logics. • Analysis of requirements based on quality metrics. • Contract-based verification and analysis, i.e. exploiting contracts to verify the architectural decomposition, to perform compositional analysis, and to analyse the safety and reliability of the system architecture. • Automated Formal verification (model checking) of requirements on the system design. (e.g. nuXmv, DIVINE, NuSMV). • Model-based specification of fault-injection and analysis of faulty scenarios with simulation (Sabotage) or model checking (xSAP) (model-based safety analysis). • Other techniques (e.g. Component Fault Trees from SysML models) for Model-based safety analysis (e.g. Medini Analyze) • Document generation	ForReq (formalisation) System Quality Analyzer (SQA) Knowledge Manager (KM) Medini Analyze (external) Sabotage (external) (ongoing and planned for P2) AMT 2.0 (external) (ongoing and planned for P2) V&V Manager DIVINE, NuSMV, nuXmv, Looney, Acacia+ (externals) RapiCov
Multi-concern Assurance	**Assurance Case Specification**	This group manages argumentation information in a modular fashion. It also includes mechanisms to support compositional assurance and assurance patterns management.	OpenCert
	Dependability Assurance	This group contains the functionality for creating and structuring the multi-concern assurance case argumentation in an understandable and maintainable way. This includes argumentations targeting various dependability attributes with support of argumentation patterns.	OpenCert
	System Dependability Co-Analysis/Co-Assessment	This group provides functionalities for analysing different quality attributes while taking care of the inter-dependences between them. This is ideally realized by inherently combined Co-Analysis and Co-Assessment methods, which take care of the inter-dependencies within the method. On the other hand, multi-concern assurance can be implemented combining separate processes with mono-concern assurance methods by a workflow tool with a subsequent interaction point activity for treating the mutual dependencies between the quality attributes.	FMVEA (external) EPF-C+BVR ConcertoFLA (external) Papyrus SSE
	Contract-based Multi-concern	This group comprises functionalities which contribute to assurance for multiple concerns	CHESS OpenCert

	assurance	by two kinds of contracts: on the one hand, component contracts, which target more than one quality attribute. On the other hand, argument contracts, which provide a means for realizing a link between related assurance cases.	
Seamless Interoperability	**Evidence management**	This module manages the full lifecycle of evidence artefacts and evidence chains. This includes evidence traceability management and impact analysis.	OpenCert
	Tool Integration Management	This module enables the exchange of data between engineering/assurance tools, e.g. between the AMASS Tool Platform and other tools developed by the AMASS partners.	OSLC
	Collaborative Work Management	This module allows different users to work at the same time with the same pieces of data, supporting the interaction of the different users.	
	Tool Quality Assessment and Characterisation	This module supports the specification and management of tool quality needs for CPS assurance and certification. It is currently supported by the Compliance Management functionality for Cross/Intra-Domain Reuse.	
Cross/Intra-Domain Reuse	**Compliance Management**	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.	OpenCert EPF
	Reuse Assistant	The reuse assistance functionality concerns intra and cross-domain reuse of assurance and certification assets. This module supports users to understand whether reuse of the assurance assets is reasonable or determine what further assurance activities (engineering, V&V, or compliance activities) are required to justify compliance in the new scenario.	OpenCert
	Process-related reuse via management of variability at process level	Functionality related to the management of variability at process level. This functionality takes as input a process, which needs to be reconfigured, and the new selections, desired by the user. As outcome, this functionality generates a new valid re-configuration of the process.	EPF-Composer and BVR VSpec, Resolution, and Realisation editors (external)
	Product-related	Functionality related to the management of variability at product level. This functionality	EPF Composer BVR Tool

	reuse via management of variability at product level	takes as input a product (more specifically, an architectural specification given in CHESSML), which needs to be tailored/reconfigured, and the new selections, desired by the user. As outcome, this functionality generates a new valid re-configuration of the architectural specification.	Small GEO Vspec (external)
	Automatic generation of process-based arguments	This functionality is related to the generation of process-based arguments from process models. It supports the strengthening of the safety case via arguments that are aimed at explaining why a process is compliant.	OpenCert
	Automatic generation of product-based arguments	This functionality is related to the generation of product-based arguments from contract-based architectural specification. It supports the strengthening of the safety case via arguments aimed at showing why the product is expected to behave safely.	OpenCert

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 implementing AMASS Case Studies

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

The wide spectrum in the 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 focuses in general 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 is 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 the 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 measurable over successive projects. The first project is likely not to have much benefit as the work must be done once, but subsequent projects can benefit from it.

2.4.2. Timing for the Prototype P1 Setting

The data required as input for this deliverable was collected last year in D1.2 [2] and the partners have been working based on it since then.

The second release of the AMASS Prototype (P1) and the training provided by the tool developers was held two months before the submission deadline of this deliverable. Since the timing was quite tight, Use Case owners have done their job being in close relationship with the tool developers via point-to-point calls as well as group-calls aimed at speeding up the knowledge transfer related to the implementation of the topics in the tools and giving each other real-time feedback.

It should be noted that typically, prototypes always require a first sprint for understanding how to install and run properly the applications, and for detecting the problems. However, for this second release of the AMASS Prototype (P1), this sprint could be shorter than the first one (Core Prototype) because the bugs and the industrial expectations remarked in the first sprint were treated and solved. Despite this positive evolution of the core tools, given the richness of the second prototype (P1), an important challenge was identified: much more functionalities have been added with respect to the ones available for the Core Prototype, released last year.

As pointed out in Section 2.4.1, to achieve meaningful measurement results, ideally, the same project should be executed twice (with and without AMASS support) and the resources and time consumption compared. Given the numerous functionalities of the AMASS Platform, this goal is not easy to achieve and could involve a high cost.

Benchmarking will add consistency and extra information about the needs of the future potential markets in different application 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 Realisation

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 controlling 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 RTUs are one of the main elements in the control system due to the fact that they execute the commands received by the control centre, acting directly over the devices placed in the field site.

Security and safety aspects are one of the primary concerns for RTU 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 platform in the lifecycle of the RTU development process, providing assistance for assurance and certification with respect to the aforementioned standards.

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

Two different usage scenarios are defined in this case study:

- US1: Managing compliance with IEC 61508, IEC 62443 and IEC 62351
- US2: Perform safety and security co-assessment

On the one hand, US1 focuses on the assessment of the RTU processes. The target for US1 is to check the compliance of the RTU processes with respect to safety and security standards. The information obtained by this scenario is very useful for the industrial partner (Schneider Electric) to identify GAPs (between what we do and we must do) and improve the RTU processes in order to align with the standards and assure the RTU product.

On the other hand, US2 addressed the RTU product assurance. This scenario is more related to the safety and security co-engineering by modelling the RTU product requirements, and evaluating the product integrity respect to safety and security aspects. Based on the relevant standards, the scenario has the objective to do the safety and security co-assessment of the RTU, analysing the requirements and identifying safety hazards, security threats and their interrelations.

The final target for both scenarios is to reduce certification time and cost for the RTU using the AMASS tools.

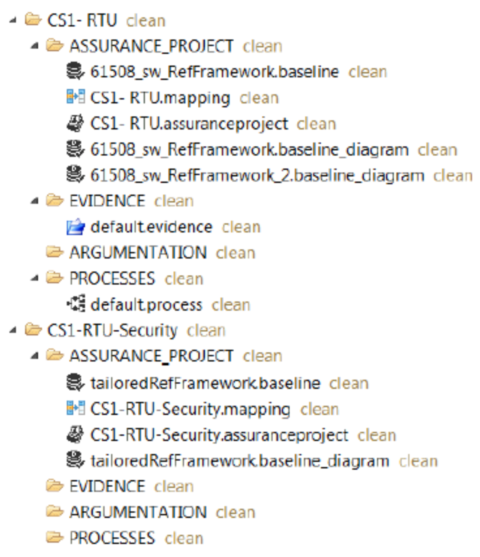
3.1.2. US1: Managing compliance with IEC 61508, IEC 62443 and IEC 62351

US1 is related to process assurance, i.e. to ensure that the RTU development process follows a given set of recommendations from the targeted standards.

The goal of this usage scenario is to enable easier understanding of these industry standards, easier checking for compliance and easier adaptation and reuse of assurance assets.

Assurance Project Creation

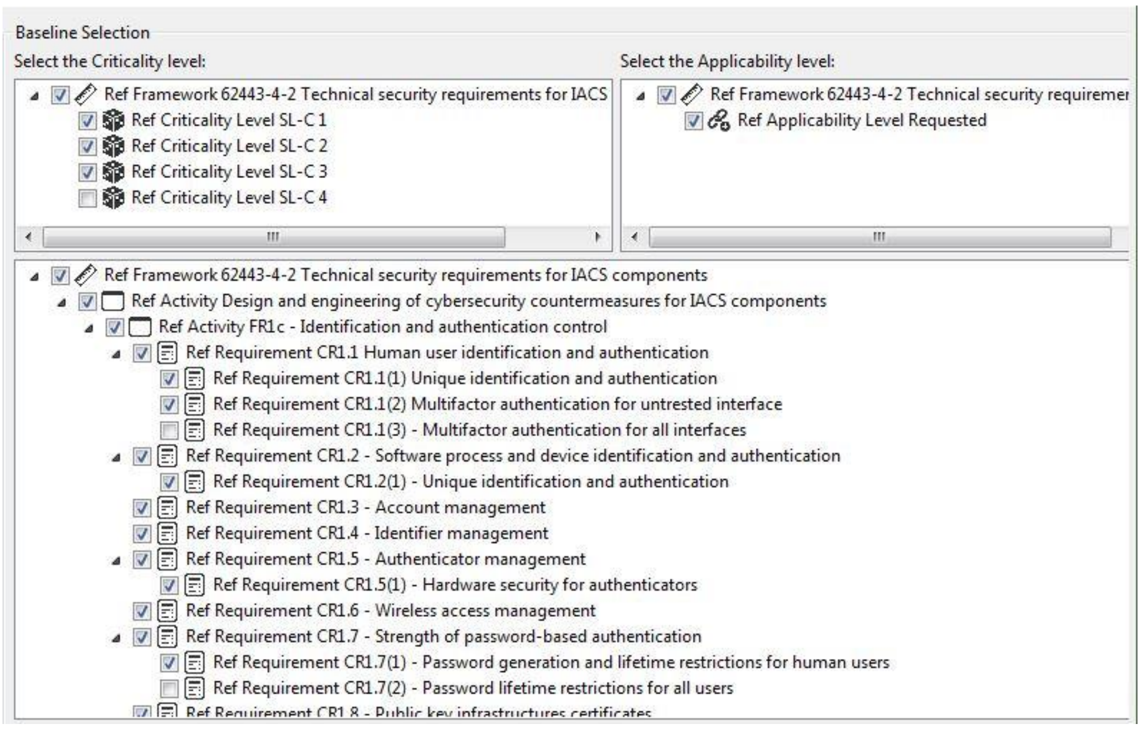
Respect to this process, in the first iteration, two assurance projects were created with the OpenCert tool: one for RTU Safety assurance (based on the standard IEC 61508) and the other for RTU Security assurance (based on the standard IEC 62443).



```
CS1- RTU clean
  ASSURANCE_PROJECT clean
    61508_sw_RefFramework.baseline clean
    CS1- RTU.mapping clean
    CS1- RTU.assuranceproject clean
    61508_sw_RefFramework.baseline_diagram clean
    61508_sw_RefFramework_2.baseline_diagram clean
  EVIDENCE clean
    default.evidence clean
  ARGUMENTATION clean
  PROCESSES clean
    default.process clean
CS1-RTU-Security clean
  ASSURANCE_PROJECT clean
    tailoredRefFramework.baseline clean
    CS1-RTU-Security.mapping clean
    CS1-RTU-Security.assuranceproject clean
    tailoredRefFramework.baseline_diagram clean
  EVIDENCE clean
  ARGUMENTATION clean
  PROCESSES clean
```

Figure 3. RTU assurance projects created

In the second iteration, new features regarding “Criticality level” and “Applicability level” were included in the assurance projects. These functions allow us to select the requirements according to the security level. In this case study, we have selected SIL-2 for safety and SL-3 for security.



Baseline Selection

Select the Criticality level:

- ✓ Ref Framework 62443-4-2 Technical security requirements for IACS
  - ✓ Ref Criticality Level SL-C 1
  - ✓ Ref Criticality Level SL-C 2
  - ✓ Ref Criticality Level SL-C 3
  - ☐ Ref Criticality Level SL-C 4

Select the Applicability level:

- ✓ Ref Framework 62443-4-2 Technical security requiremer
  - ✓ Ref Applicability Level Requested

- ✓ Ref Framework 62443-4-2 Technical security requirements for IACS components
  - ✓ Ref Activity Design and engineering of cybersecurity countermeasures for IACS components
    - ✓ Ref Activity FR1c - Identification and authentication control
      - ✓ Ref Requirement CR1.1 Human user identification and authentication
        - ✓ Ref Requirement CR1.1(1) Unique identification and authentication
        - ✓ Ref Requirement CR1.1(2) Multifactor authentication for untrested interface
        - ☐ Ref Requirement CR1.1(3) - Multifactor authentication for all interfaces
      - ✓ Ref Requirement CR1.2 - Software process and device identification and authentication
        - ✓ Ref Requirement CR1.2(1) - Unique identification and authentication
      - ✓ Ref Requirement CR1.3 - Account management
      - ✓ Ref Requirement CR1.4 - Identifier management
      - ✓ Ref Requirement CR1.5 - Authenticator management
        - ✓ Ref Requirement CR1.5(1) - Hardware security for authenticators
      - ✓ Ref Requirement CR1.6 - Wireless access management
      - ✓ Ref Requirement CR1.7 - Strength of password-based authentication
        - ✓ Ref Requirement CR1.7(1) - Password generation and lifetime restrictions for human users
        - ☐ Ref Requirement CR1.7(2) - Password lifetime restrictions for all users
      - ✓ Ref Requirement CR1.8 - Public key infrastructures certificates

Figure 4. Assurance project creation – Criticality level and Applicability level

3.1.2.1. STO2 Multi-concern Assurance

In addition, another new functionality was checked for the security project. During the creation of the assurance project, the argumentation diagram of the OpenCert tool was included.

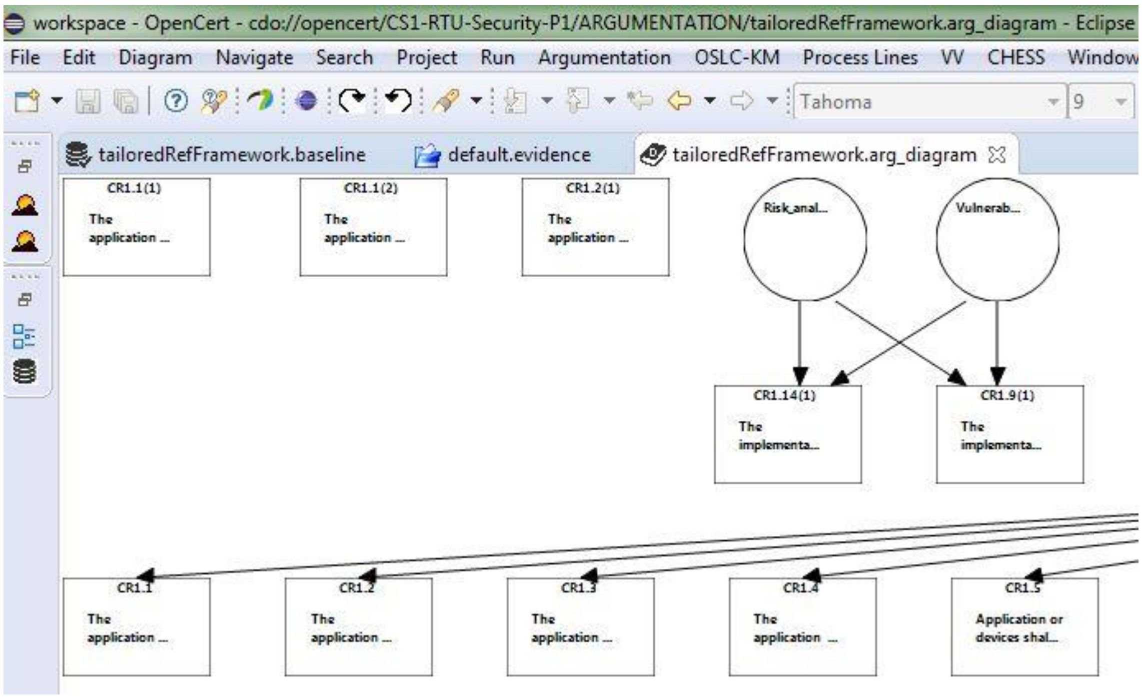

workspace - OpenCert - cdo://opencert/CS1-RTU-Security-P1/ARGUMENTATION/tailoredRefFramework.arg_diagram - Eclipse
File Edit Diagram Navigate Search Project Run Argumentation OSLC-KM Process Lines VV CHESS Window
tailoredRefFramework.baseline
default.evidence
tailoredRefFramework.arg_diagram
CR1.1(1) The application ...
CR1.1(2) The application ...
CR1.2(1) The application ...
Risk_anal...
Vulnerab...
CR1.14(1) The implementa...
CR1.9(1) The implementa...
CR1.1 The application ...
CR1.2 The application ...
CR1.3 The application ...
CR1.4 The application ...
CR1.5 Application or devices shal...

Figure 5. Assurance project creation – Argumentation diagram

Two Assurance Projects (safety and security) were defined for RTU where the baseline models (instances of reference frameworks for specific assurance) were created.

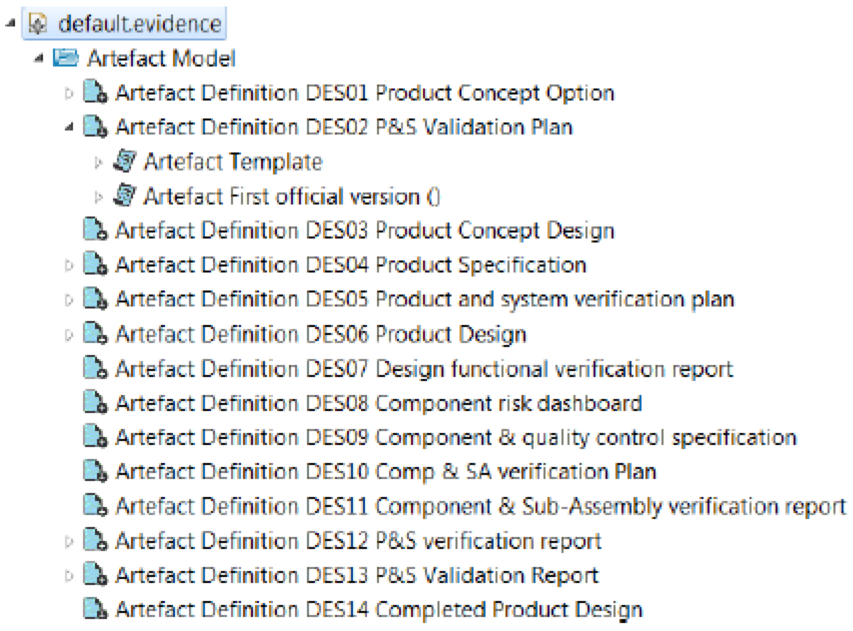
Table 3. CS1-Multi-concern Assurance: US1-Assurance Project Management (Create Assurance Project)

Realisation Scenario	**Assurance Project Management (Create Assurance Project) and Argumentation Diagram**
Scope	In iteration 1: • Creation of two Assurance Projects, one for RTU Safety assurance and the other for RTU Security assurance. In iteration 2: • Two new features “Criticality level” and “Applicability level” and Argumentation diagram included. Furthermore, and argumentation diagram has been created out of the security project.
Tool Settings	OpenCert Tools: Assurance Project Management Editor
Participants	• Data Analysis: TLV • Tool User: TLV, TEC
Activities realised	1. Create assurance projects for RTU Safety and for RTU Security. 2. When creating the Baseline models, specify the activities we focus on for the prototype benchmarking.
Usage Decisions	None
Expected Results	• Assurance Project structure and Baseline model for RTU Safety • Assurance Project structure and Baseline model for RTU Security
Conclusions	Assurance project management validated for Prototype P1.

3.1.2.2. STO3 Seamless Interoperability

Evidence Management

In the first iteration, a subset of evidence documents was included respect to the safety assurance project.



default.evidence
Artefact Model
Artefact Definition DES01 Product Concept Option
Artefact Definition DES02 P&S Validation Plan
Artefact Template
Artefact First official version ()
Artefact Definition DES03 Product Concept Design
Artefact Definition DES04 Product Specification
Artefact Definition DES05 Product and system verification plan
Artefact Definition DES06 Product Design
Artefact Definition DES07 Design functional verification report
Artefact Definition DES08 Component risk dashboard
Artefact Definition DES09 Component & quality control specification
Artefact Definition DES10 Comp & SA verification Plan
Artefact Definition DES11 Component & Sub-Assembly verification report
Artefact Definition DES12 P&S verification report
Artefact Definition DES13 P&S Validation Report
Artefact Definition DES14 Completed Product Design

Figure 6. Evidences for safety assurance project

In the second iteration, we have focused on the evidences creation for the security assurance project. Those evidences can be generated and linked from the architecture-driven assurance and multi concern-assurance (FMVEA co-analysis artefact) approaches explained in Usage Scenario 2.

Two evidence models were created for respective assurance case projects.

Table 4. CS1-Seamless Interoperability: US1-Evidence Management

Realisation Scenario	Evidence Management
Scope	In iteration 1: • Evidence documents for the safety assurance project included In iteration 2: • Evidence documents for the security assurance project included
Tool Settings	OpenCert Tools: Evidence Management Editor. SVN repository to store actual evidence documents.
Participants	• Data Analysis: TLV • Tool User: TLV, TEC • Results Analysis: TLV
Activities realised	1. Create artefact model for RTU Security 2. Create SVN repository for RTU Security 3. Collect evidence documents into the SVN repository for RTU Security 4. Specify characteristics of RTU Security artefacts 5. Collect evidence documents into the SVN repository for RTU 6. Use cross-domain functionality to reuse Artefact models from RTU Safety project in RTU Security project 7. Complete any evaluation of the artefact elements in the assurance project.
Usage Decisions	Reuse of some artefacts.
Expected Results	Evidence model and artefact repository for RTU Security.
Conclusions	Evidence management validated for Prototype P1.

3.1.2.3. STO4 Cross Intra-domain reuse

Standards Models Creation

In the first iteration, the IEC 61508 Part 3, -which applies to any software forming part of a safety-related system, was totally modelled using OpenCert. Besides, the modelling of the security standard "IEC 62443: 4-2 Technical security requirements for IACS components" was started. During the second iteration, the modelling of IEC 62443:4-2 has been finished. The structure of these standards, as well as the core concepts such as: phases, activities, artefacts, requirements and criticality levels were analysed. After that, a reference framework diagram was created for each standard.

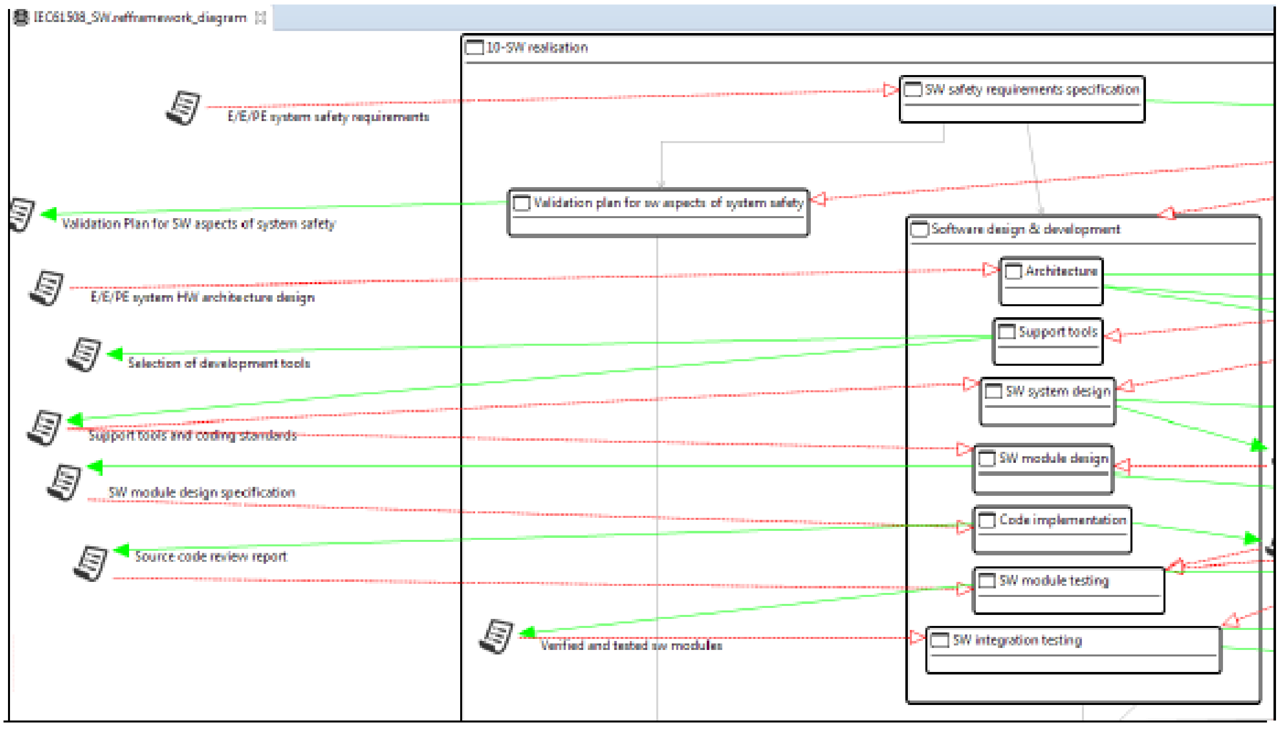

IEC61508_SW.refframework_diagram
10-SW realisation
E/E/PE system safety requirements
SW safety requirements specification
Validation plan for sw aspects of system safety
Validation Plan for SW aspects of system safety
Software design & development
E/E/PE system HW architecture design
Architecture
Support tools
Selection of development tools
SW system design
Support tools and coding standards
SW module design
SW module design specification
Code implementation
Source code review report
SW module testing
SW integration testing
Verified and tested sw modules

Figure 7. IEC61508 – Part 3 reference framework diagram (OpenCert)

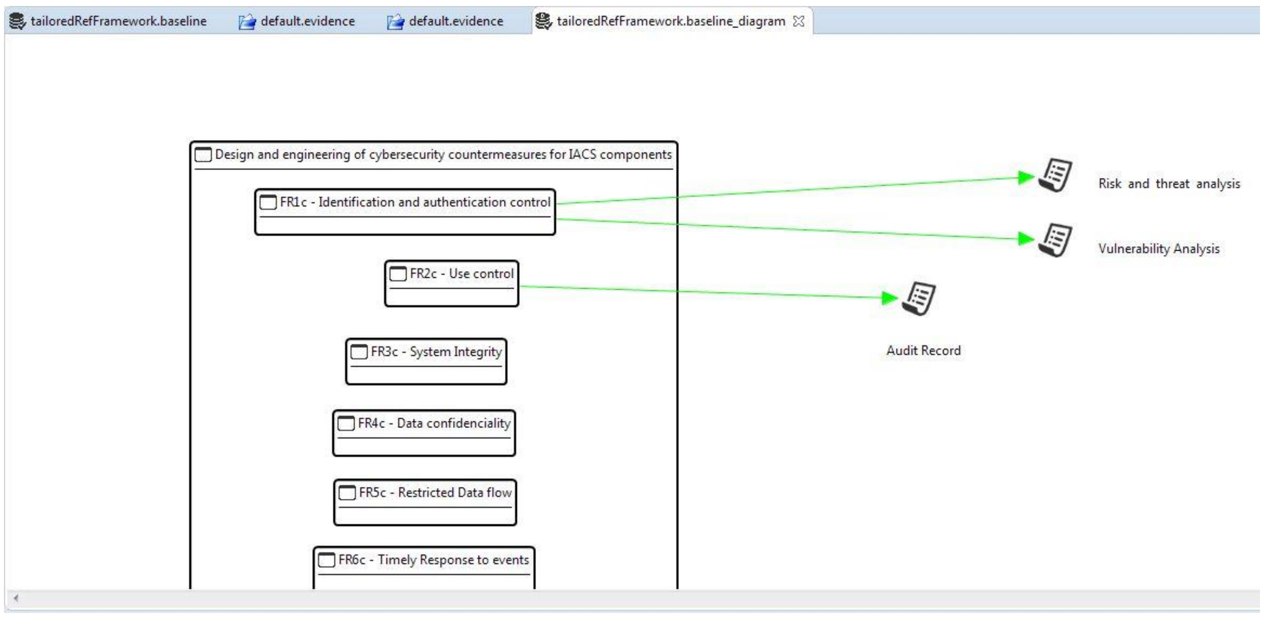
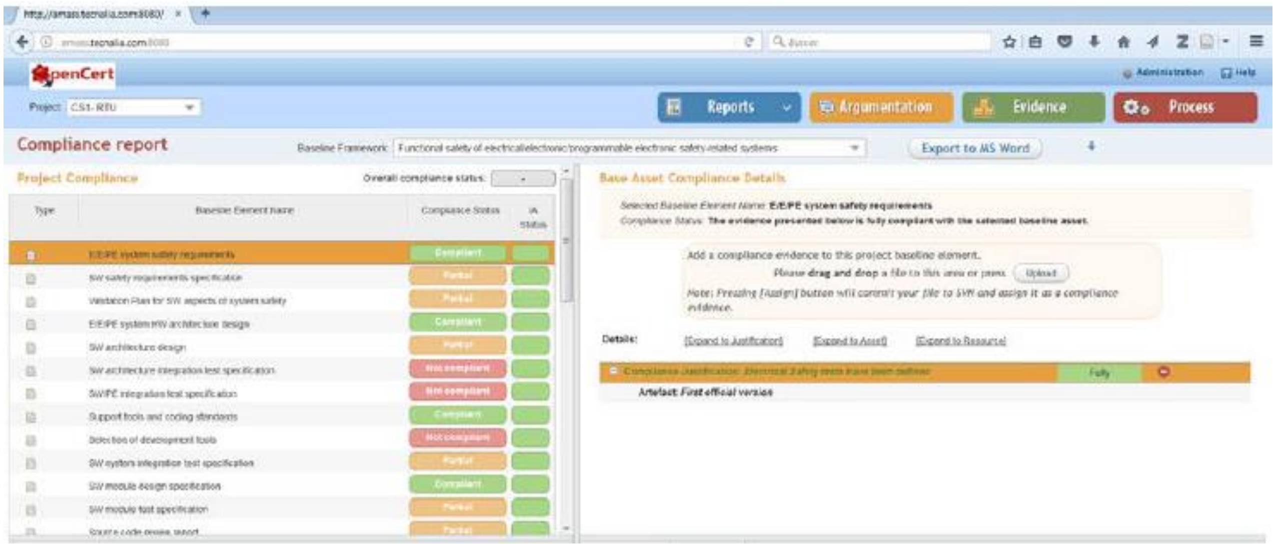

tailoredRefFramework.baseline
default.evidence
default.evidence
tailoredRefFramework.baseline_diagram
Design and engineering of cybersecurity countermeasures for IACS components
FR1c - Identification and authentication control
FR2c - Use control
FR3c - System Integrity
FR4c - Data confidenciality
FR5c - Restricted Data flow
FR6c - Timely Response to events
Risk and threat analysis
Vulnerability Analysis
Audit Record

Figure 8. IEC62443 – Part 4.2 initial reference framework diagram (OpenCert)

With respect to the standard IEC 62351, it was decided to address it in the third iteration (P2), focusing the second iteration efforts in the standard IEC 62443 (more relevant for the RTU security analysis). In the same way, the modelling of IEC 62443-4-1 will be deployed in the third iteration (Prototype P2).

Compliance Management

During the first iteration, the compliance maps of the safety and security assurance projects were created. The results were analysed using the OpenCert clients and the web application.


Project: CS1-RTU

Compliance report

Baseline Framework: Functional safety of electrical/electronic/programmable electronic safety-related systems

Export to MS Word

Project Compliance

Overall compliance status:

Type	Baseline Element name	Compliance Status	IA Status
	E/E/PE system safety requirements	Compliant	
	SW safety requirements specification	Partial	
	Validation Plan for SW aspects of system safety	Partial	
	E/E/PE system HW architecture design	Compliant	
	SW architecture design	Partial	
	SW architecture integration test specification	Not compliant	
	SW/PE integration test specification	Not compliant	
	Support tools and coding standards	Compliant	
	Selection of development tools	Not compliant	
	SW system integration test specification	Partial	
	SW module design specification	Compliant	
	SW module test specification	Partial	
	Source code review report	Partial	

Base Asset Compliance Details

Selected Baseline Element Name: **E/E/PE system safety requirements**

Compliance Status: **The evidence presented below is fully compliant with the selected baseline asset.**

Add a compliance evidence to this project baseline element.

Please **drag and drop** a file to this area or press Upload

Note: Pressing [Assign] button will commit your file to SVN and assign it as a compliance evidence.

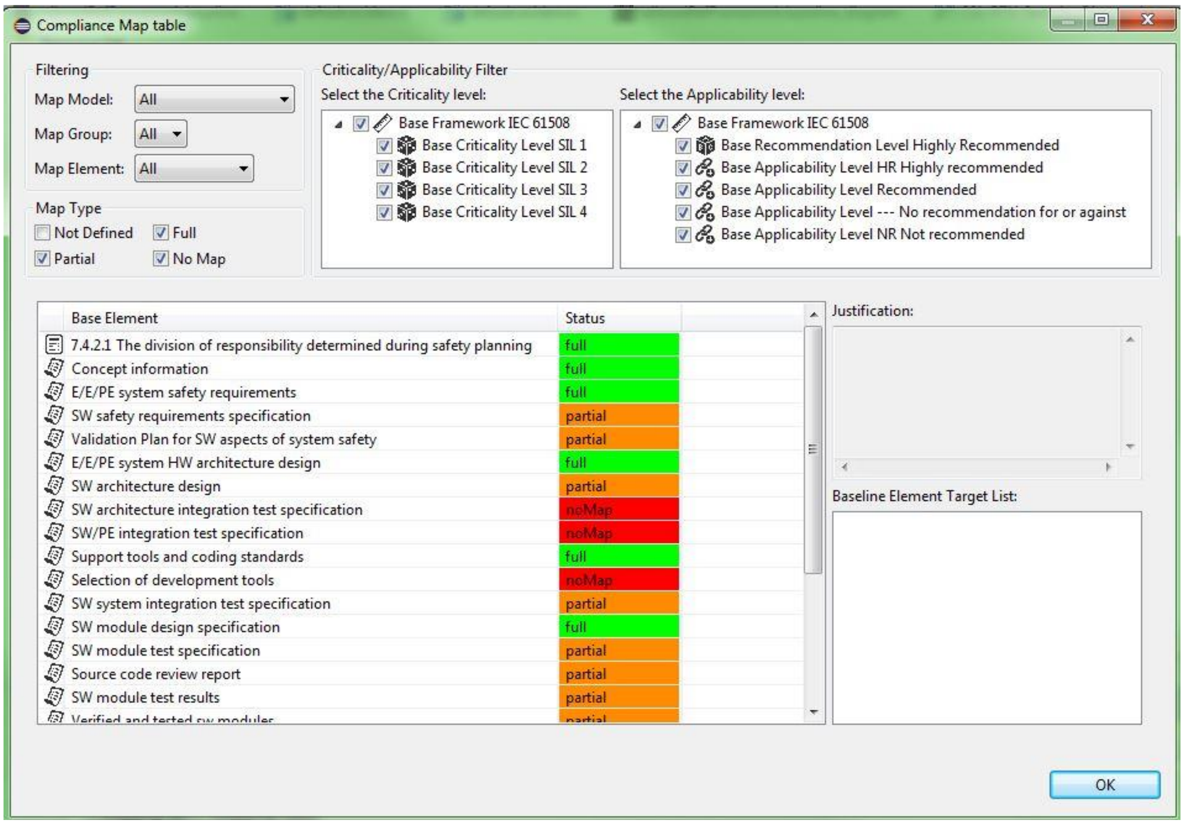
Details: [Expand to Justification] [Expand to Asset] [Expand to Resource]

Compliance Justification: [illegible] | Fully

Artefact: ***First official version***

Figure 9. Compliance report for safety assurance project

In the second iteration, the new function "Mapping Table" was used in order to obtain information about the compliance using different filters:


Compliance Map table

Filtering

Map Model: All

Map Group: All

Map Element: All

Map Type

- [ ] Not Defined
- [x] Full
- [x] Partial
- [x] No Map

Criticality/Applicability Filter

Select the Criticality level:

- [x] Base Framework IEC 61508
  - [x] Base Criticality Level SIL 1
  - [x] Base Criticality Level SIL 2
  - [x] Base Criticality Level SIL 3
  - [x] Base Criticality Level SIL 4

Select the Applicability level:

- [x] Base Framework IEC 61508
  - [x] Base Recommendation Level Highly Recommended
  - [x] Base Applicability Level HR Highly recommended
  - [x] Base Applicability Level Recommended
  - [x] Base Applicability Level --- No recommendation for or against
  - [x] Base Applicability Level NR Not recommended

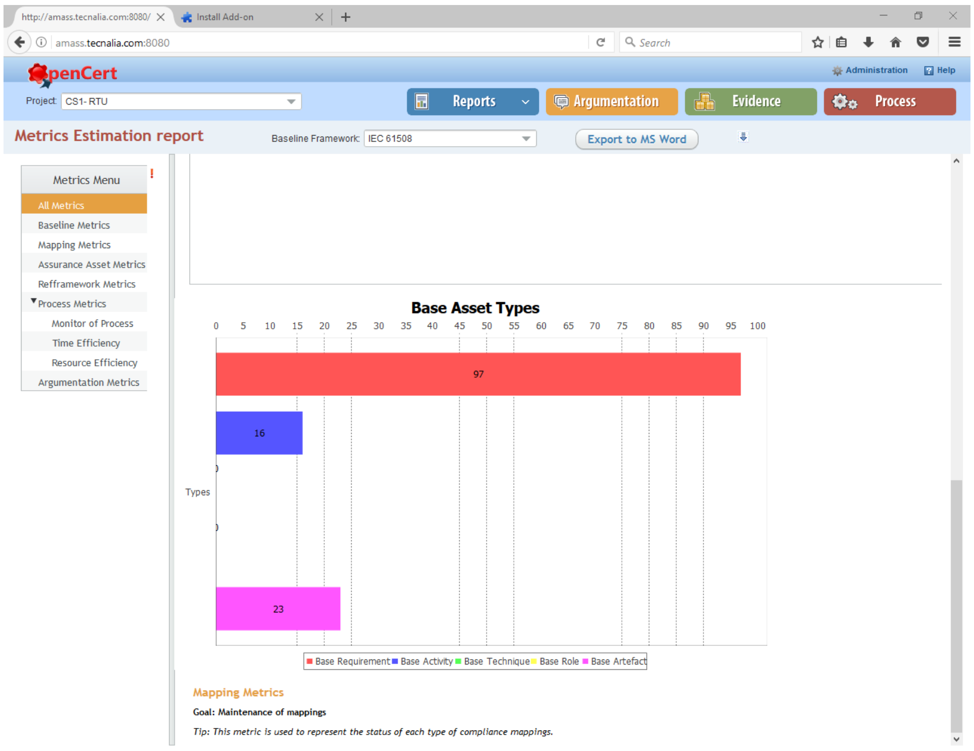
Base Element	Status
7.4.2.1 The division of responsibility determined during safety planning	full
Concept information	full
E/E/PE system safety requirements	full
SW safety requirements specification	partial
Validation Plan for SW aspects of system safety	partial
E/E/PE system HW architecture design	full
SW architecture design	partial
SW architecture integration test specification	noMap
SW/PE integration test specification	noMap
Support tools and coding standards	full
Selection of development tools	noMap
SW system integration test specification	partial
SW module design specification	full
SW module test specification	partial
Source code review report	partial
SW module test results	partial
Verified and tested sw modules	partial

Justification:

Baseline Element Target List:

Figure 10. Compliance map table

Using the web application, some improvements in compliance reports have been checked. In particular, the new charts for metrics.

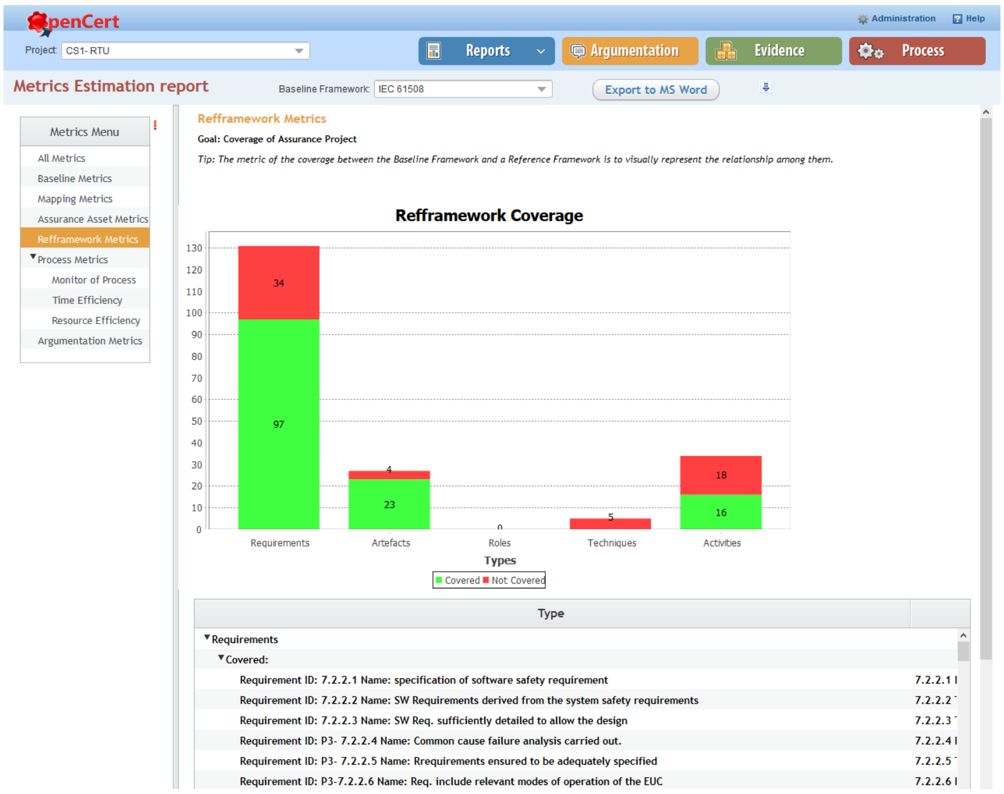


Mapping Metrics

Goal: Maintenance of mappings

Tip: This metric is used to represent the status of each type of compliance mappings.

Figure 11. New metrics for the safety assurance project (1)



Refframework Metrics

Goal: Coverage of Assurance Project

Tip: The metric of the coverage between the Baseline Framework and a Reference Framework is to visually represent the relationship among them.

Type	
▼Requirements	
▼Covered:	
Requirement ID: 7.2.2.1 Name: specification of software safety requirement	7.2.2.1
Requirement ID: 7.2.2.2 Name: SW Requirements derived from the system safety requirements	7.2.2.2
Requirement ID: 7.2.2.3 Name: SW Req. sufficiently detailed to allow the design	7.2.2.3
Requirement ID: P3- 7.2.2.4 Name: Common cause failure analysis carried out.	7.2.2.4
Requirement ID: P3- 7.2.2.5 Name: Rrequirements ensured to be adequately specified	7.2.2.5
Requirement ID: P3-7.2.2.6 Name: Req. include relevant modes of operation of the EUC	7.2.2.6

Figure 12. New metrics for the safety assurance project (2)

Table 5 illustrates the modelled two standards (IEC 61508 and IEC 62443) using OpenCert tool.

Table 5. CS1-Cross Intra-domain reuse: US1-Compliance Management (including Standard Model Creation)

Realisation Scenario	Compliance Management (including Standard Model Creation)
Scope	In iteration 1: • IEC 61508 Part 3 (safety) standard was totally modelled • IEC 62443 Part 4.2 (security) standard was partially modelled • Compliance maps of the safety and security assurance projects were created. In iteration 2: • IEC 62443 Part 4.2 (security) standard was totally modelled. • New function "Mapping Table" was used in order to obtain information about the compliance using different filters. Regarding the standard IEC 62351, it will be addressed in the third iteration.
Tool Settings	OpenCert Tools: Standards Editor, Assurance Project Management and Compliance Reporter Web Client.
Participants	• Data Analysis: TLV • Tool User: TLV, TEC • Results Analysis: TLV, TEC
Activities realised	Respect to Standard Model Creation: 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. 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). 3. Validate the interpretations by sharing the reference framework models with other safety and security experts. 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. Respect to Compliance Management: 1. Specify compliance maps for Requirements and Artefacts in both Baseline models: RTU Safety and RTU Security. 2. Analyse compliance accomplishment and gaps for both assurance projects. 3. Generate the compliance report for both assurance projects.
Usage Decisions	We will not use EPF for modelling the targeted Standards (reference frameworks). Hence, the Standards models are created from scratch.
Expected Results	• Reference Framework model for IEC 62351 • Reference Framework model for IEC 62443-4.2 • Conceptual knowledge to propose some support to filter by Safety Integrity Level (SIL) and Security Level (SL) • Compliance report for IEC 61508 • Compliance report for IEC 62443
Conclusions	Compliance Management validated for Prototype P1.

3.1.3. US2: 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[1] for safety and IEC 62351[2], IEEE 1686[3], and IEC 62443[4] for cybersecurity. After a RTU analysis, it was decided for the first iteration to focus on IEC 62443 with the option to include other standards in the second iteration depending on the need. In this sense, the ISASecure EDSA certification was used 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 intended to explore the AMASS approach for reducing certification time and cost leveraging reusable artefacts (e.g. evidence).

In the first iteration (see D1.4 [4]), the MORETO tool was introduced for model-based safety & security product requirements management with its baseline software platform architecture and some example security requirements for the configuration of devices.

MORETO, as an Enterprise Architect plugin, is an external tool to AMASS that has showed enough reliability and flexibility to model safety & security requirements applied to the RTU of the CS1. In order to integrate MORETO with the AMASS platform, the SysML output facility provides the necessary interfaces for that purpose; however, adaptations are necessary to be performed due to slightly different subsets of the modelling language between Enterprise Architect and the AMASS internal format (discussions about this topic are ongoing as of March 2018).

In this second iteration, the cybersecurity standard IEEE 1686 was implemented in MORETO and applied to the RTU, while the other controls are also further on treated on the basis of IEC 62443.

A specific novelty was the enhancement of MORETO by an automatic security requirements generation feature, which was performed successfully for the RTU. This new feature allows, based on a target security level, to enumerate the missing security requirements in a given modelled configuration.

Appendix B: MORETO gives an overview of the MORETO tool and its new features.

MORETO uses two different security standards. In the first hand is IEC 62443-4-2, used for industrial automation and control systems, and applied to cover security gaps of the network components. In the second hand, IEEE 1686 security standard is used regarding the access, operation, and configuration of the RTU device.

Figure 13 shows a simple substation, which contains different levels of structure. The top level is the control centre which is monitoring all activities in the other levels. The data comes from/ to the Operation and Engineering stations throughout communication elements (switches and routers). The RTU-CPU gets all the information coming from the Process Level (through acquisition modules) and sends it to the Station Level, so it can be processed by the Operation Station and sent also to the SCADA system located outside of the substation.

[1] IEC 61508 Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems

[2] IEC 62351 is an industry standard aimed at improving security in automation systems in the power system domain

[3] IEEE 1686 Standard for Intelligent Electronic Devices Cyber Security Capabilities

[4] IEC 62443 Industrial Network and System Security

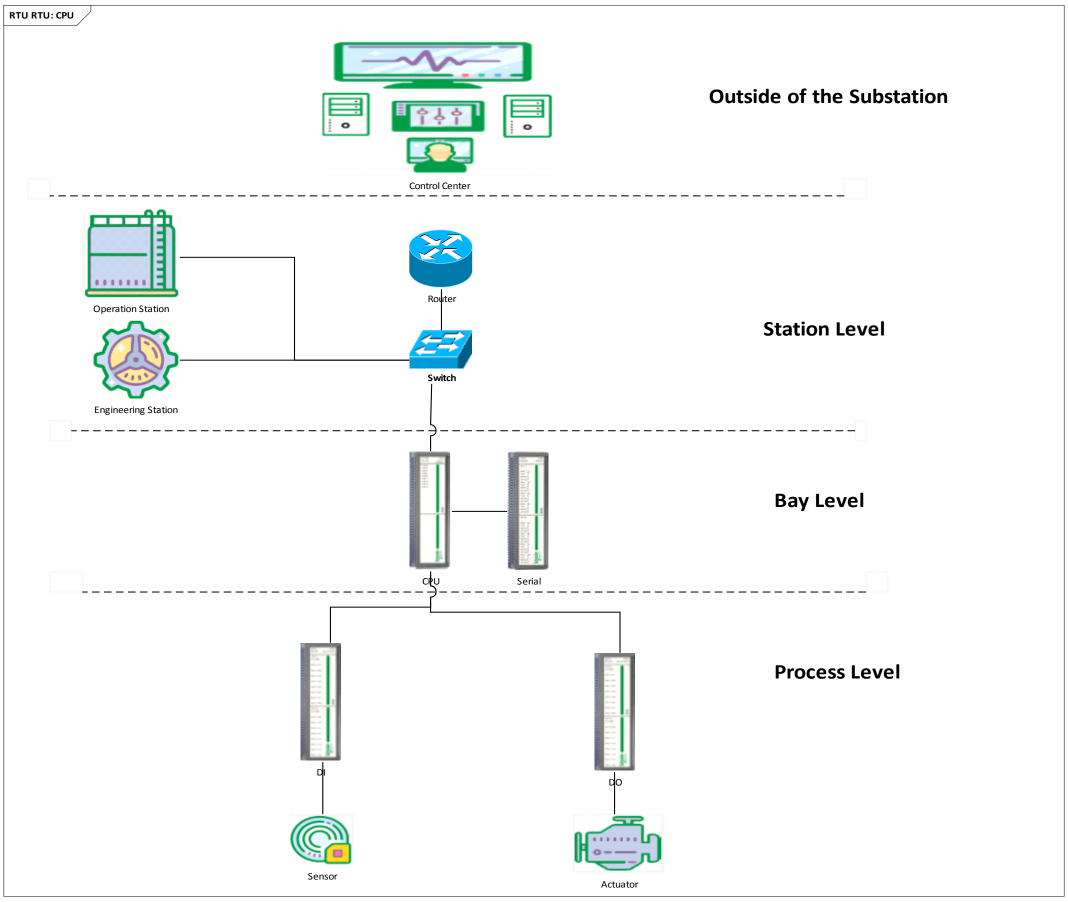


Figure 13. Substation example

Figure 13 depicts how the generation of security requirements is performed. By firing the "Security Requirement Generation" service, the user will get a list of security requirements for the following elements:

1- Switch Device:

Figure 14 shows a list of security requirements which is generated automatically by MORETO toolbox based on IEC 62443-4-2.

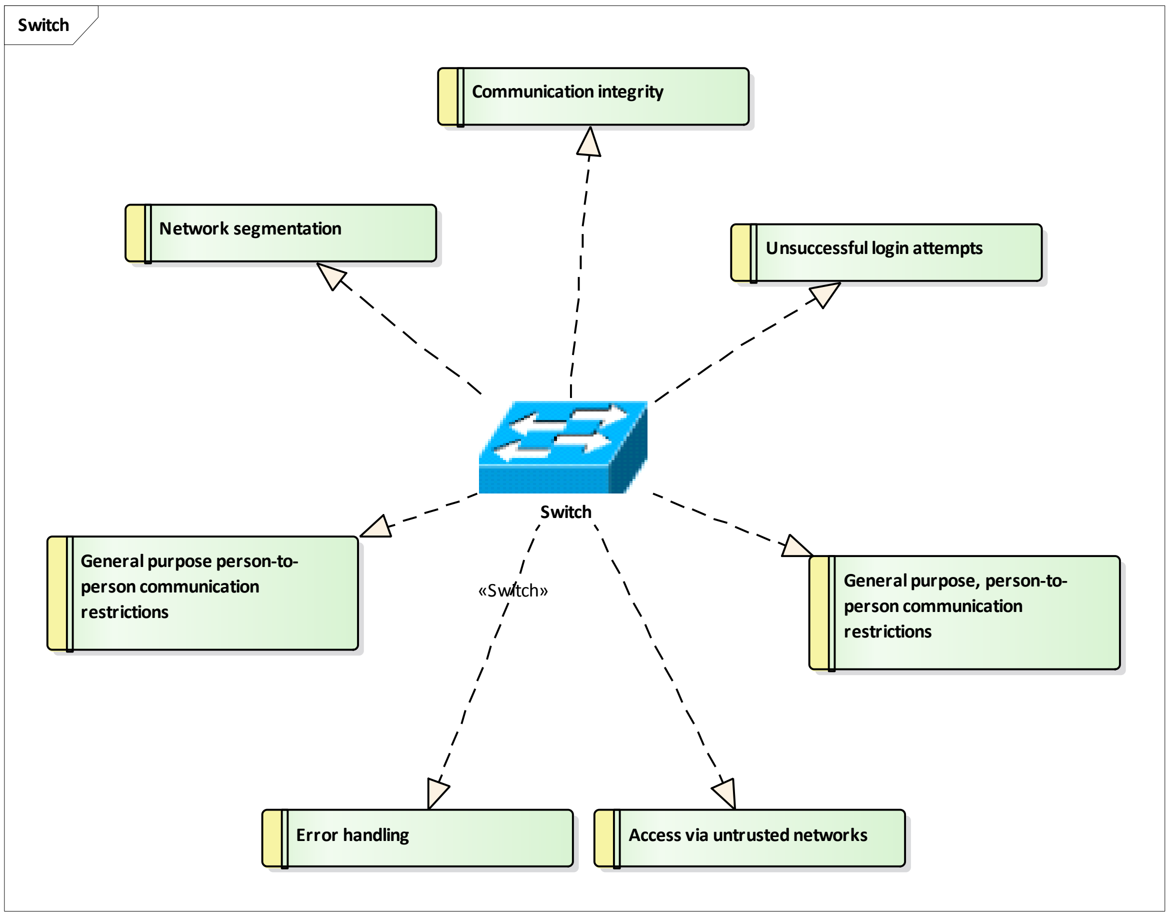

Switch
Communication integrity
Network segmentation
Unsuccessful login attempts
Switch
«Switch»
General purpose person-to-person communication restrictions
General purpose, person-to-person communication restrictions
Error handling
Access via untrusted networks

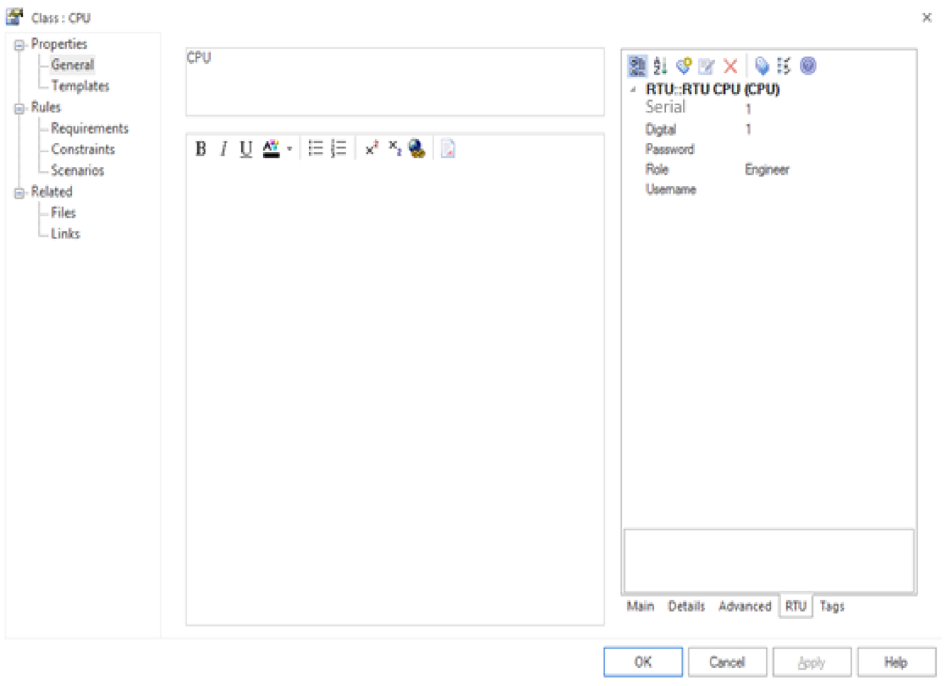
Figure 14. Switch Requirements

2- Router Device

Similarly, a list of IEC 62443-4-2 security requirements has been generated automatically by the MORETO toolbox.

3- RTU Unit

The security generation process of the RTU device is based on IEEE 1686. This process is built on the configuration of the RTU device. For example, the RTU-CPU device in Figure 13 has a set of values which are configured initially by the user to define the number of serial and digital RTU devices connected to the CPU. Likewise, the value of the role is critical and defines the access control privileges of the user. Figure 15 gives a simple description of the initial configuration values of the RTU device.



CPU

RTU::RTU CPU (CPU)

Serial	1
Digital	1
Password	
Role	Engineer
Username	

Figure 15. Configuration

Figure 15 shows some configuration values of the RTU-CPU device in Figure 13. MORETO generates a list of IEEE1686 security standards based on the following values:

- MORETO generates a list of security requirements for the serial and digital communications because their values are equal to 1.
- Generates security requirements for the username and the password; in this example, the values are missing.
- The RTU-CPU has seven different roles; the full privilege is the Admin. In this case, the Engineer role is chosen, so MORETO generates security requirements regarding that role.

Figure 16 shows a part of the generated security requirements of the RTU unit based on the IEEE 1686. MORETO generates security requirements regarding the RTU configuration parameters (i.e. password, username, role, analogue, and digital).

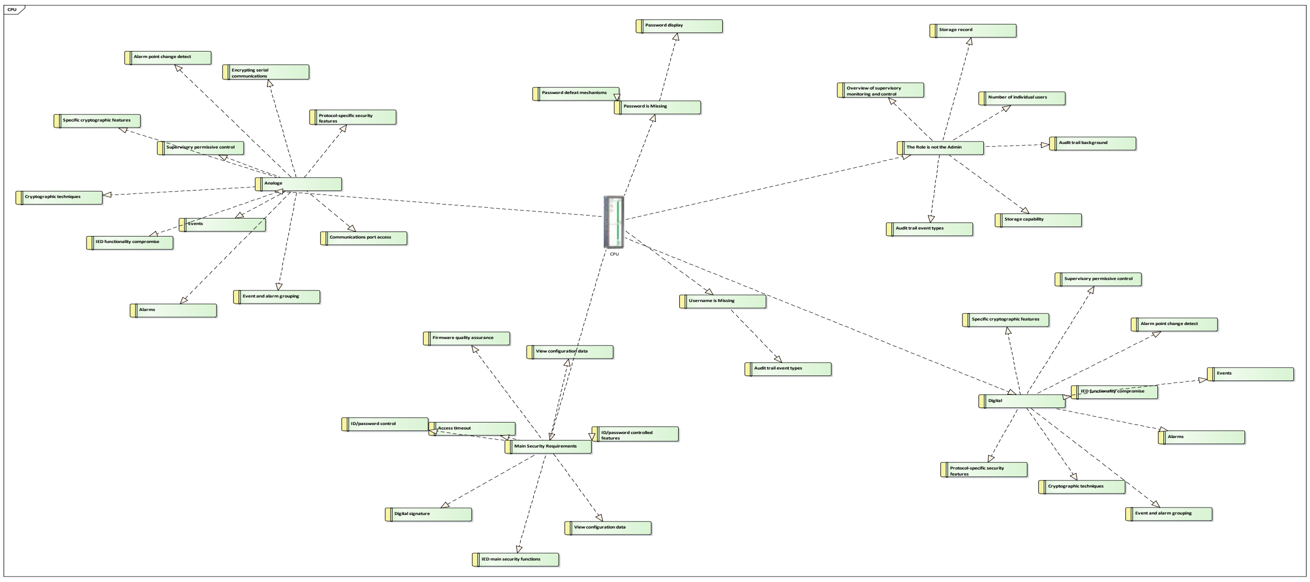

CPU
Alarm point change detect
Encrypting serial communications
Specific cryptographic features
Protocol-specific security features
Supervisory permissive control
Analoge
Cryptographic techniques
Events
IED functionality compromise
Communications port access
Alarms
Event and alarm grouping
Password display
Password defeat mechanisms
Password is Missing
CPU
Storage record
Overview of supervisory monitoring and control
Number of individual users
The Role is not the Admin
Audit trail background
Audit trail event types
Storage capability
Username is Missing
Audit trail event types
Supervisory permissive control
Specific cryptographic features
Alarm point change detect
Events
IED functionality compromise
Digital
Alarms
Protocol-specific security features
Cryptographic techniques
Event and alarm grouping
Firmware quality assurance
View configuration data
ID/password control
Access timeout
ID/password controlled features
Main Security Requirements
Digital signature
View configuration data
IED main security functions

Figure 16. Security requirements of the RTU unit based on the IEEE1886

Based on the values of these parameters, MORETO is able to generate a list of security requirements which can cover the security flaws of the given values in a category or sub-tree form. For example, the RTU in this Use-case is connected with a Digital and Analogue device. So, an example, the following Figure 17 shows the subtree of the Analogue security requirements generated under a sub-under or category named "Analogue" (cf. upper-left corner of Figure 16).

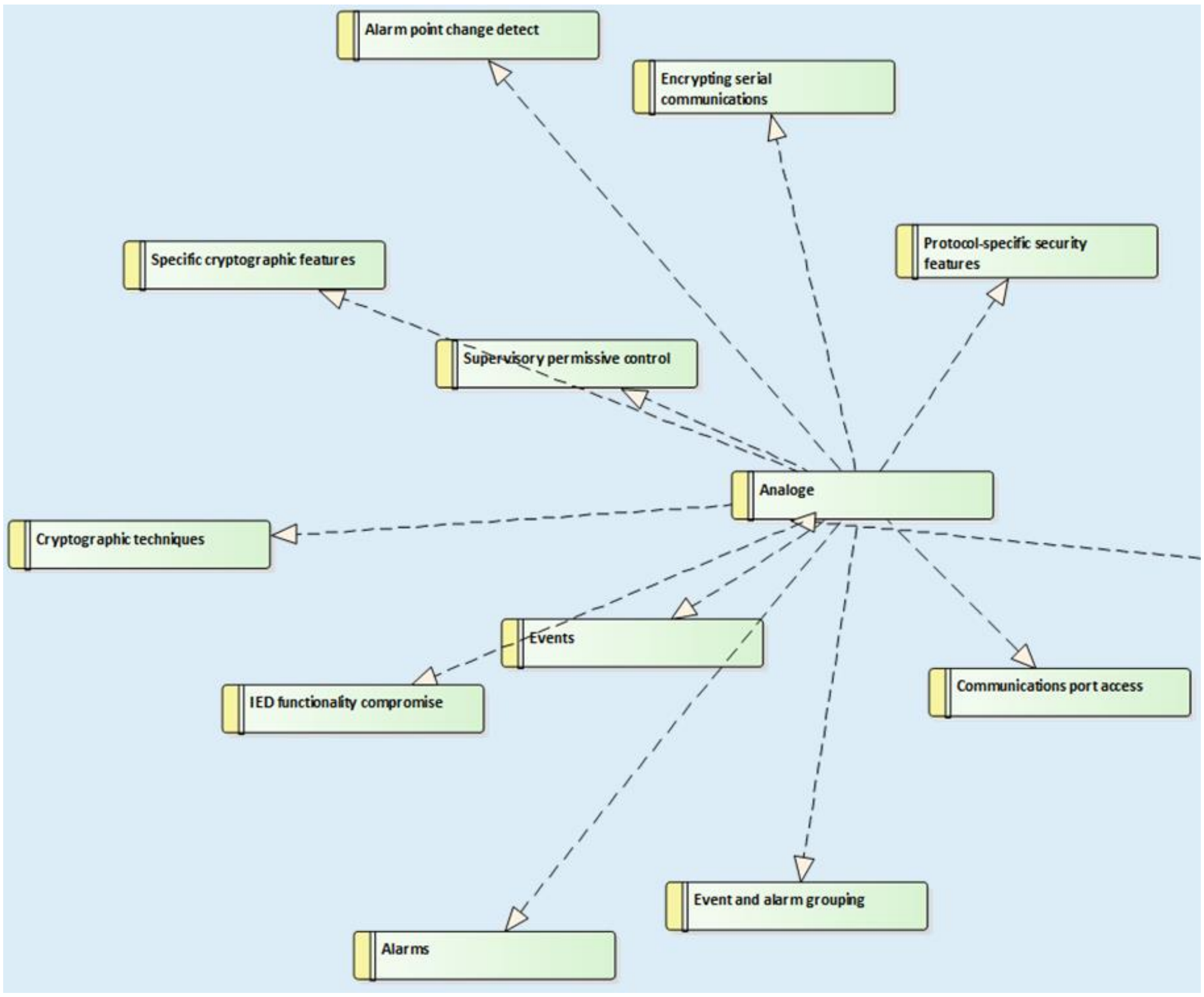

Alarm point change detect
Encrypting serial communications
Specific cryptographic features
Protocol-specific security features
Supervisory permissive control
Analoge
Cryptographic techniques
Events
IED functionality compromise
Communications port access
Event and alarm grouping
Alarms

Figure 17. The subtree of the Analogue security requirements

3.1.3.1. STO1 Architecture-Driven Assurance

The features for Architecture-Driven Assurance address a model-based safety & security product requirement management. They were provided in the first iteration and described in D1.4 [4], having significantly been enhanced since then. The tool MORETO was tailored to support the full range of IEC 62443-4-2 and the automatic generation of security requirements has been implemented. The process of security analysis and requirements generation was now exercised not only for an exemplary portion but for the entire RTU and it provided valuable results to the use case.

Table 6. CS1-Architecture-Driven Assurance: US2-Model-based requirement management

Realisation Scenario	Model-based requirement management
Scope	In Iteration 1, a subset of the following standard was used as a basis for safety and security product-related requirements. Related requirements were selected

	(additional requirements might be derived based on system specifics). • IEC 62443-4-2 In the second iteration, the following standard was used in addition: • IEEE 1686 Of high importance in the second iteration was, however, the introduction of automatic generation of missing requirements by the tool MORETO.
Tool Settings	In both iterations, the Model-based Requirement Management Tool (MORETO) was used (which is an extension of Sparx Systems Enterprise Architect), developed by AIT.
Participants	• System analysis, specifications and evaluation: TLV • Tool user and developer: AIT
Activities realised	Iteration1: • MORETO started with four diagrams to model systems for industrial automation and control systems. The RTU device was integrated but in a simple form. Import related requirements as SysML requirement diagram. • IEC 62443-4-2 is an integral part of MORETO. • The security requirements analysis process is a part of this version based on the Drag-and-Drop. • A simple automation feature is integrated with this version, using JScript language which is managed under the umbrella of the enterprise architect. • The scripts feature in EA has many limitations. • Identify and link requirements related to specific parts of the system model from (1). • Specify additional safety and security requirements based on co-analysis. • Evaluate the results of the correctness of the requirements and soundness of the approach with relation to multi-concern assurance. Iteration 2: • A new RTU diagram has been added to MORETO toolbox, which has additional features of CPU-RTU initial configurations. • The features of MORETO toolbox have been expanded by replacing JScripts with C# language. • The C# language makes MORETO more reliable and stable than the EA's script languages. • Enhancement of the contents of IEC 62443-4-2 and integration of IEEE 1686 with MORETO. • The pattern feature for all security standards has been integrated with the current version of MORETO. • Modelling of several relevant configurations in MORETO. • Creation of missing security requirements of the configuration for compliance with IEC 62443 and IEEE 1686.
Usage Decisions	Iteration 1: • MORETO is an external tool to the AMASS platform. It might be possible to import the artefacts into the AMASS platform using a third-party adapter. However, in the meantime, the same models were duplicated in Papyrus, as an internal tool for the AMASS platform. • As a starting point, the focus was set on IEC 62443-4-2 for cybersecurity. Other standards such as IEC 61508, IEC 62351 and IEEE 1686 could be considered depending on the need. Iteration2:

	• MORETO is external to AMASS and needs an integration. SysML provides an interface to the AMASS platform; necessary adaptations are currently (March 2018) under discussion with TEC. • MORETO has been extended to support also IEEE 1686 as this standard is very useful to be applied to the RTU. • Furthermore, the tool is used for automatic requirements generation; this improves efficiency significantly.
Expected Results	Iteration 1: • Product-related system and component safety and security requirement specification in the form of a list and in package of SysML diagrams. Iteration 2: • Automatic generation of system and component security requirements compliant with IEC 62443 and IEEE 1686.
Conclusions	• MORETO is a tool for security requirement analysis, allocation, and management using SysML/UML models. • MORETO toolbox applies to any system that can be modelled in SysML/UML (e.g. cyber-physical production systems CPPS). • It supports manual and automatic security requirement generation and allocation. • System model and requirements, all in one place. • Possibility to import additional requirements, or to export to different formats. • MORETO is a plug-in that can be installed and integrated with Enterprise Architect which considers one of the top modelling tools in the industry. • MORETO comes with a full installation package ready to be integrated with Enterprise Architect. The automatic generation of system and security requirements, was applied to the RTU, based on IEEE 1686 standard. For final iteration, IEC 62443 will be also applied to RTU.

3.1.3.2. STO2 Multi-concern Assurance

In AMASS, Safety & Security Co-Analysis can be implemented in more than one way:

1. Using a dedicated co-analysis tool: FMVEA

   According to D4.3 "Design of the AMASS tools and methods for multi-concern assurance (b)" [5], one available co-analysis method is FMVEA (Failure Modes, Vulnerabilities and Effects Analysis). The respective tool is currently under development and will be available in the third iteration.

2. Combining the analysis processes of separate tools by a common workflow by WEFACT.

   Also, in D4.3 [5], the method of combining multiple analysis tools by WEFACT (Workflow Engine for Analysis, Certification and Test) in order to achieve co-analysis is described. WEFACT has been available since the second iteration and is currently being extended with enhanced features.

More details about both, FMVEA and WEFACT, can be read in D4.3 [5].

Table 7. CS1-Multi-concern Assurance: US2-Safety & Security Assurance Case

Realisation Scenario	Safety & Security Assurance Case (including co-analysis)
Scope	Iteration 1: • During the first iteration, an analysis of the product security requirements from IEC 62443-4-2 and ISASecure EDSA specification were conducted, 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. FMVEA analysis was used to identify safety hazards and security threats, and their interrelations. Iteration 2: • A new tool supporting the above mentioned FMVEA method is being designed. As a base platform, Eclipse-RCP and Enterprise Architect (EA) were under discussion. Experience with other Eclipse-based development and with EA in the context of MORETO influenced the decision in favour of EA. Currently (April 2018), most of the underlying database has been designed and the user interface is currently under development.
Tool Settings	Iteration 1: • Excel sheet, optionally Microsoft Threat Modelling Tool for additional threat identification Iteration 2: • Excel sheet • Combining tools via WEFACT, e.g. MORETO or the Microsoft Threat analysis tool plus APIS FMEA. The FMVEA tool is expected for the iteration 3.
Participants	• System specification: TLV • Data analysis: AIT • Result evaluator: TLV, AIT
Activities realised	-
Usage Decisions	None specific
Expected Results	Complete safety and security analysis results in iteration 3.
Conclusions	The safety part of analyses has been postponed to iteration 3 in order to accelerate the security part of the analyse (MORETO). For the 3rd iteration, a full co-analysis with FMVEA is foreseen.

3.1.4. Coverage of AMASS Prototype P1 Architecture

Table 8 illustrates the implemented functionalities during this second iteration within the Case Study 1.

Table 8. AMASS Prototype P1 Coverage by CS1

STO	AMASS Functionality Group	Tools
Architecture-Driven Assurance	System Component Specification	MORETO
	System Architecture Modelling for Assurance	MORETO
	Architectural Patterns for Assurance	MORETO
	Contract-based Design for Assurance	-
	Activities supporting Assurance Case	-
Multi-Concern Assurance	Assurance Case Specification	OpenCert (Safety and Security Assurance Case)
	Dependability Assurance	OpenCert
	System Dependability Co-Analysis/Co-Assessment	FMVEA
	Contract-Based Multi-concern Assurance	-
Seamless	Evidence Management	OpenCert

Interoperability	Tool Integration Management	-
	Collaborative Work Management	-
	Tool Quality Assessment and Characterization	-
Cross Intra-Domain Reuse	Compliance Management	OpenCert
	Reuse Assistant	OpenCert
	Process-related reuse via management of variability at process level	-
	Process-related reuse via management of variability at product level	-
	Automatic generation of process-based arguments	-
	Automatic generation of product-based arguments	-

3.1.5. Conclusions

At this stage, the main benefits and potential improvements are identified in Table 9.

Table 9. Benefits and potential improvements for CS1

US Processes	Achievement/Benefits	Improvements/Recommendations
Standards Models Creation (OpenCert)	• Modelling of IEC 62443-4-2 and IEC 61508-3	• Modelling of IEC 62443-4-1 and IEC 62351
Assurance Project Creation (OpenCert)	• Selection of the requirements based on the "Criticality level" and "Applicability level"	• Addition of roles and users • Addition of scheduling capabilities • Wizard for project creation • Usability/User Interface • Performance
Evidence Management (OpenCert)	• Evidence creation for safety and security assurance project	• Reuse of evidences • Wizard for evidence management • Usability/User Interface • Performance
Compliance Management (OpenCert)	• Filters about compliance information using the new function "Mapping Table" • New metrics and charts	• Addition of executive summary and new charts with gaps analysis • Wizard for compliance management • Usability/User Interface • Performance • Information visualisation
Model-based requirement management (MORETO)	• New RTU diagram with additional features • Pattern feature integrated • Automatic generation of system and component security requirements compliant with IEC 62443 and IEEE 1686	• Enhancement of the RTU modelling, by differentiating control RTU and gateway • Integration of MORETO with the AMASS platform • Modelling of IEC 62443-4-1, IEC 62351 and IEC 61508 requirements
Safety & Security co-analysis (FMVEA)	• Preliminary FMVEA analysis to identify safety hazards and security threats	• Integration of FMVEA analysis in the AMASS tool
Safety & Security assurance case (MORETO)	• It has been postponed to iteration 3	• Creation of the assurance case according to WP4 results

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

3.2.1. Case Study Specification

The Case Study 2 will be based on an advanced driver assistance function (e.g. a traffic jam assistant function allowing highly automated driving of a car on highways up to a defined max speed), in which several electric drives (controller, power electronics and electric machine) act as actuators. The case study will be executed using modelling, analysis and verification tools and their respective tool integrations.

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" [1].

3.2.2. US1: Reuse of safety artefacts within a product family (Intra-domain reuse)

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. Many times, products are not designed and developed independently and they belong to a product family (e.g. product line). Even if, products from the same product line are quite similar, those product families require assurance of all "family members", which would drive efforts leading to significant price increase. Therefore, this usage scenario attempts to reusing safety assurance artefacts between different products of the same product line.

It has to be noted that this is an ongoing work which will be completed in the next iteration.

3.2.2.1. STO1 Architecture-Driven Assurance

Table 10. CS2-Architecture-Driven Assurance: US1-Reuse of safety artefacts within a product family

Realisation Scenario	Reuse of Safety Artefacts within a product family	
Scope	Model-based design of the AURIX controller	
Tool Settings	Enterprise Architecture/Medini Analyze	
Participants	Aurix Controller Architecture Model	IFX
	Analyses execution	IFX, B&M, KMT
	Evidences generation	
Activities realised	AURIX architecture modelling by using Enterprise Architecture and Medini Analyse	
	The main purpose of this Usage Scenario is to define the safety requirements for the item, the safety architecture. These specifications must be considered: • Item definition • Functional safety requirements • Technical safety requirements • Hardware safety requirements • Software safety requirements	
Usage Decisions	n.a.	
Expected Results	Complete architecture including safety information and preliminary analyses results. 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
Conclusions	Some implementations regarding reuse feasibility with the AMASS tools have been postponed to the third iteration. Once the results are obtained, the different usage scenarios will be evaluated to get metric figures.

3.2.2.2. STO3 Seamless Interoperability

Table 11. CS2-Seamless Interoperability: US1-Reuse of safety artefacts within a product family: evidence management

Realisation Scenario	Reuse of Safety Artefacts within a product family: evidence management
Scope	This usage scenario enables the user to record and retrieve consistent artefacts for a baseline system specification. Visualisation 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.). Definition and visualisation of the mappings between the evidences to be provided and the corresponding artefacts used as evidence.
Tool Settings	Medini Analyze and SVN
Participants	IFX, KMT, B&M
Activities realised	1. Impact analysis to identify reusable artefacts for a specific project. 2. Specify evidence model. 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. 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
Usage Decisions	NA
Expected Results	This usage scenario aims at demonstrating that the AMASS tool platform allows to correctly identify common reusable artefacts between specific project and generic model in order to automatically fill the Hazard Log.
Conclusions	NA

3.2.2.3. STO4 Cross Intra-domain reuse

Table 12. CS2-Cross Intra-domain reuse: US1-Reuse of Safety Artefacts within a product family: intra-domain reuse

Realisation Scenario	Reuse of Safety Artefacts within a product family: intra-domain reuse
Scope	Reuse of safety assurance artefacts within different products of the same family

Tool Settings	For now, no specific tools from the AMASS tools have been used. The usage of AMASS tools (e.g. Reuse Assistant) to address reuse will be evaluated in the next iteration.
Participants	IFX, KMT, B&M
Activities realised	First steps on project level reuseIF
Usage Decisions	NA
Expected Results	Impact Analysis of artefacts reuse
Conclusions	The reuse capabilities/functionalities of the AMASS platform will be evolved during P2 in order to improve product and process reuse.

3.2.3. Coverage of AMASS Prototype P1 Architecture

Table 13 illustrates the implemented functionalities during this second iteration within the Case Study 2.

Table 13. AMASS Prototype P1 Coverage by CS2

STO	AMASS Functionality Groups	Tools
Architecture-Driven Assurance	System Component Specification	Enterprise Architecture
	System Architecture Modelling for Assurance	Enterprise Architecture
	Architectural Patterns for Assurance	-
	Contract-based Design for Assurance	-
	Activities supporting Assurance Case	-
Multi-Concern Assurance	Assurance Case Specification	-
	Dependability Assurance	-
	System Dependability Co-Analysis/Co-Assessment	-
	Contract-Based Multi-concern Assurance	-
Seamless Interoperability	Evidence Management	SVN
	Tool Integration Management	-
	Collaborative Work Management	-
	Tool Quality Assessment and Characterization	-
Cross Intra-Domain Reuse	Compliance Management	-
	Reuse Assistant	-
	Process-related reuse via management of variability at process level	-
	Process-related reuse via management of variability at product level	-
	Automatic generation of process-based arguments	-
	Automatic generation of product-based arguments	-

3.2.4. Conclusions

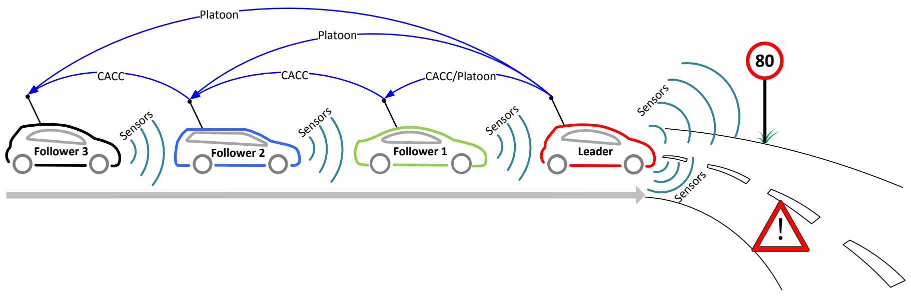
Not available.

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

3.3.1. 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.

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



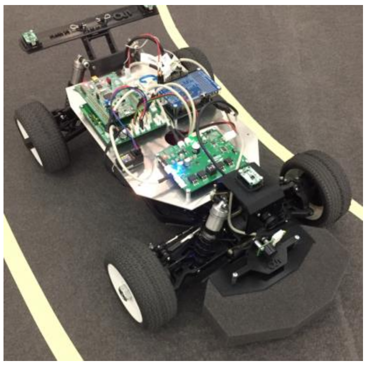


Figure 18. CS3 Main Scenario and one demonstrator

During the development of the case study, it was decided to create a (sub) case study for the validation of the vehicle powertrain, called "DC-Drive". This demonstrator is a simplified version of the electrical powertrain of one car and can used for the implementation of safety measures, regarding the functional safety of the powertrain and for fault injection of typical, technical failure modes of the powertrain (e.g. wire breaking).

Based on the initial definition of CS3 usage scenarios provided in Deliverable D1.1 [1], we selected some primary research topics and created derived usage scenarios that cover the different AMASS evaluation areas. During the first iteration (D1.4) three usage scenarios were defined: "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 system" and "US3: Systematic creation of functional and technical safety concepts based on contracts for cooperative vehicle automation". Since all of share similar concepts, they have been tackled as part of a unique usage scenario: "US1: Safety assessment for collaborative automated vehicle functions by model-based safety analysis and contracts". Furthermore, two new usage scenarios have been defined.

Thus, this is the current status regarding usage scenario definition for CS3:

- US1: Safety assessment for collaborative automated vehicle functions by model-based safety analysis and contracts
- US2: Process for development of collaborative automated vehicle functions, which considers functional safety, cybersecurity and reuse aspects.
- US3: Collection and Analysis of Assurance Information.

3.3.2. US1: Safety assessment for collaborative automated vehicle functions by model-based safety analysis and contracts

3.3.2.1. STO1 Architecture-Driven Assurance

Defined in the Table 14, the actual result of the iterations in progress will be discussed at this point, regarding the CACC/Platooning-function and the DC-Drive-validation.

Table 14. CS3-Architecture-Driven Assurance: US1- CACC/Platooning

Realisation Scenario	**CACC/Platooning**			
Scope	Model-based design and assurance of "CACC/Platooning"			
Tool Settings	SAVONA/CHESS			
Activities realised	System Design	Definition of top-level requirements	Savona	2. iteration in progress
		CACC/Platooning functional architecture	Savona	2. iteration in progress
		CACC/Platooning functional behaviour	SysML via MS Visio	2. iteration in progress
		CACC/Platooning architecture validation	Savona	1. Iteration in progress
	Safety Analyses	Hazard analysis	-	1. Iteration planning
		Functional safety conception	-	1. Iteration planning
		Contracts- Fault Injection Simulations- Monitors	SAVONA-AMT2.0-Sabotage	In progress (planned for 3rd iteration)
Usage Decisions	-			
Expected Results	System design modelling with SAVONA/CHESS and preliminary analyses results.			
Conclusions	For P2 the automation level between contract-based design (including safety and nominal behaviour), safety analysis, monitors and fault injection simulations will be further elaborated.			

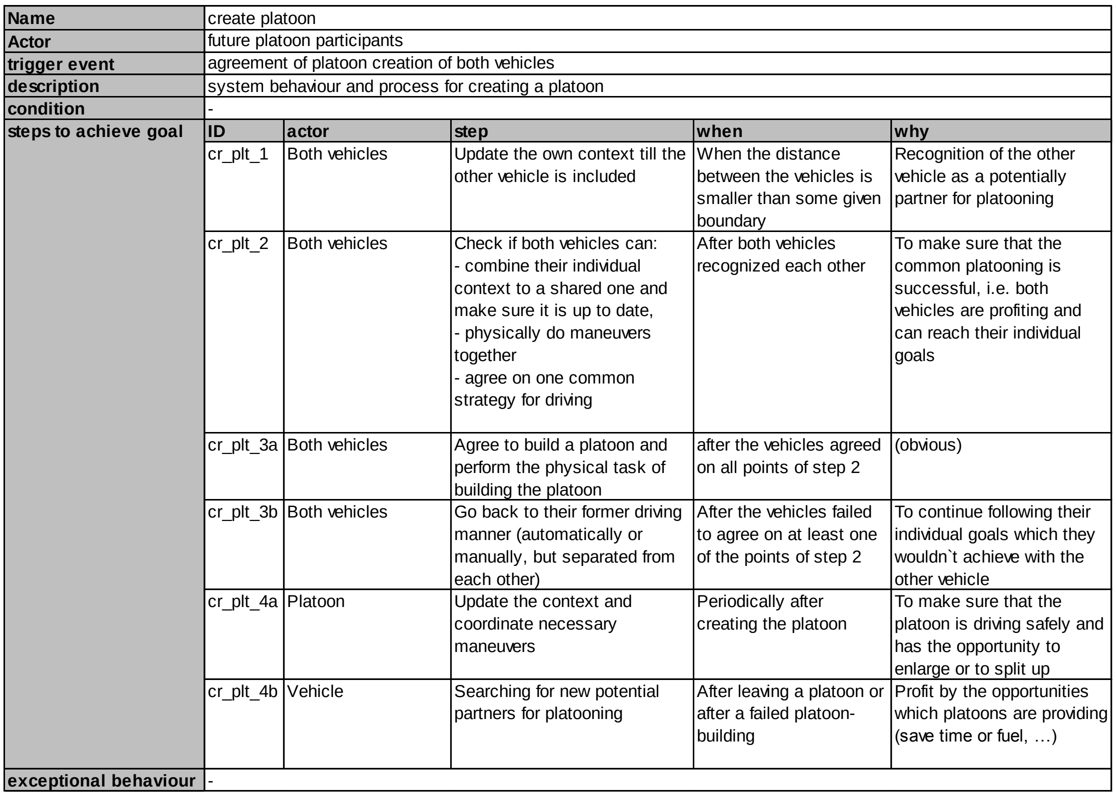
3.3.2.1.1. CACC/Platooning – System Design

Definition of Top-Level-Requirements

In the previous development iteration, several top-level requirements were defined to develop the first functions of the system regarding the following use cases:

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

In the second iteration, "running platoon", "join platoon" and "create platoon" functions are chosen. In addition, user stories describe the typical behaviour of the system, thus, they are refined and implemented to interact with the environment and the actor. Figure 19 shows an example of a typical user story:



Name	create platoon				
Actor	future platoon participants				
trigger event	agreement of platoon creation of both vehicles				
description	system behaviour and process for creating a platoon				
condition	-				
steps to achieve goal	**ID**	**actor**	**step**	**when**	**why**
	cr_plt_1	Both vehicles	Update the own context till the other vehicle is included	When the distance between the vehicles is smaller than some given boundary	Recognition of the other vehicle as a potentially partner for platooning
	cr_plt_2	Both vehicles	Check if both vehicles can: - combine their individual context to a shared one and make sure it is up to date, - physically do maneuvers together - agree on one common strategy for driving	After both vehicles recognized each other	To make sure that the common platooning is successful, i.e. both vehicles are profiting and can reach their individual goals
	cr_plt_3a	Both vehicles	Agree to build a platoon and perform the physical task of building the platoon	after the vehicles agreed on all points of step 2	(obvious)
	cr_plt_3b	Both vehicles	Go back to their former driving manner (automatically or manually, but separated from each other)	After the vehicles failed to agree on at least one of the points of step 2	To continue following their individual goals which they wouldn`t achieve with the other vehicle
	cr_plt_4a	Platoon	Update the context and coordinate necessary maneuvers	Periodically after creating the platoon	To make sure that the platoon is driving safely and has the opportunity to enlarge or to split up
	cr_plt_4b	Vehicle	Searching for new potential partners for platooning	After leaving a platoon or after a failed platoon-building	Profit by the opportunities which platoons are providing (save time or fuel, …)
exceptional behaviour	-				

Figure 19. Example of a user story for the CACC/Platooning-function

At the current state, the existing requirements will be further refined and used for the development of the system architecture. The requirements will be developed into a semi-formalized way, by doing so, functional and safety contracts will be easily created. These contracts will be part of the model-based development of the function.

<u>CACC/Platooning functional architecture</u>

The functional architecture is created in the SAVONA tool and structured under the context of multi-function integration. This means that, several vehicle functions (Like CACC, Lane Keeping, Automated parking, etc.) can easily be integrated in the functional architecture with a minimum impact on the vehicle architecture. Furthermore, all modules of the architecture are designed in a way to easily integrate the functional and safety contracts. After the definition of the first requirements, they will be converted into contracts and implemented in the functional architecture of the vehicle in SAVONA.

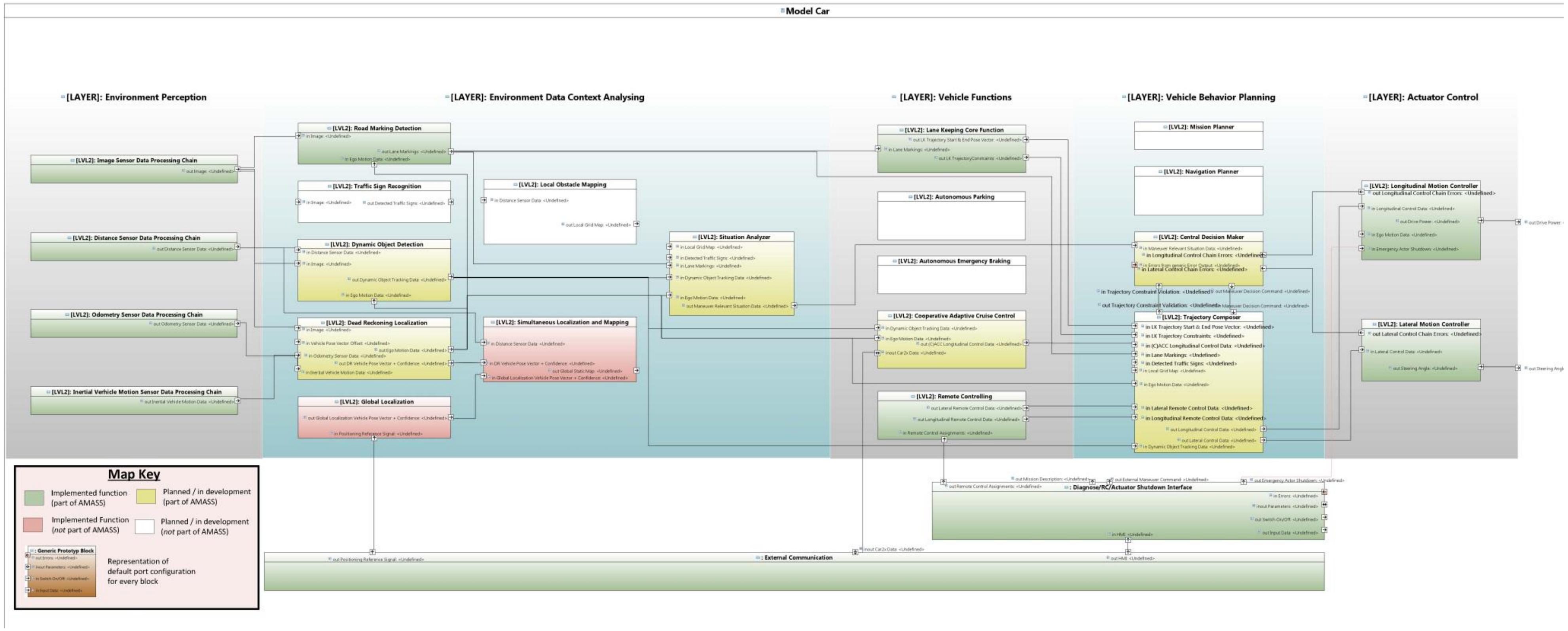

Model Car
[LAYER]: Environment Perception
[LAYER]: Environment Data Context Analysing
[LAYER]: Vehicle Functions
[LAYER]: Vehicle Behavior Planning
[LAYER]: Actuator Control
[LVL2]: Image Sensor Data Processing Chain
[LVL2]: Distance Sensor Data Processing Chain
[LVL2]: Odometry Sensor Data Processing Chain
[LVL2]: Inertial Vehicle Motion Sensor Data Processing Chain
[LVL2]: Road Marking Detection
[LVL2]: Traffic Sign Recognition
[LVL2]: Local Obstacle Mapping
[LVL2]: Dynamic Object Detection
[LVL2]: Situation Analyzer
[LVL2]: Dead Reckoning Localization
[LVL2]: Simultaneous Localization and Mapping
[LVL2]: Global Localization
[LVL2]: Lane Keeping Core Function
[LVL2]: Autonomous Parking
[LVL2]: Autonomous Emergency Braking
[LVL2]: Cooperative Adaptive Cruise Control
[LVL2]: Remote Controlling
[LVL2]: Mission Planner
[LVL2]: Navigation Planner
[LVL2]: Central Decision Maker
[LVL2]: Trajectory Composer
[LVL2]: Longitudinal Motion Controller
[LVL2]: Lateral Motion Controller
: Diagnose/RC/Actuator Shutdown Interface
: External Communication
Map Key
Implemented function (part of AMASS)
Planned / in development (part of AMASS)
Implemented Function (not part of AMASS)
Planned / in development (not part of AMASS)
: Generic Prototyp Block
Representation of default port configuration for every block

Figure 20. Current vehicle functional architecture (current progress)

CACC/Platooning functional behaviour

Due to the fact that model-based engineering includes, besides the requirement and the model structure, the description of model behaviour, several behaviour diagrams for the function are created. Unfortunately, SAVONA does not support these kinds of diagrams, so the MS Visio Tool is used for this part of development. At present, there are activity diagrams, sequence diagrams and use case diagrams available.

CACC/Platooning architecture validation

As one of the key features of SAVONA, the correctness of the functional architecture will be validated by the model checking feature.

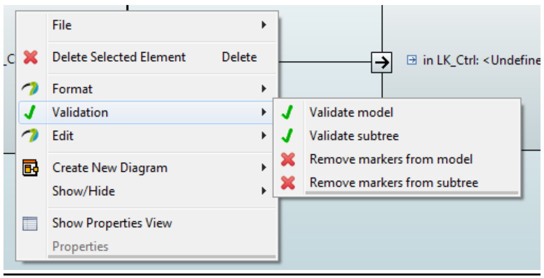


Figure 21. Example of the validation of the functional architecture in Savona

3.3.2.1.2. *CACC/Platooning – Safety Analyses*

A CACC/platooning-function has a highly degree of rigor in the assurance of safety related functionality. For the development of functional safety, regarding the safety methodology of ISO26262 for automotive system development, the following steps are planned:

Hazard analysis

Due to fact that the demonstrator only drives in a laboratory environment, only hazards, which include the interaction of the vehicles inside the platoon e.g. a rear-end-collision, will be chosen. The development of a top-level safety goal can be set by the responsible safety engineer in the same way as in traditional safety engineering by a Hazard Analysis and Risk Assessment (HARA). This obviously requires scaling it to the model cars. Results of this development can be like:

Hazard	*Safety Goal*
Unjustified brake application, which leads to a rear-end-collision with platoon member.	Ensure a sufficient time gap between platoon participants to avoid collisions.

Functional safety concept

It has to be proven that the defined safety goals hold for each mode of operation, use cases and 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.

To achieve the safety goals, we will define a functional safety concept, which mainly follows the steps defined in the figure below.

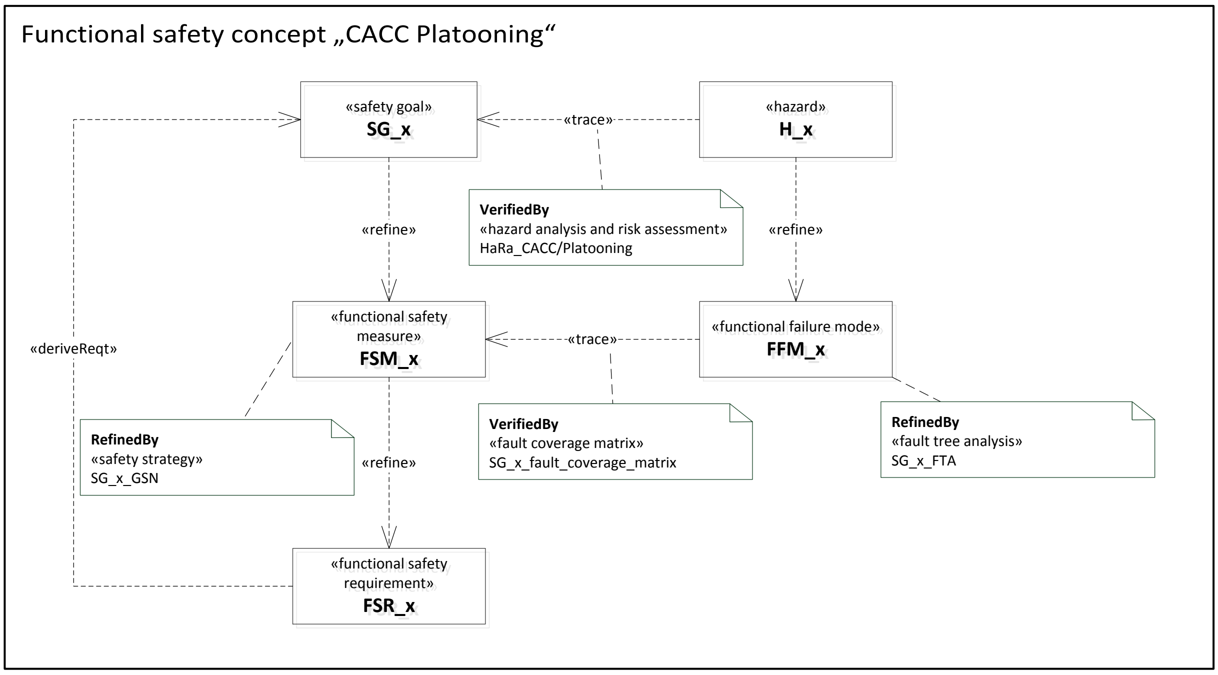


Figure 22. Methodology of functional safety requirement development for the CACC/Platooning function

The steps of this development will be repeated iteratively, until all identified failure modes are appropriately covered and the residual risk is acceptable.

3.3.3. US2: Process for development of collaborative automated vehicle functions, which considers functional safety, cybersecurity and reuse aspects

The main purpose of this Usage Scenario is to evaluate a joint process concerning functional safety and cybersecurity and the ability for cross concern reuse. The process deals with verification of collaborative automated vehicle functions and reuse of processes.

These specifications must be taken into account:

- ISO 26262 for functional safety
  SAE J3061 for cybersecurity

The cross-concern variability management and co-engineering scenario shows the usage of EPF-Composer (EPF-C) and the BVR tool to model the automotive Security-informed Safety-oriented Process Line with consideration of co-engineering. This usage scenario shows the process related to the verification of the system design of the Car2X Communication Manager unit.

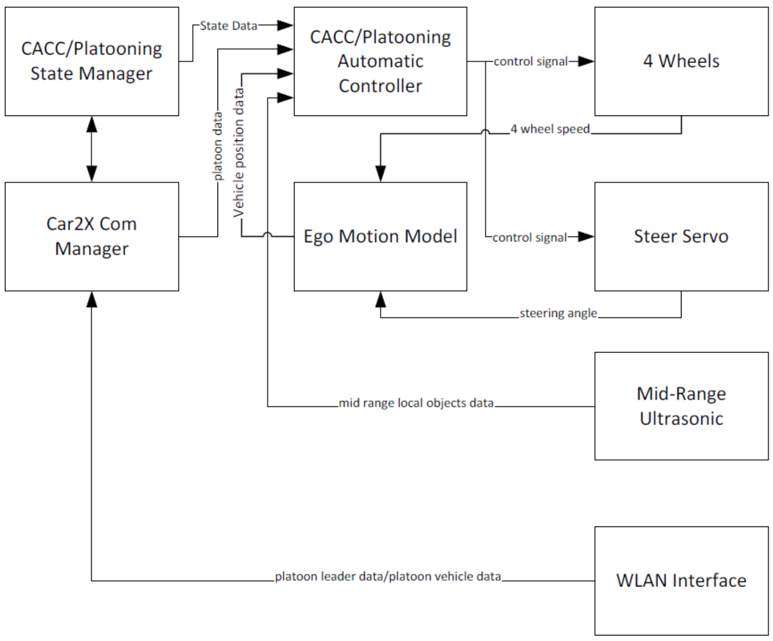
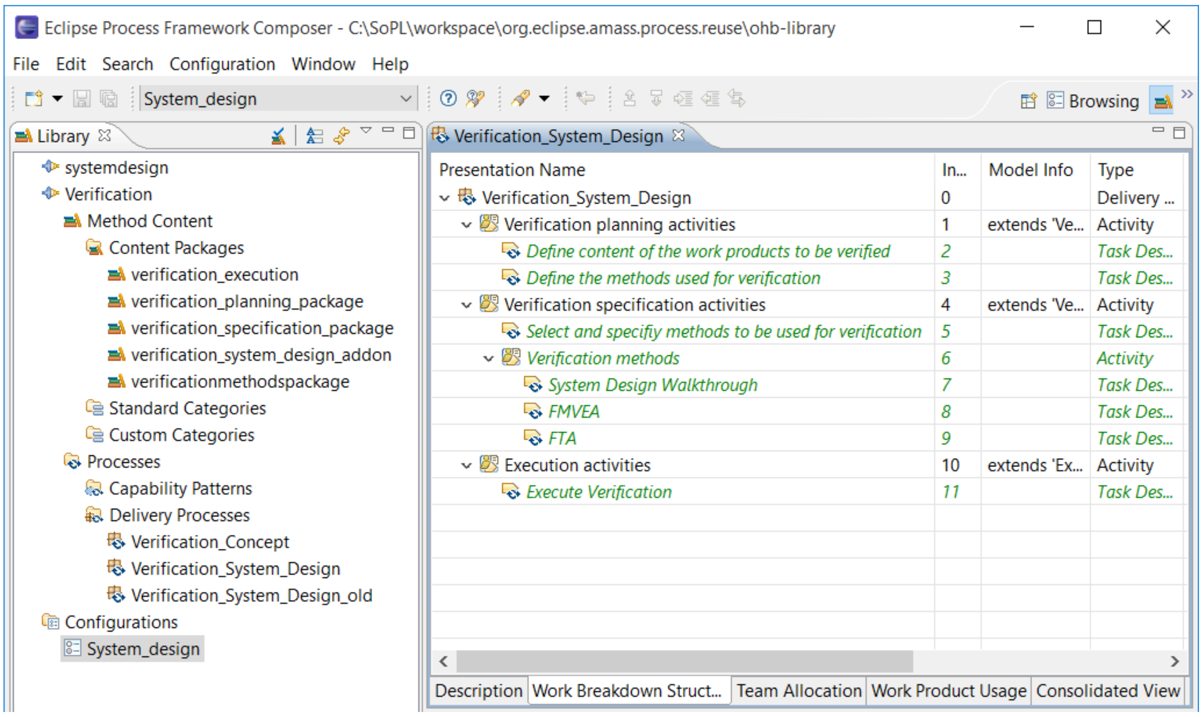


Figure 23. Simplified System Architecture in the model car [4]

The first step is the identification of standards, which are needed to implement communication between vehicles. ISO 26262 and SAE J3061 are taken into account because functional safety and cybersecurity has to be considered. A SiSoPL model related to functional safety (ISO 26262) and cybersecurity (SAE J3061) is defined. The presented solution uses the integration of EPF-C and BVR-tool. The process development (especially the base model) is done with EPF-Composer (see Figure 24). Afterwards variability aspects are managed with help of the BVR tool (see Figure 25).


Eclipse Process Framework Composer - C:\SoPL\workspace\org.eclipse.amass.process.reuse\ohb-library

File Edit Search Configuration Window Help

System_design

Browsing

Library

- systemdesign
- Verification
  - Method Content
    - Content Packages
      - verification_execution
      - verification_planning_package
      - verification_specification_package
      - verification_system_design_addon
      - verificationmethodspackage
    - Standard Categories
    - Custom Categories
  - Processes
    - Capability Patterns
    - Delivery Processes
      - Verification_Concept
      - Verification_System_Design
      - Verification_System_Design_old
- Configurations
  - System_design

Verification_System_Design

Presentation Name	In...	Model Info	Type
Verification_System_Design	0		Delivery ...
Verification planning activities	1	extends 'Ve...	Activity
Define content of the work products to be verified	*2*		*Task Des...*
Define the methods used for verification	*3*		*Task Des...*
Verification specification activities	4	extends 'Ve...	Activity
Select and specify methods to be used for verification	*5*		*Task Des...*
Verification methods	*6*		*Activity*
System Design Walkthrough	*7*		*Task Des...*
FMVEA	*8*		*Task Des...*
FTA	*9*		*Task Des...*
Execution activities	10	extends 'Ex...	Activity
Execute Verification	*11*		*Task Des...*

Description | Work Breakdown Struct... | Team Allocation | Work Product Usage | Consolidated View

Figure 24. EPF-C Work Breakdown Structure of verification process

To perform verification steps, different verification methods can be selected based on ISO 26262-4. ISO 26262-4[1] establishes both deductive (e.g. FTA) and inductive analysis methods (e.g. FMEA) which recommendation level depends on the specified ASIL (Automotive Safety Integrity Level). For example, FMEA is highly recommended for ASIL A, B, C and D, whereas FTA is recommended for ASIL B and highly recommended for C and D. The considered variability in this scenario deals with the verification method Fault Tree Analysis (FTA).

Since FTA is not highly recommended for ASIL B, the FTA would be removed from the process The ASIL is concluded once a HARA has been performed by means of a different process.

In the defined situation, the communication manager has to be developed according to ASIL B requirements. For that reason, FTA is removed from the process by the BVR tool. Figure 25 shows that FTA has the value true in the resolution diagram. This means that it will be removed from the process.

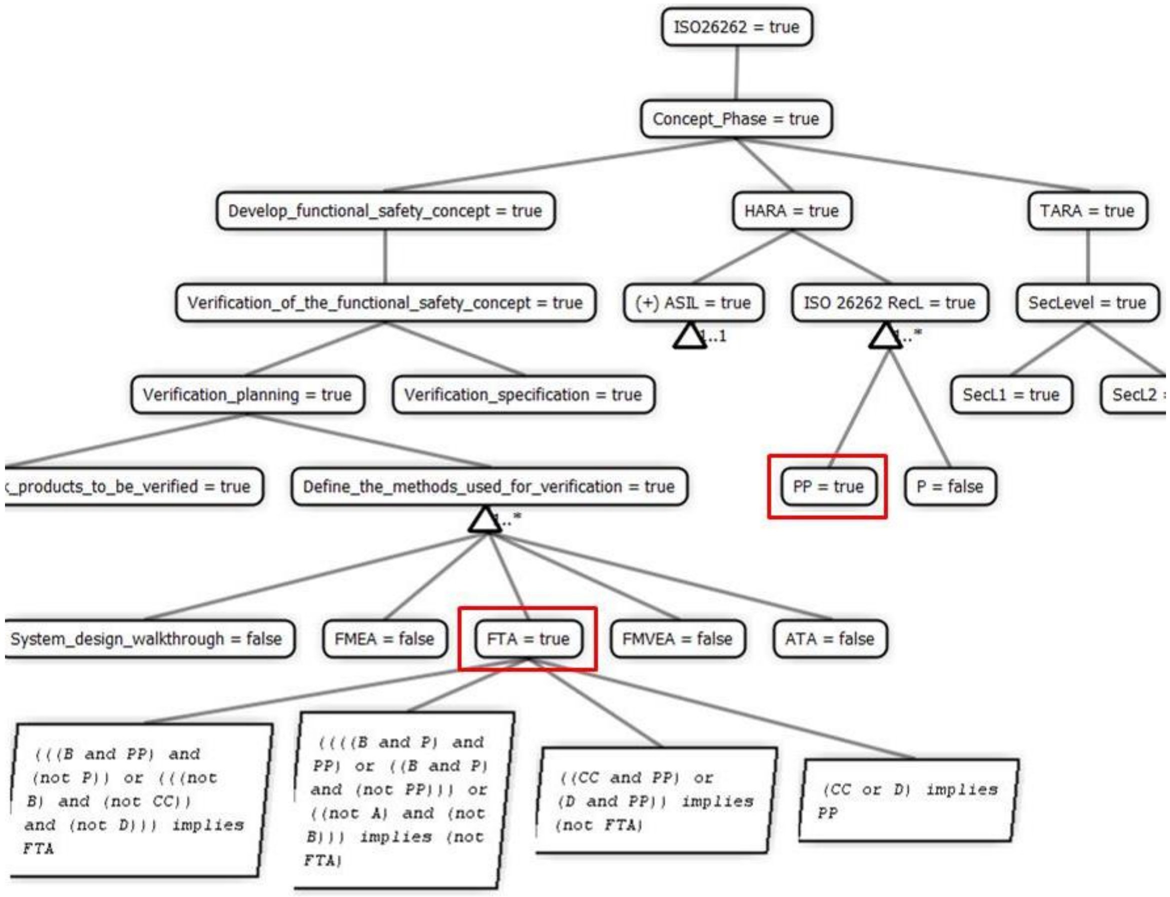

ISO26262 = true
Concept_Phase = true
Develop_functional_safety_concept = true
HARA = true
TARA = true
Verification_of_the_functional_safety_concept = true
(+) ASIL = true
1..1
ISO 26262 RecL = true
1..*
SecLevel = true
Verification_planning = true
Verification_specification = true
SecL1 = true
SecL2
products_to_be_verified = true
Define_the_methods_used_for_verification = true
1..*
PP = true
P = false
System_design_walkthrough = false
FMEA = false
FTA = true
FMVEA = false
ATA = false
(((B and PP) and (not P)) or (((not B) and (not CC)) and (not D))) implies FTA
((((B and P) and PP) or ((B and P) and (not PP))) or ((not A) and (not B))) implies (not FTA)
((CC and PP) or (D and PP)) implies (not FTA)
(CC or D) implies PP

Figure 25. BVR-Resolution diagram: integrated safety and security process (ASIL:= B)

3.3.3.1. STO2 Multi-concern Assurance

Table 15. CS3-Multi-concern Assurance: US2-Safety/security co-assessment

Realisation Scenario	Safety/security co-assessment
Scope	Development of cross concern processes (functional safety and cybersecurity)
Tool Settings	EPF-Composer, BVR Tool
Participants	VIF

[1] ISO 26262: "Road vehicles – Functional safety"

Activities realised	Identification of relevant standards Process definition with focus on cross concern activities
Usage Decisions	
Expected Results	A SiSoPL model related to functional safety (ISO 26262) and cybersecurity (SAE J3061) is defined.
Conclusions	The presented solution uses the integration of EPF-C and BVR-tool

3.3.3.2. STO4 Cross Intra-domain reuse

Table 16. CS3-Cross Intra-domain reuse: US2-Process-related reuse via management of variability at process level

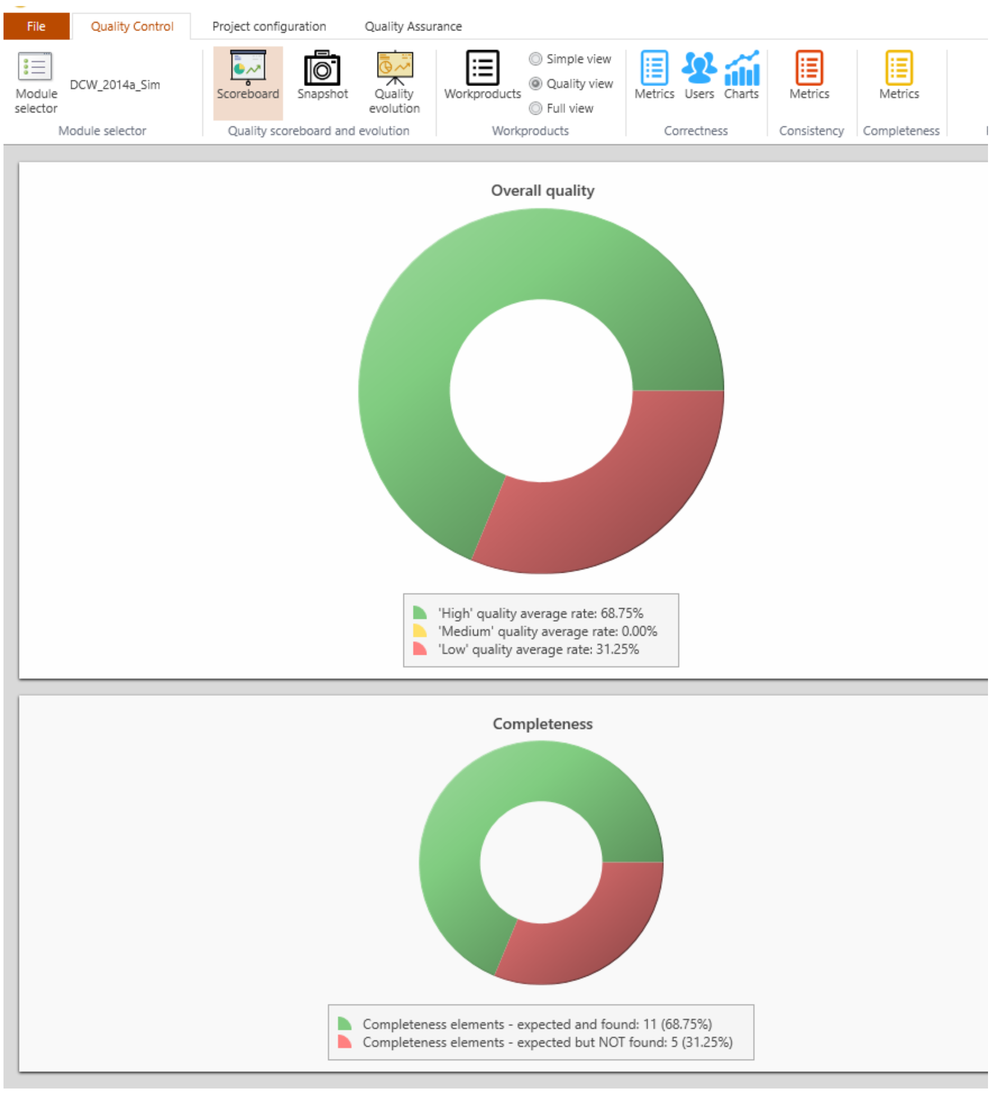
Realisation Scenario	Process-related reuse via management of variability at process level
Scope	Development of cross concern processes (functional safety and cybersecurity) Process variability
Tool Settings	EPF-Composer, BVR
Participants	VIF
Activities realised	Identification of relevant standards Process definition with focus on cross concern activities Process variability management
Usage Decisions	-
Expected Results	Process reuse between ISO 26262 and J3061
Conclusions	The so-called "Process-related reuse via management of variability at process level" functionality has been used.

3.3.4. US3: Collection and Analysis of Assurance Information

This usage scenario deals with the collection of assurance information of the use case (e.g. system models and standards) and on the analysis of their quality with TRC tools.

No work was reported on this usage scenario for the first development iteration of AMASS. For the second iteration, Simulink models have been indexed and also imported to TRC tools. Different files have been used and some of the imported models have later been analysed for quality assessment (Figure 26 and Figure 27). Finally, a partial semantic representation of ISO 26262 has been created.

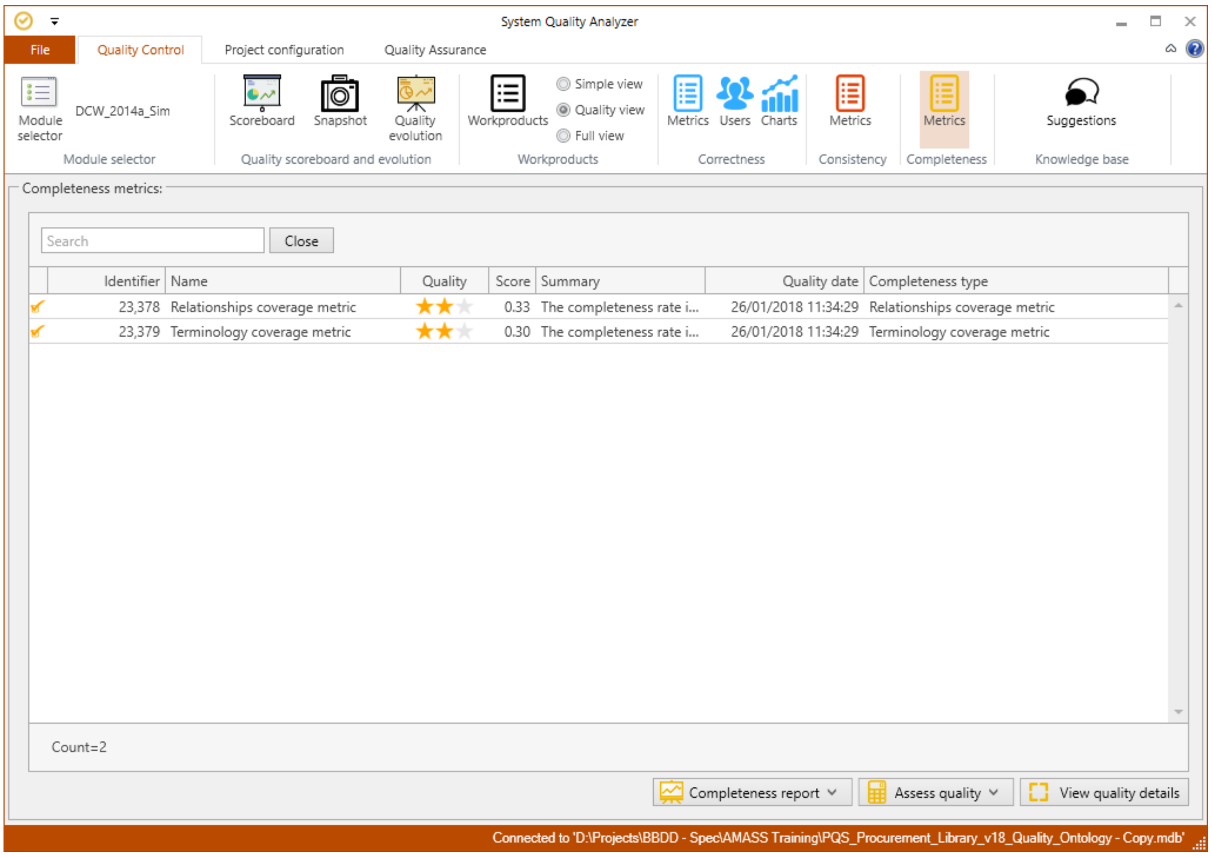
For quality analysis, SQA (System Quality Analyzer) connects to the Simulink model to extract its information, e.g. terms (components in the model), relationships (connexions between components). Next, the tool allows a user to use metrics to measure the quality of the model.



🟩 'High' quality average rate: 68.75%
🟨 'Medium' quality average rate: 0.00%
🟥 'Low' quality average rate: 31.25%

🟩 Completeness elements - expected and found: 11 (68.75%)
🟥 Completeness elements - expected but NOT found: 5 (31.25%)

Figure 26. Overall quality report



Completeness metrics:

Identifier	Name	Quality	Score	Summary	Quality date	Completeness type
23,378	Relationships coverage metric	★★☆	0.33	The completeness rate i...	26/01/2018 11:34:29	Relationships coverage metric
23,379	Terminology coverage metric	★★☆	0.30	The completeness rate i...	26/01/2018 11:34:29	Terminology coverage metric

Count=2

Connected to 'D:\Projects\BBDD - Spec\AMASS Training\PQS_Procurement_Library_v18_Quality_Ontology - Copy.mdb'

Figure 27. Detailed quality report

3.3.4.1. STO1 Architecture-Driven Assurance

Table 17. CS3-Architecture-Driven Assurance: US3- Quality Analysis of Simulink model

Realisation Scenario	**Quality Analysis of Simulink model**
Scope	Determine the quality level of a Simulink model
Tool Settings	SQA
Participants	• UC3 • TRC
Activities realised	1. Selection of metrics for quality assessment 2. Simulink model import 3. Quality evaluation execution 4. Quality results analysis
Usage Decisions	Selection of a specific, relevant model
Expected Results	Quality report
Conclusions	The quality analysis was successfully performed. The Relationship Metric revealed that the 33% of the relationship selected in the metric were not found in the model. That means that it is necessary to include two connexions between components. The metric Terminology Coverage indicated that the 70% of the terms selected in the configuration had been found in the model. In this case, it is necessary to add three components that are missing in the model.

3.3.4.2. STO3 Seamless Interoperability

Table 18. CS3-Seamless Interoperability: US3-Simulink model import with OSLC KM

Realisation Scenario	Simulink model import with OSLC KM
Scope	Import of Simulink models to TRC tools for their later analysis
Tool Settings	SQA, KM
Participants	• UC3 • TRC
Activities realised	1. Configuration of the OSLC KM connector 2. Selection of the model to import 3. Model import
Usage Decisions	Selection of a specific, relevant models
Expected Results	Imported Simulink model
Conclusions	Successful model import

3.3.4.3. STO4 Cross Intra-domain reuse

Table 19. CS3-Cross- and Intra-domain reuse: US3-Methodology to represent system artefacts: i**ndexing of Simulink models**

Realisation Scenario	Methodology to represent system artefacts: indexing of Simulink models
Scope	Semantic indexing of imported Simulink models for their later management with TRC tools
Tool Settings	KM
Participants	• UC3 • TRC
Activities realised	1. Configuration of KM indexing process 2. Selection of the model to index 3. Model indexing
Usage Decisions	Selection of a specific, relevant models
Expected Results	Indexed Simulink model
Conclusions	Successful model indexing

Table 20. CS3-Cross- and Intra-domain reuse: US3–Compliance management by means of the Semantic representation of ISO 26262

Realisation Scenario	Compliance management by means of the Semantic representation of ISO 26262
Scope	Representation of ISO 26262 with ontology-based technologies
Tool Settings	Knowledge Manager (KM; TRC)
Participants	• UC3 • TRC
Activities realised	The activities correspond to the application of the approach for semantic representation of safety standards presented in D6.5 and D6.7. 1. KM configuration for representation of ISO 26262 2. Initial specification of an ontology for ISO 26262 with its glossary 3. Partial modelling of ISO 26262 in KM, based on the metamodel for Reference Assurance Frameworks (Reference Activities, Reference Artefacts, Reference

	Artefact Relationships, Reference Activity input, Reference Activity output, etc.)
Usage Decisions	None
Expected Results	Semantic representation of ISO 26262 in KM
Conclusions	Successful representation

3.3.5. Coverage of AMASS Prototype P1 Architecture

Table 21 illustrates the implemented functionalities during this second iteration within the Case Study 3.

Table 21. AMASS Prototype P1 Coverage by CS3

STO	AMASS Functionality Groups	Tools
Architecture-Driven Assurance	System Component Specification	SAVONA/CHESS
	System Architecture Modelling for Assurance	SAVONA/CHESS
	Architectural Patterns for Assurance	-
	Contract-based Design for Assurance	SAVONA/CHESS
	Activities supporting Assurance Case	SAVONA KM Functional Verification by Simulink and AMT 2.0 monitors (ongoing) Medini Analyze Safety V&V CHESS/SAVONA-Sabotage (ongoing)
Multi-Concern Assurance	Assurance Case Specification	OpenCert
	Dependability Assurance	OpenCert (safety and security case)
	System Dependability Co-Analysis/Co-Assessment	FMVEA, EPF-C+BVR (ISO 26262 for functional safety and SAE J3061)
	Contract-Based Multi-concern Assurance	-
Seamless Interoperability	Evidence Management	-
	Tool Integration Management	SQA, KM via OSLC
	Collaborative Work Management	-
	Tool Quality Assessment and Characterization	-
Cross Intra-Domain Reuse	Compliance Management	EPF-C Semantic modelling of ISO 26262
	Reuse Assistant	-
	Process-related reuse via management of variability at process level	EPF-Composer and BVR: ISO 26262 for functional safety and SAE J3061
	Process-related reuse via management of variability at product level	-
	Automatic generation of process-based arguments	OpenCert
	Automatic generation of product-based arguments	OpenCert

3.3.6. Conclusions

At this stage, the main benefits and potential improvements are identified in Table 22.

Table 22. Benefits and potential improvements for CS3

Artefact	Achievement/Benefits	Improvements/Recommendations
System Design	• Refinement of user stories, based on experiences by the first iteration. • Semi formalized requirements definition and mapping to the architecture. • Refinement and integration of the CACC/Platooning functional architecture in the model car architecture.	• An in-tool traceability between system requirements/system structure and system behaviour (e.g. sequence- or activity-diagram-entities).
Safety Analysis	• Validation of methodology for model based safety engineering of autonomous vehicle functions. • First steps on the integration of the contract-based approach-fault injection and monitors for an early validation of safety concepts.	• Integration of safety-methodology-artefacts in Savona. • Integration of SAVONA, AMT 2.0 (monitors) and SABOTAGE (fault injection).

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

3.4.1. Case Study Specification

Sentinel-3 is an ocean and land mission to measure sea-surface topography, sea- and land-surface temperature, ocean colour and land colour with high-end accuracy and reliability. The mission will support ocean forecasting systems, as well as environmental and climate monitoring. The first satellite of the constellation (Sentinel-3A) was launched on February 16th, 2016, whereas the second launch (Sentinel-3B) is foreseen for 2018.

Each satellite is composed of six payload instruments: SRAL (Synthetic Aperture Radar Altimeter), SLSTR (Sea and Land Surface Temperature Radiometer), GNSS (Global Navigation Satellite System), MWR (Microwave Radiometer), OLCI (Ocean and Land Colour Instrument) and DORIS (Doppler Orbitography and Radiopositioning Integrated by Satellite). Figure 28 depicts these instruments together with the SMU (Satellite Management Unit), which represents the central intelligent core of the satellite at the same time as controls all the payload instruments (e.g., TC, TM, signals, etc.). The Case Study 4 (CS4) concentrates on the Ocean & Land Colour Instrument (OLCI). It represents the multi-spectral optical camera for ocean and land colour.

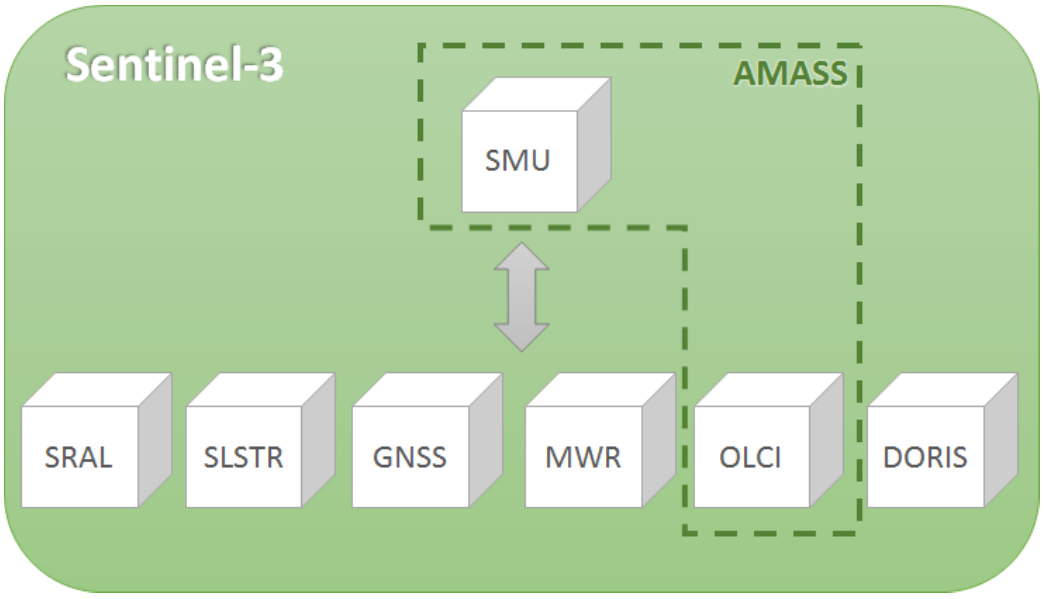


Figure 28. Sentinel-3 instruments

The CS4 standpoint has been slightly modified from the one presented in Deliverable D1.1 [1], covering not only some specific software functionalities but also a high-level view of the system architecture. The scope is to expand in order to cover the whole architecture-driven assurance process taking advantage of AMASS tools and to analyse the toolset suitability for designing space systems.

Namely, the AMASS design covers:

1. Requirements specification and formalization.
2. Design of the high-level system architecture.
3. Design of two software functionalities.
4. Conduction of safety analyses
5. Generation of the safety case.

Figure 29 shows the main elements of the OLCI instrument together with the communication links. The ICM (Instrument Control Module) is mainly responsible for the global managing of the OLCI elements and it directly communicates with the SMU. The ICM supports the software which runs on an ERC32 microprocessor with SPARC v7 architecture.

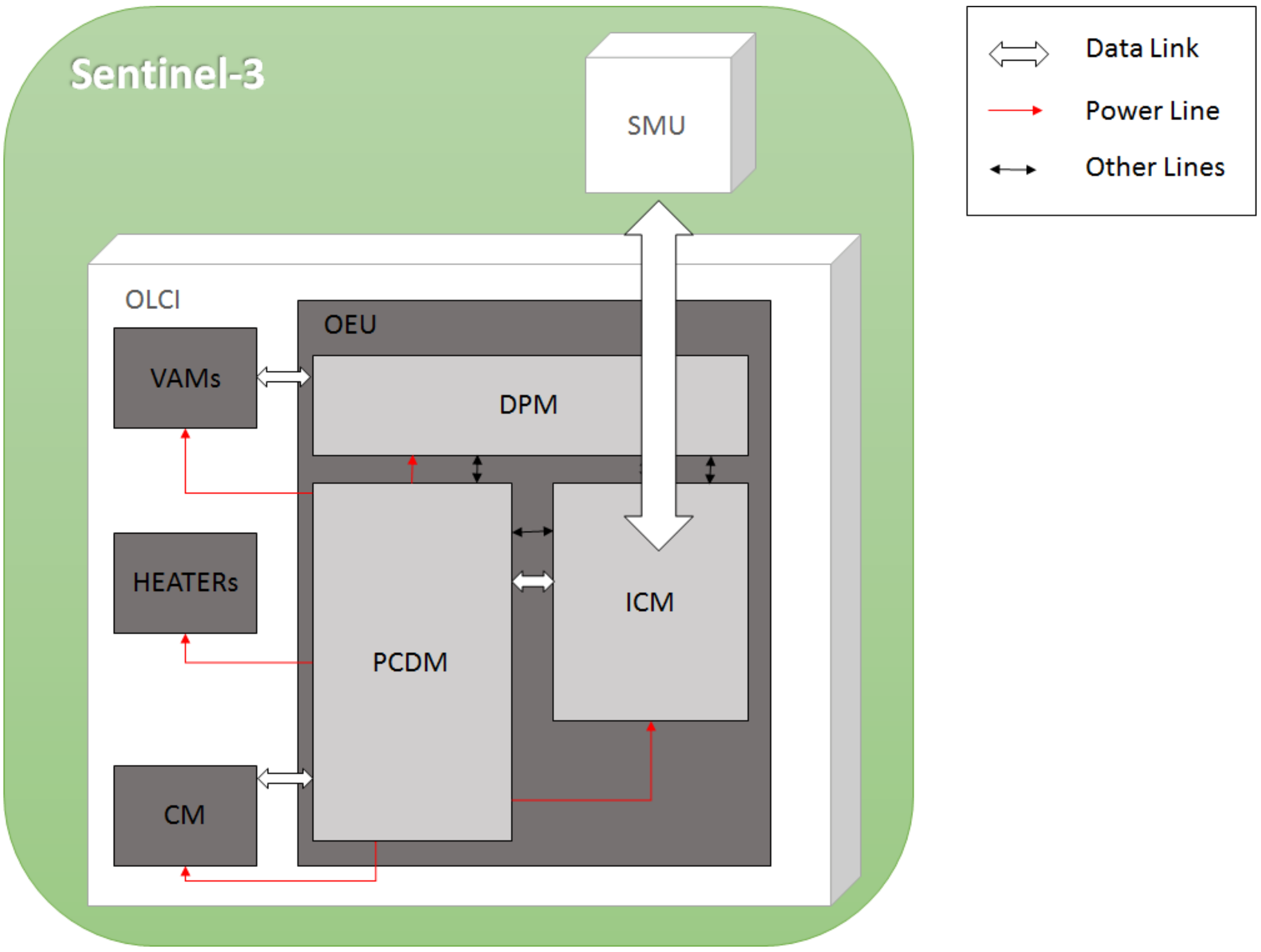


Figure 29. High-level view of the System Architecture of the OLCI instrument

The ICM software integrates both the RSW (Rescue Software) that implements a limited set of functionalities, and the OPSW (Operational Software) that implements the whole ICM functionality. RSW is classified as **CRITICAL software (Level B)**, whereas the OPSW has **MAJOR criticality (Level C)**. The software functionalities covered in CS4 are part of the operational software:

- 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.

The dependability and safety processes were carried out manually in the real development process, and conducted at the same time as the software was developed, with no tool support. In this case study, these processes are tightly coupled to the model-based design and the evidences are automatically generated from the AMASS tool framework. These evidences might derive new requirements or design constraints which can be introduced back in the model (i.e., iterative process).

As described in the Deliverable D1.1 [1], the CS4 usage scenarios will be performed over the same model-based design.

3.4.2. US1: Baseline – Architectural design (Common to all CS4 usage scenarios)

The baseline of all CS4 usage scenarios is the OLCI architectural-driven assurance model. This model will be subsequently analysed to derive the results of the following usage scenarios:

- US1.1 Assessment of components reuse using different execution platforms.
- US1.2 Re-qualification impact of modifying the hardware platform.
- US1.3 AMASS platform analyses to define safety, performance, reliability and availability requirements.

3.4.2.1. STO1 Architecture-Driven Assurance

Table 23 compiles the following information about CS4 architectural-driven assurance technical objectives:

- *Artefact*: Element/process required to fulfil a safety technical objective.
- *Tool feature*: Tool capability required to produce the case study artefact.
- *Status / Implementation phase*:
  - Current development status of each artefact. Possible values: Not started, Started, On-going, Completed, Updated). It shall be noticed that although a specific artefact is already complete, it could be updated during the third iteration, e.g., model updated according to a specific analysis needs and results.
  - The status is specified together with the implementation phase (AMASS iteration) in which that status was reached: 1st iteration, 2nd iteration, 3rd iteration.

Table 23. CS4 functionalities and status

CS4	Artefact	Tool feature	Status / Implementation phase
Architecture Driven Assurance - System Design	System requirements definition	CHESS SysML Requirements Diagram	STARTED / 1st iteration (requirements identified but not formalized)
			COMPLETED / 2nd iteration
	Requirements formalization	Definition of Formal Properties Contract-Based Approach	COMPLETED / 2nd iteration
	Requirements early verification	CHESS Requirements Semantics Analysis CHESS Validation of Contracts	IN-PROGRESS / 2nd iteration
	System architecture	CHESS SysML Block Definition Diagram	COMPLETED / 2nd iteration
	Software architecture	CHESS Class Diagram CHESS SysML Block Definition Diagram CHESS Composite Structure Diagram	COMPLETED / 1st iteration
			UPDATED / 2nd iteration (FLA behaviour added, data types and interface updated, e.g. primitive data types used)
	Functional refinement (internal hierarchical structure)	CHESS SysML Block Definition Diagram CHESS SysML Internal Block Diagram CHESS Contracts Decomposition CHESS Refined Ports CHESS Hierarchical Model View	COMPLETED / 2nd iteration

CS4	Artefact	Tool feature	Status / Implementation phase
		CHESS V&V Results View	
	Components nominal and faulty behaviour	CHESS UML State Machine Diagram	COMPLETED / 2nd iteration
Architecture Driven Assurance – Safety Analysis	Functional early verification	1) Consistency check of formal properties 2) Model checking 3) Contract-based verification of state-machines 4) Contract-refinement verification / Contracts refinement view 5) Contract-based verification of strong/weak contracts	IN-PROGRESS / 2nd iteration
	Model/Contract-based safety analysis	1) Fault Tree Analysis 2) Contract-Based Safety Analysis	IN-PROGRESS / 2nd iteration
Architecture Driven Assurance - Safety Case	Evidence generation	Safety Analyses Results	IN-PROGRESS / 2nd iteration
	Link to architectural entities	Traceability between the assurance case and the architectural entities	NOT STARTED / 2nd iteration
	Document generation	Documentation of the modelling of the system components	IN-PROGRESS / 2nd iteration
Integrity	Tool connection	Usage of external tools: OCRA, nuXmv and xSAP	IN-PROGRESS / 2nd iteration

Below are described the results obtained at the current stage of development (2nd iteration).

CS4 covers the complete architecture-driven assurance process. This process can be mainly divided into three steps:

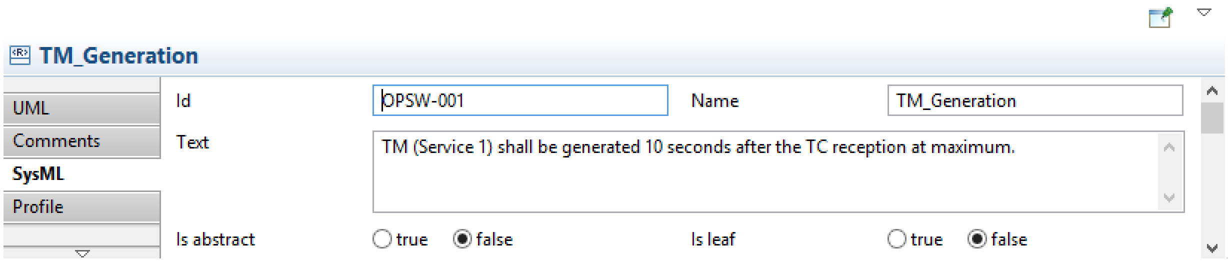
- STEP 1 – System design. Model based design of the OLCI instrument. It starts defining the system requirements in natural language, which are subsequently formalized using formal properties and contracts (including contract refinement). Traceability ensures the fulfilment and quality of these requirements. Secondly, the design covers both the system and software architectures, including nominal and faulty behaviour.
- STEP 2 – Safety analysis. Safety analysis generates the safety artefacts (e.g., model checks, consistency checks, etc.) from the system design. They are used to demonstrate the safety of the system under development.
- STEP 3 – Safety case. Collection of all the relevant output evidences from the safety process: V&V and safety analyses, traceability matrices, documentation, etc. They demonstrate and assure that the OLCI instrument is safe according to its criticality level.

The complete process is described below:

1. System Design:

    1.1. System requirements definition

Requirement are defined using natural language. Each requirement includes an identifier, a title, the text/statement (see Figure 30) and are mapped to the design entities that implement them, e.g., a formal property.


TM_Generation

Id	OPSW-001	Name	TM_Generation
Text	TM (Service 1) shall be generated 10 seconds after the TC reception at maximum.		
Is abstract	true [✓ false]	Is leaf	true [✓ false]

Figure 30. Requirements definition

Figure 31 depicts the SysML Requirements Diagram. In this case, the text of the requirements is not displayed, just the requirement title. Each requirement is associated to the design entity that satisfies it.

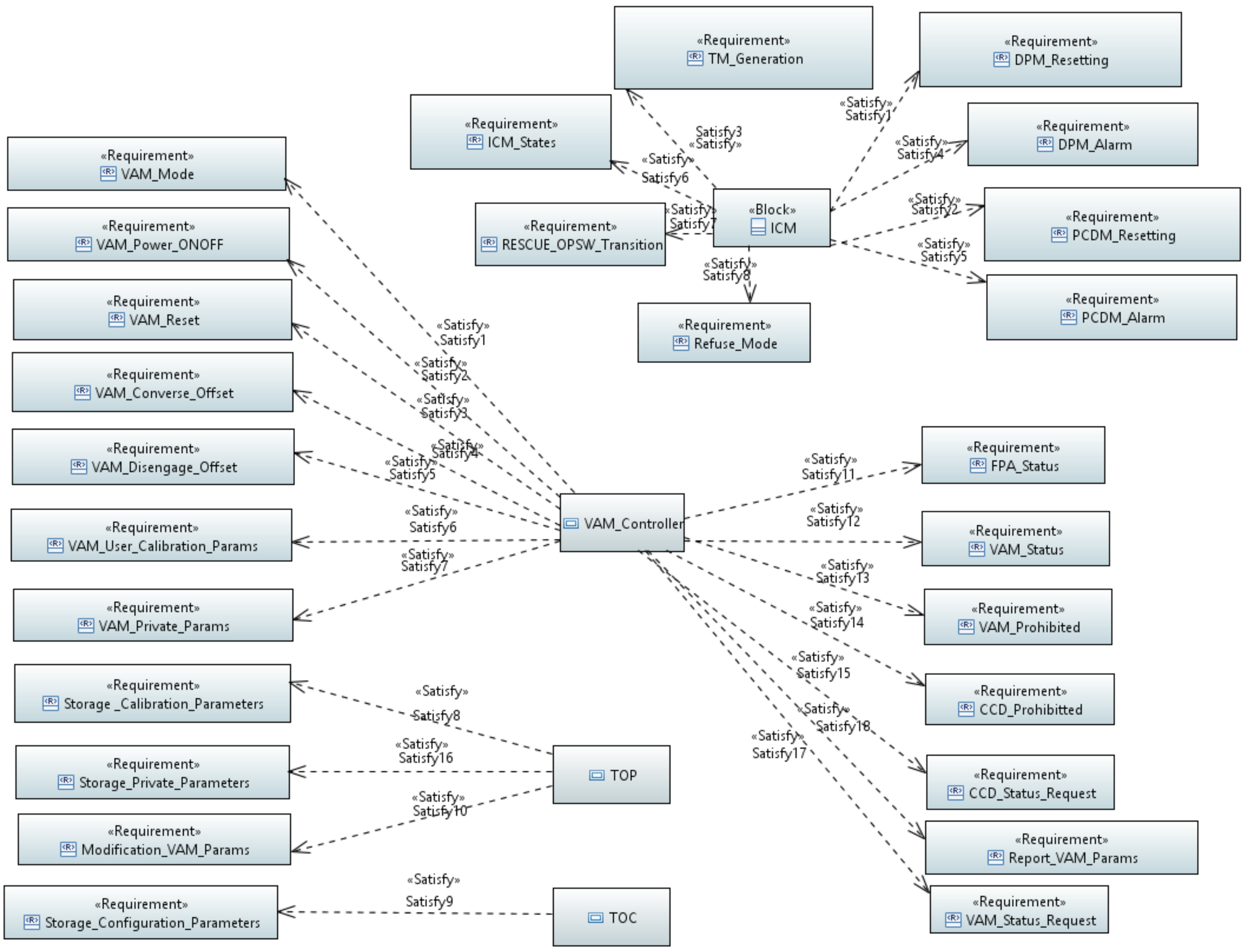
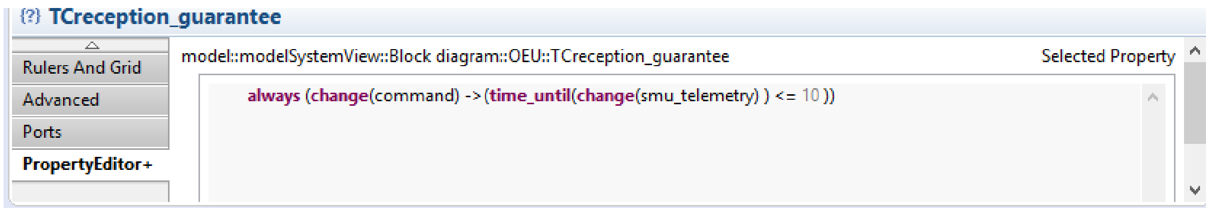


Figure 31. OLCI – Requirements diagram

1.2. Requirements formalization

Requirements are formalized using formal properties and contracts. Firstly, formal properties are defined using the OCRA language. As an example, requirement defined in Figure 30 is formalized as follows:

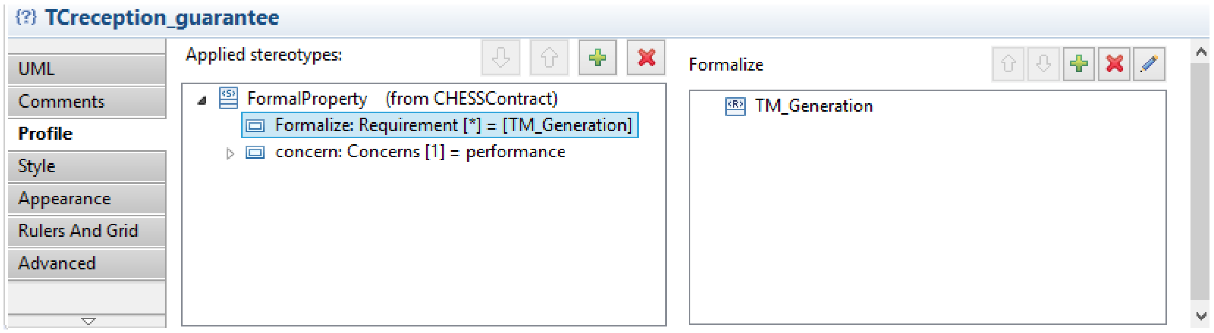


model::modelSystemView::Block diagram::OEU::TCreception_guarantee

always (change(command) ->(time_until(change(smu_telemetry)) <= 10))

Figure 32. OLCI – Requirements formalization

Secondly, formal properties are traced to the associated requirement(s):



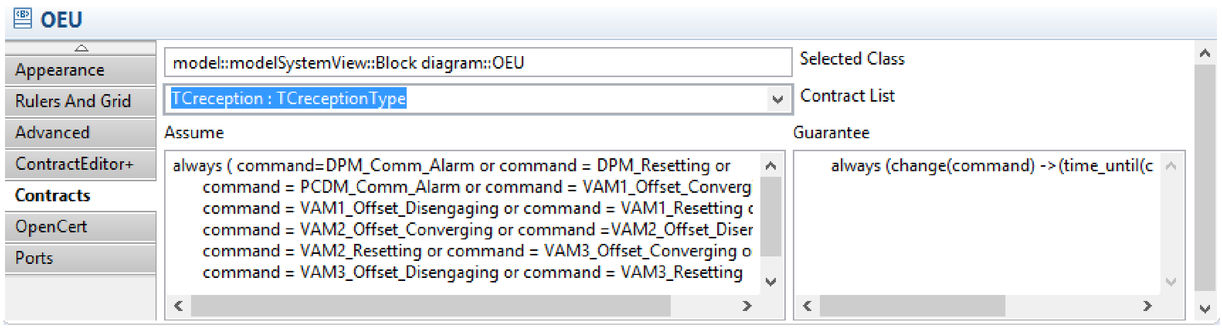
FormalProperty (from CHESSContract)
Formalize: Requirement [*] = [TM_Generation]
concern: Concerns [1] = performance

TM_Generation

Figure 33. OLCI - Requirements traceability

Apart from the specification, all formal properties include a name, a visibility and a context, i.e., the block element they belong to.

Later, formal properties can be used as contracts assumptions and guarantees. Furthermore, each contract is defined either as a Weak or a Strong Contract for this particular use case, and is bound to the corresponding block or component.



model::modelSystemView::Block diagram::OEU

TCreception : TCreceptionType

```
always ( command=DPM_Comm_Alarm or command = DPM_Resetting or
    command = PCDM_Comm_Alarm or command = VAM1_Offset_Converg
    command = VAM1_Offset_Disengaging or command = VAM1_Resetting c
    command = VAM2_Offset_Converging or command =VAM2_Offset_Diser
    command = VAM2_Resetting or command = VAM3_Offset_Converging o
    command = VAM3_Offset_Disengaging or command = VAM3_Resetting
```

always (change(command) ->(time_until(c

Figure 34. OLCI – Contracts definition

Both formal properties and contracts have been defined at the same time as the system architecture and the functional architectural refinement (see step 1.4) in an iterative way.

1.3. Requirements early verification

a) Requirements Semantic Analysis. This analysis identifies and removes requirement defects.

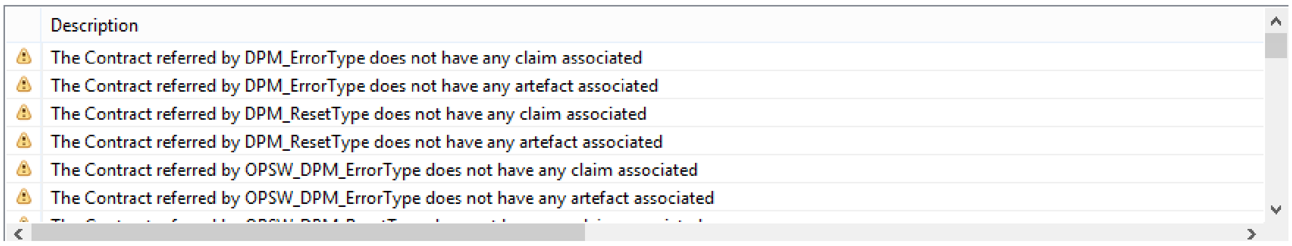
Steps: Select an *OLCI_Instrument* component in the Block Definition Diagram → Validation → V&V Manager.

Result: This functionality is not supported in current version of the AMASS tool. Currently, the execution finishes with a java.lang.NullPointerException. This error has been reported to the tool's provider to fix it in the next release.

b) Contract validation. This analysis checks the contracts correctness.

Steps: Right-click on the *OLCI_Instrument/OEU/ICM/OPSW* component → Validation → Validate contracts for assurance.

Result: The validation finishes successfully. Only some warnings appear due to the contracts have not claim/artefact associated. These warnings are not relevant for this use case as no assurance case is defined. The warnings do not affect the use case functionality.



Description
The Contract referred by DPM_ErrorType does not have any claim associated
The Contract referred by DPM_ErrorType does not have any artefact associated
The Contract referred by DPM_ResetType does not have any claim associated
The Contract referred by DPM_ResetType does not have any artefact associated
The Contract referred by OPSW_DPM_ErrorType does not have any claim associated
The Contract referred by OPSW_DPM_ErrorType does not have any artefact associated

Figure 35. OLCI– Result of validation of contracts for assurance

1.4. System/software architecture and functional refinement (internal hierarchical structure).

Firstly, data types are defined. The data types are used by both system and software architectures. This facilitates the HW-SW integration process. A set of data types was defined during the first iteration of the case study, but they have been completed and refined. For example, now UML primitive data types are used instead of new basic data types to be aligned with the tool requirements.

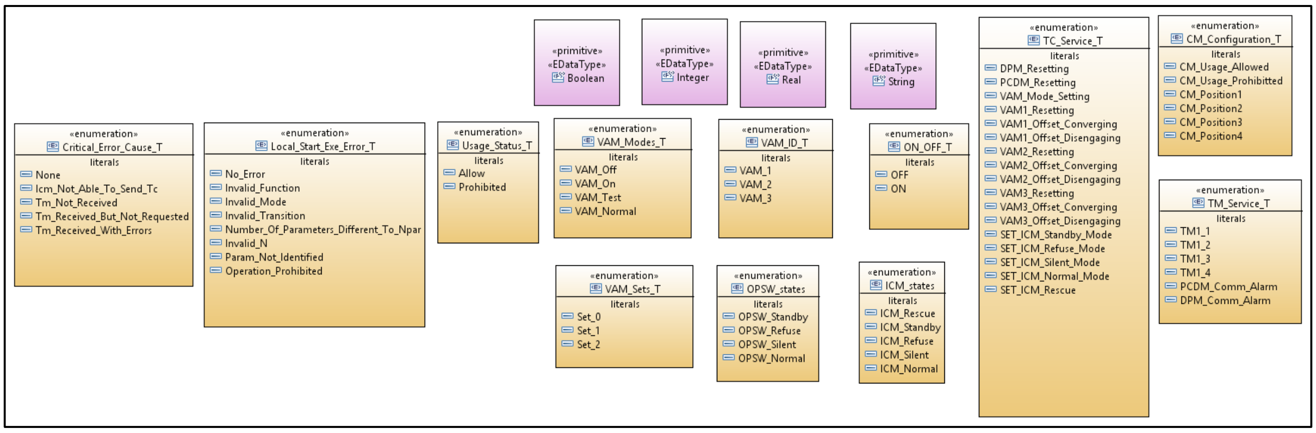


Figure 36. OLCI – Data types

Secondly, the OLCI element is represented by the *OLCI_Instrument* System Block in the Block Definition Diagram. This main block is subsequently decomposed.

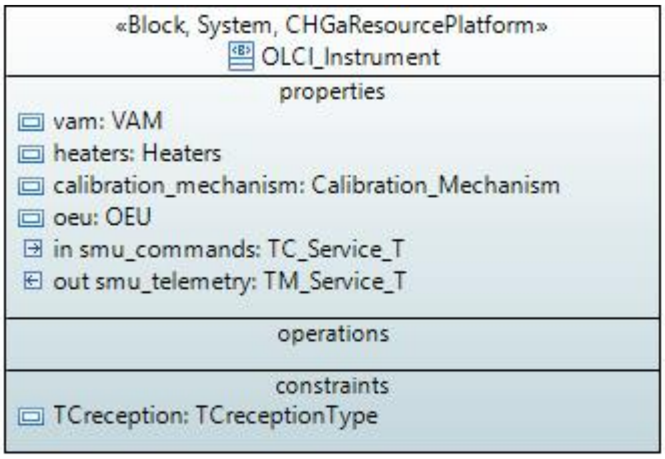

«Block, System, CHGaResourcePlatform»
OLCI_Instrument
properties
vam: VAM
heaters: Heaters
calibration_mechanism: Calibration_Mechanism
oeu: OEU
in smu_commands: TC_Service_T
out smu_telemetry: TM_Service_T
operations
constraints
TCreception: TCreceptionType

Figure 37. OLCI – System block

A hierarchical decomposition of the System Block is performed. The set of sub-blocks is defined as well as the dependencies among them. Figure 38 shows the complete Block Definition Diagram, highlighting in yellow the OPSW. The SW architecture of the OPSW is defined in the "Components View".

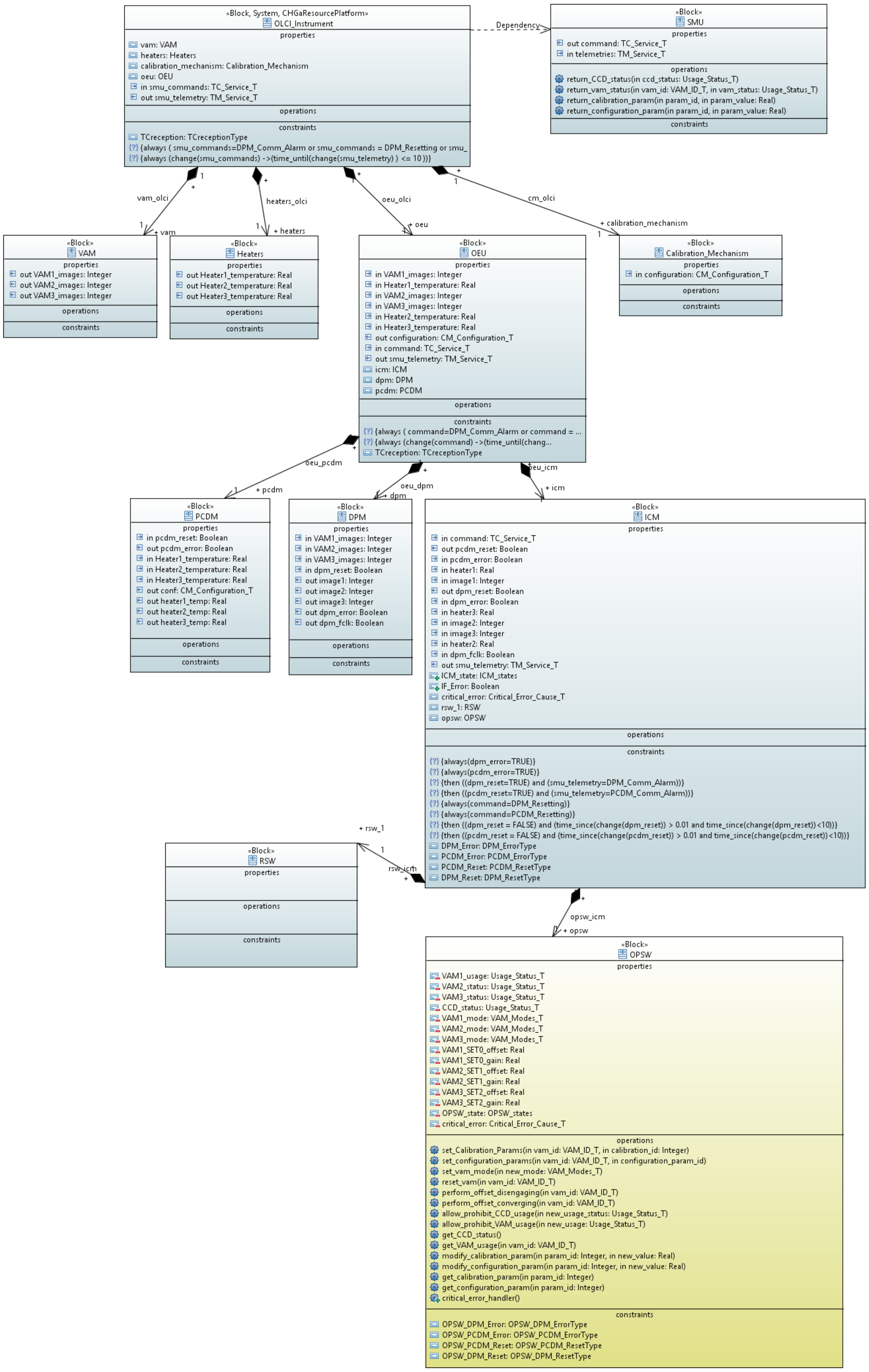

«Block, System, CHGaResourcePlatform»
OLCI_Instrument
properties
vam: VAM
heaters: Heaters
calibration_mechanism: Calibration_Mechanism
oeu: OEU
in smu_commands: TC_Service_T
out smu_telemetry: TM_Service_T
operations
constraints
TCreception: TCreceptionType
{always (smu_commands=DPM_Comm_Alarm or smu_commands = DPM_Resetting or smu_
{always (change(smu_commands) ->(time_until(change(smu_telemetry)) <= 10))}
Dependency
«Block»
SMU
properties
out command: TC_Service_T
in telemetries: TM_Service_T
operations
return_CCD_status(in ccd_status: Usage_Status_T)
return_vam_status(in vam_id: VAM_ID_T, in vam_status: Usage_Status_T)
return_calibration_param(in param_id, in param_value: Real)
return_configuration_param(in param_id, in param_value: Real)
constraints
vam_olci
1
+ vam
heaters_olci
1
+ heaters
oeu_olci
+ oeu
cm_olci
1
+ calibration_mechanism
«Block»
VAM
properties
out VAM1_images: Integer
out VAM2_images: Integer
out VAM3_images: Integer
operations
constraints
«Block»
Heaters
properties
out Heater1_temperature: Real
out Heater2_temperature: Real
out Heater3_temperature: Real
operations
constraints
«Block»
OEU
properties
in VAM1_images: Integer
in Heater1_temperature: Real
in VAM2_images: Integer
in VAM3_images: Integer
in Heater2_temperature: Real
in Heater3_temperature: Real
out configuration: CM_Configuration_T
in command: TC_Service_T
out smu_telemetry: TM_Service_T
icm: ICM
dpm: DPM
pcdm: PCDM
operations
constraints
{always (command=DPM_Comm_Alarm or command = ...
{always (change(command) ->(time_until(chang...
TCreception: TCreceptionType
«Block»
Calibration_Mechanism
properties
in configuration: CM_Configuration_T
operations
constraints
oeu_pcdm
1
+ pcdm
oeu_dpm
+ dpm
oeu_icm
+ icm
«Block»
PCDM
properties
in pcdm_reset: Boolean
out pcdm_error: Boolean
in Heater1_temperature: Real
in Heater2_temperature: Real
in Heater3_temperature: Real
out conf: CM_Configuration_T
out heater1_temp: Real
out heater2_temp: Real
out heater3_temp: Real
operations
constraints
«Block»
DPM
properties
in VAM1_images: Integer
in VAM2_images: Integer
in VAM3_images: Integer
in dpm_reset: Boolean
out image1: Integer
out image2: Integer
out image3: Integer
out dpm_error: Boolean
out dpm_fclk: Boolean
operations
constraints
«Block»
ICM
properties
in command: TC_Service_T
out pcdm_reset: Boolean
in pcdm_error: Boolean
in heater1: Real
in image1: Integer
out dpm_reset: Boolean
in dpm_error: Boolean
in heater3: Real
in image2: Integer
in image3: Integer
in heater2: Real
in dpm_fclk: Boolean
out smu_telemetry: TM_Service_T
ICM_state: ICM_states
IF_Error: Boolean
critical_error: Critical_Error_Cause_T
rsw_1: RSW
opsw: OPSW
operations
constraints
{always(dpm_error=TRUE)}
{always(pcdm_error=TRUE)}
{then ((dpm_reset=TRUE) and (smu_telemetry=DPM_Comm_Alarm))}
{then ((pcdm_reset=TRUE) and (smu_telemetry=PCDM_Comm_Alarm))}
{always(command=DPM_Resetting)}
{always(command=PCDM_Resetting)}
{then ((dpm_reset = FALSE) and (time_since(change(dpm_reset)) > 0.01 and time_since(change(dpm_reset))<10))}
{then ((pcdm_reset = FALSE) and (time_since(change(pcdm_reset)) > 0.01 and time_since(change(pcdm_reset))<10))}
DPM_Error: DPM_ErrorType
PCDM_Error: PCDM_ErrorType
PCDM_Reset: PCDM_ResetType
DPM_Reset: DPM_ResetType
+ rsw_1
1
rsw_icm
«Block»
RSW
properties
operations
constraints
opsw_icm
+ opsw
«Block»
OPSW
properties
VAM1_usage: Usage_Status_T
VAM2_status: Usage_Status_T
VAM3_status: Usage_Status_T
CCD_status: Usage_Status_T
VAM1_mode: VAM_Modes_T
VAM2_mode: VAM_Modes_T
VAM3_mode: VAM_Modes_T
VAM1_SET0_offset: Real
VAM1_SET0_gain: Real
VAM2_SET1_offset: Real
VAM2_SET1_gain: Real
VAM3_SET2_offset: Real
VAM3_SET2_gain: Real
OPSW_state: OPSW_states
critical_error: Critical_Error_Cause_T
operations
set_Calibration_Params(in vam_id: VAM_ID_T, in calibration_id: Integer)
set_configuration_params(in vam_id: VAM_ID_T, in configuration_param_id)
set_vam_mode(in new_mode: VAM_Modes_T)
reset_vam(in vam_id: VAM_ID_T)
perform_offset_disengaging(in vam_id: VAM_ID_T)
perform_offset_converging(in vam_id: VAM_ID_T)
allow_prohibit_CCD_usage(in new_usage_status: Usage_Status_T)
allow_prohibit_VAM_usage(in new_usage: Usage_Status_T)
get_CCD_status()
get_VAM_usage(in vam_id: VAM_ID_T)
modify_calibration_param(in param_id: Integer, in new_value: Real)
modify_configuration_param(in param_id: Integer, in new_value: Real)
get_calibration_param(in param_id: Integer)
get_configuration_param(in param_id: Integer)
critical_error_handler()
constraints
OPSW_DPM_Error: OPSW_DPM_ErrorType
OPSW_PCDM_Error: OPSW_PCDM_ErrorType
OPSW_PCDM_Reset: OPSW_PCDM_ResetType
OPSW_DPM_Reset: OPSW_DPM_ResetType

Figure 38. OLCI – Block Definition Diagram

The components defined in previous Block Definition Diagram are instantiated in the Internal Block Diagram and their input/output ports connected among them.

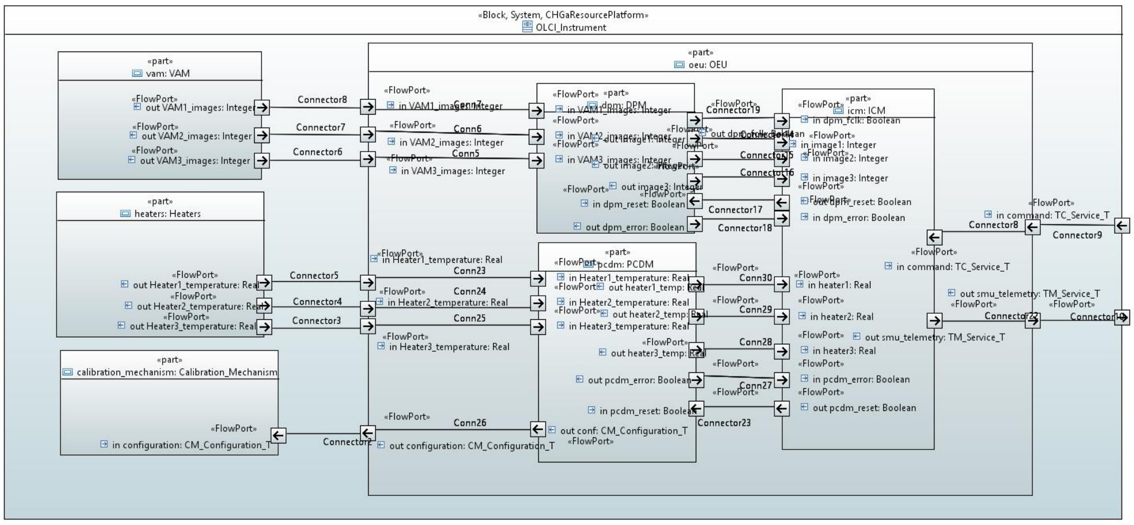

«Block, System, CHGaResourcePlatform»
OLCI_Instrument
«part» vam: VAM
out VAM1_images: Integer
out VAM2_images: Integer
out VAM3_images: Integer
«part» heaters: Heaters
out Heater1_temperature: Real
out Heater2_temperature: Real
out Heater3_temperature: Real
«part» calibration_mechanism: Calibration_Mechanism
in configuration: CM_Configuration_T
«part» oeu: OEU
«part» dpm: DPM
«part» icm: ICM
«part» pcdm: PCDM
in smu_commands: TC_Service_T
out smu_telemetry: TM_Service_T

Figure 39. OLCI – Internal Blocks Diagram

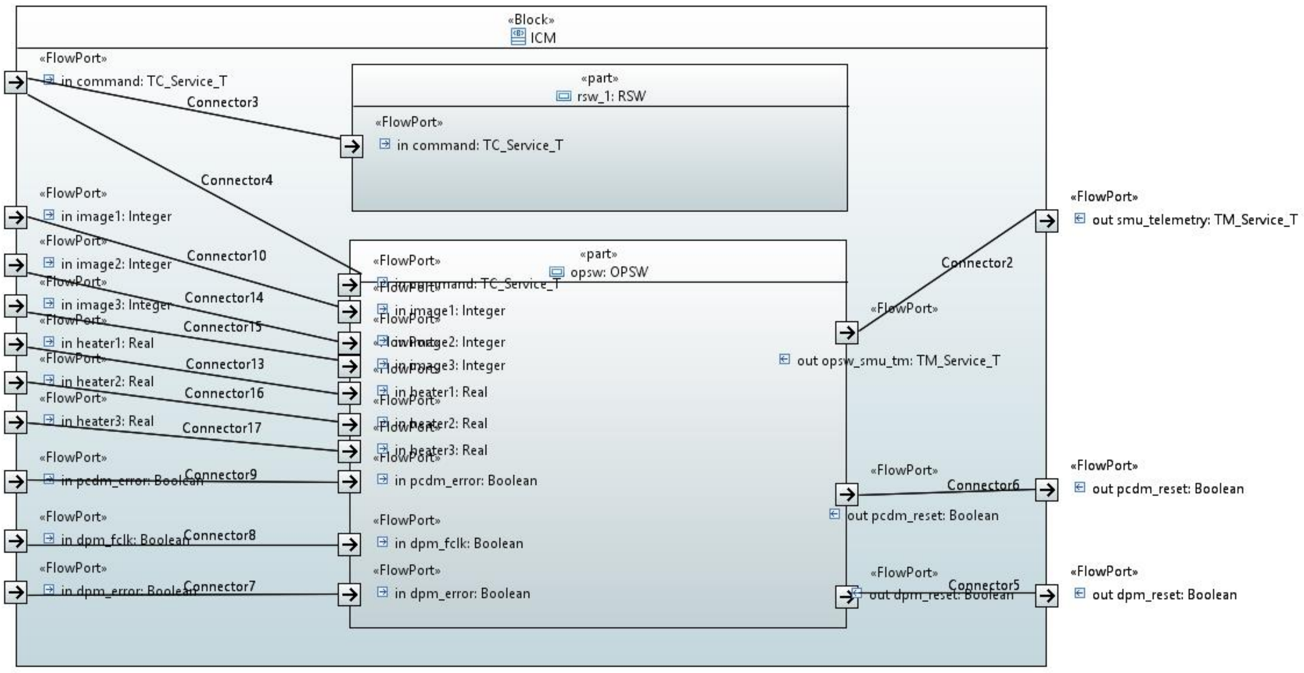
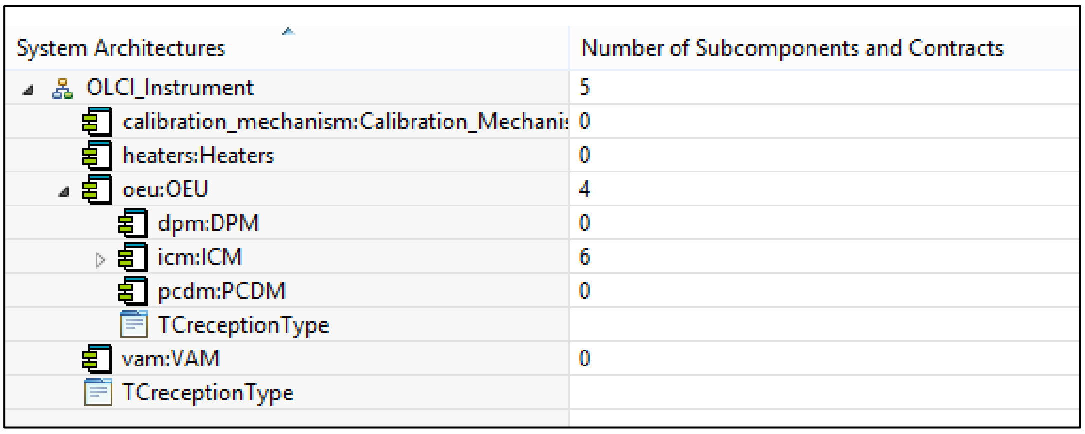


Figure 40. ICM – Internal Blocks Diagram

The system design can be visualised using different views:

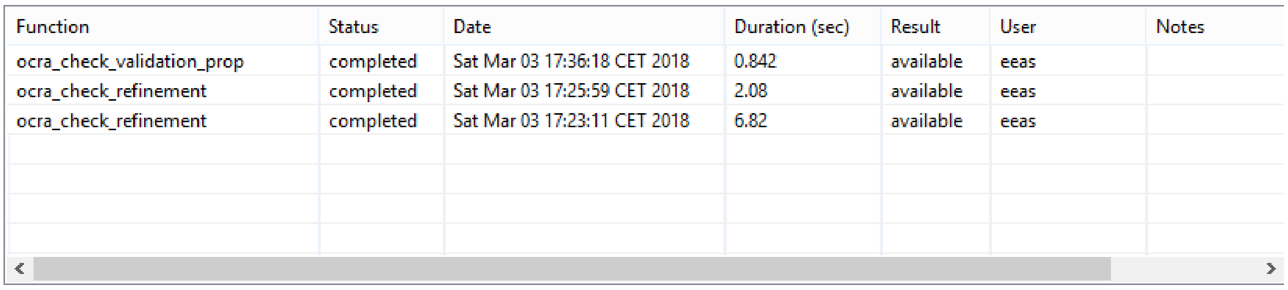
a) Hierarchical model view – It shows the hierarchical decomposition of the *OLCI_Instrument* component, as well as specifies the number of subcomponents and contracts.



System Architectures	Number of Subcomponents and Contracts
OLCI_Instrument	5
calibration_mechanism:Calibration_Mechani:	0
heaters:Heaters	0
oeu:OEU	4
dpm:DPM	0
icm:ICM	6
pcdm:PCDM	0
TCreceptionType	
vam:VAM	0
TCreceptionType	

Figure 41. OLCI – Hierarchical model view

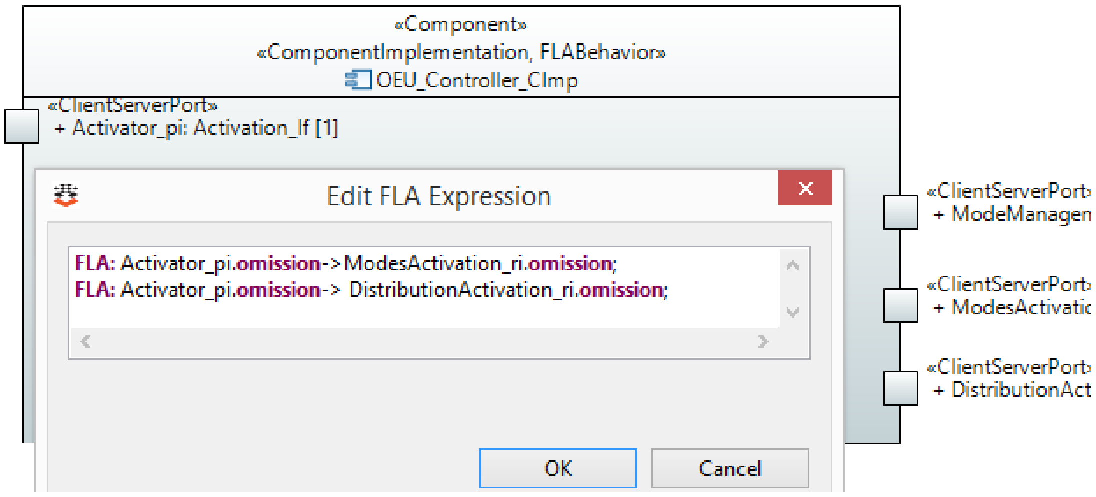
b) V&V Results View – It includes the results of the analyses executed.



Function	Status	Date	Duration (sec)	Result	User	Notes
ocra_check_validation_prop	completed	Sat Mar 03 17:36:18 CET 2018	0.842	available	eeas	
ocra_check_refinement	completed	Sat Mar 03 17:25:59 CET 2018	2.08	available	eeas	
ocra_check_refinement	completed	Sat Mar 03 17:23:11 CET 2018	6.82	available	eeas	

Figure 42. OLCI - V&V Results View

Apart from the system architecture, the OPSW software architecture is designed. The design was included in D1.4 "AMASS Demonstrators (a)" [4]. Current iteration adds FLA information. For example, the figure below shows the FLA Expressions of *OEU_Controller_CImpl*.



```
FLA: Activator_pi.omission->ModesActivation_ri.omission;
FLA: Activator_pi.omission-> DistributionActivation_ri.omission;
```

Figure 43. Edition of FLA Expressions

And the OPSW Composite Diagram is decorated with *FPTCSpecification* properties that associate specific failures (e.g., omission) to input ports.

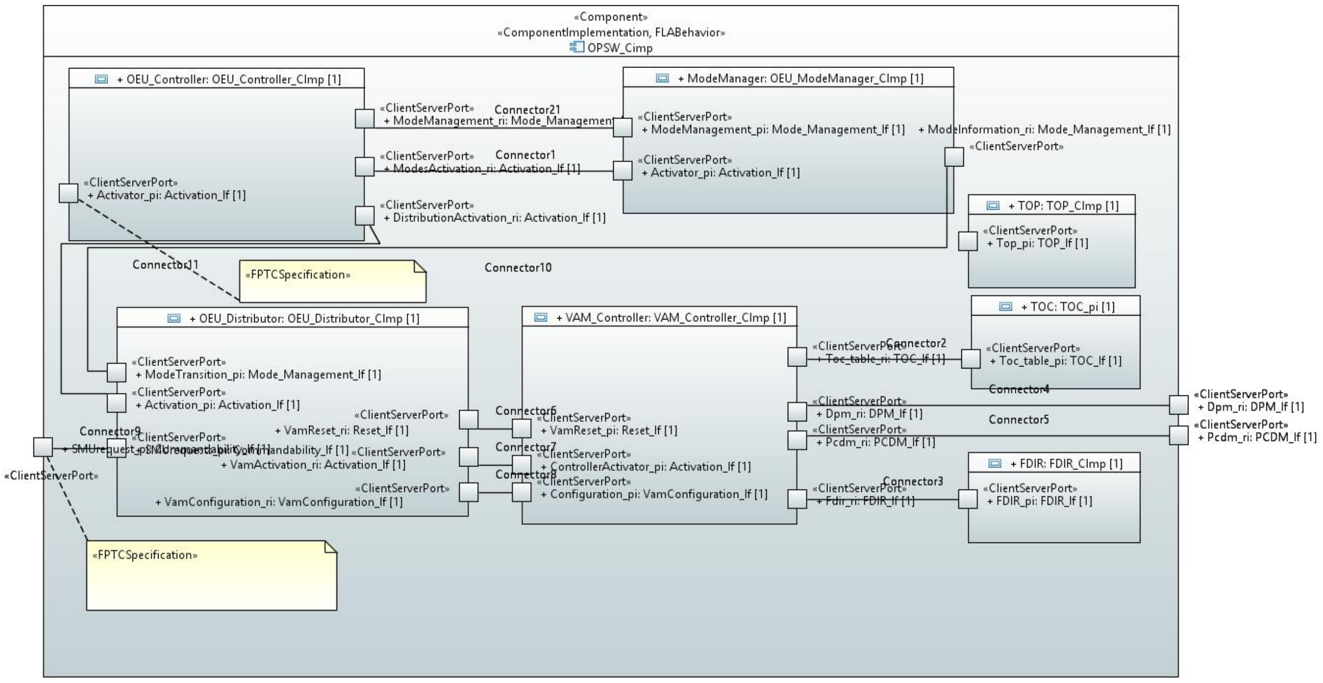

«Component»
«ComponentImplementation, FLABehavior»
OPSW_Cimp
+ OEU_Controller: OEU_Controller_CImp [1]
«ClientServerPort» + ModeManagement_ri: Mode_Management
Connector21
«ClientServerPort» + ModesActivation_ri: Activation_If [1]
Connector1
«ClientServerPort» + Activator_pi: Activation_If [1]
«ClientServerPort» + DistributionActivation_ri: Activation_If [1]
+ ModeManager: OEU_ModeManager_CImp [1]
«ClientServerPort» + ModeManagement_pi: Mode_Management_If [1]
+ ModeInformation_ri: Mode_Management_If [1] «ClientServerPort»
«ClientServerPort» + Activator_pi: Activation_If [1]
+ TOP: TOP_CImp [1]
«ClientServerPort» + Top_pi: TOP_If [1]
Connector11
«FPTCSpecification»
Connector10
+ OEU_Distributor: OEU_Distributor_CImp [1]
«ClientServerPort» + ModeTransition_pi: Mode_Management_If [1]
«ClientServerPort» + Activation_pi: Activation_If [1]
«ClientServerPort» + VamReset_ri: Reset_If [1]
«ClientServerPort» + VamActivation_ri: Activation_If [1]
«ClientServerPort» + VamConfiguration_ri: VamConfiguration_If [1]
Connector9
«ClientServerPort»
+ SMUrequest_pi: SMUrequest_ri: Commandability_If [1]
+ VAM_Controller: VAM_Controller_CImp [1]
Connector6
«ClientServerPort» + VamReset_pi: Reset_If [1]
Connector7
«ClientServerPort» + ControllerActivator_pi: Activation_If [1]
Connector8
«ClientServerPort» + Configuration_pi: VamConfiguration_If [1]
«ClientServerPort» + Toc_table_ri: TOC_If [1]
Connector2
+ TOC: TOC_pi [1]
«ClientServerPort» + Toc_table_pi: TOC_If [1]
Connector4
«ClientServerPort» + Dpm_ri: DPM_If [1]
Connector5
«ClientServerPort» + Pcdm_ri: PCDM_If [1]
«ClientServerPort» + Dpm_ri: DPM_If [1]
«ClientServerPort» + Pcdm_ri: PCDM_If [1]
«ClientServerPort» + Fdir_ri: FDIR_If [1]
Connector3
+ FDIR: FDIR_CImp [1]
«ClientServerPort» + FDIR_pi: FDIR_If [1]
«FPTCSpecification»

Figure 44. OPSW - Composite Diagram

1.5. Components nominal and faulty behaviour

Nominal and faulty behaviour is defined using UML State Machine Diagrams. In this case the behaviour of *ICM* and *OPSW* components is specified. It includes its operational modes, the events that trigger the change of modes and the effects of these changes using SMV language. In this version of the prototype the stereotypes of the faulty model are manually defined.

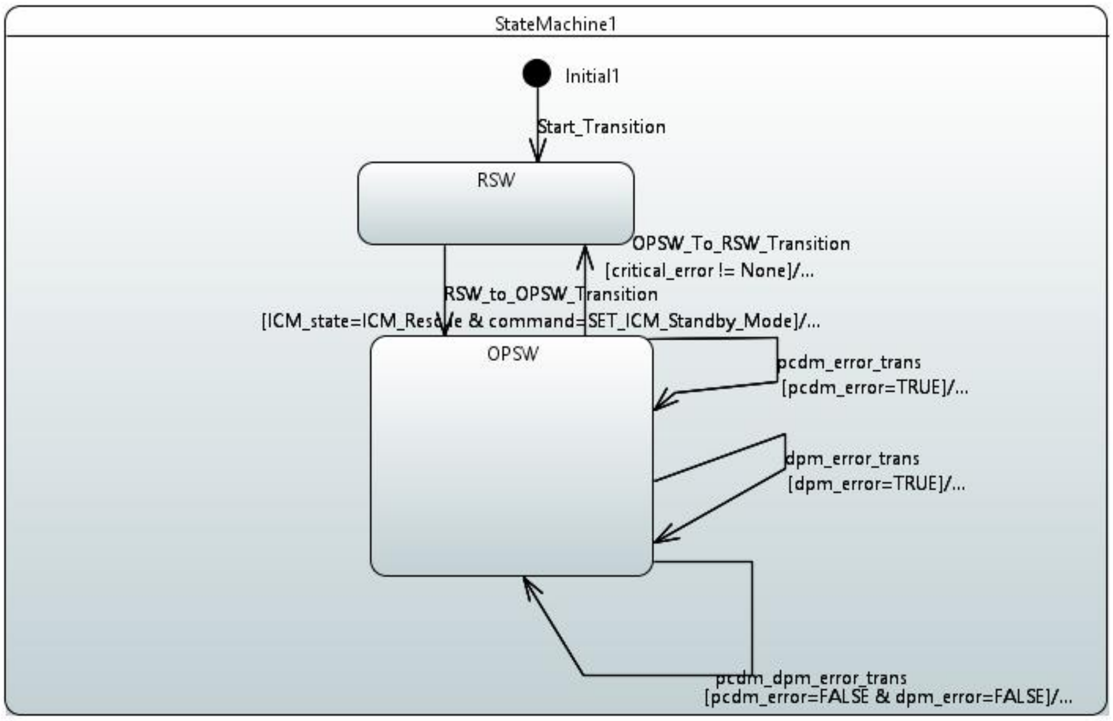


Figure 45. ICM – Nominal behaviour

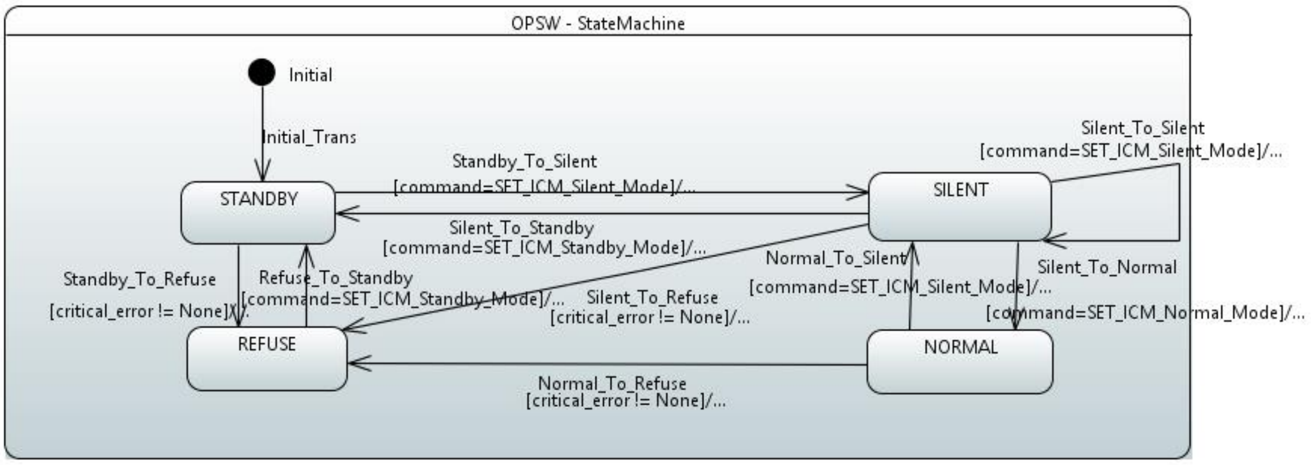

OPSW - StateMachine
Initial
Initial_Trans
STANDBY
SILENT
REFUSE
NORMAL
Standby_To_Silent [command=SET_ICM_Silent_Mode]/...
Silent_To_Standby [command=SET_ICM_Standby_Mode]/...
Silent_To_Silent [command=SET_ICM_Silent_Mode]/...
Standby_To_Refuse [critical_error != None]/...
Refuse_To_Standby [command=SET_ICM_Standby_Mode]/...
Silent_To_Refuse [critical_error != None]/...
Normal_To_Silent [command=SET_ICM_Silent_Mode]/...
Silent_To_Normal [command=SET_ICM_Normal_Mode]/...
Normal_To_Refuse [critical_error != None]/...

Figure 46. OPSW – Nominal behaviour

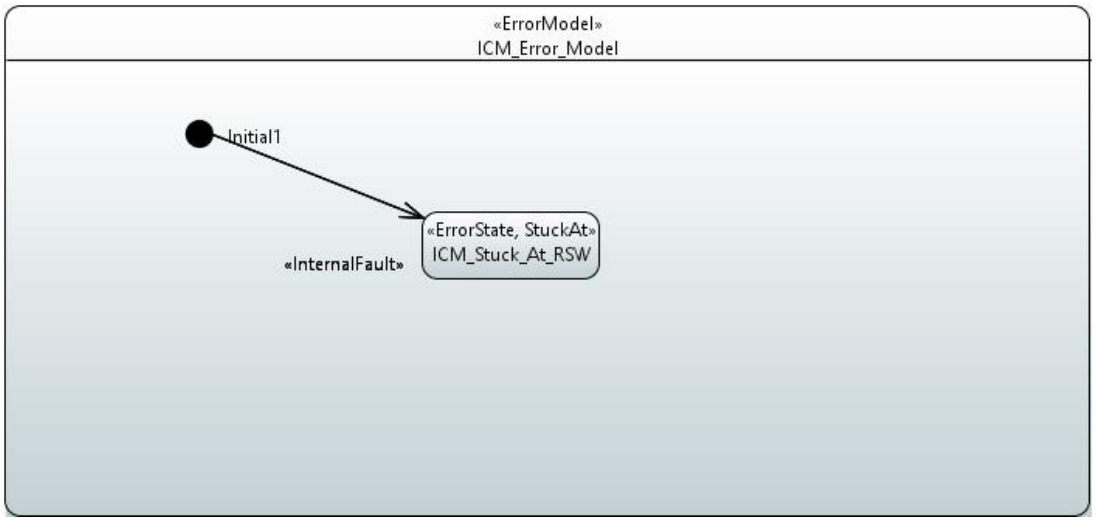


Figure 47. ICM – Faulty behaviour

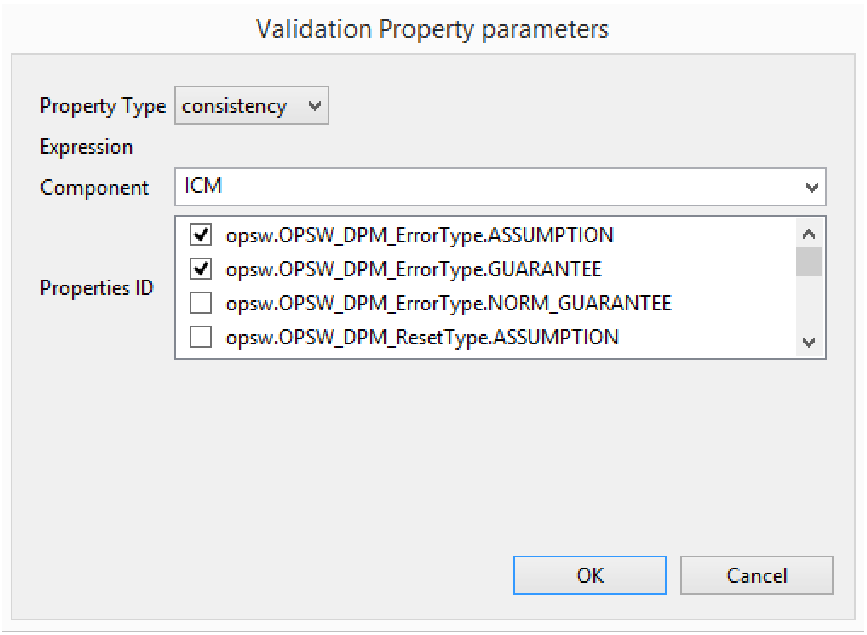
2. <u>Safety Analyses – Functional early verification</u>

The analyses tools do not support the system model as it has been designed. The following model updates/simplifications have been done to conduct the safety analyses:

- CHESS supports only state machines with discrete time. The expressions *time_until* and *time_since* have been removed to check the contract implementations. The model of time is 'Discrete' instead of 'Hybrid'.
- CHESS does not support the element Operation in the state machines. They should be replaced with events (i.e., the UML signals in CHESS). In our case, it has been replaced by the value of properties.
- CHESS does not support the definition of sate machines in non-leaf components. The ICM state machine has been removed.

2.1. Consistency check of formal properties

Steps: Select *OLCI_Instrument*/*OEU*/*ICM*/*OPSW* component → CHESS → Validation → Check validation property on selected component. Then, it is necessary to specify: i) the model of time that is being used by the system (i.e., 'Discrete'), ii) the property type (i.e., consistency, possibility or entailment), iii) the component under analysis and iv) the associated properties ID (see Figure 48).

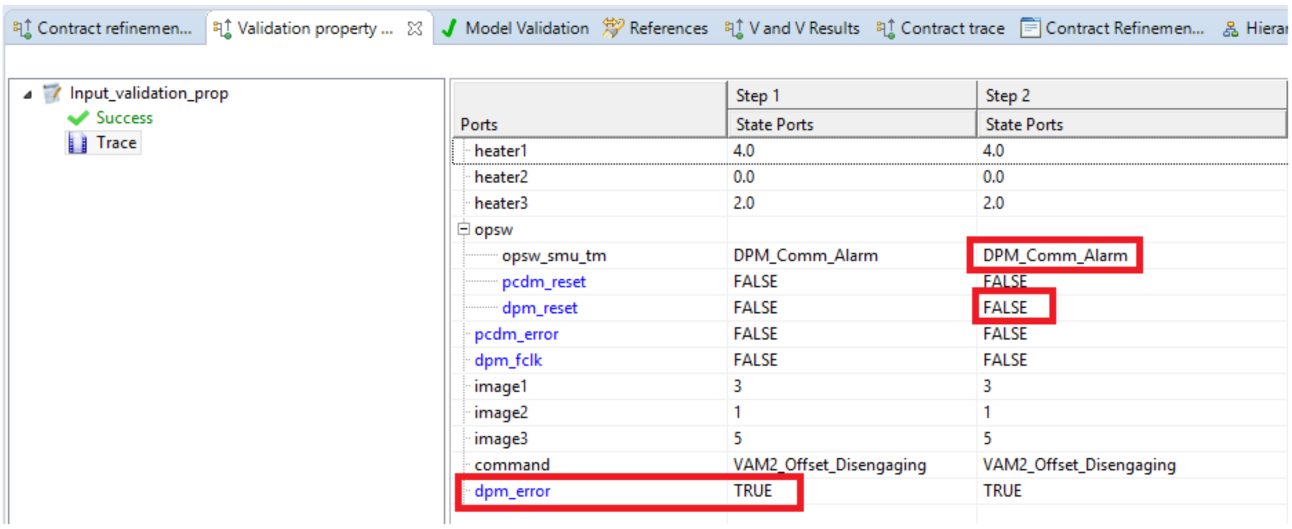

Validation Property parameters

Property Type: consistency
Expression
Component: ICM
Properties ID:
- [x] opsw.OPSW_DPM_ErrorType.ASSUMPTION
- [x] opsw.OPSW_DPM_ErrorType.GUARANTEE
- [ ] opsw.OPSW_DPM_ErrorType.NORM_GUARANTEE
- [ ] opsw.OPSW_DPM_ResetType.ASSUMPTION

Figure 48. OLCI – Validation property parameters of ICM component – consistency check

Results: The results generated for the properties defined in Figure 48 are displayed below. As it is depicted, the analysis finishes successfully. Additionally, the trace of the different ports is displayed.

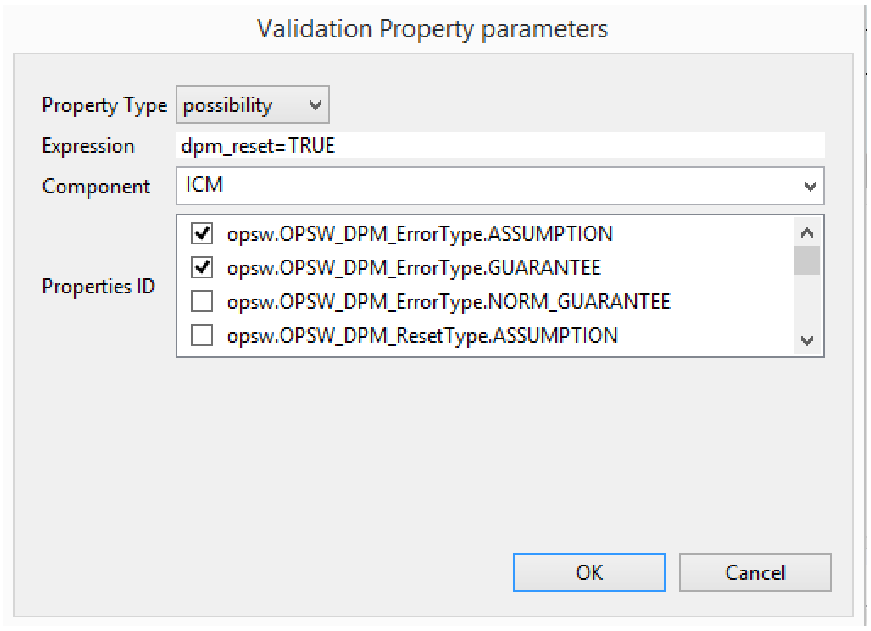
In the example below, it is checked that whenever the *dpm_error* is set to TRUE, during the next step the *opsw_smu_tm* shall be set to *DPM_Comm_Alarm* and the *dpm_reset* to FALSE.



	Step 1	Step 2
Ports	State Ports	State Ports
heater1	4.0	4.0
heater2	0.0	0.0
heater3	2.0	2.0
opsw		
opsw_smu_tm	DPM_Comm_Alarm	DPM_Comm_Alarm
pcdm_reset	FALSE	FALSE
dpm_reset	FALSE	FALSE
pcdm_error	FALSE	FALSE
dpm_fclk	FALSE	FALSE
image1	3	3
image2	1	1
image3	5	5
command	VAM2_Offset_Disengaging	VAM2_Offset_Disengaging
dpm_error	TRUE	TRUE

Figure 49. Validation property results of ICM component properties – consistency check

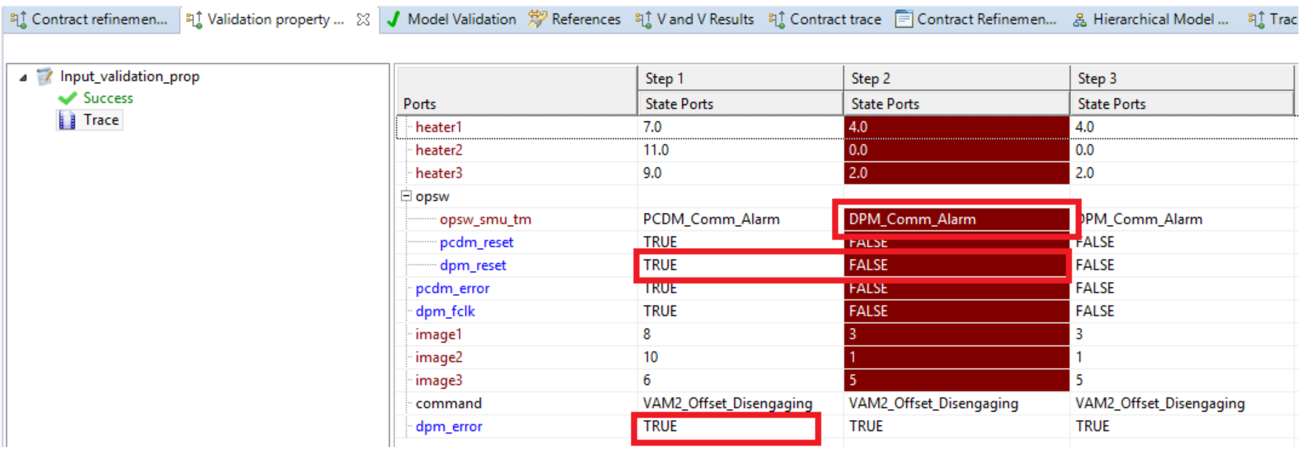
A similar validation check is made but setting the *dpm_reset* to TRUE, in order to check that it is set to FALSE in the next step. This is the analysis configuration:



Property Type: possibility
Expression: dpm_reset=TRUE
Component: ICM
Properties ID:
- [x] opsw.OPSW_DPM_ErrorType.ASSUMPTION
- [x] opsw.OPSW_DPM_ErrorType.GUARANTEE
- [ ] opsw.OPSW_DPM_ErrorType.NORM_GUARANTEE
- [ ] opsw.OPSW_DPM_ResetType.ASSUMPTION

Figure 50. OLCI – Validation property parameters of ICM component – possibility check

The result is the following:



	Step 1	Step 2	Step 3
Ports	State Ports	State Ports	State Ports
heater1	7.0	4.0	4.0
heater2	11.0	0.0	0.0
heater3	9.0	2.0	2.0
opsw			
opsw_smu_tm	PCDM_Comm_Alarm	DPM_Comm_Alarm	DPM_Comm_Alarm
pcdm_reset	TRUE	FALSE	FALSE
dpm_reset	TRUE	FALSE	FALSE
pcdm_error	TRUE	FALSE	FALSE
dpm_fclk	TRUE	FALSE	FALSE
image1	8	3	3
image2	10	1	1
image3	6	5	5
command	VAM2_Offset_Disengaging	VAM2_Offset_Disengaging	VAM2_Offset_Disengaging
dpm_error	TRUE	TRUE	TRUE

Figure 51. Validation property results of ICM component properties – possibility check

2.2. Model checking – Verification of other state machine properties apart from the contracts.

Steps: Right-click on a component → CHESS → Functional verification → Model checking on the selected component. Then, it is necessary to specify: i) the model of time that is being used by the system (i.e., 'Discrete'), as well as ii) nuXmv parameters (i.e., check type, algorithm type and property).

Results: It verifies the component behaviour. It also includes all the behaviours from the root to the leaves of the selected component. This functionality not supported in current version of the AMASS Tool. There is a bug in the plugin that converts the state machine in .smv file. The variables with enum type have the enumvalues duplicated. The tool's provider is aware of this behaviour to fix it in the next release.

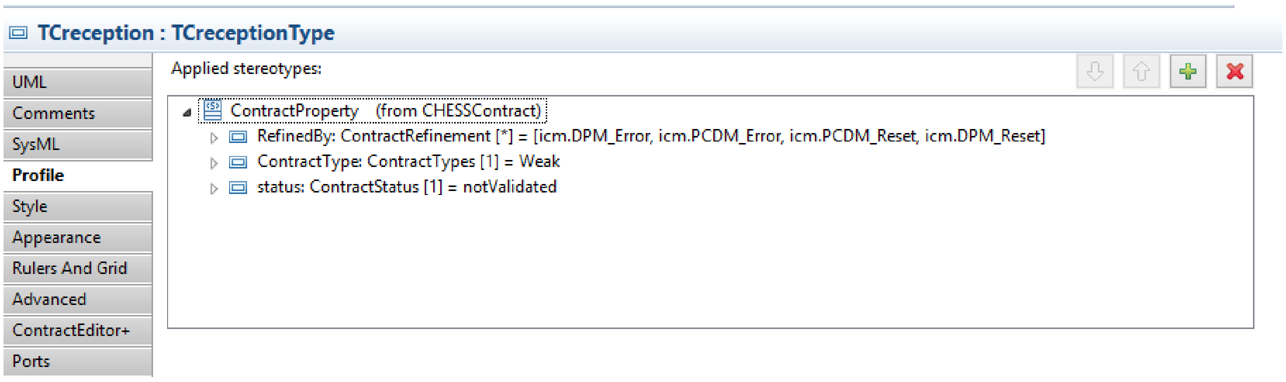
2.3. Contract-based verification of state-machines

Steps: Right click on the *ICM/OPSW* component (i.e., components with state-machines associated) → CHESS → Functional verification → Check contract implementation on selected component.

Results: Functionality not supported in current version of the AMASS Tool. This functionality not supported in current version of the AMASS Tool. There is a bug in the plugin that converts the state machine in .smv file. The variables with enum type have the enumvalues duplicated. The tool's provider is aware of this behaviour to fix it in the next release.

2.4. Contract-refinement verification

Once contracts are defined, contract refinements are configured. For example, *TMreception* is refined by a set of contracts of the ICM component.



TCreception : TCreceptionType

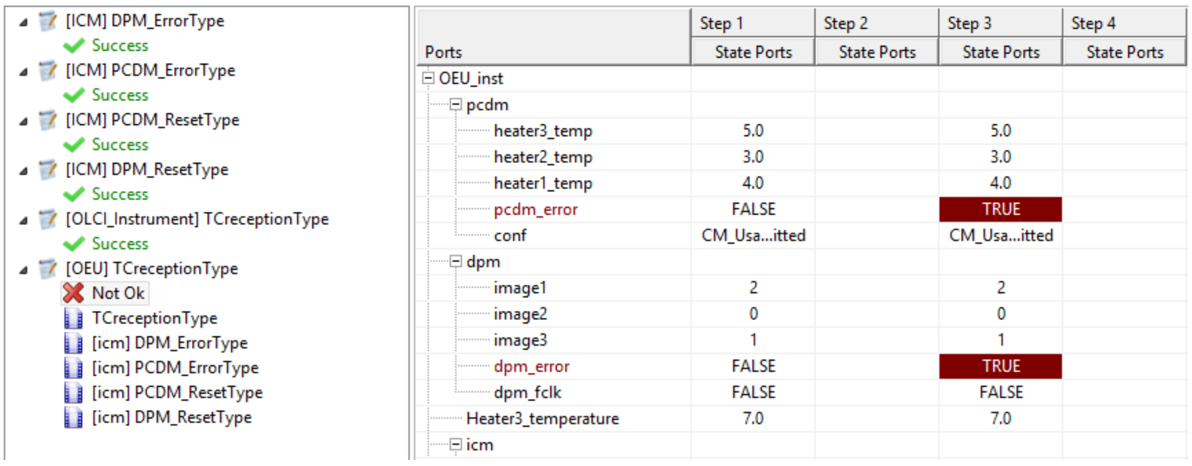
Applied stereotypes:

```
ContractProperty   (from CHESSContract)
    RefinedBy: ContractRefinement [*] = [icm.DPM_Error, icm.PCDM_Error, icm.PCDM_Reset, icm.DPM_Reset]
    ContractType: ContractTypes [1] = Weak
    status: ContractStatus [1] = notValidated
```

Figure 52. Refinement of *TCreception* contract

Steps: Right click on the *OLCI_Instrument/OEU* component (i.e., components with contracts refinement) → CHESS → Functional verification → Check contract refinement on the selected component. Then, it is necessary to specify the model of time that is being used by the system (i.e., 'Discrete').

Results: All checks finish successfully, but one check fails. In this case, a counter example is displayed. The model shall be updated to fix it.



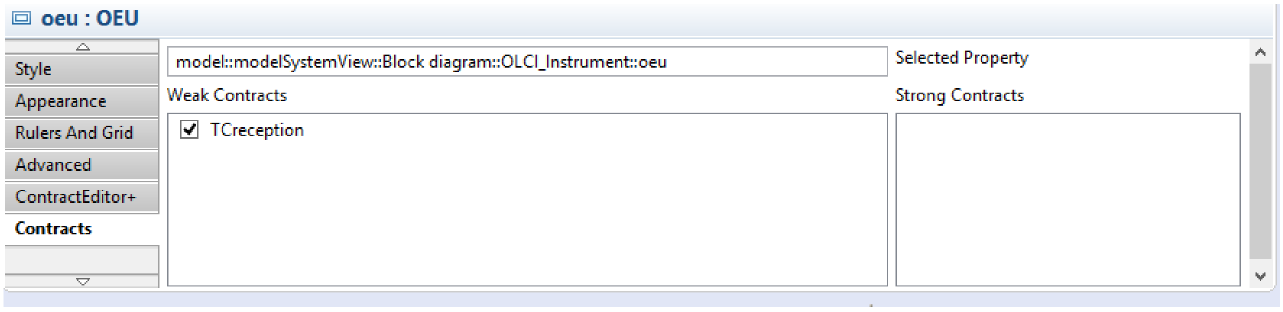
```
[ICM] DPM_ErrorType
    Success
[ICM] PCDM_ErrorType
    Success
[ICM] PCDM_ResetType
    Success
[ICM] DPM_ResetType
    Success
[OLCI_Instrument] TCreceptionType
    Success
[OEU] TCreceptionType
    Not Ok
    TCreceptionType
    [icm] DPM_ErrorType
    [icm] PCDM_ErrorType
    [icm] PCDM_ResetType
    [icm] DPM_ResetType
```

	Step 1	Step 2	Step 3	Step 4
Ports	State Ports	State Ports	State Ports	State Ports
OEU_inst				
pcdm				
heater3_temp	5.0		5.0	
heater2_temp	3.0		3.0	
heater1_temp	4.0		4.0	
pcdm_error	FALSE		TRUE	
conf	CM_Usa...itted		CM_Usa...itted	
dpm				
image1	2		2	
image2	0		0	
image3	1		1	
dpm_error	FALSE		TRUE	
dpm_fclk	FALSE		FALSE	
Heater3_temperature	7.0		7.0	
icm				

Figure 53. Contract refinement verification

2.5. Contract-based verification of strong/weak contracts

Blocks in the Block Definition Diagram can define both strong and weak contracts. When a block is instantiated (Internal Block Diagram), it shall be specified which weak contracts are applicable (see Figure 54).



oeu : OEU

Style	Appearance	Rulers And Grid	Advanced	ContractEditor+	Contracts

model::modelSystemView::Block diagram::OLCI_Instrument::oeu

Selected Property

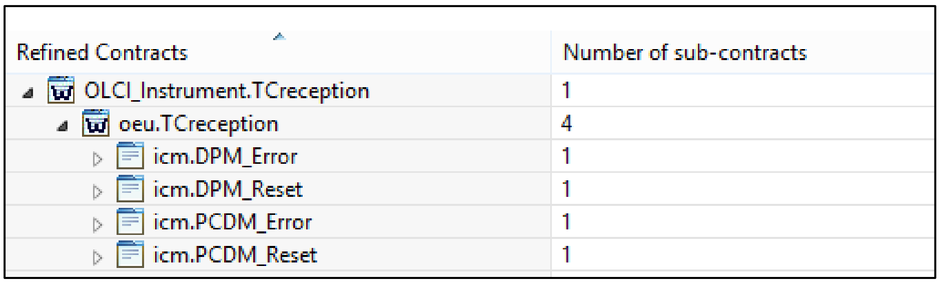
Weak Contracts

☑ TCreception

Strong Contracts

Figure 54. Specification of applicable weak contracts

Both the contract refinement and strong/weak contracts are visualized in the Contracts Refinement View:

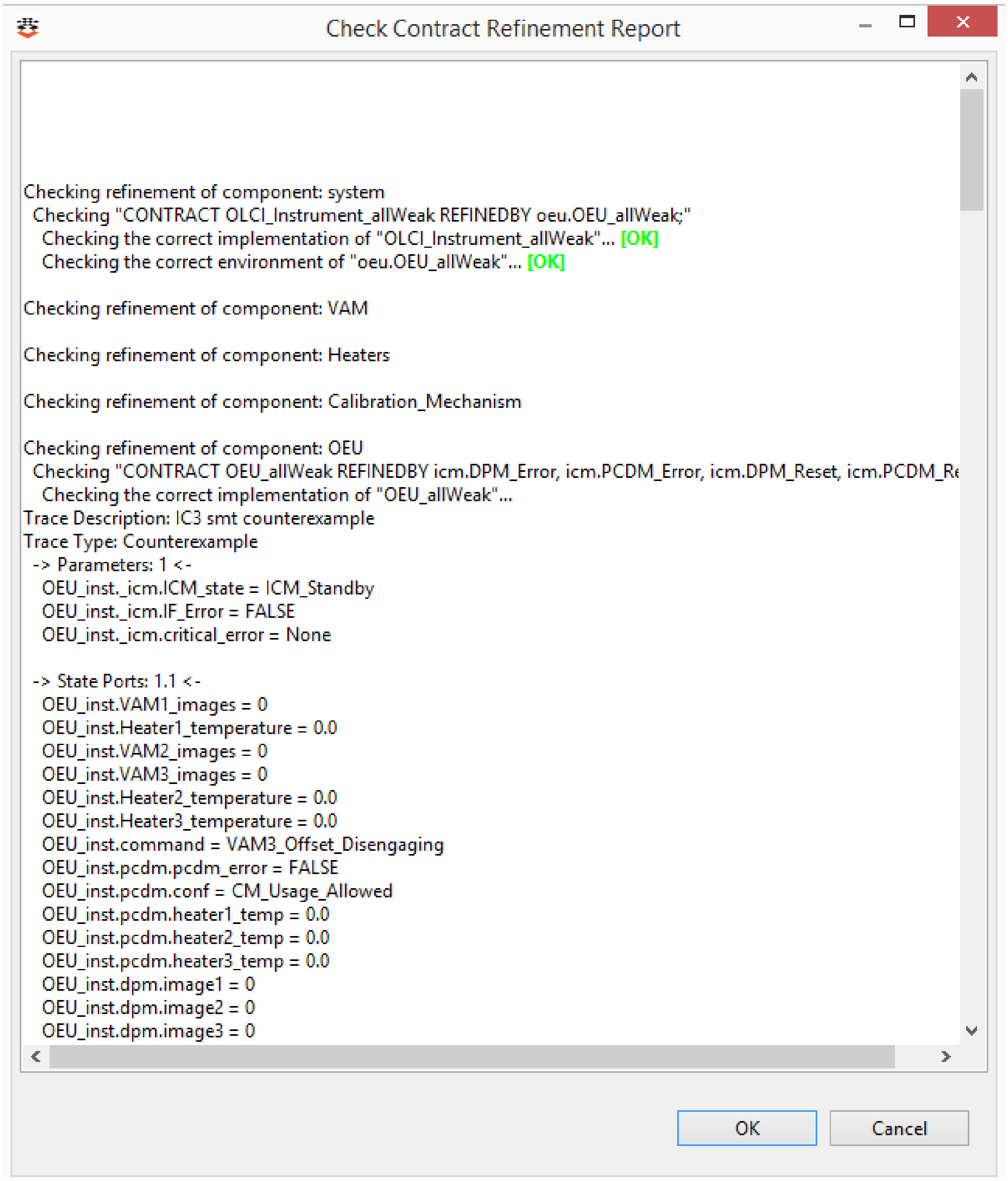


Refined Contracts	Number of sub-contracts
OLCI_Instrument.TCreception	1
oeu.TCreception	4
icm.DPM_Error	1
icm.DPM_Reset	1
icm.PCDM_Error	1
icm.PCDM_Reset	1

Figure 55. OLCI – Contracts Refinement View

Steps: CHESS (Main menu) → Analysis → Formal verifications → Contracts refinement analysis (OCRA). Then the "Analysis context" defined in the Analysis view is selected together with the root element (model::modelSystemView::Block diagram::OLCI_Instrument).

Results: All the analysis is displayed in the "Check Contract Refinement Report" (see Figure 56).



```
Checking refinement of component: system
 Checking "CONTRACT OLCI_Instrument_allWeak REFINEDBY oeu.OEU_allWeak;"
  Checking the correct implementation of "OLCI_Instrument_allWeak"... [OK]
  Checking the correct environment of "oeu.OEU_allWeak"... [OK]

Checking refinement of component: VAM

Checking refinement of component: Heaters

Checking refinement of component: Calibration_Mechanism

Checking refinement of component: OEU
 Checking "CONTRACT OEU_allWeak REFINEDBY icm.DPM_Error, icm.PCDM_Error, icm.DPM_Reset, icm.PCDM_Re
  Checking the correct implementation of "OEU_allWeak"...
Trace Description: IC3 smt counterexample
Trace Type: Counterexample
 -> Parameters: 1 <-
  OEU_inst._icm.ICM_state = ICM_Standby
  OEU_inst._icm.IF_Error = FALSE
  OEU_inst._icm.critical_error = None

 -> State Ports: 1.1 <-
  OEU_inst.VAM1_images = 0
  OEU_inst.Heater1_temperature = 0.0
  OEU_inst.VAM2_images = 0
  OEU_inst.VAM3_images = 0
  OEU_inst.Heater2_temperature = 0.0
  OEU_inst.Heater3_temperature = 0.0
  OEU_inst.command = VAM3_Offset_Disengaging
  OEU_inst.pcdm.pcdm_error = FALSE
  OEU_inst.pcdm.conf = CM_Usage_Allowed
  OEU_inst.pcdm.heater1_temp = 0.0
  OEU_inst.pcdm.heater2_temp = 0.0
  OEU_inst.pcdm.heater3_temp = 0.0
  OEU_inst.dpm.image1 = 0
  OEU_inst.dpm.image2 = 0
  OEU_inst.dpm.image3 = 0
```

Figure 56. OLCI – Check contract refinement report

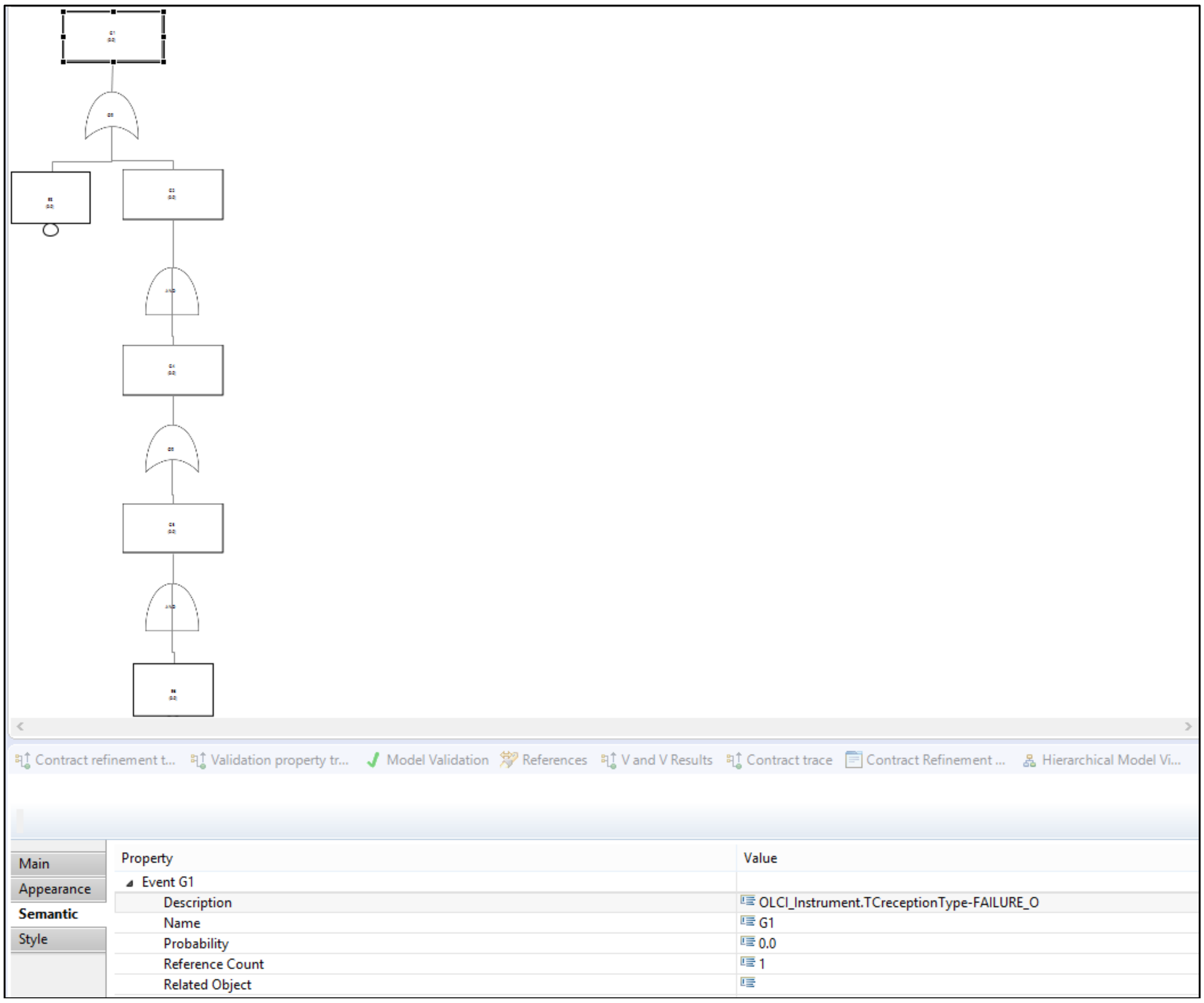
3. <u>Safety analyses – Fault Tree Generation</u>

   Functionality not supported in current version of the AMASS Tool. “xSAP FTA Analysis” option is not available on the palette (Analysis View). The tool provider has been informed to update it in the next tool release.

4. <u>Safety analyses – Contract-based safety analysis</u>

   Steps: Right click on a component of the Block Definition Diagram → CHESS → Safety Analysis → Contract-based safety analysis. Then, it is necessary to specify that a ‘Discrete’ model of time is being used by the system (continuous time is not supported).

Results: The Fault Tree result is produced. For example, the *OLCI_Instrument* fault tree for the "TCreception" contract:



Property	Value
Event G1	
Description	OLCI_Instrument.TCreceptionType-FAILURE_O
Name	G1
Probability	0.0
Reference Count	1
Related Object	

Figure 57. OLCI_Instrument - Fault tree result

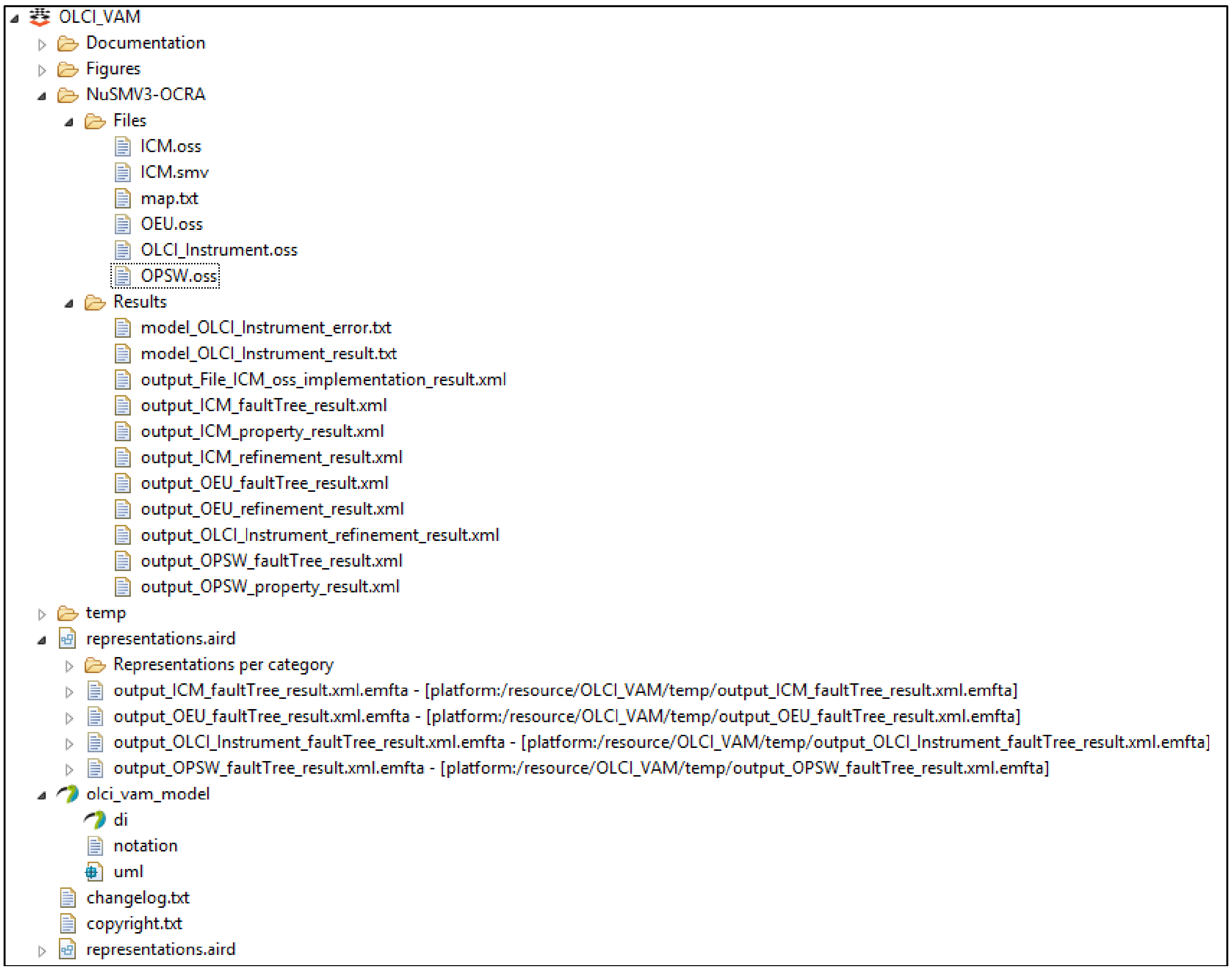
5. <u>Safety case</u>

5.1. Evidence generation

Steps: Compilation of safety analyses results. OpenCert is not used in this iteration.

Results: The following evidences are compiled:

- Validation of contracts: Stored in *OLCI_VAM/NuSMV3-OCRA/Results*
- Contract-based verification of refinement: Stored in *OLCI_VAM/NuSMV3-OCRA/Results*
- Contract-based verification of refinement of strong and weak contracts: Stored in *OLCI_VAM/NuSMV3-OCRA/Results*
- Contract-based safety analysis: Stored in *OLCI_VAM/NuSMV3-OCRA/Results*
- Fault tree: *OLCI_VAM/representation.aird*
- Document generation: Stored in *OLCI_VAM/Documentation*
- Requirements traceability: *olci_vam_model*



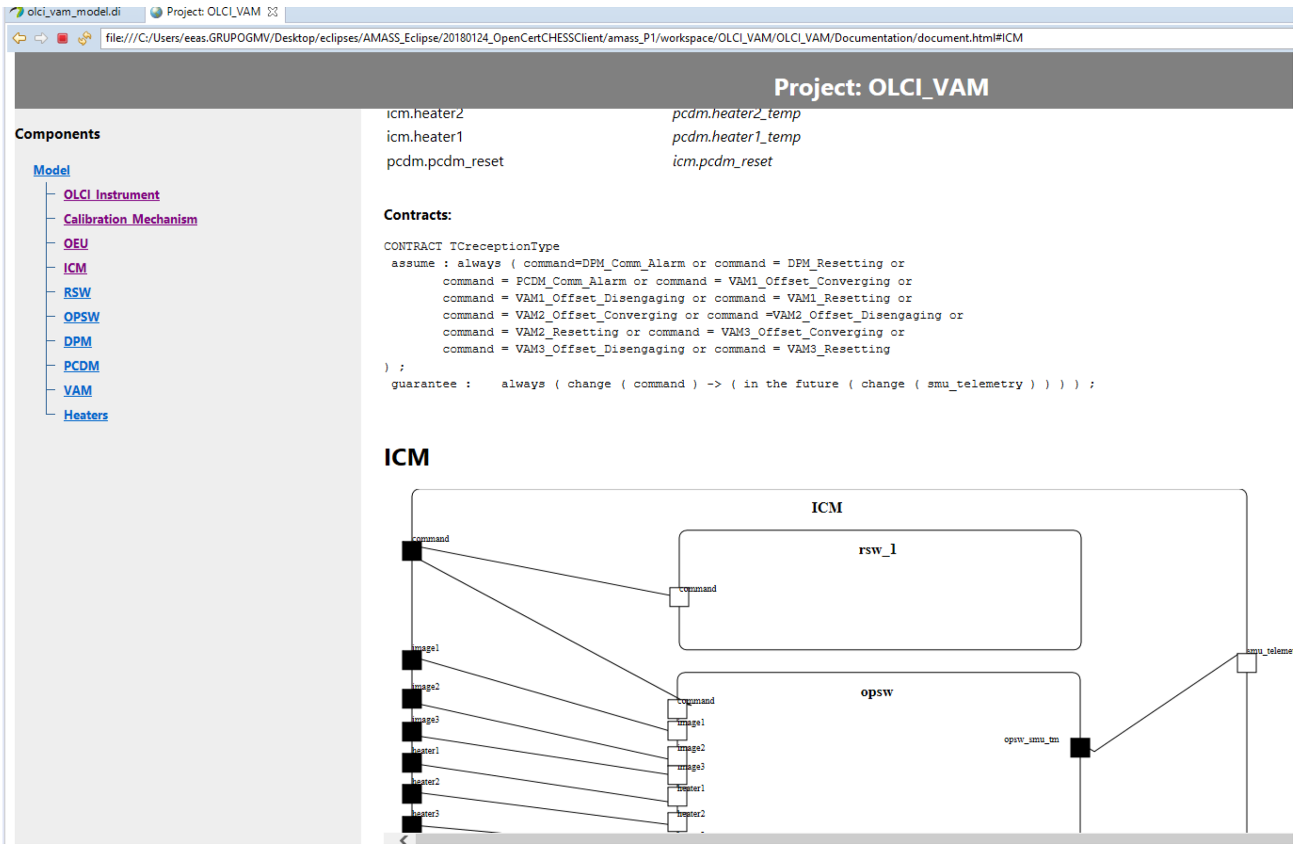
```
OLCI_VAM
  Documentation
  Figures
  NuSMV3-OCRA
    Files
      ICM.oss
      ICM.smv
      map.txt
      OEU.oss
      OLCI_Instrument.oss
      OPSW.oss
    Results
      model_OLCI_Instrument_error.txt
      model_OLCI_Instrument_result.txt
      output_File_ICM_oss_implementation_result.xml
      output_ICM_faultTree_result.xml
      output_ICM_property_result.xml
      output_ICM_refinement_result.xml
      output_OEU_faultTree_result.xml
      output_OEU_refinement_result.xml
      output_OLCI_Instrument_refinement_result.xml
      output_OPSW_faultTree_result.xml
      output_OPSW_property_result.xml
  temp
  representations.aird
    Representations per category
    output_ICM_faultTree_result.xml.emfta - [platform:/resource/OLCI_VAM/temp/output_ICM_faultTree_result.xml.emfta]
    output_OEU_faultTree_result.xml.emfta - [platform:/resource/OLCI_VAM/temp/output_OEU_faultTree_result.xml.emfta]
    output_OLCI_Instrument_faultTree_result.xml.emfta - [platform:/resource/OLCI_VAM/temp/output_OLCI_Instrument_faultTree_result.xml.emfta]
    output_OPSW_faultTree_result.xml.emfta - [platform:/resource/OLCI_VAM/temp/output_OPSW_faultTree_result.xml.emfta]
  olci_vam_model
    di
    notation
    uml
  changelog.txt
  copyright.txt
  representations.aird
```

Figure 58. Storage of safety evidences

5.2. Document generation

Steps: Right-click on a component of the Block Definition Diagram → CHESS → Safety Case → Document generation → Generate documentation on the selected component. Then, it is necessary to specify: i) the model of time is being used by the system (i.e., 'Discrete'), ii) the output directory and iii) the document format (i.e, 'html').

Results: the documentation is stored in the output folder. Figure 59 shows part of the documents generated for the OLCI instrument.



Project: OLCI_VAM

icm.heater2	*pcdm.heater2_temp*
icm.heater1	*pcdm.heater1_temp*
pcdm.pcdm_reset	*icm.pcdm_reset*

Contracts:

```
CONTRACT TCreceptionType
 assume : always ( command=DPM_Comm_Alarm or command = DPM_Resetting or
        command = PCDM_Comm_Alarm or command = VAM1_Offset_Converging or
        command = VAM1_Offset_Disengaging or command = VAM1_Resetting or
        command = VAM2_Offset_Converging or command =VAM2_Offset_Disengaging or
        command = VAM2_Resetting or command = VAM3_Offset_Converging or
        command = VAM3_Offset_Disengaging or command = VAM3_Resetting
) ;
 guarantee :    always ( change ( command ) -> ( in the future ( change ( smu_telemetry ) ) ) ) ;
```

ICM

Figure 59. *OLCI_Instrument* documentation

6. Integrity

   Tool connection with external tools. In particular OCRA for contract-based analysis, nuXmv for model checking and xSAP for model-based safety analysis.

The table below summarizes the information related to the realisation of this scenario:

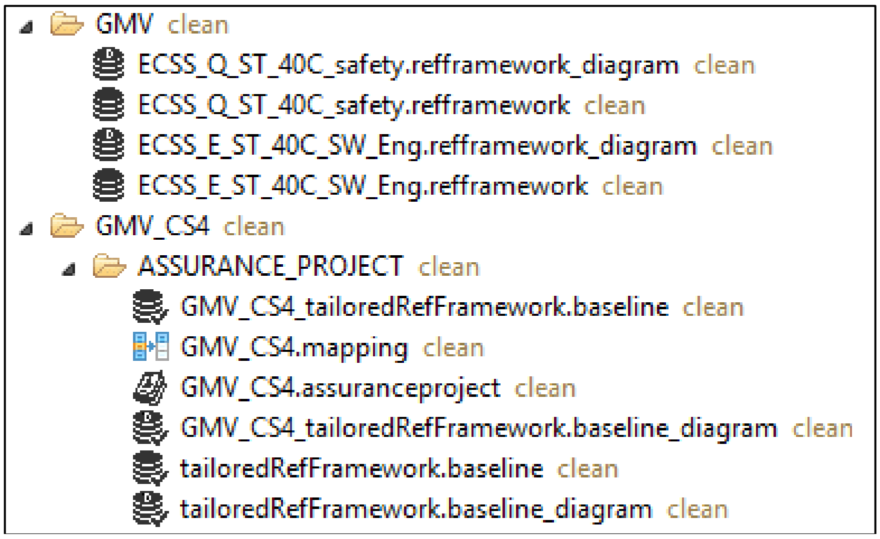
Table 24. CS4-Architecture-Driven Assurance: US1-Baseline: Architectural Design

Realisation Scenario	**Baseline: Architectural Design**	
Scope	Model-based design of the OLCI (system level) and OLCI application software (selected functionalities) using the AMASS platform.	
Tool Settings	CHESS / Papyrus / OCRA / xSAP / nuXmv	
Participants	System and Software Design	GMV
	Analyses execution	GMV
	Evidences generation	GMV
	Tool support	FBK, INT, TEC, RPT
	UC assessment	TEC, TASE
Activities realised	02/2017	AMASS first prototype tools installation and configuration
	02/2017	SW architecture using Papyrus and CHESS
	02/2018	AMASS second prototype tool installation and configuration
	02/2018	Requirements formalization (CHESS), system and

		software architecture design (CHESS), analyses execution (CHESS, OCRA, xSAP, nuXmv) and safety case (CHESS)
Usage Decisions	Focus on the system architecture.	
Expected Results	Complete architecture including safety information and preliminary analyses results.	
Conclusions	The analyses will be conducted using the third tool prototype (due to the maturity level of the tools). Once the results are obtained, the different usage scenarios will be evaluated to get metric figures.	

3.4.2.2. STO2 Multi-concern Assurance

CS4 main focus will be the Architecture-Driven Assurance process. Nevertheless, a simplified assurance case specification is foreseen to check the relationship with the system component specification. Currently, two standards are being modelled: the ECSS-Q-ST-40C and the ECSS-E-ST-40C. Additionally, the assurance project has been created.



```
GMV clean
    ECSS_Q_ST_40C_safety.refframework_diagram clean
    ECSS_Q_ST_40C_safety.refframework clean
    ECSS_E_ST_40C_SW_Eng.refframework_diagram clean
    ECSS_E_ST_40C_SW_Eng.refframework clean
GMV_CS4 clean
    ASSURANCE_PROJECT clean
        GMV_CS4_tailoredRefFramework.baseline clean
        GMV_CS4.mapping clean
        GMV_CS4.assuranceproject clean
        GMV_CS4_tailoredRefFramework.baseline_diagram clean
        tailoredRefFramework.baseline clean
        tailoredRefFramework.baseline_diagram clean
```

Figure 60. CS4 standards and assurance project

Below, the ECSS-Q-ST-40C diagram is depicted:

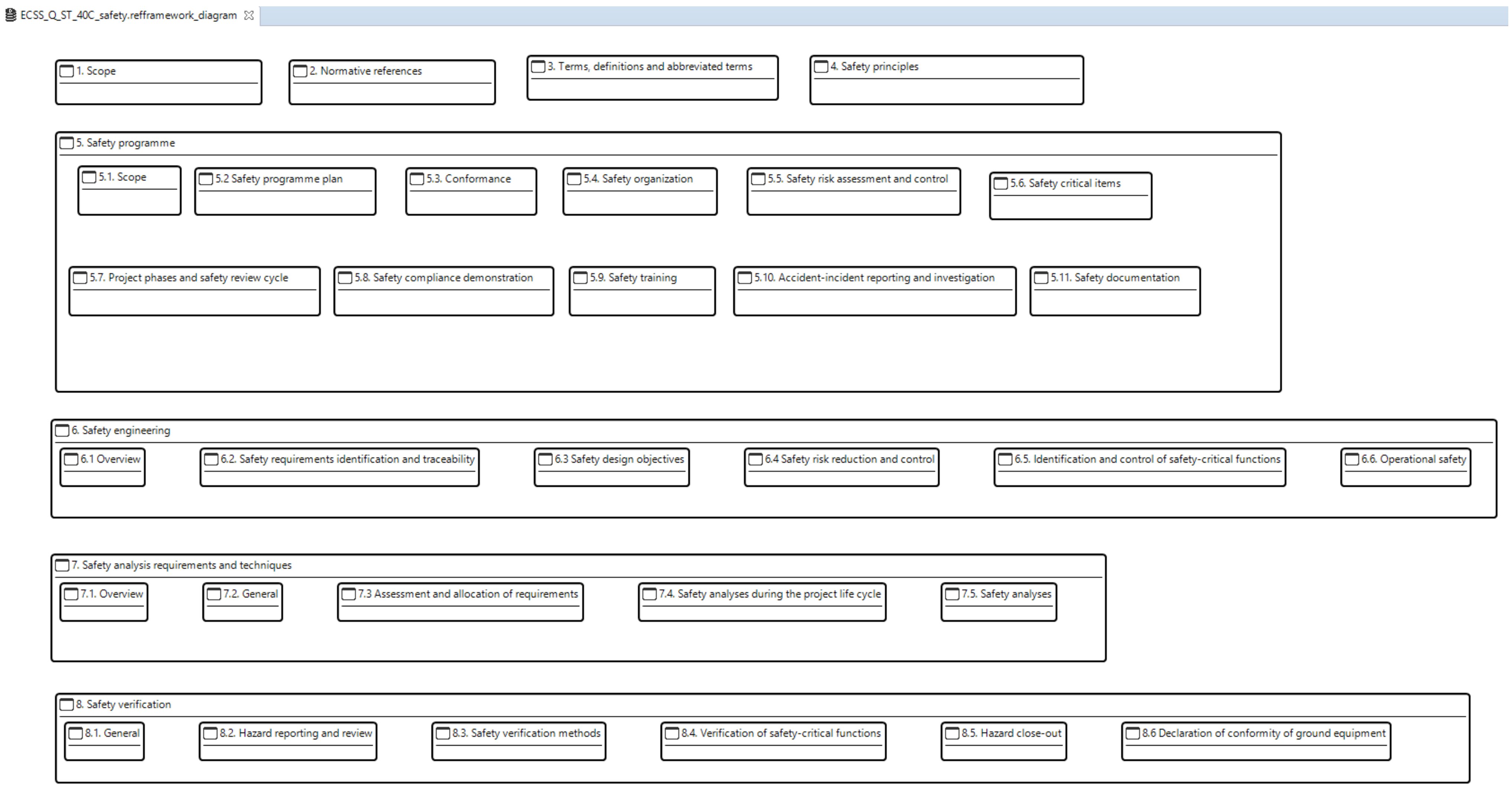

ECSS_Q_ST_40C_safety.refframework_diagram
1. Scope
2. Normative references
3. Terms, definitions and abbreviated terms
4. Safety principles
5. Safety programme
5.1. Scope
5.2 Safety programme plan
5.3. Conformance
5.4. Safety organization
5.5. Safety risk assessment and control
5.6. Safety critical items
5.7. Project phases and safety review cycle
5.8. Safety compliance demonstration
5.9. Safety training
5.10. Accident-incident reporting and investigation
5.11. Safety documentation
6. Safety engineering
6.1 Overview
6.2. Safety requirements identification and traceability
6.3 Safety design objectives
6.4 Safety risk reduction and control
6.5. Identification and control of safety-critical functions
6.6. Operational safety
7. Safety analysis requirements and techniques
7.1. Overview
7.2. General
7.3 Assessment and allocation of requirements
7.4. Safety analyses during the project life cycle
7.5. Safety analyses
8. Safety verification
8.1. General
8.2. Hazard reporting and review
8.3. Safety verification methods
8.4. Verification of safety-critical functions
8.5. Hazard close-out
8.6 Declaration of conformity of ground equipment

Figure 61. ECSS-Q-ST-40C standard

The simplified assurance case specification will be available at the end of the 3rd iteration.

Table 25. CS4-Multi-concern Assurance: US1–Baseline: Architectural design

Realisation Scenario	**Baseline: Architectural Design**	
Scope	Definition of the assurance case model.	
Tool Settings	Reference Framework Editor / OpenCert	
Participants	Standards Specification	GMV
	Assurance project	GMV
	Tool support	TEC
	UC assessment	TEC
Activities realised	02/2018	The activities of the assurance case specification have started.
Usage Decisions	Just a simplified assurance case specification will be developed to check its relationship with the system components model.	
Expected Results	Simplified assurance case specification linked to the CHESS model.	
Conclusions	-	

3.4.3. Coverage of AMASS Prototype P1 Architecture

Table 26 illustrates the implemented functionalities during this second iteration within the Case Study 4.

Table 26. AMASS Prototype P1 Coverage by CS4

STO	AMASS Functionality Groups	Tools
Architecture-Driven Assurance	System Component Specification	CHESS
	System Architecture Modelling for Assurance	CHESS
	Architectural Patterns for Assurance	-
	Contract-based Design for Assurance	CHESS/OCRA
	Activities supporting Assurance Case	CHESS, OCRA, xSAP, nuXmv
Multi-Concern Assurance	Assurance Case Specification	OpenCert
	Dependability Assurance	-
	System Dependability Co-Analysis/Co-Assessment	Concerto FLA
	Contract-Based Multi-concern Assurance	-
Seamless Interoperability	Evidence Management	-
	Tool Integration Management	V&V Manager and OSLC Automation V&V Tool Integration
	Collaborative Work Management	-
	Tool Quality Assessment and Characterization	-
Cross Intra-Domain Reuse	Compliance Management	OpenCert
	Reuse Assistant	-
	Process-related reuse via management of variability at process level	-
	Process-related reuse via management of variability at product level	-
	Automatic generation of process-based arguments	-

	Automatic generation of product-based arguments	-

3.4.4. Conclusions

At this stage, the main benefits and potential improvements are identified in Table 27.

Table 27. Benefits and potential improvements for CS4

Artefact	Achievement/Benefits	Improvements/Recommendations
Requirements specification and formalization	• Complete requirements specification at model level. • Formalization allows requirements verification at early development stages. • Requirements linked to design entities, formal properties and contracts. • Traceability.	• Formal specification does not support operations (*PropertyEditor+*) and many functionalities are service-oriented.
Design of system/software architecture	• Flexibility: Many features available (e.g. hierarchy). • The tool allows defining both static and dynamic architectures. • System architecture decorated with non-functional properties. • All the information defined in a single model. It improves consistency.	• Improve the GUI, for example: ○ Too many options available: could be some of them be filtered (e.g., depending on the role?) ○ Similar analyses in different menus ○ Accessing state machine diagrams from the system block diagram would be useful ○ The procedure to define Guard/ Effects can be improved/simplified ○ F4 capability is not intuitive • Tool maturity, e.g. fix bugs. • At system level, only flow ports are available.
Conduction of safety analyses	• Simulation of different scenarios (setting the value of specific properties). • The V&V and safety analyses provide evidences of the fulfilment of the project requirements. • Evidences directly obtained from the model. • In line with space needs.	• Tool maturity, e.g. fix bugs. • Tool limitations (only discrete type, state machines do not support i) operations and ii) state machines in non-leaf components).
Safety case	• In line with space needs. • Evidences and documentation. generation automatically produced from the model.	• Document generation ○ Allow the usage of a specific template ○ Allow the designer to select the information to be documented ○ Support different formats (doc, pdf) • Evidence generation ○Configure the folder path
Multi-concern	• In space, ECSS tailoring is commonly	

Artefact	Achievement/Benefits	Improvements/Recommendations
Assurance Model	requested according to the project requirements. • The tool can be used to define this tailoring.	

3.5. Case Study 5: Railway domain: Platform Screen Doors Controller

3.5.1. 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.

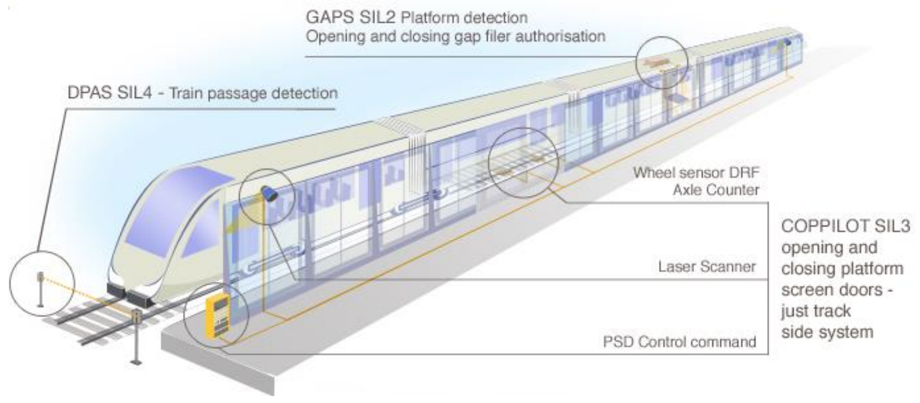


Figure 62. Coppilot system with its different subsystems: laser scanner, steel wheel sensor and the door control command

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 of 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 EN 50126, 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.

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

3.5.2. US1: Generation of Frama-C asserted C code from B models

3.5.2.1. STO3 Seamless Interoperability

This usage scenario is aimed at demonstrating 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. This second generation is based on an in-house, SIL4 ready hardware (CLEARSY Safety Platform).

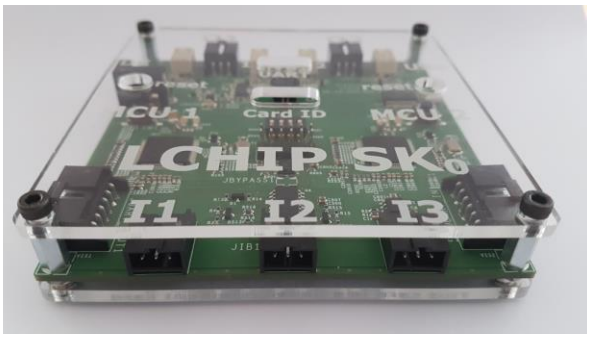
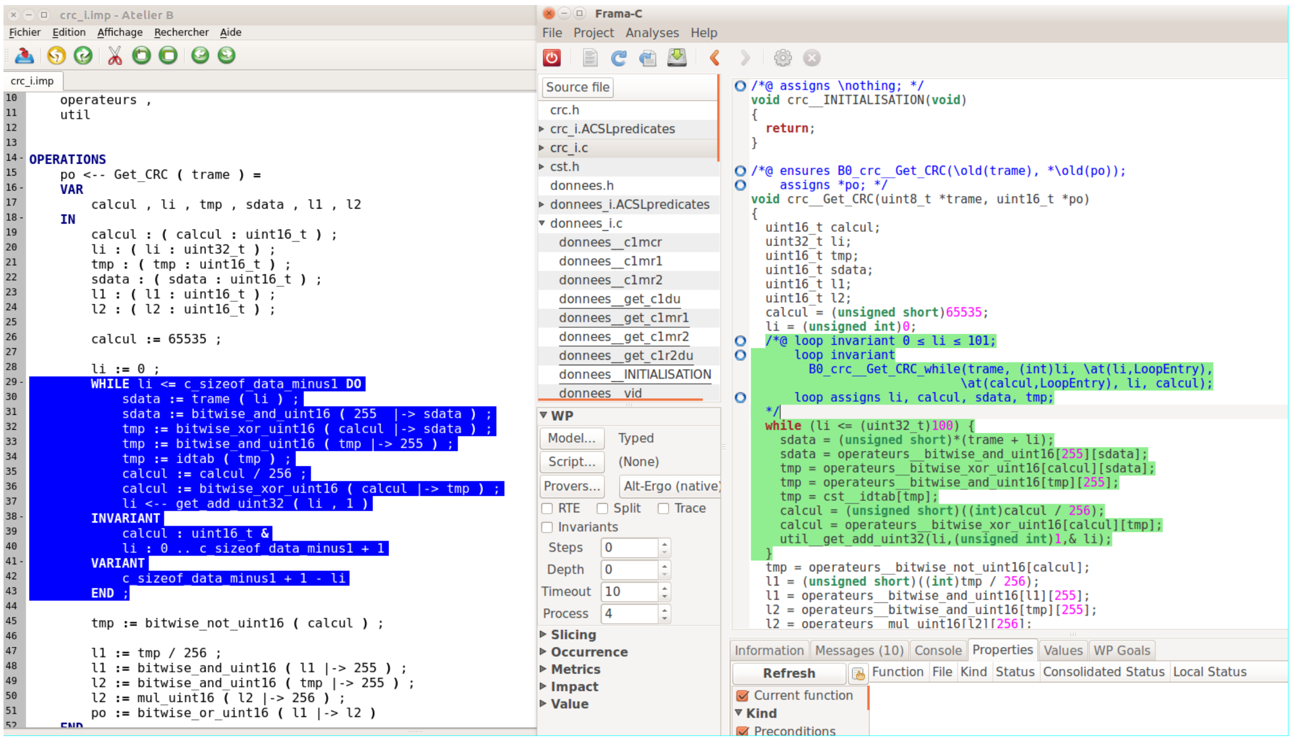


Figure 63. CLEARSY Safety Platform starter kit SK0

The experiment is performed on the software in charge of the safety of the device. It performs the bootload of the binaries and the verification of their correctness (CRC, no memory overlap between the 2 binaries, memory allocation compatible with memory map, etc.). Then it is in charge of the sequencing of the 2 binaries, the check of the memory integrity (CRCs), the verification of the identity of the values stored in the variables for both instances, the same verification performed with the other microcontroller, the instruction checking that verifies that the microcontroller is able to execute all the instructions used by the binaries, etc. Some verifications are delayed over several cycles. The resulting software contains all the features required to detect a divergent behaviour among the 4 software instances (4oo4 SW) and the 2 microcontrollers (2oo2 HW).



```
    operateurs ,
    util


OPERATIONS
    po <-- Get_CRC ( trame ) =
    VAR
        calcul , li , tmp , sdata , l1 , l2
    IN
        calcul : ( calcul : uint16_t ) ;
        li : ( li : uint32_t ) ;
        tmp : ( tmp : uint16_t ) ;
        sdata : ( sdata : uint16_t ) ;
        l1 : ( l1 : uint16_t ) ;
        l2 : ( l2 : uint16_t ) ;

        calcul := 65535 ;

        li := 0 ;
        WHILE li <= c_sizeof_data_minus1 DO
            sdata := trame ( li ) ;
            sdata := bitwise_and_uint16 ( 255  |-> sdata ) ;
            tmp := bitwise_xor_uint16 ( calcul |-> sdata ) ;
            tmp := bitwise_and_uint16 ( tmp |-> 255 ) ;
            tmp := idtab ( tmp ) ;
            calcul := calcul / 256 ;
            calcul := bitwise_xor_uint16 ( calcul |-> tmp ) ;
            li <-- get_add_uint32 ( li , 1 )
        INVARIANT
            calcul : uint16_t &
            li : 0 .. c_sizeof_data_minus1 + 1
        VARIANT
            c_sizeof_data_minus1 + 1 - li
        END ;

        tmp := bitwise_not_uint16 ( calcul ) ;

        l1 := tmp / 256 ;
        l1 := bitwise_and_uint16 ( l1 |-> 255 ) ;
        l2 := bitwise_and_uint16 ( tmp |-> 255 ) ;
        l2 := mul_uint16 ( l2 |-> 256 ) ;
        po := bitwise_or_uint16 ( l1 |-> l2 )
    END
```

```
/*@ assigns \nothing; */
void crc__INITIALISATION(void)
{
  return;
}

/*@ ensures B0_crc__Get_CRC(\old(trame), *\old(po));
    assigns *po; */
void crc__Get_CRC(uint8_t *trame, uint16_t *po)
{
  uint16_t calcul;
  uint32_t li;
  uint16_t tmp;
  uint16_t sdata;
  uint16_t l1;
  uint16_t l2;
  calcul = (unsigned short)65535;
  li = (unsigned int)0;
  /*@ loop invariant 0 ≤ li ≤ 101;
      loop invariant
        B0_crc__Get_CRC_while(trame, (int)li, \at(li,LoopEntry),
                              \at(calcul,LoopEntry), li, calcul);
      loop assigns li, calcul, sdata, tmp;
  */
  while (li <= (uint32_t)100) {
    sdata = (unsigned short)*(trame + li);
    sdata = operateurs__bitwise_and_uint16[255][sdata];
    tmp = operateurs__bitwise_xor_uint16[calcul][sdata];
    tmp = operateurs__bitwise_and_uint16[tmp][255];
    tmp = cst__idtab[tmp];
    calcul = (unsigned short)((int)calcul / 256);
    calcul = operateurs__bitwise_xor_uint16[calcul][tmp];
    util__get_add_uint32(li,(unsigned int)1,& li);
  }
  tmp = operateurs__bitwise_not_uint16[calcul];
  l1 = (unsigned short)((int)tmp / 256);
  l1 = operateurs__bitwise_and_uint16[l1][255];
  l2 = operateurs__bitwise_and_uint16[tmp][255];
  l2 = operateurs__mul_uint16[l2][256];
```

Figure 64. Example of assertion generation from a B model (left) to the corresponding C code (right)

Table 28. CS5-Seamless Interoperability: 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 including target specific code generator, Frama-C
Participants	• Leader: CLS • CEA
Activities realised	• Definition of the requirements that define the formal verification framework

	• Writing of a specification document that defines the BXML file format (persisting format for B models) • Definition of the process [generation of assertions from B models] • Proof of concept on significant, non-trivial properties and source code from Stockholm PSD project) • Selection of the parts of the B model to consider [ongoing] • Definition of the translation rules [ongoing] • Specification of the B2ACSL translator [ongoing]
Usage Decisions	Definition of target code: generation of assertions for generated code but also for third party code (extension to). Definition of input models: implementations only.
Expected Results	Conformance established automatically by formal proof.
Conclusions	Positive partial assessment so far.

3.5.3. US2: Support for system-level model, including safety and security aspects

This usage scenario is aimed at ensuring that the AMASS platform is able to be used for the modelling of our systems. During the first year, the modelling was centred on the COPPILOT system (platform screen doors controller). During the second year the focus was more on the CLEARSY Safety Platform that is now at the heart of the new COPPILOT systems. It is not a system by itself but rather a building block to be integrated in future systems to be certified, with a certification kit. The CLEARSY Safety Platform relies on the smart integration of formal methods, redundant code generation and compilation, and a hardware platform that ensures a safe execution of the software.

The CLEARSY Safety Platform will be provided with a certification kit including design documentation, safety principles + justification, testing performed and exported constraints. The content is not yet fixed and requires external advice form a certification body (Bureau Veritas, CERTIFER). We are considering here to which extent the AMASS tools would contribute to the certification kit by considering EN 50129 first than IEC 61508 in a second time (safety critical automation at large).

Both functional modelling and contract-based design were performed during this year by using PAPYRUS and CHESS tools. The functional modelling of single board was completed by dividing a board into separate half-boards, by making clear that one processor provides energy for one half board and command for the other half one, etc. The contract-based design allowed to introduce constraints like the outputs are in a permissive state if and only if the 2 processors are alive, the intra and inter CPU verifications are OK, etc.

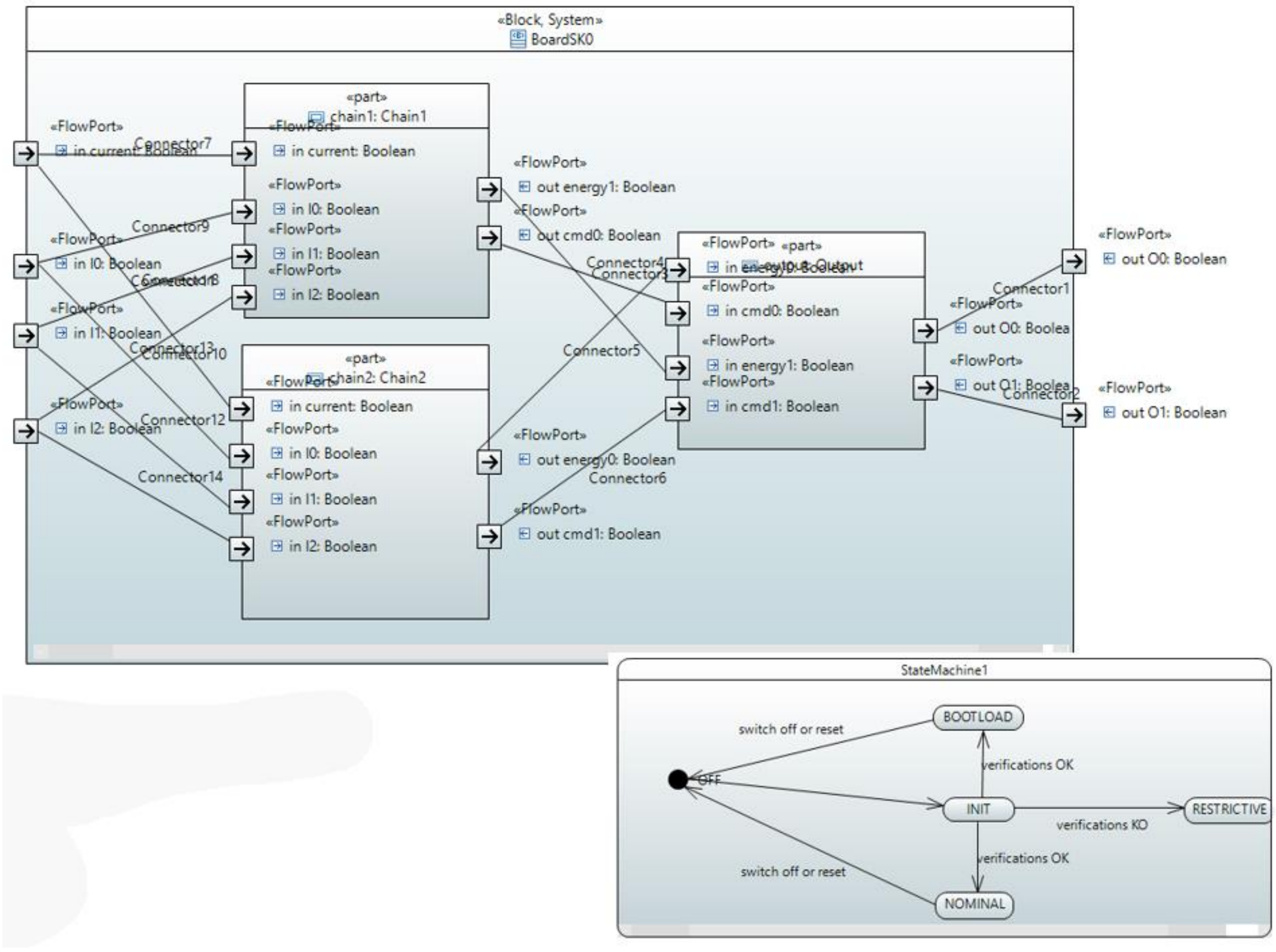


Figure 65. Top-level block specification of the CLEARSY Safety Platform

3.5.3.1. STO1 Architecture-Driven Assurance

Table 29. CS5-Architecture-Driven Assurance: US2-Model_based System component specification

Realisation Scenario	Model_based System component specification
Scope	The system components are specified including system requirements
Tool Settings	Papyrus, CHESS
Participants	• Leader: CLS • CEA
Activities realised	1. Analysis of the COPPILOT specifications 2. System modelling: context/environment, requirements, system architecture, high level functional decomposition, traceability 3. Refinement of architecture models: include different configurations for the architecture 4. Analysis of the Safety ClearSy Platform
Usage Decisions	
Expected Results	Architecture analysis
Conclusions	Positive partial assessment so far

3.5.3.2. STO2 Multi-concern Assurance

Table 30. CS5-Multi-concern Assurance: US2-Security Assessment

Realisation Scenario	Security Assessment
Scope	Security assessment for PSD systems
Tool Settings	Papyrus
Participants	• Leader: CLS • CEA
Activities realised	1. Identification of the required standards and existing templates 2. Definition of the safety requirements 3. Preliminary list of security concerns 4. Extension of the analysis to the ClearSy Safety Platform
Usage Decisions	Definition of an *a priori* vulnerabilities list How to conduct a preliminary security analysis?
Expected Results	Contribution to the safety case Directions for integrating security analysis to safety case
Conclusions	

3.5.4. Coverage of AMASS Prototype P1 Architecture

Table 31 illustrates the implemented functionalities during this second iteration within the Case Study 5.

Table 31. AMASS Prototype P1 Coverage by CS5

STO	AMASS Functionality Groups	Tools
Architecture-Driven Assurance	System Component Specification	CHESS
	System Architecture Modelling for Assurance	CHESS
	Architectural Patterns for Assurance	-
	Contract-based Design for Assurance	CHESS/OCRA
	Activities supporting Assurance Case	OCRA
Multi-Concern Assurance	Assurance Case Specification	OpenCert
	Dependability Assurance	-
	System Dependability Co-Analysis/Co-Assessment	Papyrus SSE
	Contract-Based Multi-concern Assurance	-
Seamless Interoperability	Evidence Management	-
	Tool Integration Management	Generation of Frama-C asserted C code from B models Atelier B formal IDE including target specific code generator Frama-C V&V Tool Integration
	Collaborative Work Management	-
	Tool Quality Assessment and Characterization	-
Cross Intra-Domain Reuse	Compliance Management	-
	Reuse Assistant	-
	Process-related reuse via management of variability at process level	-

	Process-related reuse via management of variability at product level	-
	Automatic generation of process-based arguments	-
	Automatic generation of product-based arguments	-

3.5.5. Conclusions

The AMASS prototype P1 has proved to be usable for modelling parts of our systems. Several improvements on the contract-based design are expected in P2 to model more precisely and distribute contracts over sub-components.

The user documentation is OK (documentation and videos). Several internal users have been able to jump start into “P1” without significant problem.

3.6. Case Study 6: Railway domain: Automatic Train Control Formal Verification

3.6.1. Case Study Specification

Alstom Signalling develops safety critical signalling systems for railway application (mass transit or main lines). These systems shall comply with international safety standards such as CENELEC EN50126/8/9, specific regional safety regulations and technical specification for interoperability (e.g. ERTMS specification in Europe). Among these safety critical systems are Automatic Train Control systems which are the topic of the Alstom case study.

The objective of this case study is to create a safety assurance project for an Automatic Train Control signalling system that includes formal proof demonstration (instead of classical workbench tests).

This safety assurance project shall include all artefacts required by the EN 50129 Generic Application Safety Case. These artefacts could be a reference to a document, table, diagram or text. The application of the EN 50129 requires independence between the designer, the verifier (V&V) and the safety validation team.

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

The Alstom Case Study includes four different Usage Scenarios:

- US1: Assurance Project Creation
- US2: System Design, V&V and Dependability Assessment
- US3: Evidence Management
- US4: Compliance Management

As Alstom joined the consortium at a later date, none of these Usage Scenarios were tackled during the first iteration. The objectives and work plan for the second iteration are described in the following paragraphs.

3.6.2. US1: Assurance Project Creation

This activity is related to the creation and the setting of the assurance project in the AMASS platform.
The output shall be:

- Creation of the roles and definition of credentials (implementation of the independence between Design, V&V and Safety roles).
- Workflow definition and allocation: define the stream of activities and allocate activities to actors.
- Creation of the assurance project artefact structure: reference to a document, table, text, diagram, etc.
- EN 50129 and EN 50128 clauses hierarchically captured (Hypothesis: Standards are recorded in the AMASS platform within a library for reuse purpose).

During the second iteration, the objective for this usage scenario is to use two of the main AMASS tools functionalities: Architecture-Driven Assurance and Intra-domain reuse.

The following figure represents the Case Study workflow. It indicates under each artefact that shall be captured by the AMASS platform (boxes with painted black circles) which main functionality of the tool is used, that is to say Architecture-Drive Assurance, Intra-domain Reuse, or a combination of both, for this case study.

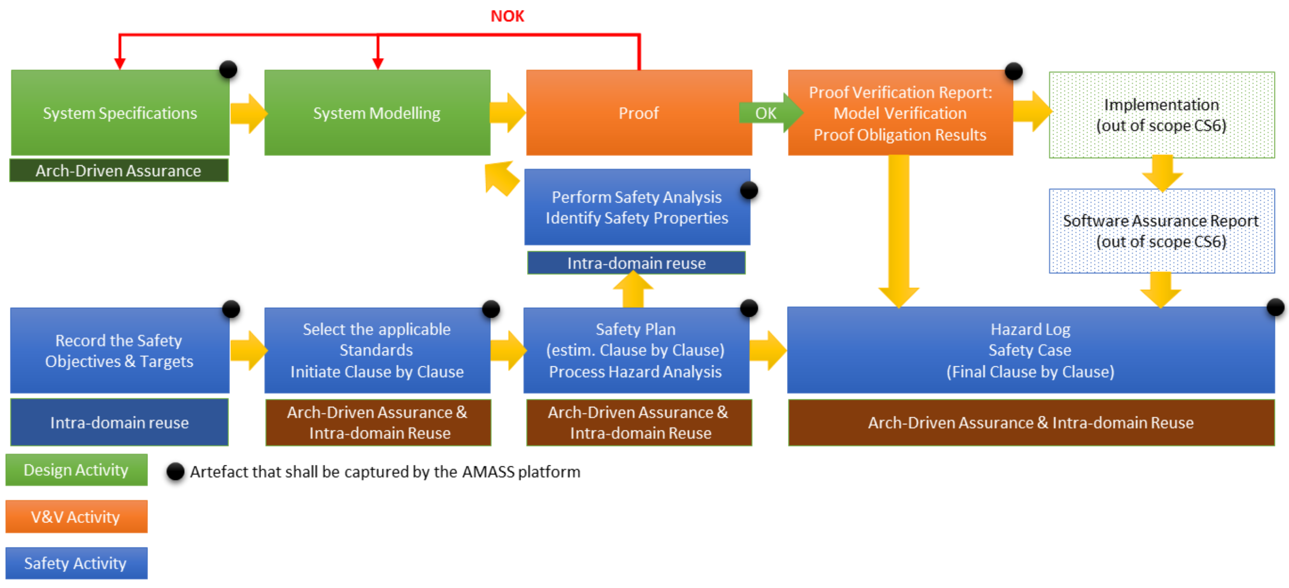

NOK
System Specifications
Arch-Driven Assurance
System Modelling
Proof
OK
Proof Verification Report: Model Verification Proof Obligation Results
Implementation (out of scope CS6)
Software Assurance Report (out of scope CS6)
Perform Safety Analysis Identify Safety Properties
Intra-domain reuse
Record the Safety Objectives & Targets
Intra-domain reuse
Select the applicable Standards Initiate Clause by Clause
Arch-Driven Assurance & Intra-domain Reuse
Safety Plan (estim. Clause by Clause) Process Hazard Analysis
Arch-Driven Assurance & Intra-domain Reuse
Hazard Log Safety Case (Final Clause by Clause)
Arch-Driven Assurance & Intra-domain Reuse
Design Activity
V&V Activity
Safety Activity
Artefact that shall be captured by the AMASS platform

Figure 66. Case Study 6 workflow

This workflow is applicable to the four Usage Scenarios of this Case Study. The use of the AMASS tool functionalities for this Usage Scenario is described in the two tables below. Similar tables are used for the three other Usage Scenarios.

3.6.2.1. STO4 Cross Intra-domain reuse

Table 32. CS6- Cross Intra-domain reuse: US1-Assurance Project Creation

Realisation Scenario	Assurance Project Creation
Scope	This Usage Scenario aims at defining a generic workflow for Safety Assurance Projects based on the three CENELEC standards applicable to the railway signalling system considered (EN 50126/8/9).
Tool Settings	AMASS Tools (Reference Framework Editor, Assurance Project Management Editor)
Participants	ALS
Activities realised	• Create AMASS project in the AMASS tools • Model CENELEC standards • Define safety assurance project workflow using CENELEC standards clauses and Alstom process • Creation of the project artefact structure
Usage Decisions	NA
Expected Results	The main expected result of this usage scenario is to create an appropriate workflow and assurance case model structure that fit the Alstom process needs.
Conclusions	NA

Table 33. CS6-Cross Intra-Domain Reuse: US1-Reuse of assurance artefacts

Realisation Scenario	Reuse of assurance artefacts
Scope	This Usage Scenario aims at defining a generic workflow for Safety Assurance Projects based on the three CENELEC standards applicable to the railway signalling system considered (EN 50126/8/9).
Tool Settings	AMASS Tools (Reference Framework Editor, Assurance Project Management Editor)

Participants	ALS
Activities realised	1. Create AMASS project in the AMASS tools 2. Model CENELEC standards 3. Define assurance project workflow using CENELEC standards clauses and Alstom process 4. Creation of the project artefact structure
Usage Decisions	NA
Expected Results	The main expected result of this usage scenario is to create a reusable workflow that will define each the necessary activity of the Assurance Project and guide the work process of the Safety Assurance Manager. It will also help generate a clause by clause analysis of the relevant standards for a given Assurance Project.
Conclusions	NA

3.6.3. US2: System Design, V&V and Dependability Assessment

This usage scenario corresponds to the main activities performed by the actors during the project. The Safety Assurance Manager, the Design Leader and the V&V Leader follow the workflow assistant to add requested data at each step of the process.

During the second iteration, the objective for this usage scenario is to use two of the main AMASS tools functionalities: Architecture-Driven Assurance and Intra-domain reuse.

3.6.3.1. STO1 Architecture-Driven Assurance

Table 34. CS6-Architecture-Driven Assurance: US2-System Design, V&V and Dependability Assessment

Realisation Scenario	System Design, V&V and Dependability Assessment
Scope	This is the basic usage; the actors follow the process with the workflow assistant and provide their baseline artefacts when necessary.
Tool Settings	Papyrus/SysML
Participants	ALS
Activities realised	1. Designer activity (DI as per CENELEC roles) a. Model SysML of the system b. Ensure consistency and traceability with the formal model of the system 2. Verifier activity (VER as per CENELEC roles) a. Log traceability verification of models b. Log proof obligations results (proof report) 3. Validator activity (VAL as per CENELEC roles) a. Record Safety Plan (including CENELEC clause by clause analysis initialization) b. Perform Hazard Analyses c. Fill Hazard Log using DI and VER artefacts recorder in AMASS tool d. Record Safety Case
Usage Decisions	NA
Expected Results	This Usage Scenario shall demonstrate that the safety demonstration of the Automatic Train Control system developed using formal methods and formal verification can be implemented using the Assurance Project Management Editor from the AMASS Tools.
Conclusions	NA

3.6.3.2. STO4 Cross Intra-domain reuse

Table 35. CS6-Cross Intra-domain reuse: US2-System Design, V&V and Dependability Assessment

Realisation Scenario	System Design, V&V and Dependability Assessment
Scope	This is the basic usage; the actors follow the process and provide their baseline artefacts when necessary.
Tool Settings	AMASS Tools (Assurance Project Management Editor)
Participants	ALS
Activities realised	1. Create new specific project using an existing model
Usage Decisions	NA
Expected Results	This usage scenario shall demonstrate that the Assurance Project Workflow defined in US1 can easily be reused for different projects that use similar development processes.
Conclusions	NA

3.6.4. US3: Evidence Management

This Usage Scenario's goal is to manage the evidence of the safety demonstration. The evidence to record is shown in Figure 66, it is the different blocks with painted black circle:

- Safety Objectives and Targets, recorded as text or table.
- Standards clause (as library) recorded as table.
- System specification, recorded as references to documents.
- Safety Plan and Process Hazard Analysis, recorded as references to document.
- System Safety Analysis and Safety Properties Models recorded as reference document [or table (one line per system requirement analysis, and one line per safety property model)].
- Proof Verification Report, recorded as references to document [or table if the granularity of the artefacts set for the project allow the capture of each specific safety properties to prove, see above].
- Hazard Log and Safety Case recorded as reference to a document.

The configuration management of evidence shall allow each actor to manage local or working copy of artefacts and to freeze a baseline of artefacts when necessary.

During the second iteration, the objective for this usage scenario is to use two of the main AMASS tools functionalities: Architecture-Driven Assurance and Intra-domain reuse.

3.6.4.1. STO1 Architecture-Driven Assurance

Table 36. CS6-Architecture-Driven Assurance: US3-Evidence Management

Realisation Scenario	Evidence Management
Scope	This usage scenario enables the user to record and retrieve consistent artefacts for a baseline system specification.
Tool Settings	AMASS tools (Evidence Management Editor)
Participants	ALS
Activities realised	1. Create traceability links between DI and VER artefacts 2. Automatically generate Hazard Log for VAL activities
Usage Decisions	NA
Expected Results	This usage scenario aims at demonstrating that the Hazard Log document can be

	automatically generated based on the different links between DI and VER activities already implemented through the AMASS Tool platform.
Conclusions	NA

3.6.4.2. STO4 Cross Intra-domain reuse

Table 37. CS6-Cross Intra-domain reuse: US3-Evidence Management

Realisation Scenario	Evidence Management
Scope	This usage scenario enables the user to record and retrieve consistent artefacts for a baseline system specification.
Tool Settings	AMASS tools (Evidence Management Editor)
Participants	ALS
Activities realised	1. Impact analysis to identify reusable artefacts for a specific project 2. Automatic completion of existing evidence for the Hazard Log
Usage Decisions	NA
Expected Results	This usage scenario aims at demonstrating that the AMASS tool platform allows to correctly identify common reusable artefacts between specific project and generic model in order to automatically fill the Hazard Log.
Conclusions	NA

3.6.5. US4: Compliance Management

The compliance management is performed by Safety Assurance manager directly within the EN 50128 and EN 50129 tables. For each clause, the Safety Assurance manager provides a justification (not applicable because …) or an artefact of one baseline process assurance project (reference to a document, table, text or diagram).

There are two steps in the CS6 process to perform compliance:

- During the Safety Plan redaction: to perform estimated standards compliance (the plan to reach the compliance).
- During the safety case redaction: to perform the resulted standards compliance (how the project reaches the compliance).

During the second iteration, the objective for this usage scenario is to use two of the main AMASS tools functionalities: Architecture-Driven Assurance and Intra-domain reuse.

3.6.5.1. STO1 Architecture-Driven Assurance

Table 38. CS6-Architecture-Driven Assurance: US4-Compliance Management

Realisation Scenario	Compliance Management
Scope	This usage scenario aims at facilitating the Safety Assurance Manager's job when writing a Safety Plan or a Safety Case according to CENELEC standard EN 50129 by automatically linking a number of documents (or document extracts) to CENELEC clauses for the safety demonstration.
Tool Settings	AMASS tools (Evidence Management Editor)
Participants	ALS
Activities realised	1. Use the previously defined CENELEC standards models (especially EN 50129) 2. Define reusable links between CENELEC standard clauses and evidence categories. 3. Generate automatically a clause by clause table for a given type of assurance

	project.
Usage Decisions	NA
Expected Results	This usage scenario shall demonstrate that the clause by clause analysis of CENELEC standard (or part of it at least) can be automatically generated through the use of AMASS tools.
Conclusions	NA

3.6.5.2. STO4 Cross Intra-domain reuse

Table 39. CS6-Cross Intra-domain reuse: US4-Compliance Management

Realisation Scenario	Compliance Management
Scope	This usage scenario aims at facilitating the Safety Assurance Manager's job when writing a Safety Plan or a Safety Case according to CENELEC standard EN 50129 by automatically linking a number of documents (or document extracts) to CENELEC clauses for the safety demonstration.
Tool Settings	AMASS tools (Evidence Management Editor)
Participants	ALS
Activities realised	1. Use the previously defined CENELEC standards models (especially EN 50129) 2. Define reusable links between CENELEC standard clauses and evidence categories. 3. Generate automatically a clause by clause table for a given type of assurance project.
Usage Decisions	NA
Expected Results	This usage scenario shall demonstrate that the process used for the safety demonstration of the Alstom Automatic Train Control can be capitalized and that part of it can be automatically integrated in the clause by clause analysis of the CENELEC standards in order to be reused for other similar projects.
Conclusions	NA

3.6.6. Coverage of AMASS Prototype P1 Architecture

Table 40 illustrates the implemented functionalities during this second iteration within the Case Study 6.

Table 40. Prototype P1 Coverage by CS6

STO	AMASS Functionality Groups	Tools
Architecture-Driven Assurance	System Component Specification	Papyrus/SysML
	System Architecture Modelling for Assurance	-
	Architectural Patterns for Assurance	-
	Contract-based Design for Assurance	Requirements formalization (external)
	Activities supporting Assurance Case	Requirements early validation, Functional Early Verification, model-based safety analysis (external tools)
Multi-Concern Assurance	Assurance Case Specification	OpenCert
	Dependability Assurance	-
	System Dependability Co-Analysis/Co-Assessment	-
	Contract-Based Multi-concern Assurance	-

Seamless Interoperability	Evidence Management	-
	Tool Integration Management	-
	Collaborative Work Management	-
	Tool Quality Assessment and Characterization	-
Cross Intra-Domain Reuse	Compliance Management	Modelling of CENELEC EN 50126, EN 50128, EN 50129.
	Reuse Assistant	-
	Process-related reuse via management of variability at process level	-
	Process-related reuse via management of variability at product level	-
	Automatic generation of process-based arguments	OpenCert
	Automatic generation of product-based arguments	-

3.6.7. Conclusions

At this stage, the main benefits and potential improvements are identified in Table 41.

Table 41. Benefits and potential improvements for CS6

Artefact	Achievement/Benefits	Improvements/Recommendations
Standard modelling	• Modelling of CENELEC EN 50126 in progress. The first model allows to capitalize industrial knowledge on the way Alstom complies with the standard. • Straightforward and user-friendly interface	• The graphical representation of the standard modelling is often difficult to read because of numerous concepts to model in a standard.
Assurance Project Creation	• NA	• The addition of a workflow assistant in order to guide the user through the assurance Project Creation could be beneficial
Overall comments	• The user manual is well written and allows new users to easily start working with the tool. Videos are a useful guidance.	• Some technical difficulties prevent a smooth use of the tool ○ Frequent latencies ○ Instabilities (software crash, nullpointer exceptions) • A 32bits edition of the OpenCert platform should be released (Alstom IT constraint) • Alstom security prevents accessing remote server. Local database for local use is requested for Alstom.

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

3.7.1. Case Study Specification

The Human Machine Interface available to pilots in cockpits can provide several means of communication, like cursor control devices, menu-based controls, touch screens, voice recognition and voice activated controls. The touch screens contain areas that, when touched by the cockpit crew, initiate actions. The Liquid Crystal Display on the touch screen also displays aircraft system information to provide the crew with information that can be used to guide control actions or to provide situational awareness. The recognizer component analyses and classifies touch events and individual gestures.

This Case Study will focus mainly on the following 3 Usage Scenarios:

- US1: Application of aerospace industrial standards for safety assessments
- US2: Automation of the verification objectives
- US3: Reuse of assurance artefacts from automotive technology into the avionics domains

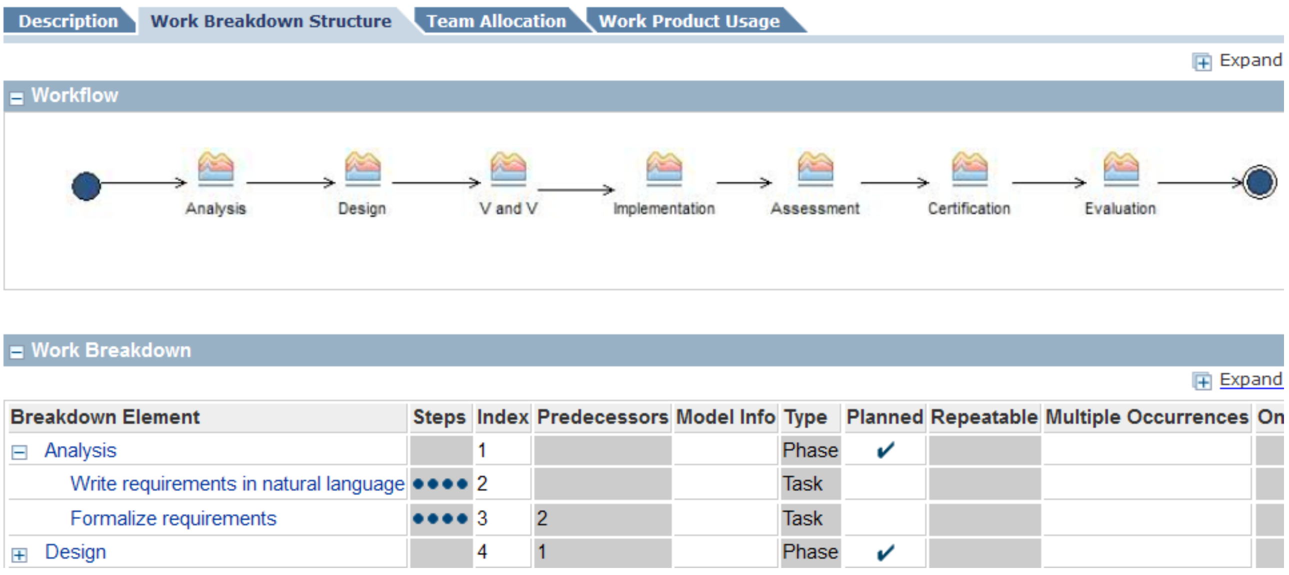
For a detailed description on the case study see the Deliverable "D1.1. Case studies description and business impact" [1].

3.7.2. US1: Application of aerospace industrial standards for safety assessments

The safety assessment must consider the nature of the multi-mode interaction and since the safety assessment should be semi-automated, the safety requirements must be captured in the formal machine readable representation.

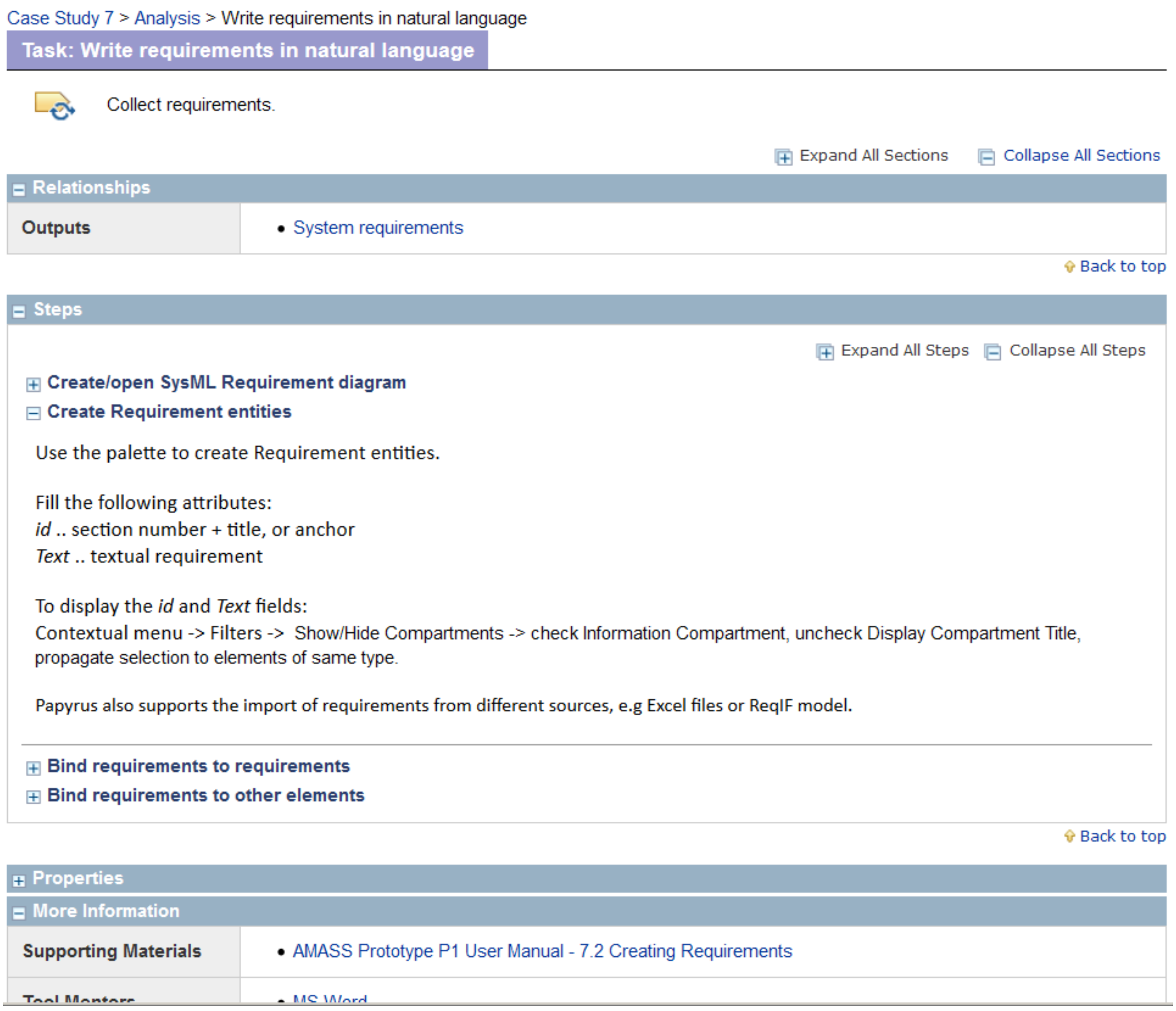
In the first iteration, the requirements for the gesture recognition component were elicited and consolidated.

In the second iteration, the EPF-Composer tool from the AMASS Platform was used to define the development process, see Figure 67 and Figure 68.



Breakdown Element	Steps	Index	Predecessors	Model Info	Type	Planned	Repeatable	Multiple Occurrences	On
⊟ Analysis		1			Phase	✓			
Write requirements in natural language	●●●●	2			Task				
Formalize requirements	●●●●	3	2		Task				
⊞ Design		4	1		Phase	✓			

Figure 67. Development process specification in EPF-Composer, top-level process structure



Case Study 7 > Analysis > Write requirements in natural language

Task: Write requirements in natural language

Collect requirements.

Expand All Sections Collapse All Sections

Relationships

Outputs	• System requirements

Back to top

Steps

Expand All Steps Collapse All Steps

Create/open SysML Requirement diagram

Create Requirement entities

Use the palette to create Requirement entities.

Fill the following attributes:
id .. section number + title, or anchor
Text .. textual requirement

To display the *id* and *Text* fields:
Contextual menu -> Filters -> Show/Hide Compartments -> check Information Compartment, uncheck Display Compartment Title, propagate selection to elements of same type.

Papyrus also supports the import of requirements from different sources, e.g Excel files or ReqIF model.

Bind requirements to requirements

Bind requirements to other elements

Back to top

Properties

More Information

Supporting Materials	• AMASS Prototype P1 User Manual - 7.2 Creating Requirements
Tool Mentors	• MS Word

Figure 68. Process description in EPF-Composer, example steps of a task

The Papyrus tool with the CHESS extension was used to capture the requirements (see Figure 69) according to the EPF process definition mentioned above.

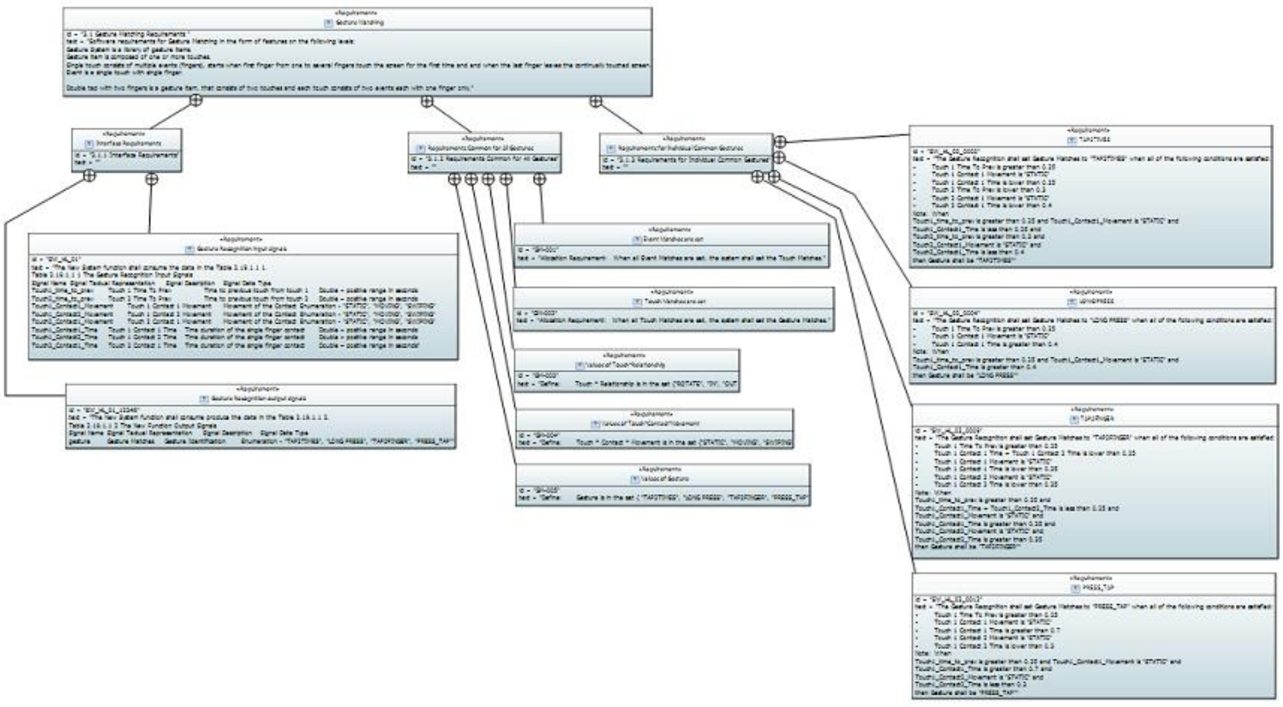


Figure 69. The textual requirements for the developed system (the readability was decreased intentionally)

The Papyrus and CHESS were also used to draw a conceptual model of the system, which is shown in the Figure 70 below.

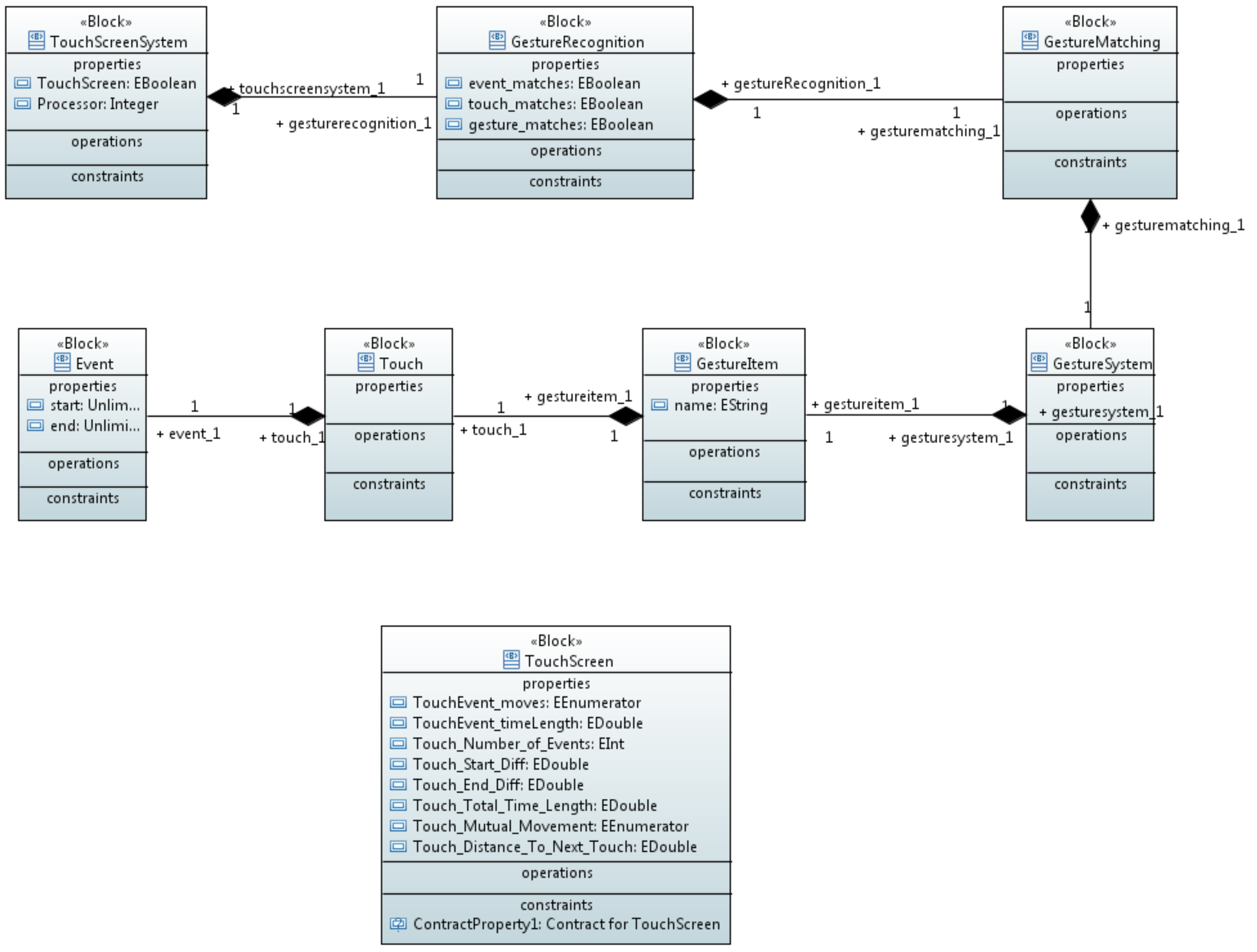


Figure 70. Block Definition Diagram of the Touch screen system

3.7.2.1. STO1 Architecture-Driven Assurance

Table 42. CS7-Architecture-Driven Assurance: US1-Model-based System Component Specification

Realisation Scenario	**Model-based System Component Specification**
Scope	The system components will be specified including system requirements. The following standards will be applied: • *Safety:* SAE ARP 4761 – EUROCAE ED-135 – Guidelines and methods for conducting the safety assessment process on civil airborne systems and equipment. • *System:* SAE ARP 4754A – EUROCAE ED-79A – Guidelines for development of civil aircraft and systems.
Tool Settings	• Mathworks Matlab Simulink – system architecture only. • Sparx Systems Enterprise Architect – 3 View Systems Engineering. • AMASS Platform – Papyrus and CHESS to import the SysML models (requirements, architecture, BDD). • Internal tools to specify system requirements and contracts.
Participants	Leader: HON
Activities realised	1. Author safety requirements 2. Formalize safety requirements 3. Create system architecture – 3 views of the system 4. Allocate requirements to system.
Usage Decisions	Decide which part of the system will be used in Simulink and which in Enterprise Architect. Decide how to allocate the requirements in order to enable automated verification.
Expected Results	System architecture, formal safety requirement specification.
Conclusions	

Table 43. CS7- Architecture-Driven Assurance: US1-Safety Assessment

Realisation Scenario	**Safety Assessment**
Scope	Automated safety assessment for: • Complete MMI system • Touch recognition
Tool Settings	• Mathworks Matlab Simulink – system architecture only. • Sparx Systems Enterprise Architect – 3 View Systems Engineering. • AMASS Platform – Papyrus and CHESS to import the SysML models (requirements, architecture, BDD) this allows to use safety assessment tools to get minimal cut sets and other safety assessment artefacts. • Tools for automated safety assessment.
Participants	• Leader: HON • Tool providers: UOM, FBK
Activities realised	1. Identify system failures. 2. Select appropriate system architecture pattern to reach design assurance level. 3. Apply fault injection mechanism. 4. Compute fault propagation automatically using for example FBK and UOM tools. 5. Perform safety assessment, for example generate Fault Tree Analysis (FTA).

	6. Evaluate the results.
Usage Decisions	Which parts of safety assessment should be automated. Which safety assessment tools used.
Expected Results	A list of safety hazards.
Conclusions	

3.7.2.2. STO3 Seamless Interoperability

Table 44. CS7-Seamless Interoperability: US1-Evidence Assessment

Realisation Scenario	**Evidence Assessment**
Scope	Collect and manage evidence artefacts required to fulfil the selected standards
Tool Settings	OpenCert Tools: Evidence Management
Participants	• Leader: HON • TEC
Activities realised	1. Create artefact model for safety. 2. Collect evidence documents. 3. Initial verification and judgment of the quality of the evidence. 4. Perform evaluation of results based on D1.3 [3]. 5. Report the result to the customer.
Usage Decisions	How to measure baseline process – a process performed by separate team?
Expected Results	Evidence model and artefact repository.
Conclusions	

3.7.2.3. STO4 Cross Intra-domain reuse

Table 45. CS7-Cross Intra-domain reuse: US1-Compliance Management

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

Table 46. CS7-Cross- and Intra-domain reuse: US1-Compliance Management

Realisation Scenario	**Compliance Management**

Scope	Representation of DO-178C with ontology-based technologies
Tool Settings	Knowledge Manager (KM; TRC)
Participants	• UC3 • TRC
Activities realised	The activities correspond to the application of the approach for semantic representation of safety standards presented in D6.5 and D6.7. 1. KM configuration for representation of DO-178C 2. Initial specification of an ontology for DO-178C with its glossary 3. Modelling of DO-178C development process in KM, based on the metamodel for Reference Assurance Frameworks (Reference Activities, Reference Artefacts, Reference Artefact Relationships, Reference Activity input, Reference Activity output, etc.)
Usage Decisions	None
Expected Results	Semantic representation of DO-178C in KM
Conclusions	Successful representation

3.7.3. US2: Automation of verification objectives

Automation of the formal requirements using formal methods will be performed to save time, cost and cost of poor quality. High-level system will be modelled and the corresponding (i.e. high-level) requirements will be semantically checked and formally verified against the low-level requirements written in Simulink or C system design using model checking and testing. The plan is to allow verification of knowledge based system behind the multi-modal interaction.

During the first Iteration, the automated semantic analysis of formal requirements was augmented with the reliability checking.

During the second Iteration, the development of the V&V Manager has started. It communicates with the Verification Server. The V&V Manager composes requests for formal verification from the contents of the select model element (Contract or Block) and sends those requests to the Verification Server, see Figure 71.

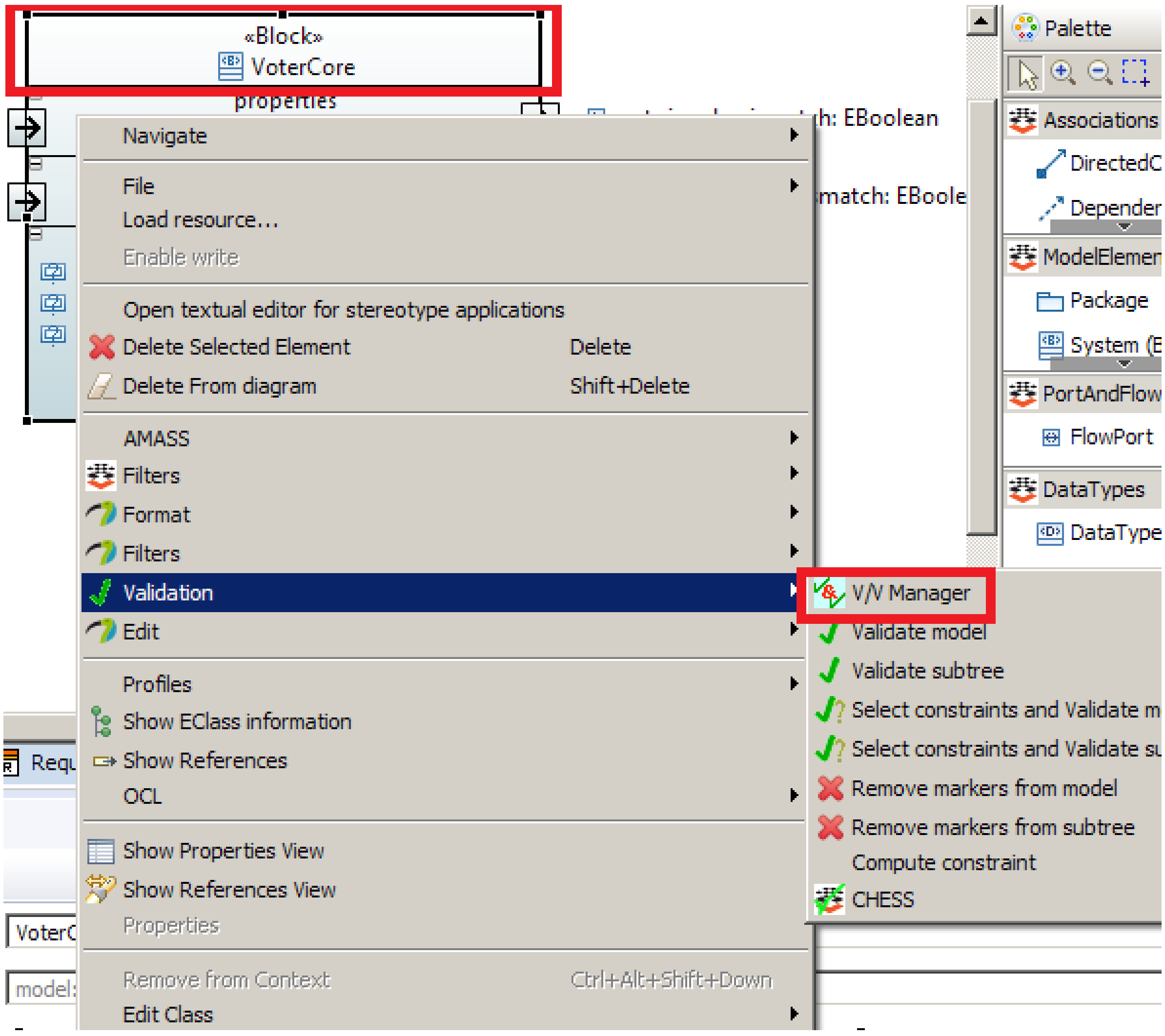
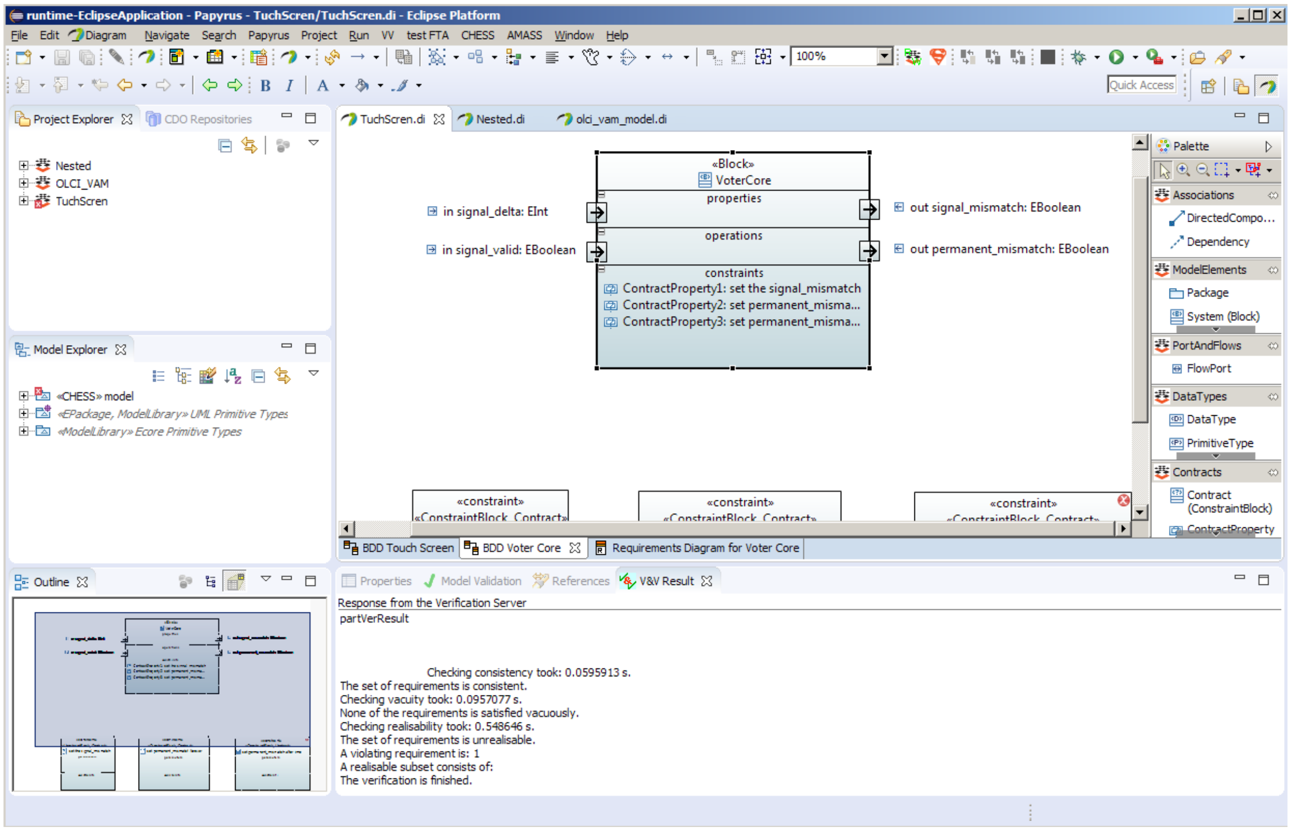


Figure 71. Invocation of the V&V Manager from the contextual menu of a block

Depending on the incoming request the Verification Server invokes various kinds of semantic analysis of requirements or the verification of a model against the requirements. When the work managed by the Verification Server is completed, the result is sent back to the V&V Manager, which displays it in the V&V Result View, see Figure 72.


runtime-EclipseApplication - Papyrus - TuchScren/TuchScren.di - Eclipse Platform
Project Explorer
Model Explorer
Outline
«Block» VoterCore
in signal_delta: EInt
in signal_valid: EBoolean
out signal_mismatch: EBoolean
out permanent_mismatch: EBoolean
ContractProperty1: set the signal_mismatch
ContractProperty2: set permanent_misma...
ContractProperty3: set permanent_misma...
BDD Touch Screen
BDD Voter Core
Requirements Diagram for Voter Core
Palette
V&V Result

```
Response from the Verification Server
partVerResult


                    Checking consistency took: 0.0595913 s.
The set of requirements is consistent.
Checking vacuity took: 0.0957077 s.
None of the requirements is satisfied vacuously.
Checking realisability took: 0.548646 s.
The set of requirements is unrealisable.
A violating requirement is: 1
A realisable subset consists of:
The verification is finished.
```

Figure 72. The V&V Result View containing the result of the semantic analysis of requirements

The approach for the knowledge-based part of the Multi-modal Interaction use case is to create the automated translation from the representation of the rules into the formal language to enable automation of the formal verification process.

Succinctness check finds the requirements that could be simplified while still defining the same behaviour.

Technology is based on using the mutated requirements and when equivalent succinct version is found this is showed to the engineer.

When system design is created, this technology can also help with creating alternative more demanding version of the requirement, which is still satisfied by the underlying system design.

Engineer decides if the intention was the succinct version of the requirement or the more demanding version (suggested by the tool or written manually by the engineer).

The decision cannot be made automatically since original intention could be wrong and therefore the tool cannot rely on the other artefacts.

Example from Touch subsystem:

> The *Gesture Recognition* shall set Gesture Matches to “T2F” when all of the following conditions are satisfied:
>
> o Occur To Prev is greater than 1.25
> o Cont 1 + Cont 2 is lower than 0.7
> o Cont 1 Movement is “STATIC”
> o Cont 1 is lower than 0.7
> o Cont 2 Movement is “STATIC”
> o Cont 2 is lower than 0.8

In this simple example the succinctness check finds statically without creating mutated requirements that the red conditions are implied by the green condition and thus are redundant. In this case the succinct version is the requirement without the red conditions.

3.7.3.1. STO1 Architecture-Driven Assurance

Table 47. CS7-Architecture-Driven Assurance: US2-Model-based System Component Specification

Realisation Scenario	Model-based System Component Specification
Scope	The system components will be specified including system requirements. The following standards will be applied: *System:* SAE ARP 4754A – EUROCAE ED-79A – Guidelines for development of civil aircraft and systems
Tool Settings	• Mathworks Matlab Simulink – system architecture only • Sparx Systems Enterprise Architect – 3 View Systems Engineering • Internal tools to specify system requirements and contracts
Participants	Leader: HON
Activities realised	1. Author system requirements 2. Formalize behavioural system requirements 3. Create system architecture – 3 views of the system 4. Allocate requirements to the system.
Usage Decisions	Decide which part of the system will be used in Simulink and which in Enterprise Architect. Decide how to allocate the requirements in order to enable automated verification.
Expected Results	System architecture, formal system requirement specification.
Conclusions	

Table 48. CS7- Architecture-Driven Assurance: US2-Automated Verification

Realisation Scenario	Automated Verification
Scope	Automated formal semantic verification and validation of requirements. Contribution to the following objectives: • Enforced verifiability (DO-178C A-3.4) • Ensured conformance to requirement standards (DO-178C A-3.5) • Decrease the number of defects (ARP 4761 objective, DO-178C A-3.2) • Automated formal verification that requirements comply with system. • Contribution to the objectives: DO178C A-4.1. and when possible to DO178C A-3.1
Tool Settings	• Mathworks Matlab Simulink – system architecture only • Sparx Systems Enterprise Architect – 3 View Systems Engineering • AMASS Platform – Papyrus and CHESS to import the SysML models (requirements, architecture, BDD) this allows to use V&V Manager to verify requirements early on. • Internal tool for formalisation, analysis and brokerage of verification tasks: ForReq • Tools for automated formal verification: DIVINE, NuSMV, nuXmv, Looney, Acacia+
Participants	• Leader: HON • TEC

	• Tool providers: UOM, FBK
Activities realised	1. Automated semantic requirement analysis 2. Create system design 3. Automated formal verification of requirement compliance with system design 4. Automated generation of test cases if requested 5. Evaluate the results
Usage Decisions	Which parts of safety assessment should be automated. Which safety assessment tools used.
Expected Results	Verification results, measurements of the injected, detected and remove defects in all development phases.
Conclusions	

3.7.3.2. STO3 Seamless Interoperability

Table 49. CS7-Seamless Interoperability: US2-Evidence Assessment

Realisation Scenario	**Evidence Assessment**
Scope	Collect and manage evidence artefacts required to fulfil the selected standards.
Tool Settings	OpenCert Tools: Evidence Management
Participants	• Leader: HON • TEC
Activities realised	1. Create artefact model for the system. 2. Collect evidence documents 3. Initial verification and judgment of the quality of the evidence 4. Perform evaluation of results based on D1.3 [3] 5. Report the result to the customer.
Usage Decisions	How to measure baseline process – a process performed by separate team?
Expected Results	Evidence model and artefact repository.
Conclusions	

3.7.3.3. STO4 Cross Intra-domain reuse

Table 50. CS7-Cross Intra-domain reuse: US2-Compliance Management

Realisation Scenario	**Compliance Management**
Scope	Evaluate compliance of artefacts as per the avionics standards.
Tool Settings	OpenCert Tools: Assurance Project Management
Participants	• Leader: HON • TEC
Activities realised	1. Definition of the development plan 2. Definition of tasks and tools to be integrated. 3. Evaluation of the development lifecycle based on AMASS evaluation framework 4. Reporting of compliance results to the customer 5. Analyse compliance accomplishment
Usage Decisions	None
Expected Results	Compliance report complying with standards: • SAE ARP 4761 – EUROCAE ED-135

	• SAE ARP 4754A – EUROCAE ED-79° • RTCA DO-178C – EUROCAE ED-12C
Conclusions	

3.7.4. Coverage of AMASS Prototype P1 Architecture

Table 51 illustrates the implemented functionalities during this second iteration within the Case Study 7.

Table 51. AMASS Prototype P1 Coverage by CS7

STO	AMASS Functionality Groups	Tools
Architecture-Driven Assurance	System Component Specification	CHESS
	System Architecture Modelling for Assurance	CHESS
	Architectural Patterns for Assurance	-
	Contract-based Design for Assurance	CHESS + OCRA
	Activities supporting Assurance Case	DIVINE, NuSMV, nuXmv, Looney, Acacia+ V&V Manager
Multi-Concern Assurance	Assurance Case Specification	-
	Dependability Assurance	-
	System Dependability Co-Analysis/Co-Assessment	EPF-C+ OpenCert
	Contract-Based Multi-concern Assurance	-
Seamless Interoperability	Evidence Management	OpenCert
	Tool Integration Management	V&V Manager and OSLC Automation V&V Tool Integration
	Collaborative Work Management	-
	Tool Quality Assessment and Characterization	-
Cross Intra-Domain Reuse	Compliance Management	OpenCert/Knowledge Manager (semantics mapping)
	Reuse Assistant	-
	Process-related reuse via management of variability at process level	-
	Process-related reuse via management of variability at product level	-
	Automatic generation of process-based arguments	OpenCert
	Automatic generation of product-based arguments	OpenCert

3.7.5. Conclusions

At this stage, the main benefits and potential improvements are identified in Table 52.

Table 52. Benefits and potential improvements for CS7

Artefact	Achievement/Benefits	Improvements/Recommendations
Requirements specification and formalization	• Improved requirement grammar allows more complex real-time requirements authoring. • Requirements linked to design entities, formal properties and contracts	• Formal requirement grammar is yet not part of *PropertyEditor*. Standalone XText implementation of requirement authoring is being developed.

Artefact	Achievement/Benefits	Improvements/Recommendations
	• Traceability	
Automated Verification	• V&V Manager allows verify requirement semantic analysis early on using contracts and *FormalProperties* – consistency, redundancy, realisability and missing requirements. • Subsystem is newly supported.	• Support to verify system design and architecture design will be added to V&V Manager. • Extend scalability of the requirement semantic analysis especially for realisability checking.
Design of system/software architecture	• Tool allow interchange of the design artefacts with other tools. • All the information defined in a single model.	• Improve the GUI, for example: o Sometimes some options are missing and random clicking is needed to make it reappear again. o Similar analyses in different menus • Tool maturity is very low.
Conduction of safety analyses	• The V&V and safety analyses provide minimal cut set. • Evidences directly obtained from the model.	• Tool maturity is very low. • Need a lot of manual effort to perform the analysis completely. • Tool limitations (only discrete type, state machines do not support i) operations and ii) state machines in non-leaf components).

3.8. Case Study 8: Automotive domain: Telematics function

3.8.1. Case Study Specification

This case study focuses on component-based (element-out-of-context) multi-concern assurance, analysis and assessment. The intended item is an automated driving function (ADC item) and specifically its ability to determine when the vehicle's geographical position is on a road where it can use the automated driving function. The element-out-of-context is a positioning component where safety, security and performance are critical quality attributes. The component will have cybersecurity goals up to Cybersecurity Assurance Level (CAL) 4, the highest security criticality level, and safety goals up to Automotive safety integrity level (ASIL) D, the highest safety integrity level in ISO 26262.

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

3.8.2. US1: Multi-concern assurance case for safety/security (US1 MCAC)

This scenario is about creation of an assurance case and processes for multiple standards (safety and security). This scenario is mainly related to assurance case specification, evidence management and compliance management.

During the first Iteration, a safety case according to ISO 26262 and most of its artefacts was created in Word and Excel.

During the second Iteration, transferred much of the safety case information into OpenCert and system modelling was done in SysML using Papyrus. A security case has been started and is also modelled in OpenCert. The assurance case in OpenCert includes modelling of reference frameworks for ISO 26262 and automotive cybersecurity, argumentation, evidence and process modelling, and finally compliance mapping. All parts are not complete.

It can be noted that, for both iterations, the assurance cases are partial, i.e. not all safety goals, components, etc. have been elaborated. The focus is on including all parts of an assurance case rather than completing all parts of the rather complex functionality of the ADC item and positioning element.

Usage scenario US1 is represented by the activities labelled with a yellow tag US1 in Figure 73. All the activities in US1 are started during the second iteration. The corresponding functionalities in OpenCert have been validated and feedback collected, i.e. reference framework modelling, assurance project creation, minor work on equivalence mapping, GSN argument modelling (including multi-concern argumentation), evidence model creation, process modelling in EPF and import to OpenCert, some work on compliance mapping, and some report generation from the web interface. In addition, the case study has done set-up and management of an own OpenCert CDO repository.

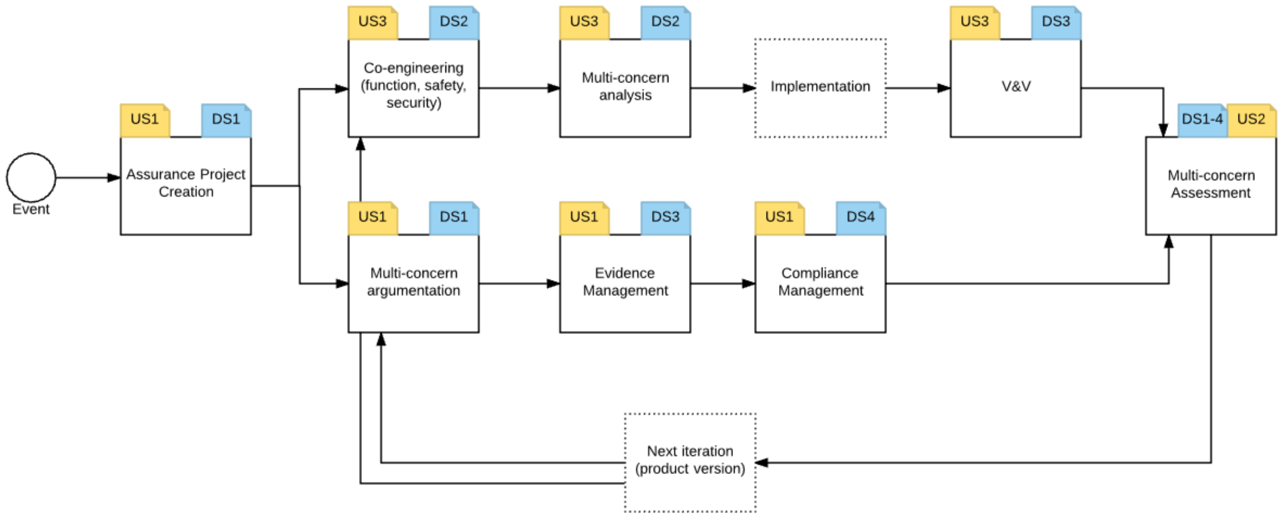


Figure 73. Overview of CS8 usage scenarios

3.8.2.1. STO2 Multi-concern Assurance

To model and build a case for multi-concern assurance, an essential part is to have a reference frame work for the standards one aims to adhere to. Figure 74 is an example of the model of the reference frame work in OpenCert that covers the automotive cybersecurity concern of the of the multi-concern assurance case. Figure 75 is showing a view of the corresponding work done for safety, in this case ISO26262. The modelling of argumentation for compliance of the multi-concern is also done in OpenCert (see Figure 76). One also needs to substantiate the argumentation with evidence, an example of this can be seen in Figure 77. All these activities and artefact will eventually tie in together to form a solid claim for a fulfilled multi-concern assurance case covering Safety, Security and Availability.

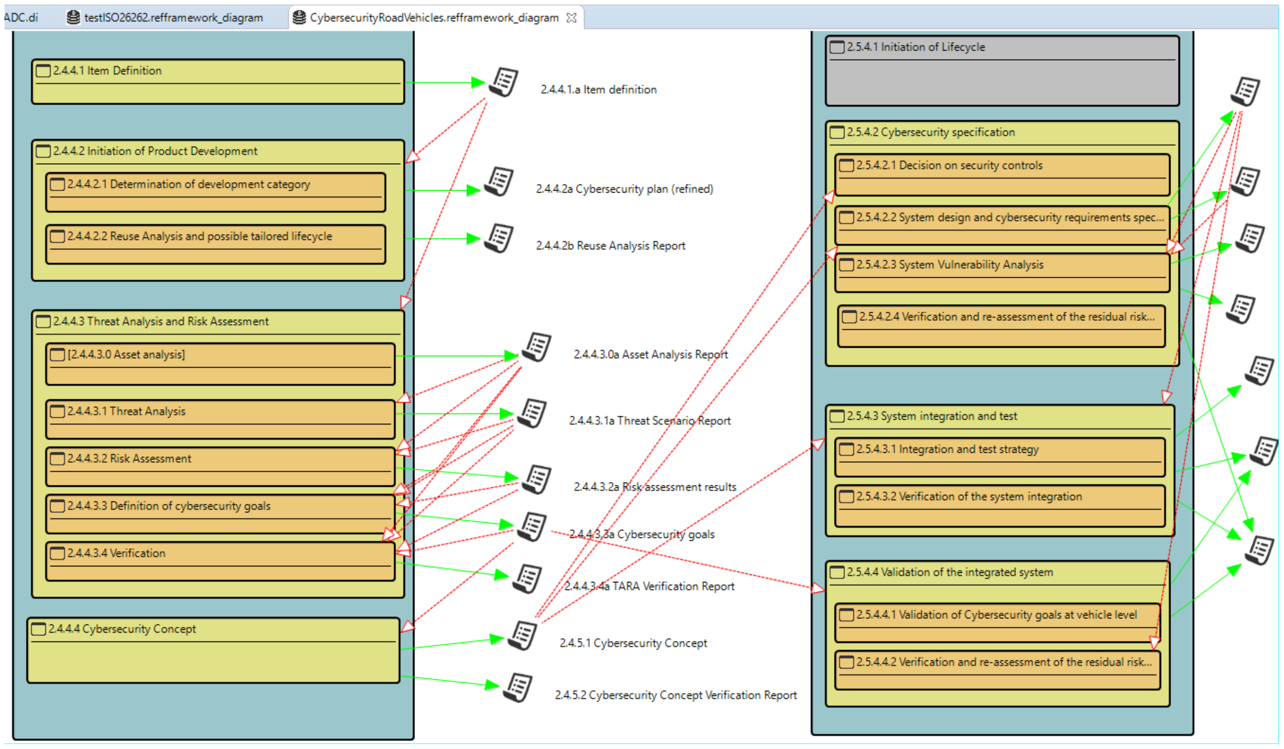

ADC.di
testISO26262.refframework_diagram
CybersecurityRoadVehicles.refframework_diagram
2.4.4.1 Item Definition
2.4.4.2 Initiation of Product Development
2.4.4.2.1 Determination of development category
2.4.4.2.2 Reuse Analysis and possible tailored lifecycle
2.4.4.3 Threat Analysis and Risk Assessment
[2.4.4.3.0 Asset analysis]
2.4.4.3.1 Threat Analysis
2.4.4.3.2 Risk Assessment
2.4.4.3.3 Definition of cybersecurity goals
2.4.4.3.4 Verification
2.4.4.4 Cybersecurity Concept
2.4.4.1.a Item definition
2.4.4.2a Cybersecurity plan (refined)
2.4.4.2b Reuse Analysis Report
2.4.4.3.0a Asset Analysis Report
2.4.4.3.1a Threat Scenario Report
2.4.4.3.2a Risk assessment results
2.4.4.3.3a Cybersecurity goals
2.4.4.3.4a TARA Verification Report
2.4.5.1 Cybersecurity Concept
2.4.5.2 Cybersecurity Concept Verification Report
2.5.4.1 Initiation of Lifecycle
2.5.4.2 Cybersecurity specification
2.5.4.2.1 Decision on security controls
2.5.4.2.2 System design and cybersecurity requirements spec...
2.5.4.2.3 System Vulnerability Analysis
2.5.4.2.4 Verification and re-assessment of the residual risk...
2.5.4.3 System integration and test
2.5.4.3.1 Integration and test strategy
2.5.4.3.2 Verification of the system integration
2.5.4.4 Validation of the integrated system
2.5.4.4.1 Validation of Cybersecurity goals at vehicle level
2.5.4.4.2 Verification and re-assessment of the residual risk...

Figure 74. Part of automotive cybersecurity reference framework model in CS8

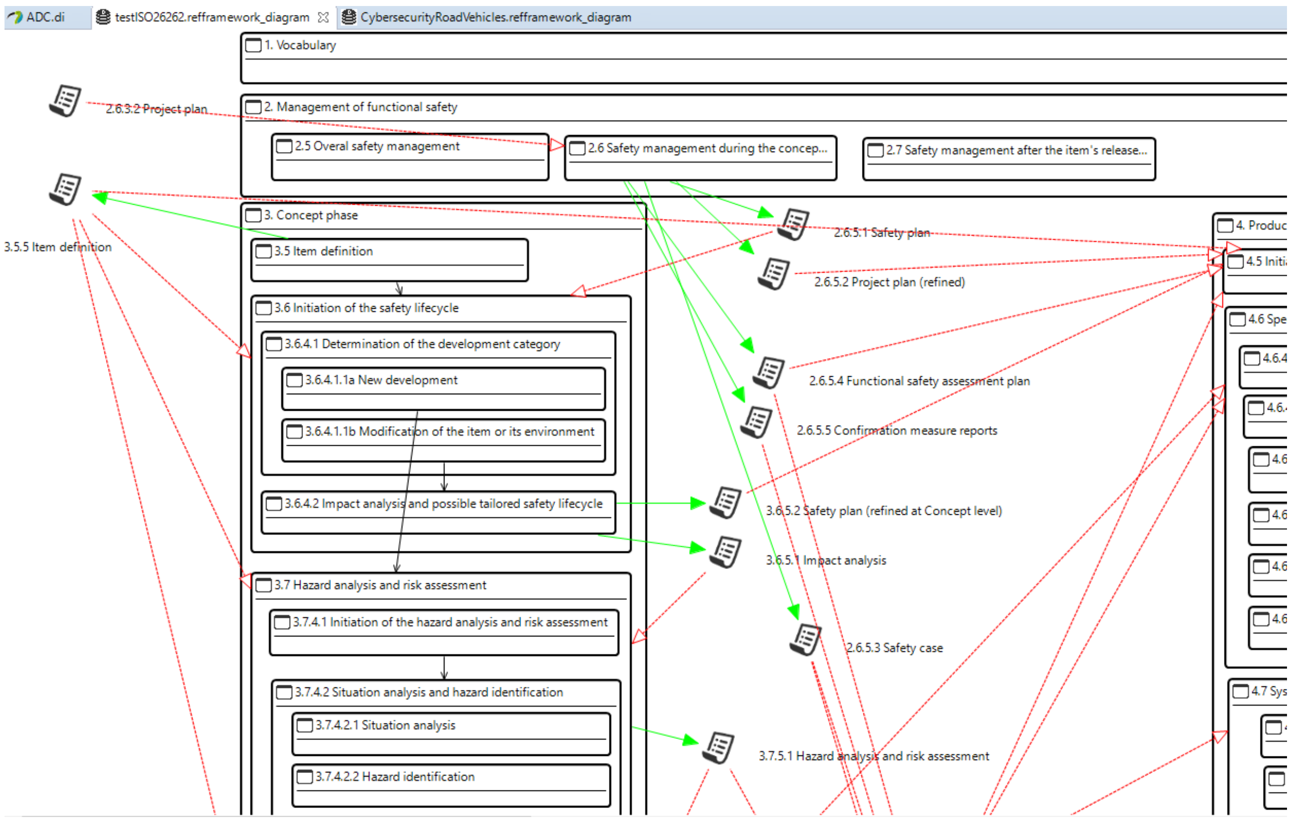

ADC.di
testISO26262.refframework_diagram
CybersecurityRoadVehicles.refframework_diagram
1. Vocabulary
2.6.3.2 Project plan
2. Management of functional safety
2.5 Overal safety management
2.6 Safety management during the concep...
2.7 Safety management after the item's release...
3.5.5 Item definition
3. Concept phase
3.5 Item definition
2.6.5.1 Safety plan
2.6.5.2 Project plan (refined)
4. Produc
4.5 Initi
3.6 Initiation of the safety lifecycle
3.6.4.1 Determination of the development category
3.6.4.1.1a New development
3.6.4.1.1b Modification of the item or its environment
3.6.4.2 Impact analysis and possible tailored safety lifecycle
2.6.5.4 Functional safety assessment plan
2.6.5.5 Confirmation measure reports
3.6.5.2 Safety plan (refined at Concept level)
3.6.5.1 Impact analysis
3.7 Hazard analysis and risk assessment
3.7.4.1 Initiation of the hazard analysis and risk assessment
2.6.5.3 Safety case
3.7.4.2 Situation analysis and hazard identification
3.7.4.2.1 Situation analysis
3.7.4.2.2 Hazard identification
3.7.5.1 Hazard analysis and risk assessment
4.6 Spe
4.6.4
4.6.
4.6
4.6
4.6
4.6
4.7 Sys

Figure 75. Part of ISO 26262 reference framework model

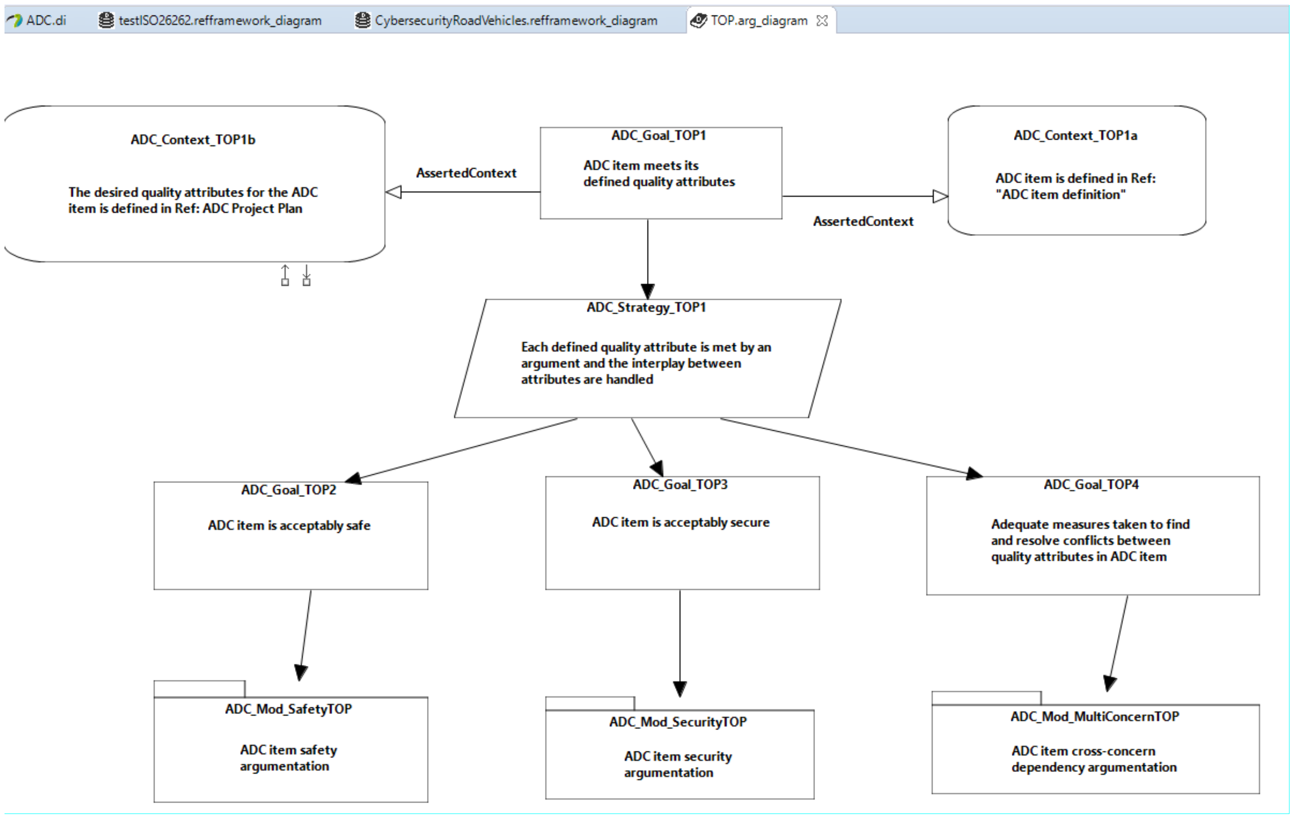

ADC.di
testISO26262.refframework_diagram
CybersecurityRoadVehicles.refframework_diagram
TOP.arg_diagram
ADC_Context_TOP1b
The desired quality attributes for the ADC item is defined in Ref: ADC Project Plan
AssertedContext
ADC_Goal_TOP1
ADC item meets its defined quality attributes
AssertedContext
ADC_Context_TOP1a
ADC item is defined in Ref: "ADC item definition"
ADC_Strategy_TOP1
Each defined quality attribute is met by an argument and the interplay between attributes are handled
ADC_Goal_TOP2
ADC item is acceptably safe
ADC_Goal_TOP3
ADC item is acceptably secure
ADC_Goal_TOP4
Adequate measures taken to find and resolve conflicts between quality attributes in ADC item
ADC_Mod_SafetyTOP
ADC item safety argumentation
ADC_Mod_SecurityTOP
ADC item security argumentation
ADC_Mod_MultiConcernTOP
ADC item cross-concern dependency argumentation

Figure 76. Top module in argumentation for CS8 ADC item

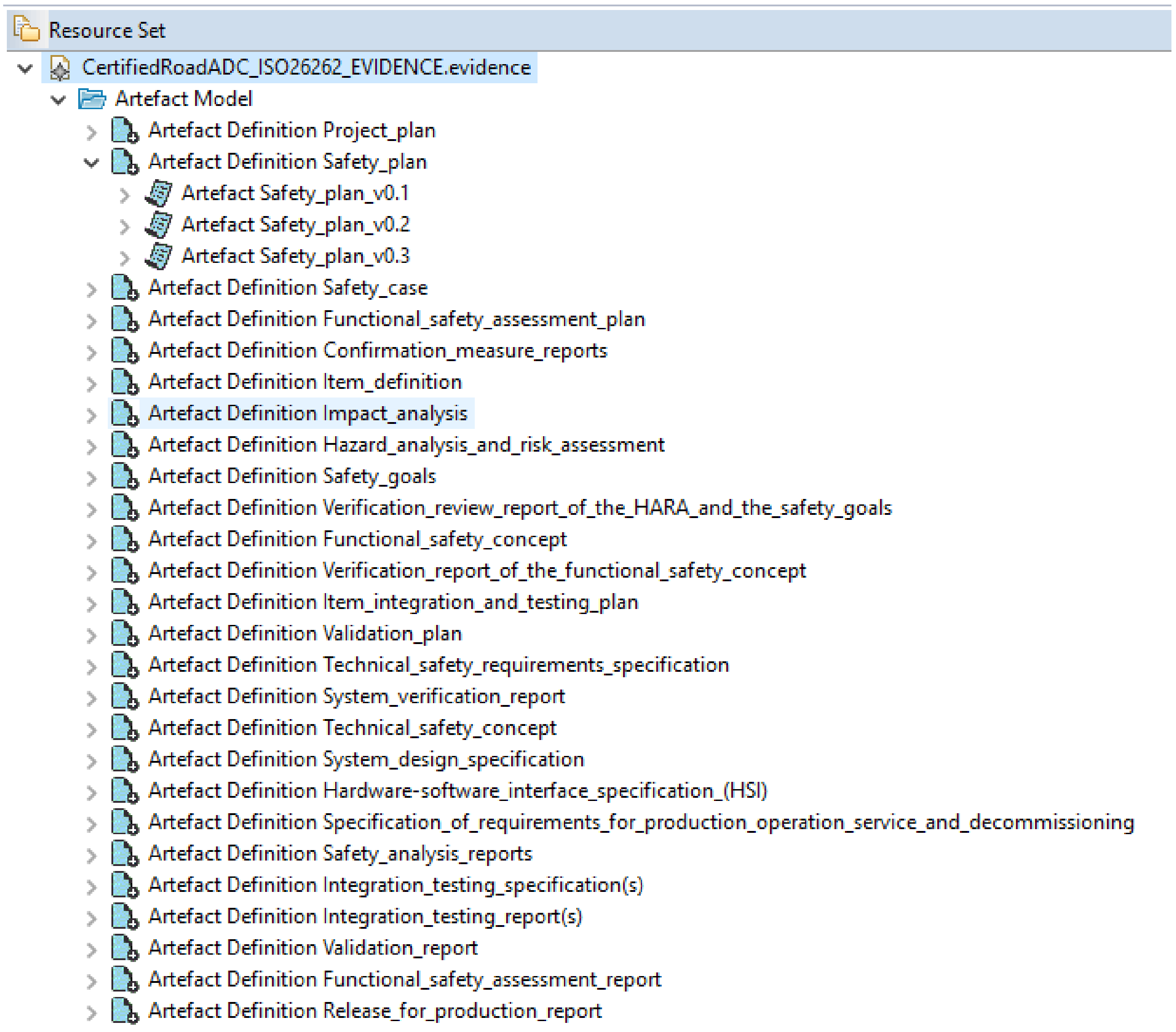

Resource Set
CertifiedRoadADC_ISO26262_EVIDENCE.evidence
Artefact Model
Artefact Definition Project_plan
Artefact Definition Safety_plan
Artefact Safety_plan_v0.1
Artefact Safety_plan_v0.2
Artefact Safety_plan_v0.3
Artefact Definition Safety_case
Artefact Definition Functional_safety_assessment_plan
Artefact Definition Confirmation_measure_reports
Artefact Definition Item_definition
Artefact Definition Impact_analysis
Artefact Definition Hazard_analysis_and_risk_assessment
Artefact Definition Safety_goals
Artefact Definition Verification_review_report_of_the_HARA_and_the_safety_goals
Artefact Definition Functional_safety_concept
Artefact Definition Verification_report_of_the_functional_safety_concept
Artefact Definition Item_integration_and_testing_plan
Artefact Definition Validation_plan
Artefact Definition Technical_safety_requirements_specification
Artefact Definition System_verification_report
Artefact Definition Technical_safety_concept
Artefact Definition System_design_specification
Artefact Definition Hardware-software_interface_specification_(HSI)
Artefact Definition Specification_of_requirements_for_production_operation_service_and_decommissioning
Artefact Definition Safety_analysis_reports
Artefact Definition Integration_testing_specification(s)
Artefact Definition Integration_testing_report(s)
Artefact Definition Validation_report
Artefact Definition Functional_safety_assessment_report
Artefact Definition Release_for_production_report

Figure 77. Evidence model for CS8 ADC item

Table 53. CS8-Multi-concern Assurance: US1-Multi-concern assurance case for safety/security

Realisation Scenario	Multi-concern assurance case for safety/security
Scope	Creation of an assurance case for multiple standards (safety: ISO 26262 and automotive cybersecurity).
Tool Settings	OpenCert and EPF
Participants	SPS, COM
Activities realised	• Modelling of reference frameworks for ISO 26262 and automotive cybersecurity. • Creation of multi-concern assurance cases for ADC item and positioning element. • Argumentation for ADC item and start of argumentation for positioning element. • Evidence model for ADC item. • Process model for ADC item. • Some compliance mapping.
Usage Decisions	The safety and security cases are partial. After the initial complete system model, the assurance case is aimed at completing what is needed for analysing

	the impact on one element and one safety goal.
Expected Results	A multi-concern assurance case possible to assess in US2.
Conclusions	Feedback from using the tools has been collected for the tool developers. Completion of the scenario and metrics collection are left for the third iteration.

3.8.3. US2: 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 usage scenario US1. This scenario is mainly related to evidence management and compliance management.

US3 is a smaller scenario focused on a functional safety assessment (FSA) of the work in US1 and US3, i.e. whereas the other scenarios look at assurance from the viewpoint of the developing organization creating the assurance case, the purpose of this scenario is to evaluate if the assurance case is suitable from the viewpoint of an independent assessor performing a functional safety assessment. This scenario follows after US1 and US3 (see Figure 73) and since these scenarios are not complete yet this scenario has not yet been performed and will be left for the third iteration.

3.8.4. US3: Multi-concern specification, analysis, assurance (US2 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 in usage scenarios 1 and 2 is used.

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

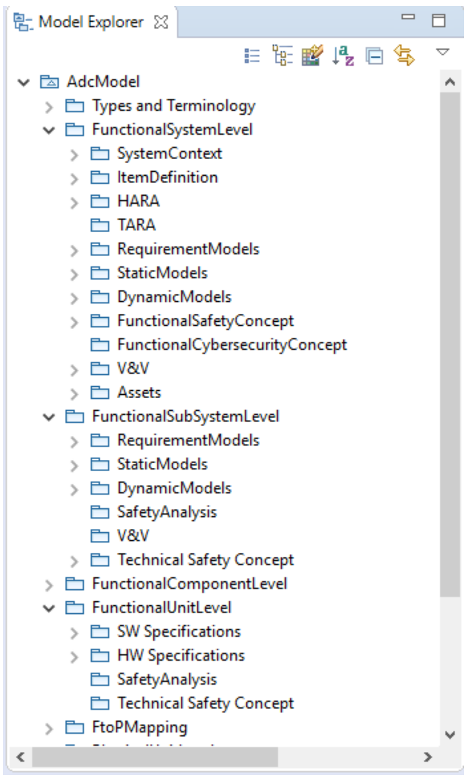
During the first iteration, an early version of system specification, functional safety concept and technical safety concept was done in word.

During the second iteration, a SysML/Papyrus model of the ADC item has been created. This includes item context, concept, functional, and technical implementation level, as well as requirements.

Usage scenario US3 is represented by the activities labelled with a yellow tag US3 in Figure 73. So far no AMASS platform specific functionality has been used for this scenario.

3.8.4.1. STO1 Architecture-Driven Assurance

The industry need for building an assurance case with elements out of context (EooC) has been identified and the feasibility of this, within the AMASS framework, will be investigated, i.e. how to handle requirements (Figure 79) and system modelling (Figure 78, Figure 80) with EooC in mind.


Model Explorer

- AdcModel
  - Types and Terminology
  - FunctionalSystemLevel
    - SystemContext
    - ItemDefinition
    - HARA
    - TARA
    - RequirementModels
    - StaticModels
    - DynamicModels
    - FunctionalSafetyConcept
    - FunctionalCybersecurityConcept
    - V&V
    - Assets
  - FunctionalSubSystemLevel
    - RequirementModels
    - StaticModels
    - DynamicModels
    - SafetyAnalysis
    - V&V
    - Technical Safety Concept
  - FunctionalComponentLevel
  - FunctionalUnitLevel
    - SW Specifications
    - HW Specifications
    - SafetyAnalysis
    - Technical Safety Concept
  - FtoPMapping

Figure 78. Information captured in SysML model

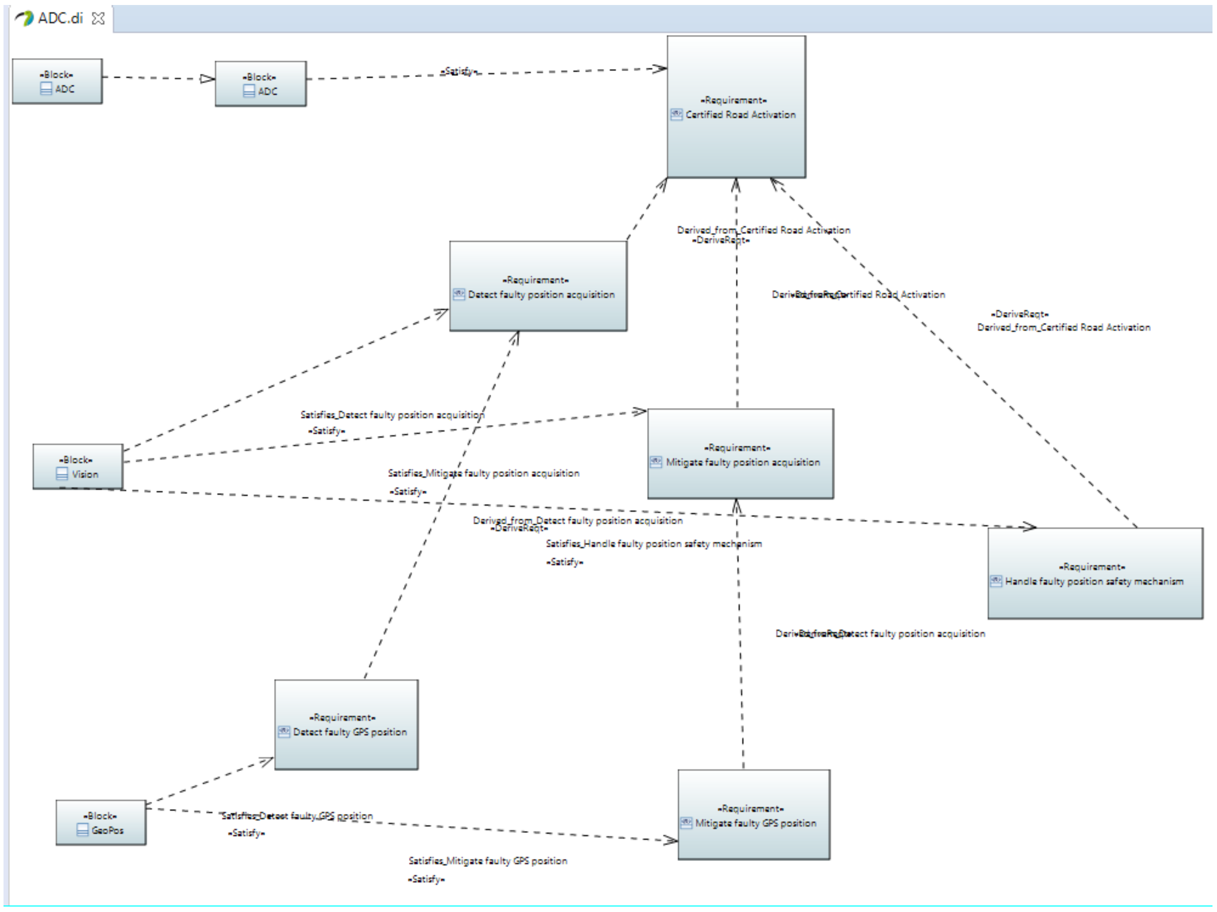

ADC.di
«Block» ADC
«Block» ADC
«Satisfy»
«Requirement» Certified Road Activation
Derived_from_Certified Road Activation
«DeriveReqt»
«Requirement» Detect faulty position acquisition
«DeriveReqt» Derived_from_Certified Road Activation
«DeriveReqt»
Derived_from_Certified Road Activation
Satisfies_Detect faulty position acquisition
«Satisfy»
«Block» Vision
«Requirement» Mitigate faulty position acquisition
Satisfies_Mitigate faulty position acquisition
«Satisfy»
Derived_from_Detect faulty position acquisition
«DeriveReqt»
Satisfies_Handle faulty position safety mechanism
«Satisfy»
«Requirement» Handle faulty position safety mechanism
«DeriveReqt» Derived_from_Detect faulty position acquisition
«Requirement» Detect faulty GPS position
«Requirement» Mitigate faulty GPS position
«Block» GeoPos
Satisfies_Detect faulty GPS position
«Satisfy»
Satisfies_Mitigate faulty GPS position
«Satisfy»

Figure 79. Safety requirements and traceability for safety goal “ADC may only be activated on enabled certified roads”

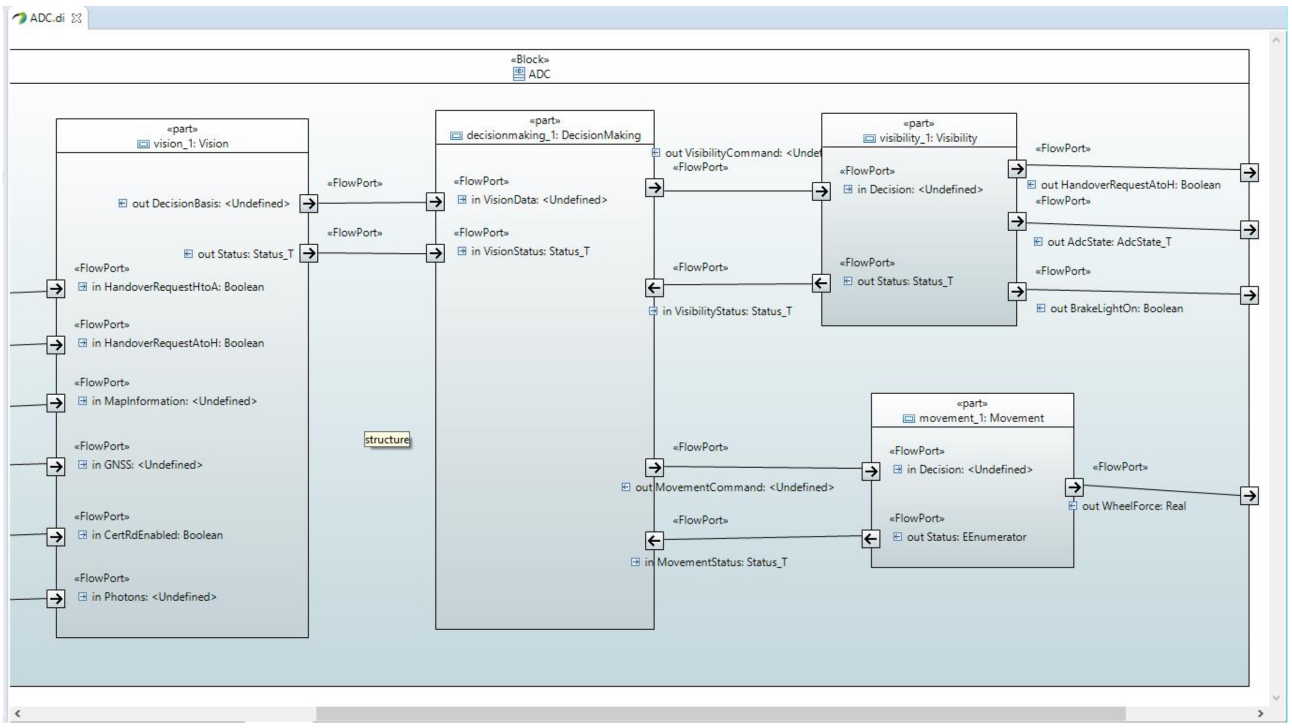

ADC.di
«Block» ADC
«part» vision_1: Vision
out DecisionBasis: <Undefined>
out Status: Status_T
in HandoverRequestHtoA: Boolean
in HandoverRequestAtoH: Boolean
in MapInformation: <Undefined>
in GNSS: <Undefined>
in CertRdEnabled: Boolean
in Photons: <Undefined>
«part» decisionmaking_1: DecisionMaking
in VisionData: <Undefined>
in VisionStatus: Status_T
out VisibilityCommand: <Undef
in VisibilityStatus: Status_T
out MovementCommand: <Undefined>
in MovementStatus: Status_T
«part» visibility_1: Visibility
in Decision: <Undefined>
out Status: Status_T
out HandoverRequestAtoH: Boolean
out AdcState: AdcState_T
out BrakeLightOn: Boolean
«part» movement_1: Movement
in Decision: <Undefined>
out Status: EEnumerator
out WheelForce: Real
«FlowPort»
structure

Figure 80. Top-level system specification for ADC item

Table 54. CS8-Architecture-Driven Assurance: US3-Multi-concern specification, analysis, assurance

Realisation Scenario	Multi-concern specification, analysis, assurance
Scope	Model and specify a safety and security-related item and element-out-of-context
Tool Settings	SysML/Papyrus (CHESS planned for iteration 3)
Participants	COM, SPS
Activities realised	• System model for ADC item created from concept to technical level. • Requirements included in system model, nominal, safety, (security)
Usage Decisions	ADC item modelled in plain SysML. CHESS specific features only to be used on element-out-of-context, but this has had to be delayed until the next iteration.
Expected Results	System model for the ADC item and positioning element in the case study.
Conclusions	AMASS platform features have not been used yet for this scenario but will come into play in the last iteration.

3.8.4.2. STO2 Multi-concern Assurance

The activities and artefact will eventually tie in together to form a solid claim for a fulfilled multi-concern assurance case covering Safety, Security and (Availability), however in order to establish a balance between the concerns and identify synergies from co-engineering, an analysis has to be employed to support these aspects.

Table 55. CS8-Multi-concern Assurance: US3-Multi-concern specification, analysis, assurance

Realisation Scenario	Multi-concern specification, analysis, assurance
Scope	Co-engineering and co-analysis for multiple quality attributes
Tool Settings	SysML/Papyrus
Participants	COM, SPS
Activities realised	• Co-engineering: System model includes safety information (safety concept, safety requirements), we also begun to add security.

	• Beginning of multi-concern analysis in the concept phase (TARA/HARA) but most of multi-concern analysis remains to be added.
Usage Decisions	Keep as much information as possible in the system model, but e.g. some parts of the item definition is still in separate Word document.
Expected Results	System specification including safety and security concerns.
Conclusions	AMASS platform features have not yet been used for this scenario but will come into play in the last iteration. No suitable tool for our chosen concept-phase co-analysis method.

3.8.5. Coverage of AMASS Prototype P1 Architecture

Table 56 illustrates the implemented functionalities during this second iteration within the Case Study 8.

Table 56. AMASS Prototype P1 Coverage by CS8

STO	AMASS Functionality Groups	Tools
Architecture-Driven Assurance	System Component Specification	Papyrus/SysML
	System Architecture Modelling for Assurance	Papyrus/SysML
	Architectural Patterns for Assurance	-
	Contract-based Design for Assurance	-
	Activities supporting Assurance Case	-
Multi-Concern Assurance	Assurance Case Specification	OpenCert
	Dependability Assurance	OpenCert
	System Dependability Co-Analysis/Co-Assessment	OpenCert
	Contract-Based Multi-concern Assurance	-
Seamless Interoperability	Evidence Management	OpenCert
	Tool Integration Management	-
	Collaborative Work Management	-
	Tool Quality Assessment and Characterization	-
Cross Intra-Domain Reuse	Compliance Management	EPF-C. Import to OpenCert
	Reuse Assistant	-
	Process-related reuse via management of variability at process level	-
	Process-related reuse via management of variability at product level	-
	Automatic generation of process-based arguments	-
	Automatic generation of product-based arguments	-

3.8.6. Conclusions

At this stage, the main benefits and potential improvements are identified in Table 57.

Table 57. CS8 Benefits and potential improvements

Artefact	Achievement/Benefits	Improvements/Recommendations
Reference framework (standards)	• Standards can be modelled; user interface is straight-forward and works well.	• For multi-concern assurance, could consider adding a dependence relationship between standards in

Artefact	Achievement/Benefits	Improvements/Recommendations
modelling and compliance mapping in OpenCert	• Tailoring when importing to assurance project allows for e.g. safety/security-element-out-of-context.	addition to the existing equivalence relationship. • Compliance mapping is tedious, the GUI could probably be improved to make this more efficient.
Argument modelling in OpenCert	• Arguments can be built with GSN notation. • Possibility to use modules and contracts (agreements) greatly improves flexibility / management of complexity. • Addition of notation for multi-concern arguments is useful.	• Some GUI improvements could be made: o Would like to be able to use any diagram as a module instead of using templates view only. o Would like to be able to color-code the boxes, e.g. for difference concerns, to improve readability.
Evidence modelling in OpenCert	• Evidence modelling connects artefacts with the assurance case.	• Unclear how to handle versions especially for artefacts which are continuously updated (such as test results from continuous integration systems) since versions are manually managed in the evidence editor.
Process modelling in OpenCert	• Modelling of the (executed) work process.	• Bugs in import from EPF
Process modelling in EPF	• Modelling of process (plans).	
OpenCert (overall)	• Ties together all information for an assurance case.	• Overall visualization of an assurance case could be improved. For instance, some sort of dashboard or graphical view showing the interconnection of the models used in the assurance case and highlighting errors/incomplete parts. • Re-use of information could be improved. For instance, the evidence and process models could be partly populated with information from the tailored reference model or argument model. • Versioning of models/diagrams would be necessary to comply with configuration management requirements in some standards. • Flexibility to import/export assurance cases or copy/paste diagrams within an assurance case would make the tool more flexible to work with.

3.9. Case Study 9: Air Traffic Management domain: Safety-Critical SW Lifecycle of a Monitoring System for NavAid

3.9.1. Case Study Specification

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

a. 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).
b. 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).

The Assurance Level for the sub-system shall be AL ≥ 4, out of a scale from 1 (for software that could cause or contribute to the failure of the ground-based system resulting in a catastrophic failure condition) to 6 (for software that could cause or contribute to the failure of the ground-based system resulting in no effect on the system).

For a detailed description on the case study see the Deliverable D1.4 "AMASS-demonstrators(a)" [4].

3.9.2. US1: System/Software Design and Safety Analysis (SWD)

During this phase (SWD) of the SW development process:

- the tools of AMASS P1 will be used to verify and validate the SW module interaction (through contracts and formal methods), in order to help the qualification/certification process at the architecture level. Such activity shall cover:
  - the 'System Design' functionality;
  - the 'System Component Specification' functionality group belonging to the AMASS Platform Basic Building Blocks;
- the tools of AMASS P1 (or P2) will be used to conduct safety analyses (FMEA and/or FTA), taking as input the architectural design based on contracts and formal methods (as mentioned above) and evaluating the contracts. Such activity shall cover:
  - the 'Safety Analysis' functionality;
  - again, the 'System Component Specification' functionality group belonging to the AMASS Platform Basic Building Blocks.

The architecture description (in terms of SW modules interaction, contracts etc.) has been modelled by means of CHESS in order to verify and validate the consistency of the architecture.

The same description of the architecture will be used, through the version of CHESS associated to AMASS P1 or AMASS P2, to conduct the safety analysis (FMEA and/or FTA).

The following AMASS functionalities (all belonging to STO1 Architecture-Driven Assurance) will be validated/verified through the Usage Scenario 1:

- System Design → System Definition: through the definition of the system architecture and of the SW modules interactions (contracts etc.);
- System Design → Functional Early Verification: through the functional verification of the system architecture and of the SW modules interaction;
- System Design → Functional Refinement: through the architecture refinement following the previous step.
- Safety Analysis → Simulation-based Fault Injection + Model-Based Safety Analysis + Contract-Based Safety Analysis: through the conduction of a FMEA and/or of a FTA.

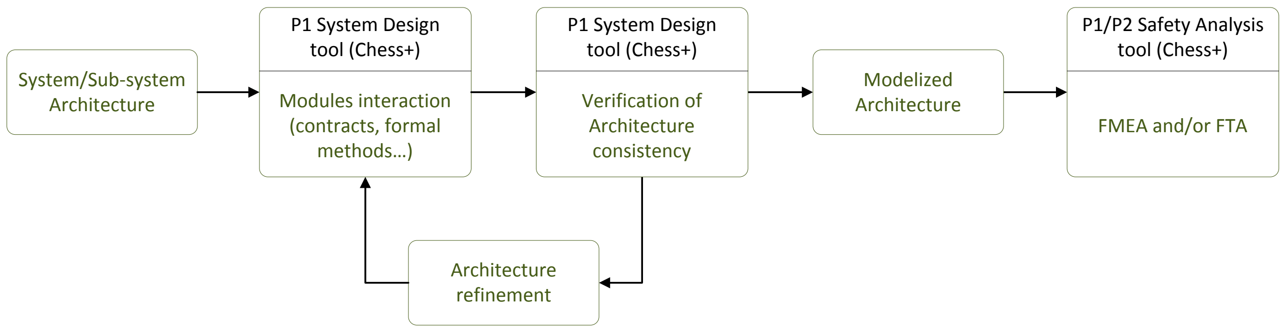

System/Sub-system Architecture
P1 System Design tool (Chess+)
Modules interaction (contracts, formal methods…)
P1 System Design tool (Chess+)
Verification of Architecture consistency
Modelized Architecture
P1/P2 Safety Analysis tool (Chess+)
FMEA and/or FTA
Architecture refinement

Figure 81. Architecture-driven assurance

3.9.2.1. STO1 Architecture-Driven Assurance

Within the functionality group "System Component Specification", two projects were defined:

- an Assurance Project, consisting in defining the System/Software Architecture, including module interactions, and in verifying the architecture consistency;
- a Dependability Assessment, consisting in typical safety analysis (FMEA and/or FTA).

Table 58. CS9-Architecture-Driven Assurance: US1-Assurance Project Creation

Realisation Scenario	Assurance Project Creation
Scope	Verification and validation of the SW modules interaction (through contracts and formal methods), in order to help the qualification/certification process at the architecture level.
Tool Settings	CHESS, Papyrus and OCRA
Participants	• Tool Provider: FBK, TEC • Tool User and Data Analysis: THI
Activities	1. Create CHESS project 2. Architecture modelling (modules interaction, through contracts and formal methods) 3. Verification of the modules interaction (architecture consistency) 4. Contracts/architecture refinement
Usage Decisions	
Expected Results	Successful architecture definition and V&V
Conclusions	

Table 59. CS9-Architecture-Driven Assurance: US1-Dependability Assessment

Realisation Scenario	Dependability Assessment
Scope	Conduction of a typical safety analysis (FMEA and/or FTA).
Tool Settings	Papyrus, Ocra and xSAP
Participants	• Tool Provider: FBK, TEC • Tool User and Data Analysis: THI
Activities	1. Create CHESS project 2. Architecture modelling (modules interaction, through contracts and formal methods) 3. FMEA (simulation-based fault injection) and/or FTA
Usage Decisions	
Expected Results	Improve design validation and safety, reducing the effort.
Conclusions	

3.9.3. US2: Safety Case (SWV)

During this phase (SWV) of the SW development process, the basic blocks “Compliance Management” and “Evidence Management” of 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.

The related activities consisted of the modelisation of the objectives of the software standard (for CNS/ATM systems) ED-109, through basic AMASS tools (OpenCert). As a result, a set of evidences was collected, which shall be mapped to the actual artefacts collected at the end of the SW development, to check the fulfilment of all the objectives.

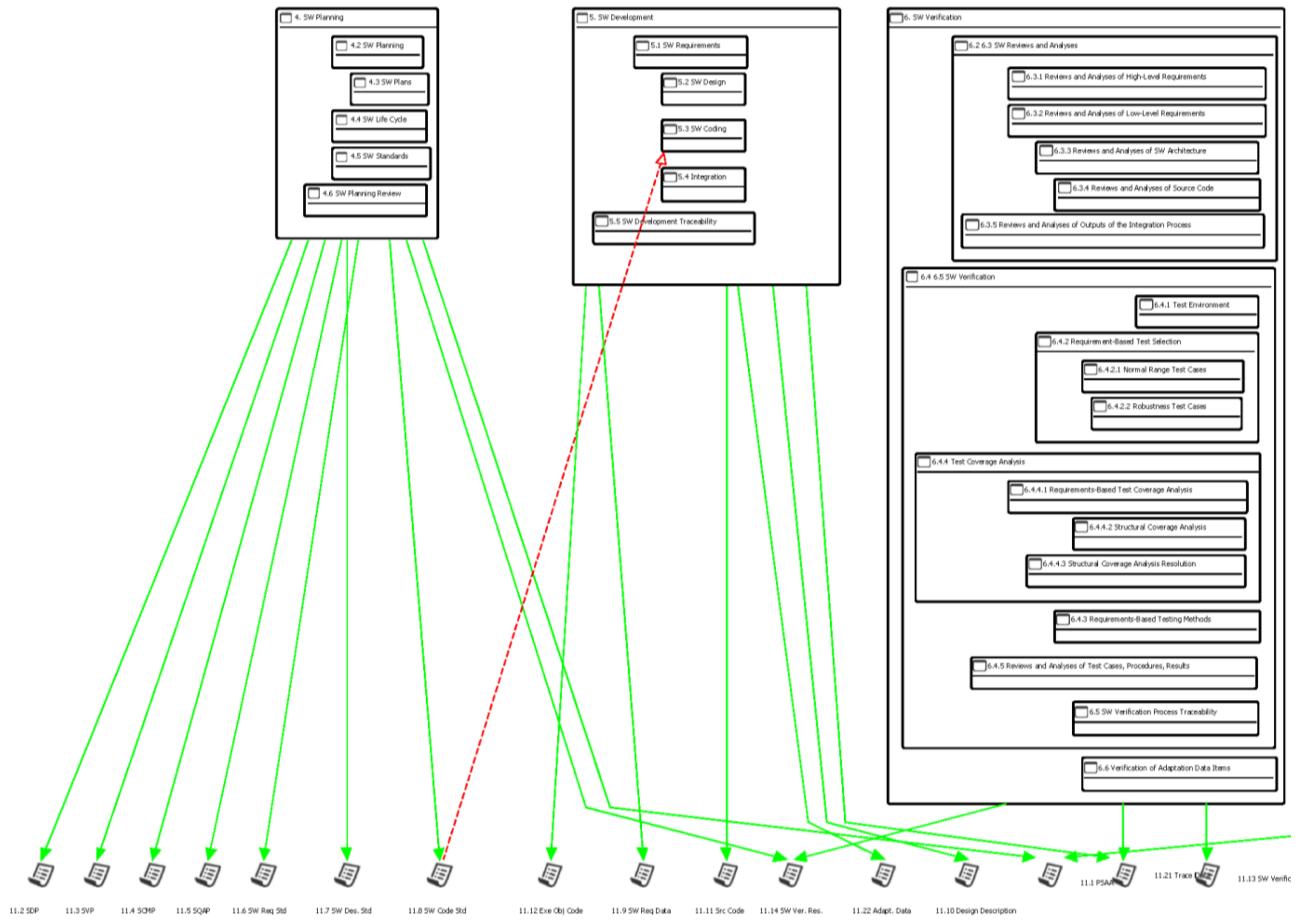

4. SW Planning
4.2 SW Planning
4.3 SW Plans
4.4 SW Life Cycle
4.5 SW Standards
4.6 SW Planning Review
5. SW Development
5.1 SW Requirements
5.2 SW Design
5.3 SW Coding
5.4 Integration
5.5 SW Development Traceability
6. SW Verification
6.2 6.3 SW Reviews and Analyses
6.3.1 Reviews and Analyses of High-Level Requirements
6.3.2 Reviews and Analyses of Low-Level Requirements
6.3.3 Reviews and Analyses of SW Architecture
6.3.4 Reviews and Analyses of Source Code
6.3.5 Reviews and Analyses of Outputs of the Integration Process
6.4 6.5 SW Verification
6.4.1 Test Environment
6.4.2 Requirement-Based Test Selection
6.4.2.1 Normal Range Test Cases
6.4.2.2 Robustness Test Cases
6.4.4 Test Coverage Analysis
6.4.4.1 Requirements-Based Test Coverage Analysis
6.4.4.2 Structural Coverage Analysis
6.4.4.3 Structural Coverage Analysis Resolution
6.4.3 Requirements-Based Testing Methods
6.4.5 Reviews and Analyses of Test Cases, Procedures, Results
6.5 SW Verification Process Traceability
6.6 Verification of Adaptation Data Items
11.2 SDP
11.3 SVP
11.4 SCMP
11.5 SQAP
11.6 SW Req Std
11.7 SW Des. Std
11.8 SW Code Std
11.12 Exe Obj Code
11.9 SW Req Data
11.11 Src Code
11.14 SW Ver. Res.
11.22 Adapt. Data
11.10 Design Description
11.1 PSAA
11.21 Trace Data
11.13 SW Verific

Figure 82. ED 109 Model

The so-called “Automatic generation of process-based arguments” AMASS functionality (belonging to STO4) will be validated/verified through the Usage Scenario 2:

- Safety Case → Evidence Generation; a two-step process will be implemented:
    - modelisation of the ED-109 objectives (Standard Modelling and Compliance Management)
    - comparison, at the end of the SW developments, of the generated evidences vs. the ED-109 requirements (Compliance Mapping)

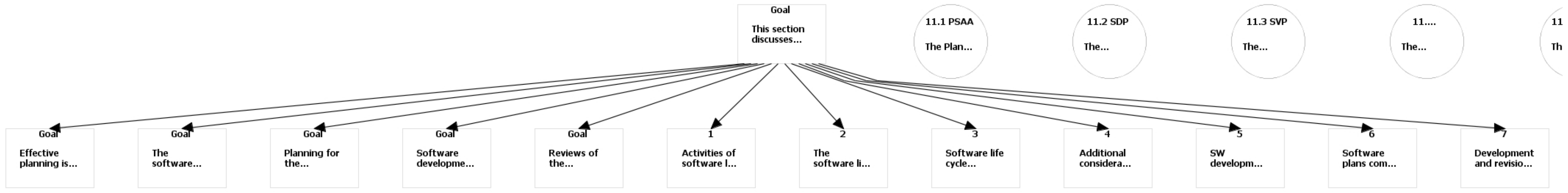


Figure 83. Compliance Management

3.9.3.1. STO4 Cross-Intra Domain Reuse

Table 60. CS9-Cross-Intra Domain Reuse: US2-Compliance Management

Realisation Scenario	Compliance Management
Scope	Modelization of the ED-109 objectives and association, to each objective, of the preliminary conditions necessary to the fulfilment of the objective.
Tool Settings	OpenCert
Participants	• Tool Provider: TEC • Tool User and Data Analysis: THI
Activities	1. Create OpenCert project 2. ED-109 objectives modelling 3. Collection of the required evidences
Usage Decisions	
Expected Results	Simplification of the process aimed at certifying the alignment with the applicable standards.
Conclusions	

Table 61. CS9-Cross-Intra Domain Reuse : US2-Evidence Generation and Evidence Management

Realisation Scenario	Evidence Generation and Evidence Management
Scope	Comparison, at the end of the SW developments, of the generated evidences (artefacts) with the evidences required by the ED-109
Tool Settings	OpenCert
Participants	• Tool Provider: TEC • Tool User and Data Analysis: THI
Activities	1. Collection, during the SW development, of the required artefacts, through the AMASS tool 2. Data comparison, analysis and validation
Usage Decisions	
Expected Results	Simplification of the process aimed at certifying the alignment with the applicable standards.
Conclusions	

3.9.4. Coverage of Prototype P1 AMASS Architecture

Table 62 illustrates the implemented functionalities during this second iteration within the Case Study 9.

Table 62. AMASS Prototype P1 Coverage by CS9

STO	AMASS Functionality Groups	Tools
Architecture-Driven Assurance	System Component Specification	CHESS
	System Architecture Modelling for Assurance	CHESS
	Architectural Patterns for Assurance	-
	Contract-based Design for Assurance	CHESS+OCRA
	Activities supporting Assurance Case	Papyrus, OCRA and xSAP
Multi-Concern	Assurance Case Specification	-

Assurance	Dependability Assurance	-
	System Dependability Co-Analysis/Co-Assessment	-
	Contract-Based Multi-concern Assurance	-
Seamless Interoperability	Evidence Management	OpenCert
	Tool Integration Management	-
	Collaborative Work Management	-
	Tool Quality Assessment and Characterization	-
Cross Intra-Domain Reuse	Compliance Management	OpenCert
	Reuse Assistant	-
	Process-related reuse via management of variability at process level	-
	Process-related reuse via management of variability at product level	-
	Automatic generation of process-based arguments	-
	Automatic generation of product-based arguments	-

3.9.5. Conclusions

At this stage, the main benefits and potential improvements are identified in Table 63.

Table 63. Benefits and potential improvements for CS9

Artefact	Achievement/Benefits	Improvements/Recommendations
Standards Models Creation (OpenCert)	• Modelling of ED-109 EUROCAE standard	• Performance
Assurance Project Creation (OpenCert)	• Selection of the ED 109 objectives based on the "Assurance level".	• Wizard for project creation • Usability/User Interface • Performance • Prevent meaningless selections during project creation
Evidence modelling in OpenCert	• Evidence modelling connects artefacts with the assurance case	• Unclear how to handle versions especially for artefacts which are continuously updated
Design of system/software architecture	• All the information defined in a single model	• Sometimes some options are missing and random clicking is needed to make it reappear again. • Similar analyses in different menus • Tool maturity is very low
Dependability assessment	• Safety analysis integrated with all the other development phases	

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

3.10.1. Case Study Specification

The objective of this Case Study is to validate the different architectures and related tools developed in AMASS. For this second iteration, feedback will be provided. For the third 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 US1 and US2 follow the same Data Collection structure described in [2]. 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, meanwhile the modified ones will be the only ones to be restudied to check its compatibility, not affected the firsts 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 D1.1 “Case studies description and business impact” [1].

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

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 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.

The System that has been modelled using the AMASS solutions is a basic SW module of the BSW. The BSW is the basic layer implemented for MPSoC architectures, or FPGA-based. BSW is responsible for managing the startup and initial HW checks, and starting the load of the ASW (Application Software). It runs a self-test for verifying main and cache memories, timers and interrupts, generating and sending a test report by using telemetry mechanism. Moreover, the BSW process tele-commands that have been sent to command the satellite payload. The architectures mentioned above for US1 and US2 make use both of the BSW.

3.10.2. US1 & US2: BSW modelling for SSDP & reconfigurable FPGA architectures

3.10.2.1. STO1 Architecture-Driven Assurance

BSW module has been modelled as a CHESS project to cover Architecture-Driven Assurance (STO1).
This task includes the construction of basic building blocks of the BSW. These blocks are named the BSW Host, the PUS and the TMTC. The functions accomplished by them are described below.

BSW Host. It is the basic building block of the BSW is the Host SW component. The BSW Host:

- Executes TC (Tele-command) packets
- Verifies TC execution
- Configures TMTC interface
- Satisfies the PUS format.

PUS. Performs the following functions:

- Parsing and validation (acknowledge) of TC packets.
- Build TM (Telemetries) packets and redirect them to the TMTC block.
- Receives TC packets from TMTC for parsing.

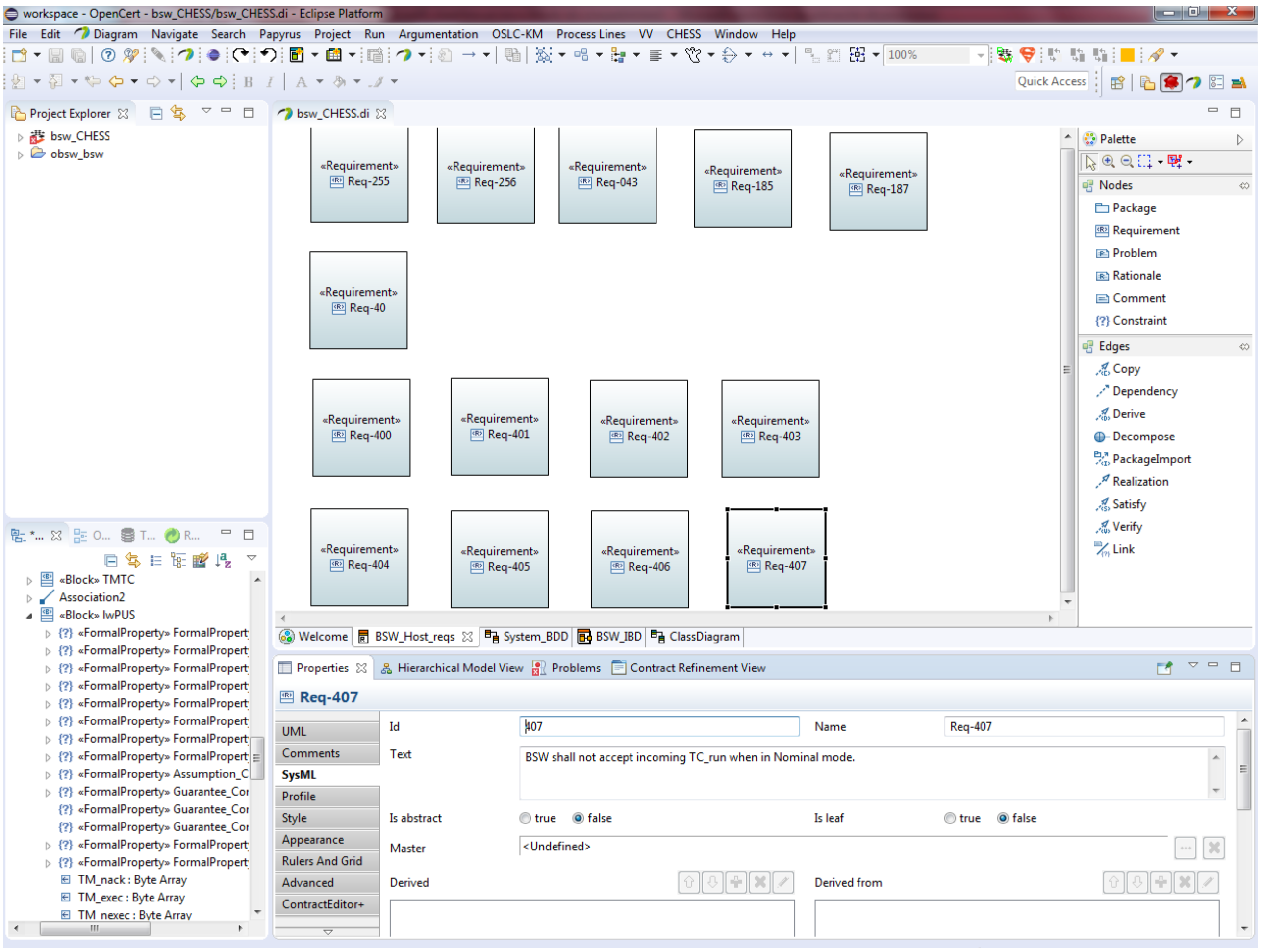
TMTC. Responsible for Tele-command and Tele-metries sending and receiving:

- Sends TCs to the PUS block for acceptance.
- Sends TM packets to external modulator devices.

3.10.2.1.1. System Design

System Definition

First step has been to add relevant BSW requirements in a SysML Requirements diagram, adding a requirement name, identification and text description.



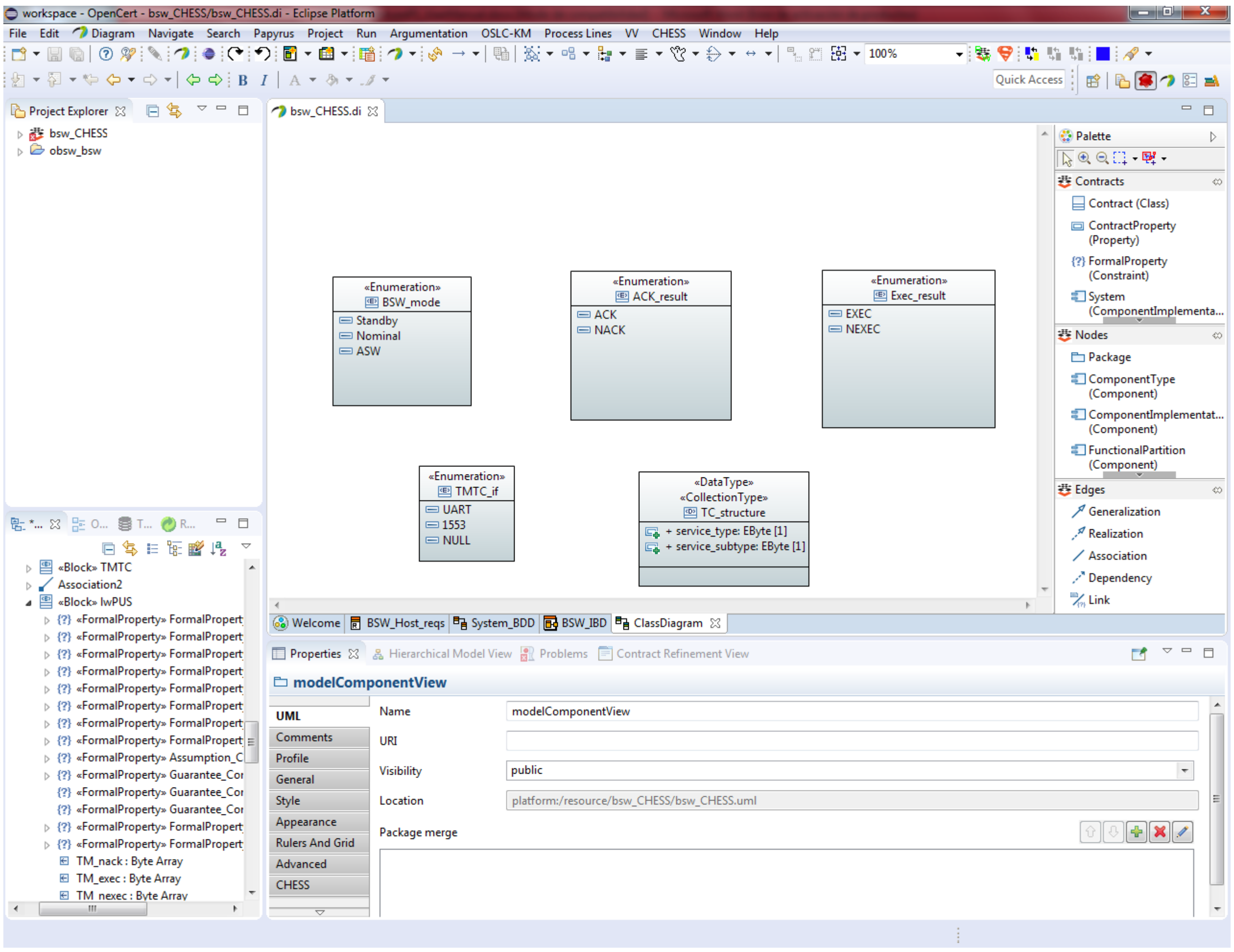
Req-407

Id: 407
Name: Req-407
Text: BSW shall not accept incoming TC_run when in Nominal mode.
Is abstract: true [✓ false]
Is leaf: true [✓ false]
Master: <Undefined>
Derived:
Derived from:

Figure 84. CS10 Requirements diagram

Secondly, a Class diagram has been created, in order to define data structures and enumerations to be later referenced in the Block Definition Diagram. These data correspond to:

- BSW modes
- TC packet acknowledge and execution result
- TMTC interface possible configurations


workspace - OpenCert - bsw_CHESS/bsw_CHESS.di - Eclipse Platform
File Edit Diagram Navigate Search Papyrus Project Run Argumentation OSLC-KM Process Lines VV CHESS Window Help
Project Explorer
bsw_CHESS
obsw_bsw
bsw_CHESS.di
«Enumeration» BSW_mode
Standby
Nominal
ASW
«Enumeration» ACK_result
ACK
NACK
«Enumeration» Exec_result
EXEC
NEXEC
«Enumeration» TMTC_if
UART
1553
NULL
«DataType» «CollectionType» TC_structure
+ service_type: EByte [1]
+ service_subtype: EByte [1]
Palette
Contracts
Contract (Class)
ContractProperty (Property)
FormalProperty (Constraint)
System (ComponentImplementa...
Nodes
Package
ComponentType (Component)
ComponentImplementat... (Component)
FunctionalPartition (Component)
Edges
Generalization
Realization
Association
Dependency
Link
«Block» TMTC
Association2
«Block» IwPUS
«FormalProperty» FormalPropert
«FormalProperty» Guarantee_Cor
«FormalProperty» Assumption_C
TM_nack : Byte Array
TM_exec : Byte Array
TM_nexec : Byte Array
Welcome
BSW_Host_reqs
System_BDD
BSW_IBD
ClassDiagram
Properties
Hierarchical Model View
Problems
Contract Refinement View
UML
Comments
Profile
General
Style
Appearance
Rulers And Grid
Advanced
CHESS

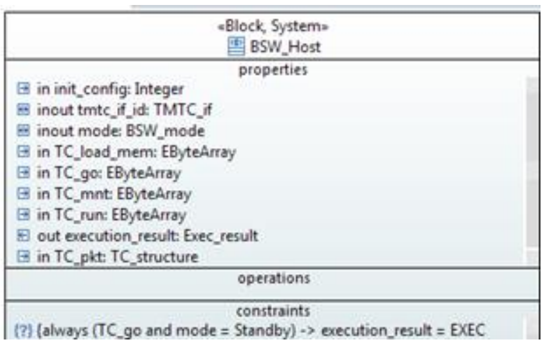
modelComponentView

Name: modelComponentView
URI:
Visibility: public
Location: platform:/resource/bsw_CHESS/bsw_CHESS.uml
Package merge

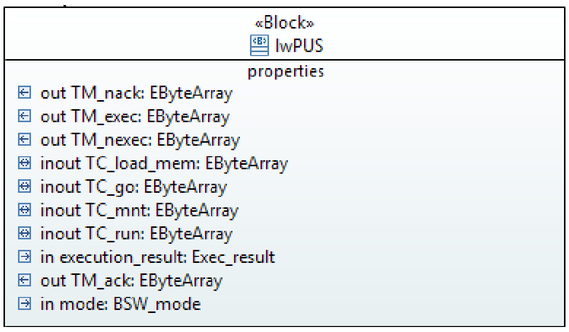
Figure 85. CS10 Class Diagram

Next, the BDD diagram includes the modelled blocks, their associated data in the form of CHESS flowports, and their association relationships for the BSW system. These hierarchical associations among system blocks are needed since otherwise the contract refinement cannot be performed afterwards.

For the BSW_Host, the declared flowports are shown below:

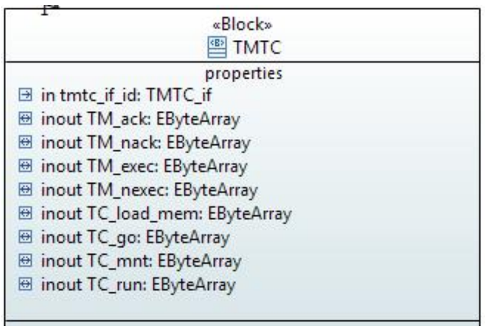


For the PUS block, flowports are associated with each type of TC or TM, the TC execution result received from the BSW_Host, and the BSW working mode:



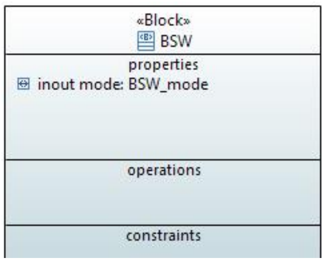
«Block»
lwPUS
properties
out TM_nack: EByteArray
out TM_exec: EByteArray
out TM_nexec: EByteArray
inout TC_load_mem: EByteArray
inout TC_go: EByteArray
inout TC_mnt: EByteArray
inout TC_run: EByteArray
in execution_result: Exec_result
out TM_ack: EByteArray
in mode: BSW_mode

The TMTC properties include the interface configuration (that will indicate the port interface to be used to transmit TM packets) and different types of TM/TC packets that can be received or transmitted:



«Block»
TMTC
properties
in tmtc_if_id: TMTC_if
inout TM_ack: EByteArray
inout TM_nack: EByteArray
inout TM_exec: EByteArray
inout TM_nexec: EByteArray
inout TC_load_mem: EByteArray
inout TC_go: EByteArray
inout TC_mnt: EByteArray
inout TC_run: EByteArray

Finally, it has been implemented a general BSW block that contains system properties as the BSW mode (Nominal or Standby):



«Block»
BSW
properties
inout mode: BSW_mode
operations
constraints

An IBD diagram has been created in the CHESS model to include the ports among system blocks. This diagram clearly reflects the flow of TCs and TMs in our modelled architecture. The TMTC part sends TCs received from external module, and the PUS successfully parsed TCs are finally forwarded to the BSW_Host block. Also, the TMTC receives TM packets generated by the PUS block, to be sent to external modulators.

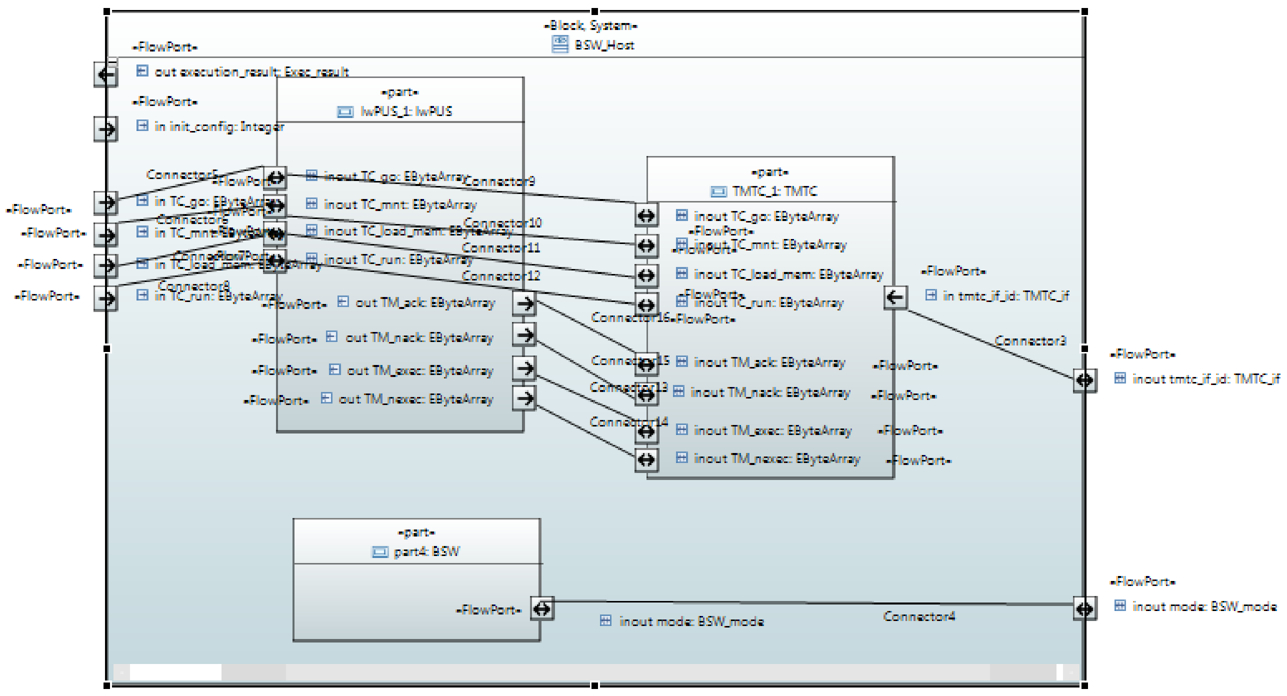
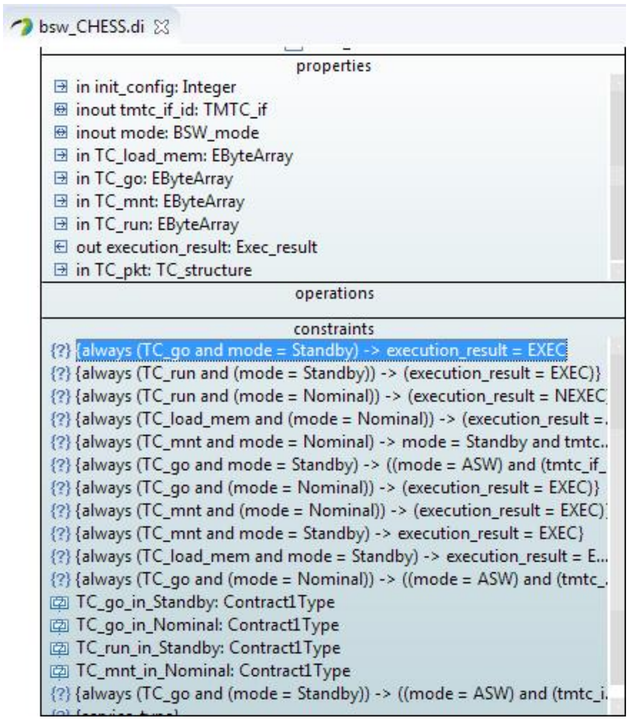


Figure 86. CS10 Interface Diagram

Requirements formalization

After system blocks definition, the relevant requirements for CS10 have been written as formal properties and contracts.

The BSW_Host formal properties reflect the behaviour of the BSW_Host component when it receives accepted TC packets. The function of the BSW_Host is 1) Deciding whether a TC packet must be executed or not. 2) Executing the command, performing the needed actions in the BSW. 3) Commanding the needed TM responses to the PUS module. Additionally, contract properties have been created in BSW_Host block.


bsw_CHESS.di

properties

- in init_config: Integer
- inout tmtc_if_id: TMTC_if
- inout mode: BSW_mode
- in TC_load_mem: EByteArray
- in TC_go: EByteArray
- in TC_mnt: EByteArray
- in TC_run: EByteArray
- out execution_result: Exec_result
- in TC_pkt: TC_structure

operations

constraints

- {?} {always (TC_go and mode = Standby) -> execution_result = EXEC
- {?} {always (TC_run and (mode = Standby)) -> (execution_result = EXEC)}
- {?} {always (TC_run and (mode = Nominal)) -> (execution_result = NEXEC
- {?} {always (TC_load_mem and (mode = Nominal)) -> (execution_result =.
- {?} {always (TC_mnt and mode = Nominal) -> mode = Standby and tmtc..
- {?} {always (TC_go and mode = Standby) -> ((mode = ASW) and (tmtc_if_
- {?} {always (TC_go and (mode = Nominal)) -> (execution_result = EXEC)}
- {?} {always (TC_mnt and (mode = Nominal)) -> (execution_result = EXEC)
- {?} {always (TC_mnt and mode = Standby) -> execution_result = EXEC}
- {?} {always (TC_load_mem and mode = Standby) -> execution_result = E...
- {?} {always (TC_go and (mode = Nominal)) -> ((mode = ASW) and (tmtc_
- TC_go_in_Standby: Contract1Type
- TC_go_in_Nominal: Contract1Type
- TC_run_in_Standby: Contract1Type
- TC_mnt_in_Nominal: Contract1Type
- {?} {always (TC_go and (mode = Standby)) -> ((mode = ASW) and (tmtc_i.

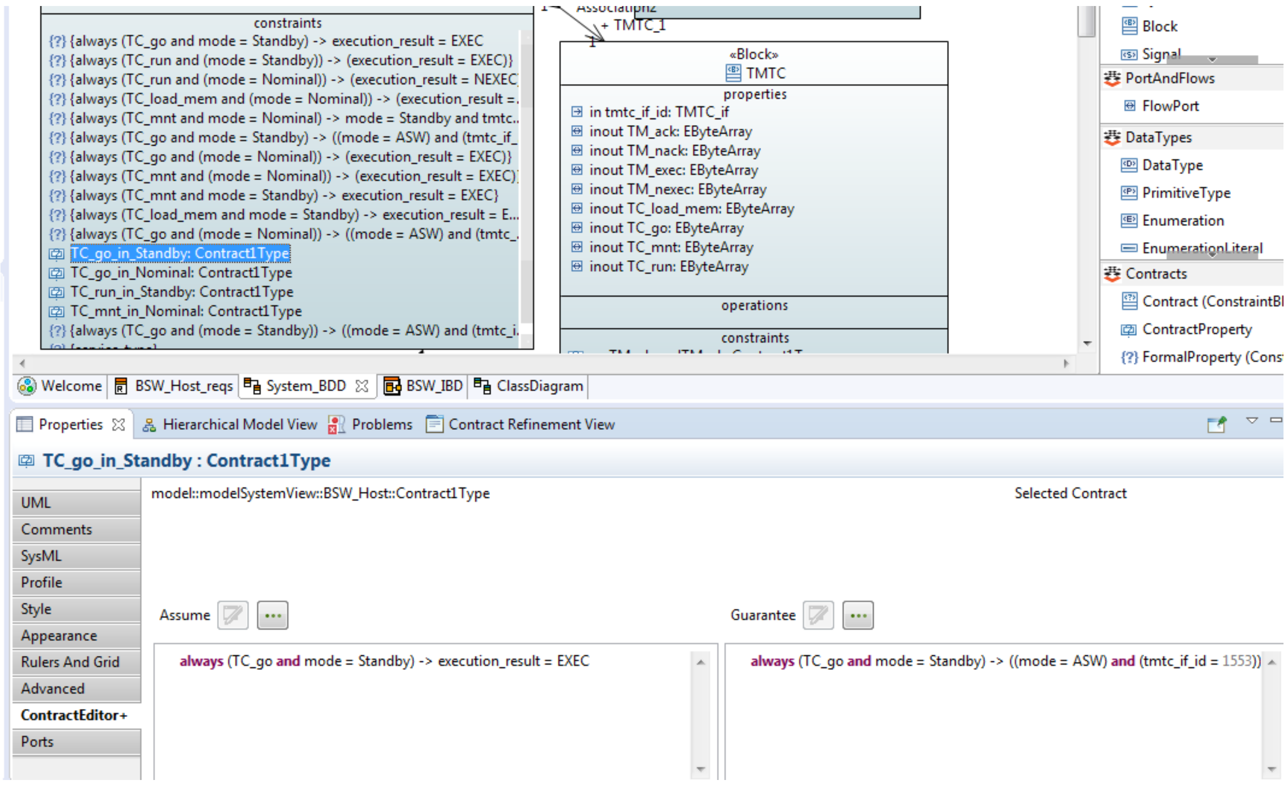
Figure 87. CS10 Formal Properties (1)

For example, following formal property:

always (TC_go **and** mode = Standby) -> execution_result = EXEC

specifies that in case the BSW is in Standby mode and telecommand TC_go is received, the BSW_Host must execute this command, because it is compatible with this BSW status.

For BSW_Host block, each contract property has been constructed using an existing formal property as assumption and another as guarantee. For example, the following contract in figure below takes as *assumption* that a telecommand (TC_go) has been correctly executed by the BSW_Host, and the *guarantee* for this contract is that the BSW_Host shall switch to ASW mode (this meands that the boot SW changes execution pointer to the memory address of the Application SW module) and configures TMTC interface to 1553 protocol (so as corresponding telemetry TM packet can be sent by the other modules):


constraints
{?} {always (TC_go and mode = Standby) -> execution_result = EXEC
{?} {always (TC_run and (mode = Standby)) -> (execution_result = EXEC)}
{?} {always (TC_run and (mode = Nominal)) -> (execution_result = NEXEC
{?} {always (TC_load_mem and (mode = Nominal)) -> (execution_result =.
{?} {always (TC_mnt and mode = Nominal) -> mode = Standby and tmtc..
{?} {always (TC_go and mode = Standby) -> ((mode = ASW) and (tmtc_if_
{?} {always (TC_go and (mode = Nominal)) -> (execution_result = EXEC)}
{?} {always (TC_mnt and (mode = Nominal)) -> (execution_result = EXEC)
{?} {always (TC_mnt and mode = Standby) -> execution_result = EXEC}
{?} {always (TC_load_mem and mode = Standby) -> execution_result = E...
{?} {always (TC_go and (mode = Nominal)) -> ((mode = ASW) and (tmtc_.
TC_go_in_Standby: Contract1Type
TC_go_in_Nominal: Contract1Type
TC_run_in_Standby: Contract1Type
TC_mnt_in_Nominal: Contract1Type
{?} {always (TC_go and (mode = Standby)) -> ((mode = ASW) and (tmtc_i.

«Block»
TMTC
properties
in tmtc_if_id: TMTC_if
inout TM_ack: EByteArray
inout TM_nack: EByteArray
inout TM_exec: EByteArray
inout TM_nexec: EByteArray
inout TC_load_mem: EByteArray
inout TC_go: EByteArray
inout TC_mnt: EByteArray
inout TC_run: EByteArray
operations
constraints

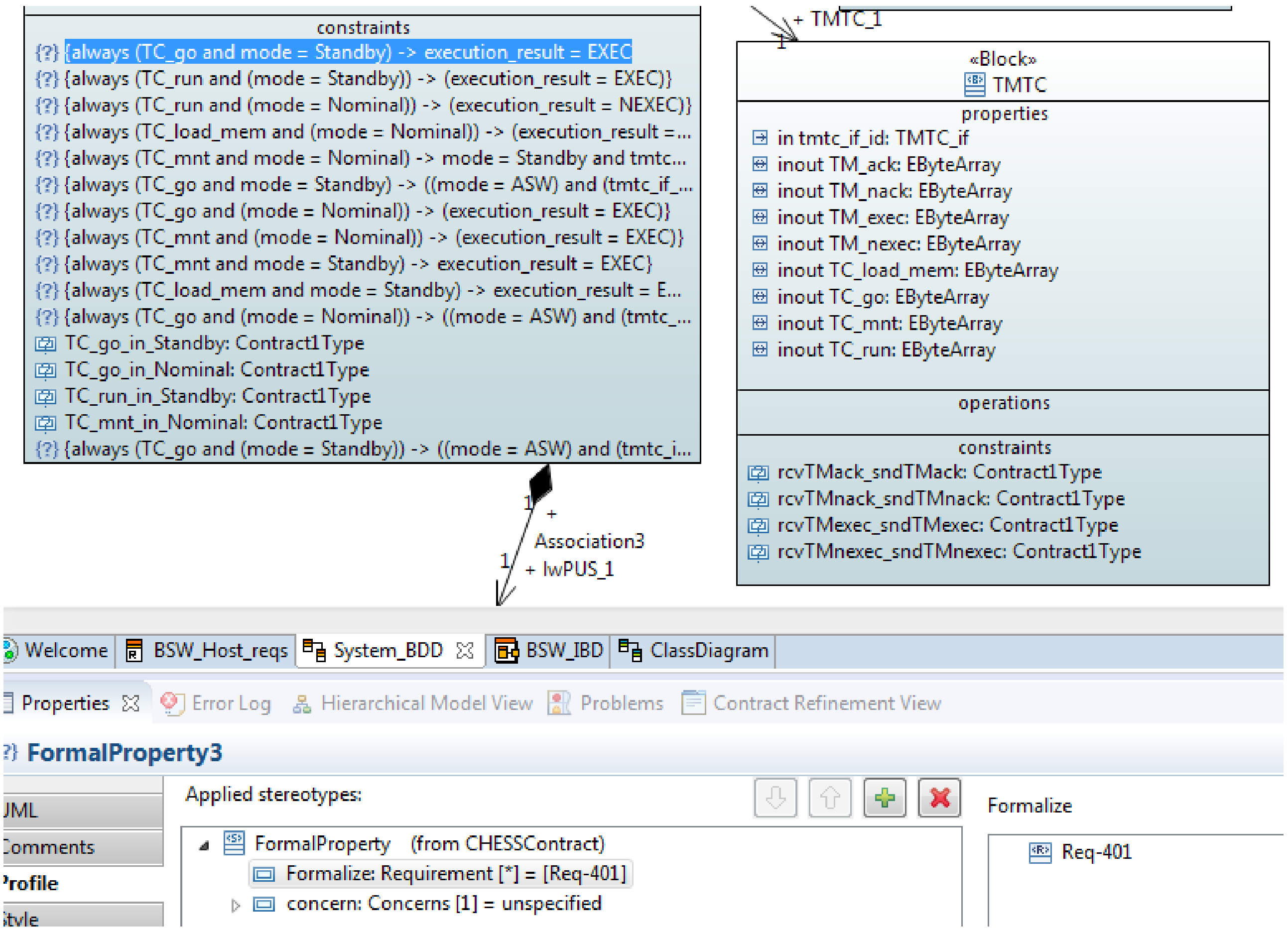
TC_go_in_Standby : Contract1Type
model::modelSystemView::BSW_Host::Contract1Type
Selected Contract

Assume: **always** (TC_go **and** mode = Standby) -> execution_result = EXEC

Guarantee: **always** (TC_go **and** mode = Standby) -> ((mode = ASW) **and** (tmtc_if_id = 1553))

Figure 88. CS10 Contract (1)

Mapping of requirements to formal properties. In this task it has been chosen, for each formal property of BSW_Host, the specific requirement that is being formalized.



constraints
{?} {always (TC_go and mode = Standby) -> execution_result = EXEC
{?} {always (TC_run and (mode = Standby)) -> (execution_result = EXEC)}
{?} {always (TC_run and (mode = Nominal)) -> (execution_result = NEXEC)}
{?} {always (TC_load_mem and (mode = Nominal)) -> (execution_result =...
{?} {always (TC_mnt and mode = Nominal) -> mode = Standby and tmtc...
{?} {always (TC_go and mode = Standby) -> ((mode = ASW) and (tmtc_if_...
{?} {always (TC_go and (mode = Nominal)) -> (execution_result = EXEC)}
{?} {always (TC_mnt and (mode = Nominal)) -> (execution_result = EXEC)}
{?} {always (TC_mnt and mode = Standby) -> execution_result = EXEC}
{?} {always (TC_load_mem and mode = Standby) -> execution_result = E...
{?} {always (TC_go and (mode = Nominal)) -> ((mode = ASW) and (tmtc_...
TC_go_in_Standby: Contract1Type
TC_go_in_Nominal: Contract1Type
TC_run_in_Standby: Contract1Type
TC_mnt_in_Nominal: Contract1Type
{?} {always (TC_go and (mode = Standby)) -> ((mode = ASW) and (tmtc_i...

+ TMTC_1

«Block»
TMTC
properties
in tmtc_if_id: TMTC_if
inout TM_ack: EByteArray
inout TM_nack: EByteArray
inout TM_exec: EByteArray
inout TM_nexec: EByteArray
inout TC_load_mem: EByteArray
inout TC_go: EByteArray
inout TC_mnt: EByteArray
inout TC_run: EByteArray
operations
constraints
rcvTMack_sndTMack: Contract1Type
rcvTMnack_sndTMnack: Contract1Type
rcvTMexec_sndTMexec: Contract1Type
rcvTMnexec_sndTMnexec: Contract1Type

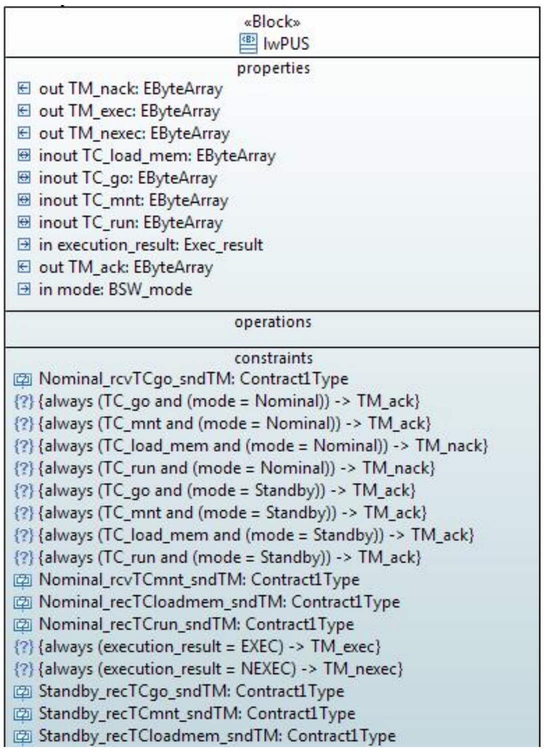
FormalProperty3

Applied stereotypes:
FormalProperty (from CHESSContract)
Formalize: Requirement [*] = [Req-401]
concern: Concerns [1] = unspecified

Formalize
Req-401

Figure 89. CS10 Requirements mapping

The PUS formal properties refer to positive or negative acknowledgement of TC commands. As specified in the system requirements TCs are accepted or not according to the BSW state where they are received. The PUS block must generate TM packets with ACK result. Therefore, for the contract properties, these formal properties are taken as *assumptions*, while the *guarantees* will be the sending of the TM packet informing about the TC execution to the TMTC block.



«Block»
lwPUS

properties

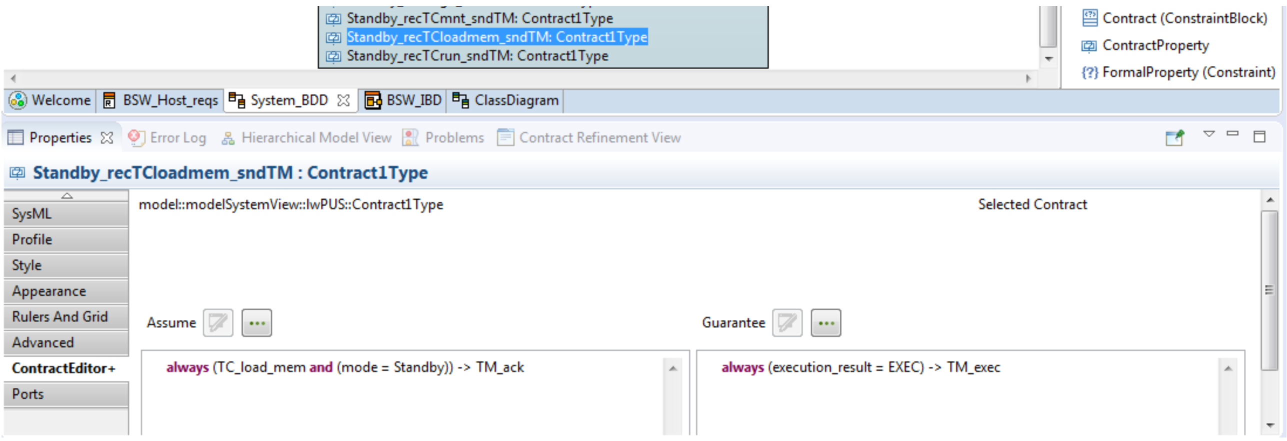
out TM_nack: EByteArray
out TM_exec: EByteArray
out TM_nexec: EByteArray
inout TC_load_mem: EByteArray
inout TC_go: EByteArray
inout TC_mnt: EByteArray
inout TC_run: EByteArray
in execution_result: Exec_result
out TM_ack: EByteArray
in mode: BSW_mode

operations

constraints

Nominal_rcvTCgo_sndTM: Contract1Type
{always (TC_go and (mode = Nominal)) -> TM_ack}
{always (TC_mnt and (mode = Nominal)) -> TM_ack}
{always (TC_load_mem and (mode = Nominal)) -> TM_nack}
{always (TC_run and (mode = Nominal)) -> TM_nack}
{always (TC_go and (mode = Standby)) -> TM_ack}
{always (TC_mnt and (mode = Standby)) -> TM_ack}
{always (TC_load_mem and (mode = Standby)) -> TM_ack}
{always (TC_run and (mode = Standby)) -> TM_ack}
Nominal_rcvTCmnt_sndTM: Contract1Type
Nominal_recTCloadmem_sndTM: Contract1Type
Nominal_recTCrun_sndTM: Contract1Type
{always (execution_result = EXEC) -> TM_exec}
{always (execution_result = NEXEC) -> TM_nexec}
Standby_recTCgo_sndTM: Contract1Type
Standby_recTCmnt_sndTM: Contract1Type
Standby_recTCloadmem_sndTM: Contract1Type

Figure 90. CS10 Formal Properties (2)



Standby_recTCmnt_sndTM: Contract1Type
Standby_recTCloadmem_sndTM: Contract1Type
Standby_recTCrun_sndTM: Contract1Type

Contract (ConstraintBlock)
ContractProperty
FormalProperty (Constraint)

Standby_recTCloadmem_sndTM : Contract1Type

model::modelSystemView::lwPUS::Contract1Type

Selected Contract

Assume: always (TC_load_mem and (mode = Standby)) -> TM_ack

Guarantee: always (execution_result = EXEC) -> TM_exec

Figure 91. CS10 Contract (2)

The TMTC constraints consist only in a set of contract properties, describing the assumptions that must be fulfilled for guaranteeing the sending of a TM packet. These assumptions are the reception of the corresponding TM acknowledge packet by the TMTC and a proper interface configuration.

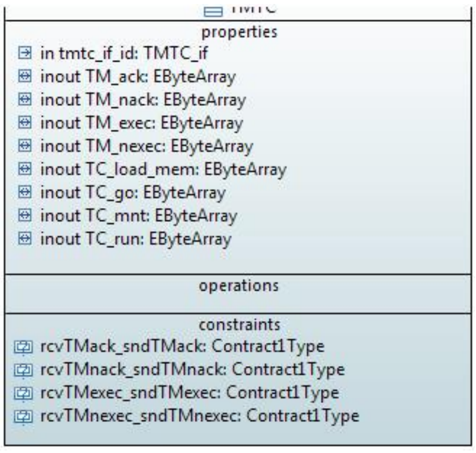
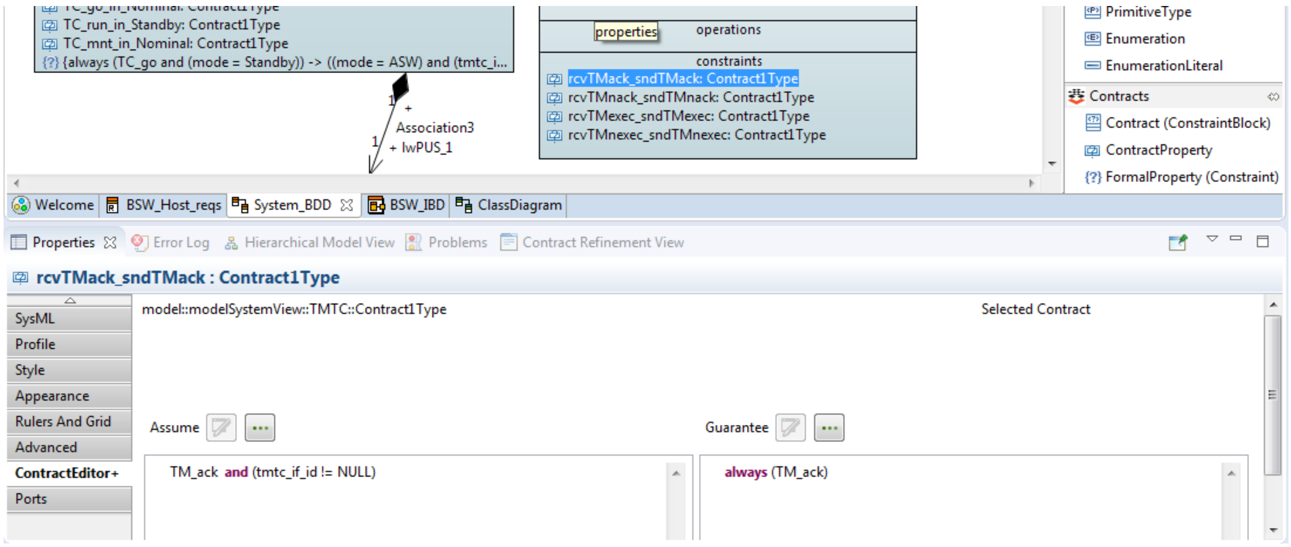

properties
in tmtc_if_id: TMTC_if
inout TM_ack: EByteArray
inout TM_nack: EByteArray
inout TM_exec: EByteArray
inout TM_nexec: EByteArray
inout TC_load_mem: EByteArray
inout TC_go: EByteArray
inout TC_mnt: EByteArray
inout TC_run: EByteArray
operations
constraints
rcvTMack_sndTMack: Contract1Type
rcvTMnack_sndTMnack: Contract1Type
rcvTMexec_sndTMexec: Contract1Type
rcvTMnexec_sndTMnexec: Contract1Type

Figure 92. CS10 Formal Properties (3)


TC_run_in_Standby: Contract1Type
TC_mnt_in_Nominal: Contract1Type
{always (TC_go and (mode = Standby)) -> ((mode = ASW) and (tmtc_i...

Association3
+ lwPUS_1

properties operations
constraints
rcvTMack_sndTMack: Contract1Type
rcvTMnack_sndTMnack: Contract1Type
rcvTMexec_sndTMexec: Contract1Type
rcvTMnexec_sndTMnexec: Contract1Type

PrimitiveType
Enumeration
EnumerationLiteral
Contracts
Contract (ConstraintBlock)
ContractProperty
FormalProperty (Constraint)

Welcome | BSW_Host_reqs | System_BDD | BSW_IBD | ClassDiagram

Properties | Error Log | Hierarchical Model View | Problems | Contract Refinement View

rcvTMack_sndTMack : Contract1Type

SysML
Profile
Style
Appearance
Rulers And Grid
Advanced
ContractEditor+
Ports

model::modelSystemView::TMTC::Contract1Type

Selected Contract

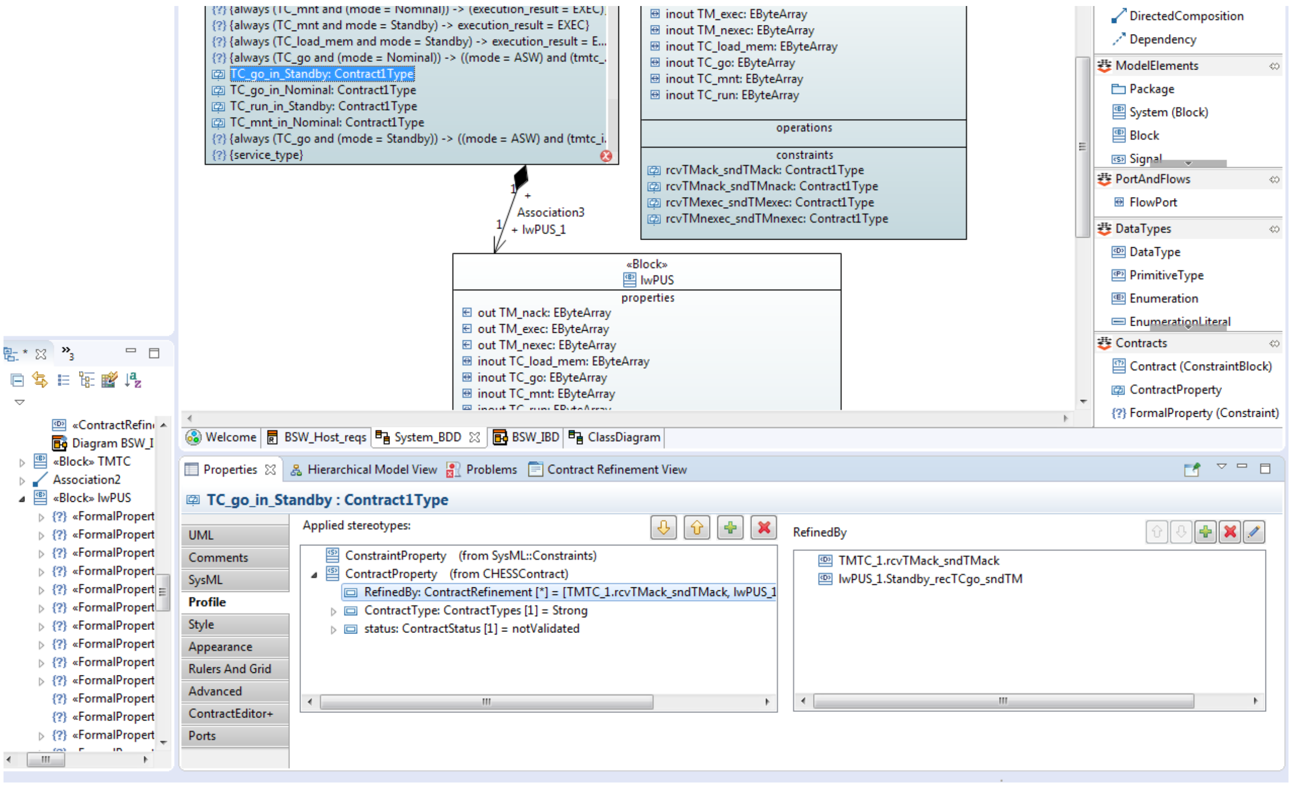
Assume: TM_ack and (tmtc_if_id != NULL)

Guarantee: always (TM_ack)

Figure 93. CS10 Contract (3)

Functional refinement

BSW_Host contracts refinement. Each contract property in BSW_Host block can be refined by other contracts defined in the dependant blocks. This is the way to assure that to fully cover a system requirement.



TC_go_in_Standby : Contract1Type

Applied stereotypes:
- ConstraintProperty (from SysML::Constraints)
- ContractProperty (from CHESSContract)
  - RefinedBy: ContractRefinement [*] = [TMTC_1.rcvTMack_sndTMack, IwPUS_1
  - ContractType: ContractTypes [1] = Strong
  - status: ContractStatus [1] = notValidated

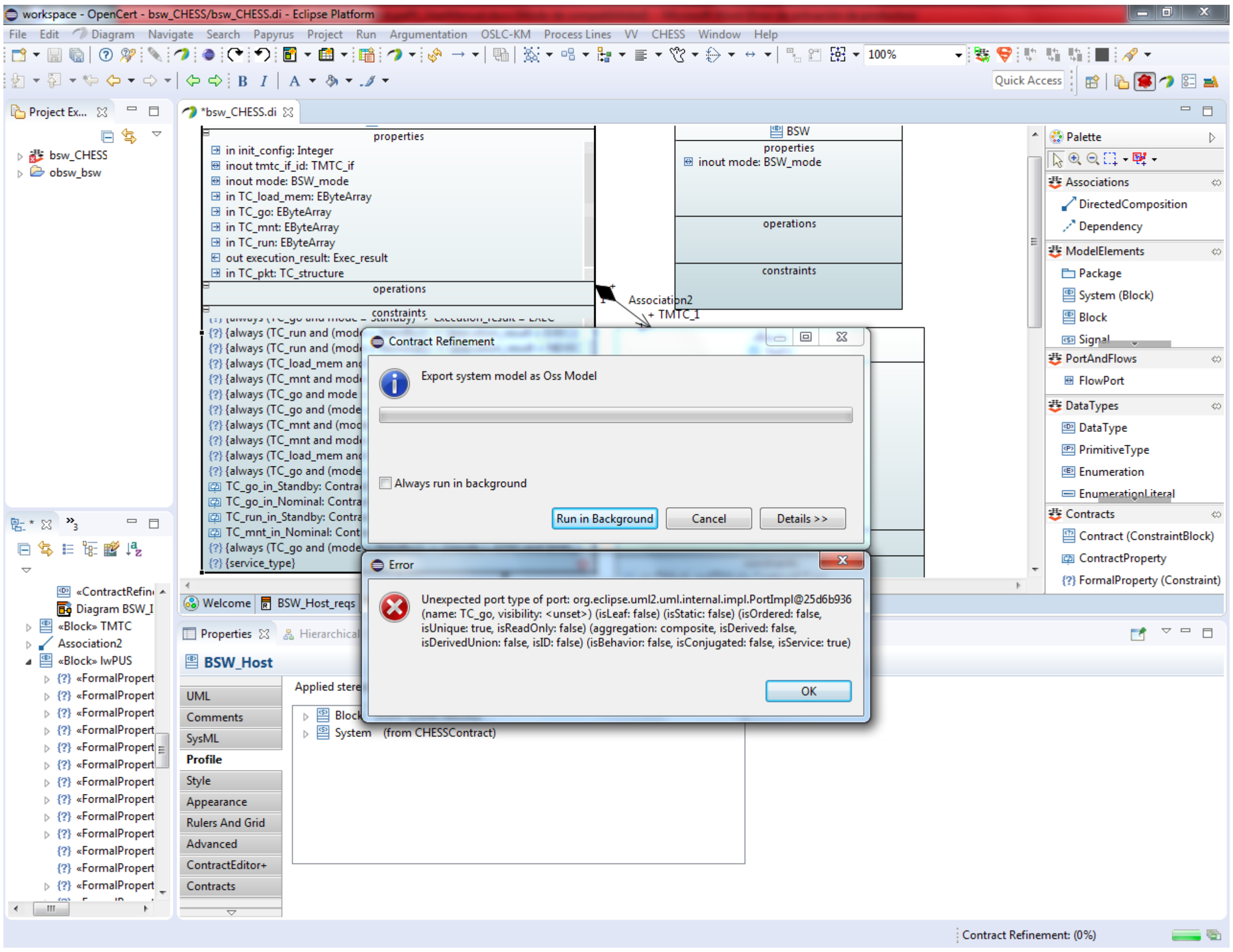
RefinedBy
- TMTC_1.rcvTMack_sndTMack
- IwPUS_1.Standby_recTCgo_sndTM

Figure 94. CS10 Contract refinement

V&V Tools

This functionality, intended to perform checks on existing formal properties, contract implementation and refinements, is not currently working with CS10 defined model, since the tool reports an invalid port type (for EByte array type).

For this issue the tool developer has been informed of our necessity and a solution will be fixed for the third iteration.


properties
in init_config: Integer
inout tmtc_if_id: TMTC_if
inout mode: BSW_mode
in TC_load_mem: EByteArray
in TC_go: EByteArray
in TC_mnt: EByteArray
in TC_run: EByteArray
out execution_result: Exec_result
in TC_pkt: TC_structure

Contract Refinement

Export system model as Oss Model

Always run in background

Error

```
Unexpected port type of port: org.eclipse.uml2.uml.internal.impl.PortImpl@25d6b936
(name: TC_go, visibility: <unset>) (isLeaf: false) (isStatic: false) (isOrdered: false,
isUnique: true, isReadOnly: false) (aggregation: composite, isDerived: false,
isDerivedUnion: false, isID: false) (isBehavior: false, isConjugated: false, isService: true)
```

workspace - OpenCert - bsw_CHESS/bsw_CHESS.di - Eclipse Platform
Run in Background
Cancel
Details >>
OK
BSW_Host
Palette
Contract Refinement: (0%)

Figure 95. V&V validation

Table 64. CS10-Architecture-Driven Assurance: US1&US2-Assurance Project Management (Create Assurance Project)

Realisation Scenario	**Assurance Project Management (Create Assurance Project)**
Scope	Creation of a CHESS project for BSW architecture. The scope for this second prototype is the modelling of main BSW functional blocks, both used by US1 and US2. Specify and map requirements.
Tool Settings	CHESS Tools
Participants	• Data Analysis: TAS • Tool User: TAS
Activities realised	1. Create CHESS project including requirements, BDD, IBD diagrams. 2. Specification of formal properties. 3. Mapping of requirements to formal properties. 4. Composition of contracts based on defined formal properties. 5. Contracts refinement.
Usage Decisions	None
Expected Results	• BSW Project structure for Architecture-Driven assurance.
Conclusions	• System architecture modelling and contract-based assurance successful. • V&V check failed (port type not yet supported by AMASS P1 prototype).

3.10.2.2. STO2 Multi-Concern Assurance

3.10.2.2.1. Define your Assurance Case architecture

The goals of software product assurance are to provide adequate confidence to the customer and to the supplier that the developed or procured/reused software satisfies its requirements throughout the system lifetime. In particular, that the software is developed to perform properly and safely in its operational environment, meeting the quality objectives agreed for the project.

In Space Domain the assurance architecture is guided by the ECSS-Q-ST 80C standard. This Standard defines a set of software product assurance requirements to be used for the development and maintenance of software for space systems. Space systems include manned and unmanned spacecraft, launchers, payloads, experiments and their associated ground equipment and facilities. Software includes the software component of firmware.

This Standard also applies to the development or reuse of non-deliverable software which affects the quality of the deliverable product or service provided by a space system, if the service is implemented by software.

ECSS-Q-ST-80 interfaces with space engineering and management, which are addressed in the Engineering (-E) and Management (-M) branches of the ECSS System, and explains how they relate to the software product assurance processes.

This standard complements ECSS-E-ST-40 “Space engineering — Software general requirements”, with product assurance aspects, integrated in the space system software engineering processes as defined in ECSS-E-ST-40. Together the two standards specify all processes for space software development.

By the use of specific tools we can model this standard. The main blocks are identified as classes in UML language. Inside the classes we can define subclasses which reach more specific tasks:

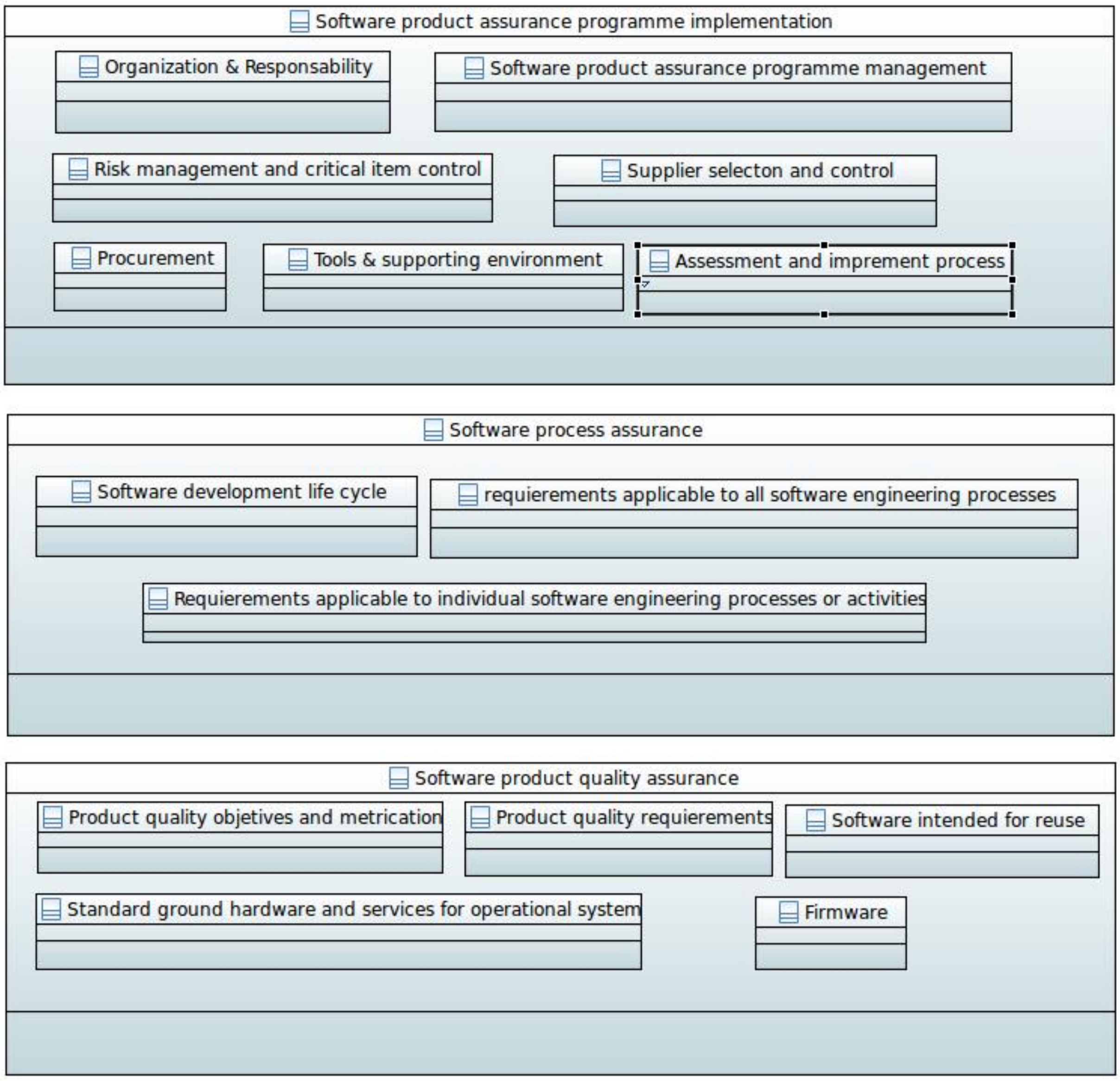

Software product assurance programme implementation
Organization & Responsability
Software product assurance programme management
Risk management and critical item control
Supplier selecton and control
Procurement
Tools & supporting environment
Assessment and imprement process
Software process assurance
Software development life cycle
requierements applicable to all software engineering processes
Requierements applicable to individual software engineering processes or activities
Software product quality assurance
Product quality objetives and metrication
Product quality requierements
Software intended for reuse
Standard ground hardware and services for operational system
Firmware

Figure 96. Composite Structure Diagram

The subclasses inside the main ones aims to achieve more specific goals. By the use of the UML tool we can identify parts of this classes named "components". The next figure shows the components of these subclasses:

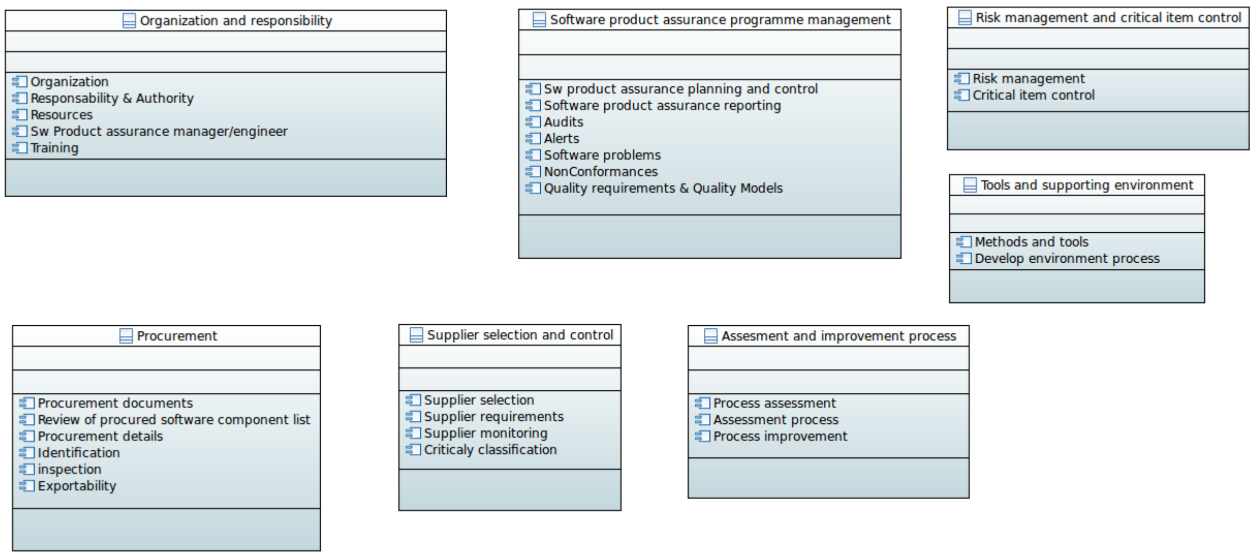

Organization and responsibility
Organization
Responsability & Authority
Resources
Sw Product assurance manager/engineer
Training
Software product assurance programme management
Sw product assurance planning and control
Software product assurance reporting
Audits
Alerts
Software problems
NonConformances
Quality requirements & Quality Models
Risk management and critical item control
Risk management
Critical item control
Tools and supporting environment
Methods and tools
Develop environment process
Procurement
Procurement documents
Review of procured software component list
Procurement details
Identification
inspection
Exportability
Supplier selection and control
Supplier selection
Supplier requirements
Supplier monitoring
Criticaly classification
Assesment and improvement process
Process assessment
Assessment process
Process improvement

Figure 97. Subclases related to Software product assurance programme implementation

Software process assurance needs also Software development life cycle, structure represented as a flow diagram or kind of state machine which the condition to jump is always “ready and accepted”.

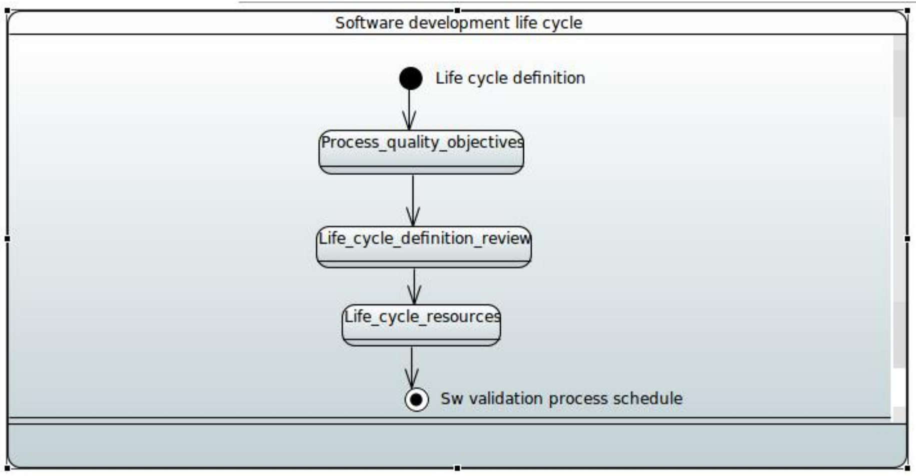

Software development life cycle
Life cycle definition
Process_quality_objectives
Life_cycle_definition_review
Life_cycle_resources
Sw validation process schedule

Figure 98. Life cycle for SW product

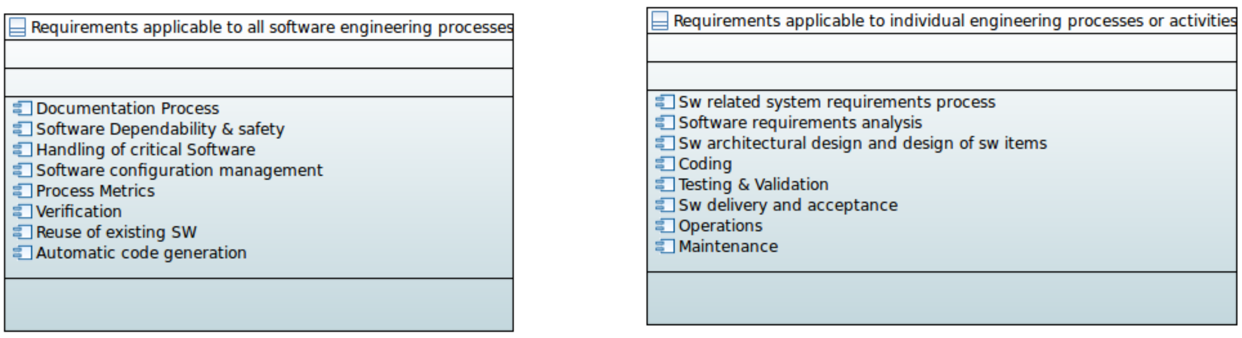

Requirements applicable to all software engineering processes
Documentation Process
Software Dependability & safety
Handling of critical Software
Software configuration management
Process Metrics
Verification
Reuse of existing SW
Automatic code generation
Requirements applicable to individual engineering processes or activities
Sw related system requirements process
Software requirements analysis
Sw architectural design and design of sw items
Coding
Testing & Validation
Sw delivery and acceptance
Operations
Maintenance

Figure 99. Subclasses and components for Software process assurance main class.

Components inside the subclasses represents the actions to be treated more deeply to fill the standard requirements. Here are attached the components from each subclass and its dependency:

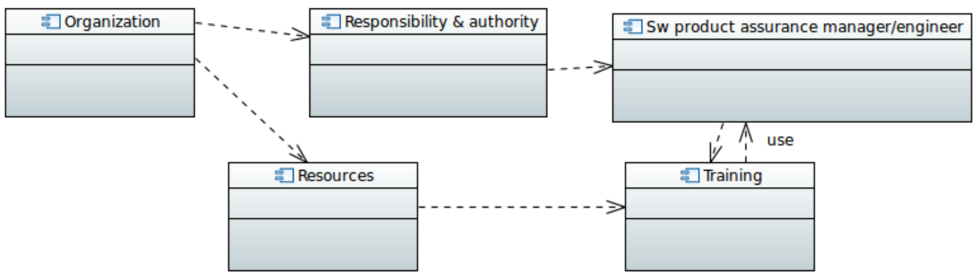

Organization
Responsibility & authority
Sw product assurance manager/engineer
Resources
Training
use

Figure 100. Components of Organization & responsibility subclass

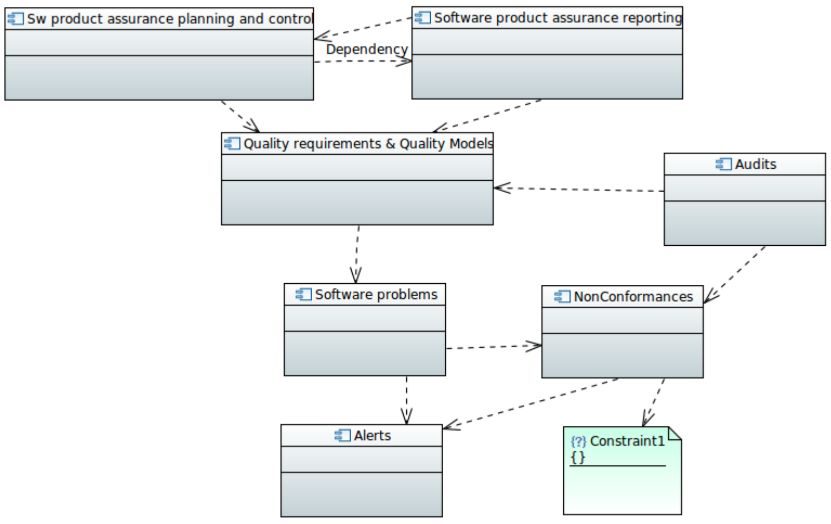

Sw product assurance planning and control
Software product assurance reporting
Dependency
Quality requirements & Quality Models
Audits
Software problems
NonConformances
Alerts
{?} Constraint1
{}

Figure 101. Components of SW product assurance programme management

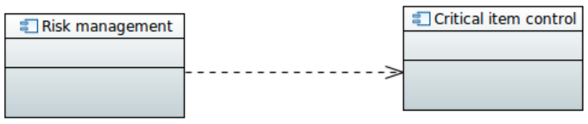


Figure 102. Components of risk management and critical item control

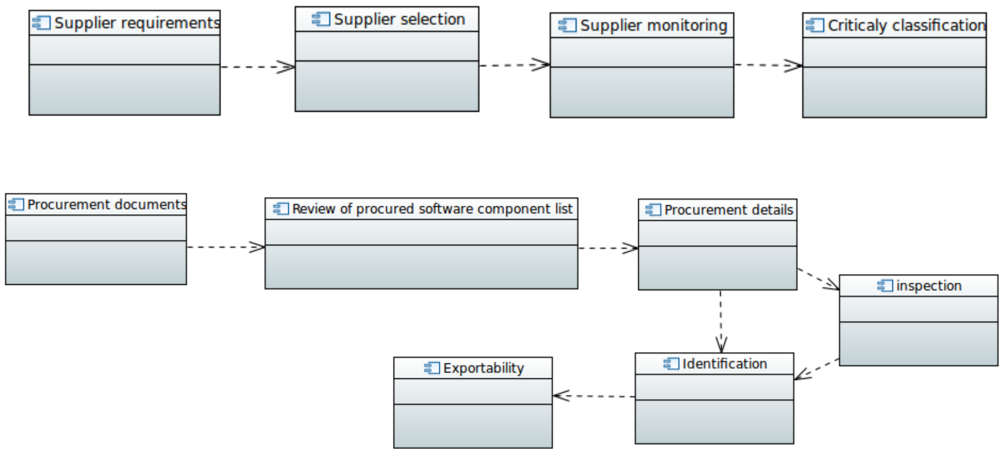


Figure 103. Components of Procurement subclass

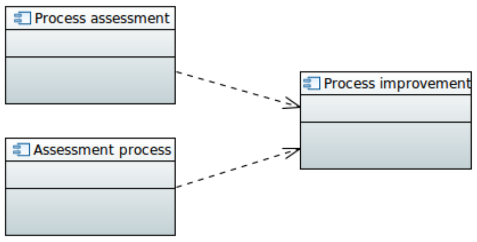


Figure 104. Components Assessment and implement process

For the Software process assurance, we can see:

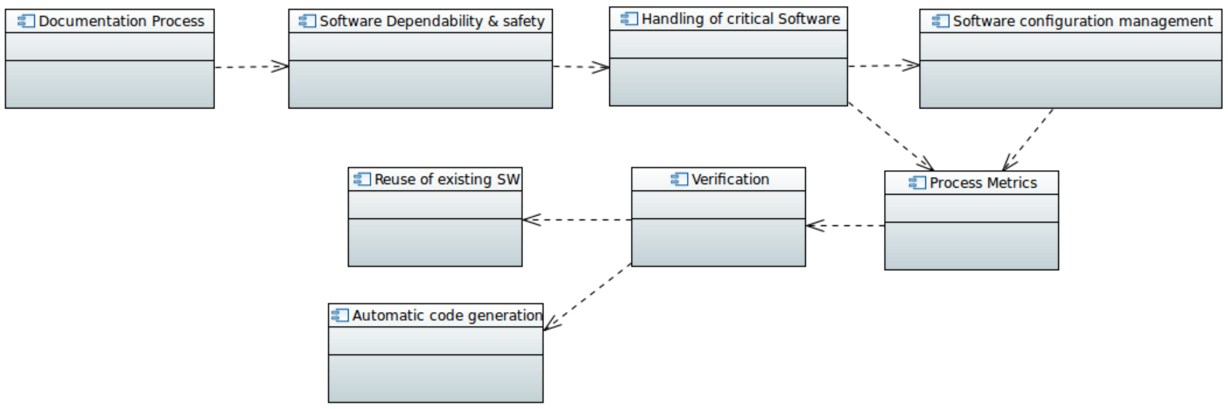

Documentation Process
Software Dependability & safety
Handling of critical Software
Software configuration management
Reuse of existing SW
Verification
Process Metrics
Automatic code generation

Figure 105. Components of Requirements applicable to all Sw engineering processes

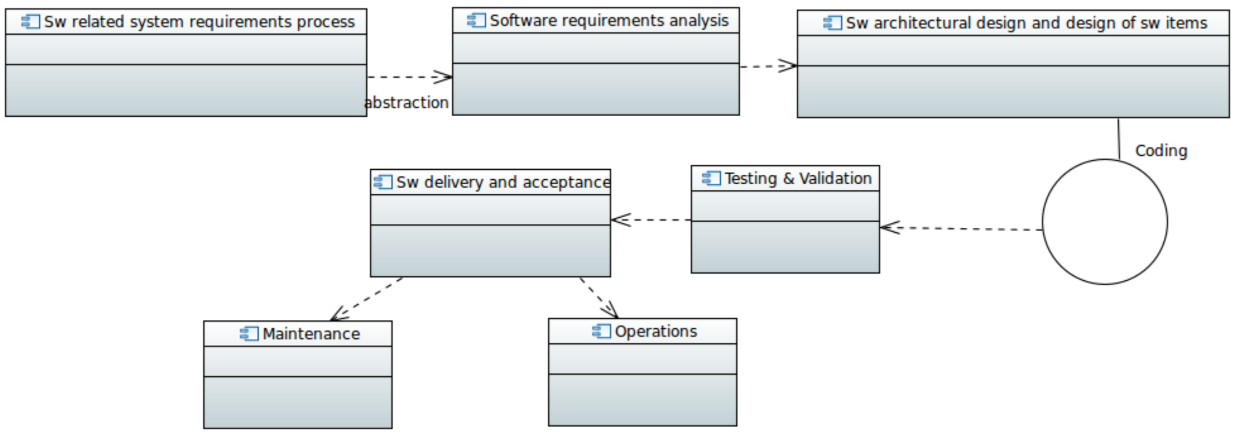

Sw related system requirements process
abstraction
Software requirements analysis
Sw architectural design and design of sw items
Coding
Sw delivery and acceptance
Testing & Validation
Maintenance
Operations

Figure 106. Components of Requirements applicable to individual engineering processes or activities.

In Space Domain every software development process must follow the ECSS-E-ST-40C, but the quality assurance process is guided by ECSS-Q-ST-80C. ECSS-Q-ST-80C refers on how the requirements described in the ECSS-E-ST-40C are taking into account and how are being implemented.

From the assurance architecture we can extrapolate when and who is the responsible of doing these activities. Once the assurance structure is clear, we pass to model with OpenCert TAS-E Usage Scenario:

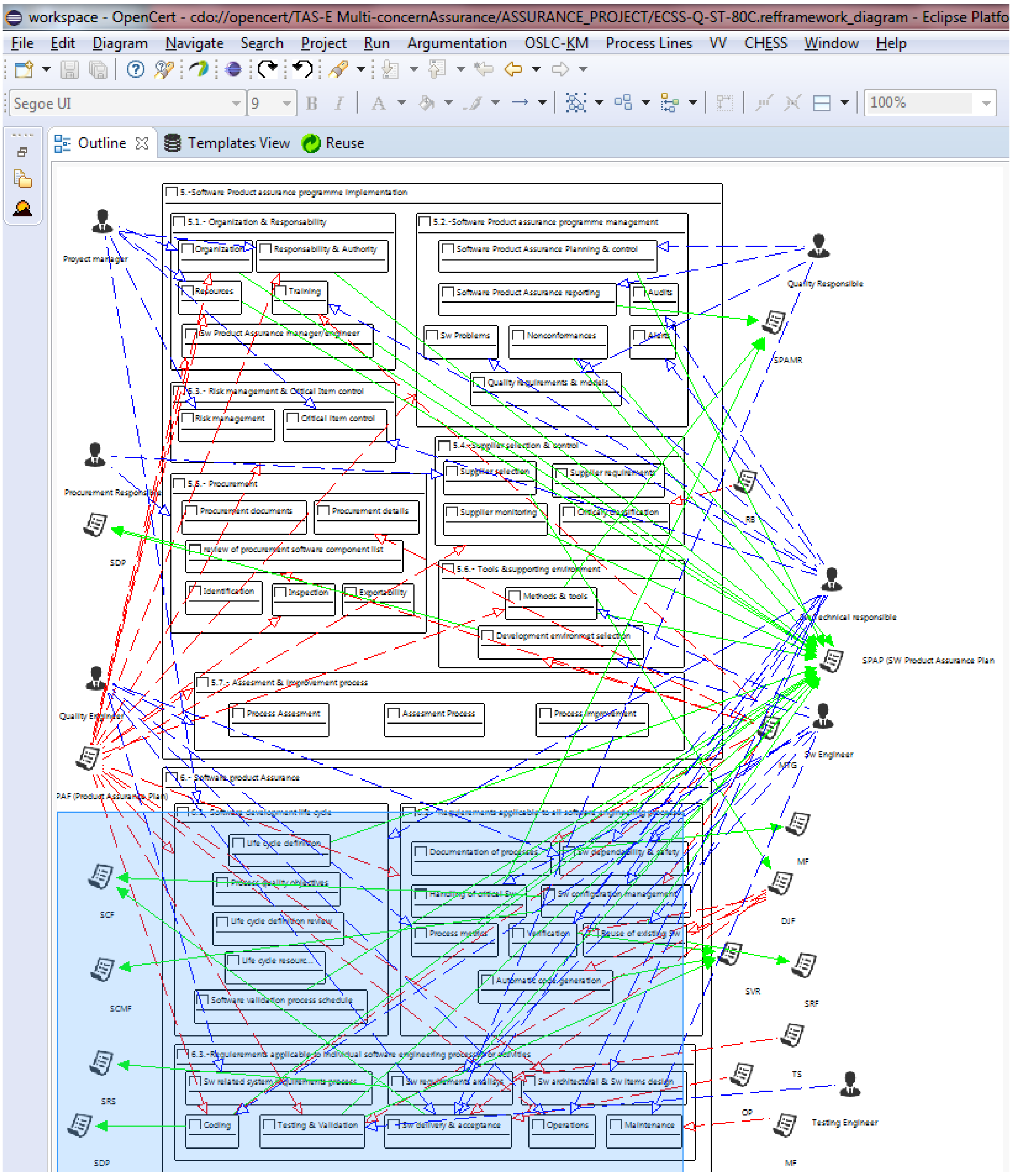

workspace - OpenCert - cdo://opencert/TAS-E Multi-concernAssurance/ASSURANCE_PROJECT/ECSS-Q-ST-80C.refframework_diagram - Eclipse Platfo
File Edit Diagram Navigate Search Project Run Argumentation OSLC-KM Process Lines VV CHESS Window Help
Segoe UI
9
100%
Outline
Templates View
Reuse
5.-Software Product assurance programme implementation
5.1.- Organization & Responsability
Organization
Responsability & Authority
Resources
Training
Sw Product Assurance manager/engineer
5.2.-Software Product assurance programme management
Software Product Assurance Planning & control
Software Product Assurance reporting
Audits
Sw Problems
Nonconformances
Alerts
Quality requirements & models
5.3.- Risk management & Critical Item control
Risk management
Critical Item control
5.4.- Supplier selection & control
Supplier selection
Supplier requirements
Supplier monitoring
Criticaly classification
5.5.- Procurement
Procurement documents
Procurement details
review of procurement software component list
Identification
Inspection
Exportability
5.6.- Tools & supporting environment
Methods & tools
Development environment selection
5.7.- Assesment & improvement process
Process Assesment
Assesment Process
Process improvement
6.- Software product Assurance
6.1.- Software development life cycle
Life cycle definition
Process quality objectives
Life cycle definition review
Life cycle resourc...
Software validation process schedule
6.2.- Requirements applicable to all software engineering processes
Documentation of processes
Sw dependability & safety
Handling of critical Sw
Sw configuration management
Process metrics
Verification
Reuse of existing Sw
Automatic code generation
6.3.-Requirements applicable to individual software engineering processes or activities
Sw related system requirements process
Sw requirements analysis
Sw architectural & Sw items design
Coding
Testing & Validation
Sw delivery & acceptance
Operations
Maintenance
Project manager
Procurement Responsible
SDP
Quality Engineer
PAF (Product Assurance Plan)
SCF
SCMF
SRS
SDP
Quality Responsible
SPAMR
RB
Technical responsible
SPAP (SW Product Assurance Plan)
Sw Engineer
MFG
MF
DJF
SVR
SRF
TS
OP
Testing Engineer
MF

Figure 107. Assurance Architecture overview

Above are some pictures to show the assurance architecture bigger and with higher resolution:

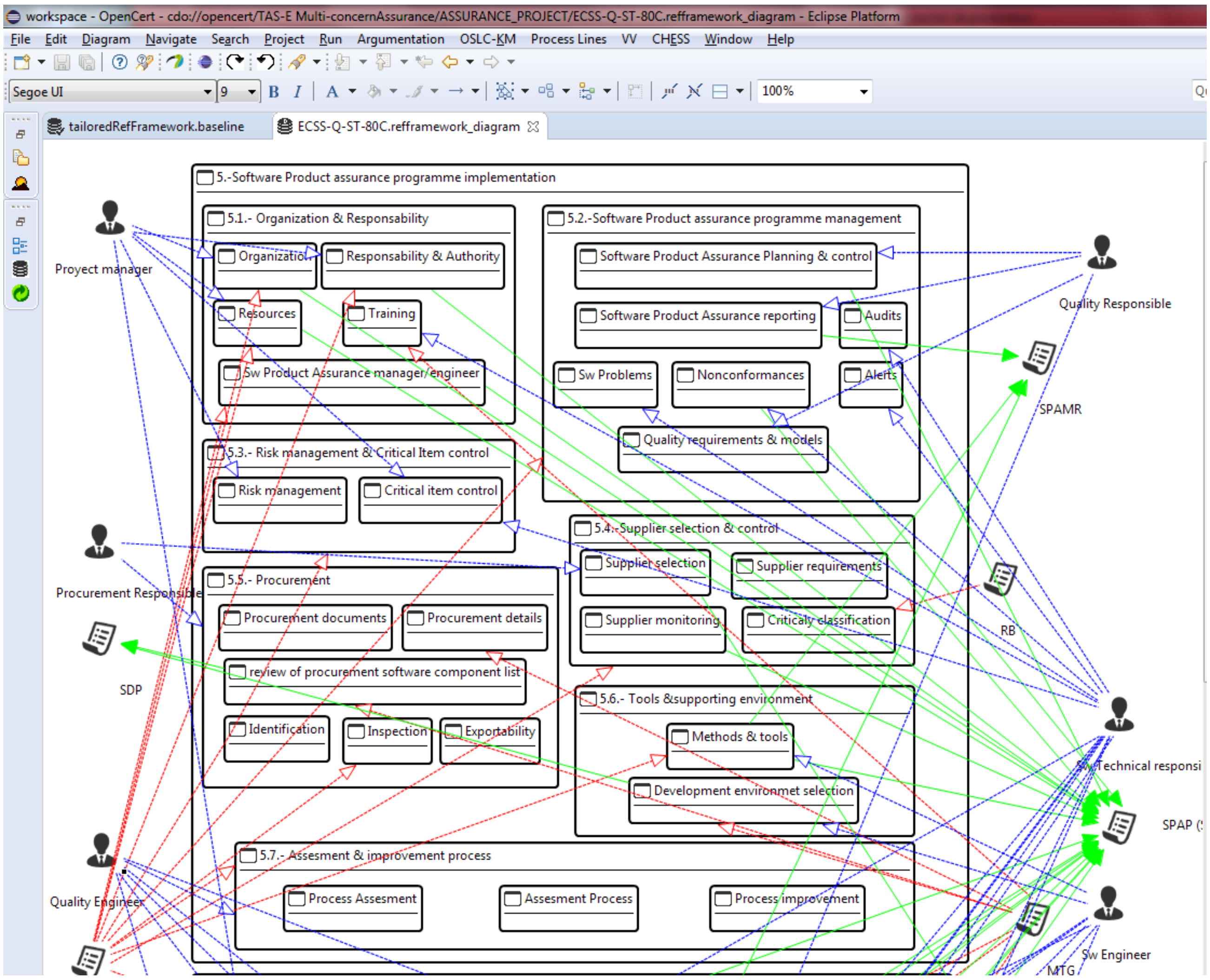

workspace - OpenCert - cdo://opencert/TAS-E Multi-concernAssurance/ASSURANCE_PROJECT/ECSS-Q-ST-80C.refframework_diagram - Eclipse Platform
File Edit Diagram Navigate Search Project Run Argumentation OSLC-KM Process Lines VV CHESS Window Help
Segoe UI
9
100%
tailoredRefFramework.baseline
ECSS-Q-ST-80C.refframework_diagram
5.-Software Product assurance programme implementation
5.1.- Organization & Responsability
Organizatio
Responsability & Authority
Resources
Training
Sw Product Assurance manager/engineer
5.2.-Software Product assurance programme management
Software Product Assurance Planning & control
Software Product Assurance reporting
Audits
Sw Problems
Nonconformances
Alerts
Quality requirements & models
5.3.- Risk management & Critical Item control
Risk management
Critical item control
5.4.-Supplier selection & control
Supplier selection
Supplier requirements
Supplier monitoring
Criticaly classification
5.5.- Procurement
Procurement documents
Procurement details
review of procurement software component list
Identification
Inspection
Exportability
5.6.- Tools &supporting environment
Methods & tools
Development environmet selection
5.7.- Assesment & improvement process
Process Assesment
Assesment Process
Process improvement
Proyect manager
Quality Responsible
SPAMR
Procurement Responsible
SDP
RB
Technical responsi
SPAP (
Quality Engineer
Sw Engineer
MTG

Figure 108. ECSS-Q-80C Assurance Architecture for TAS-E (1)

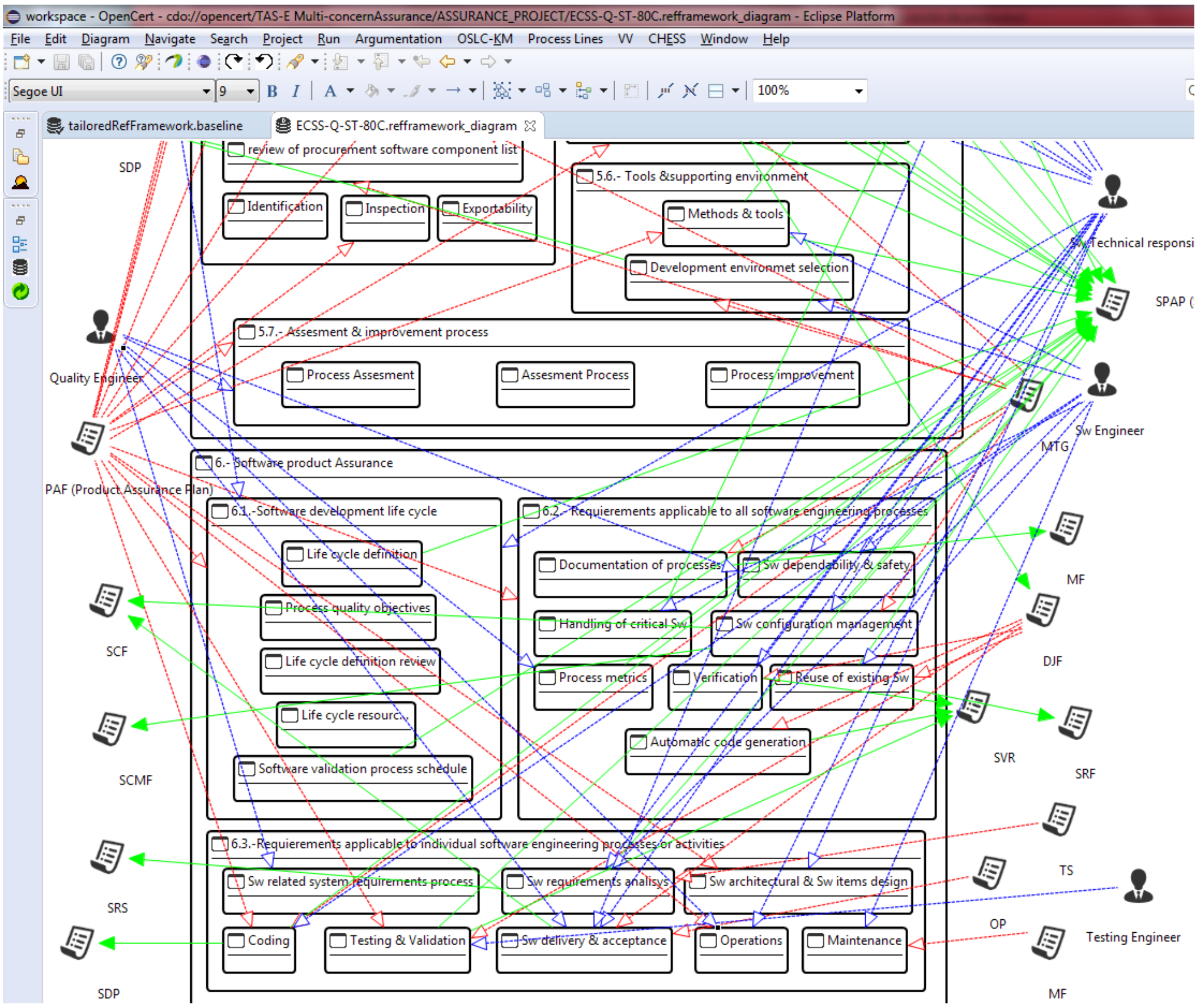
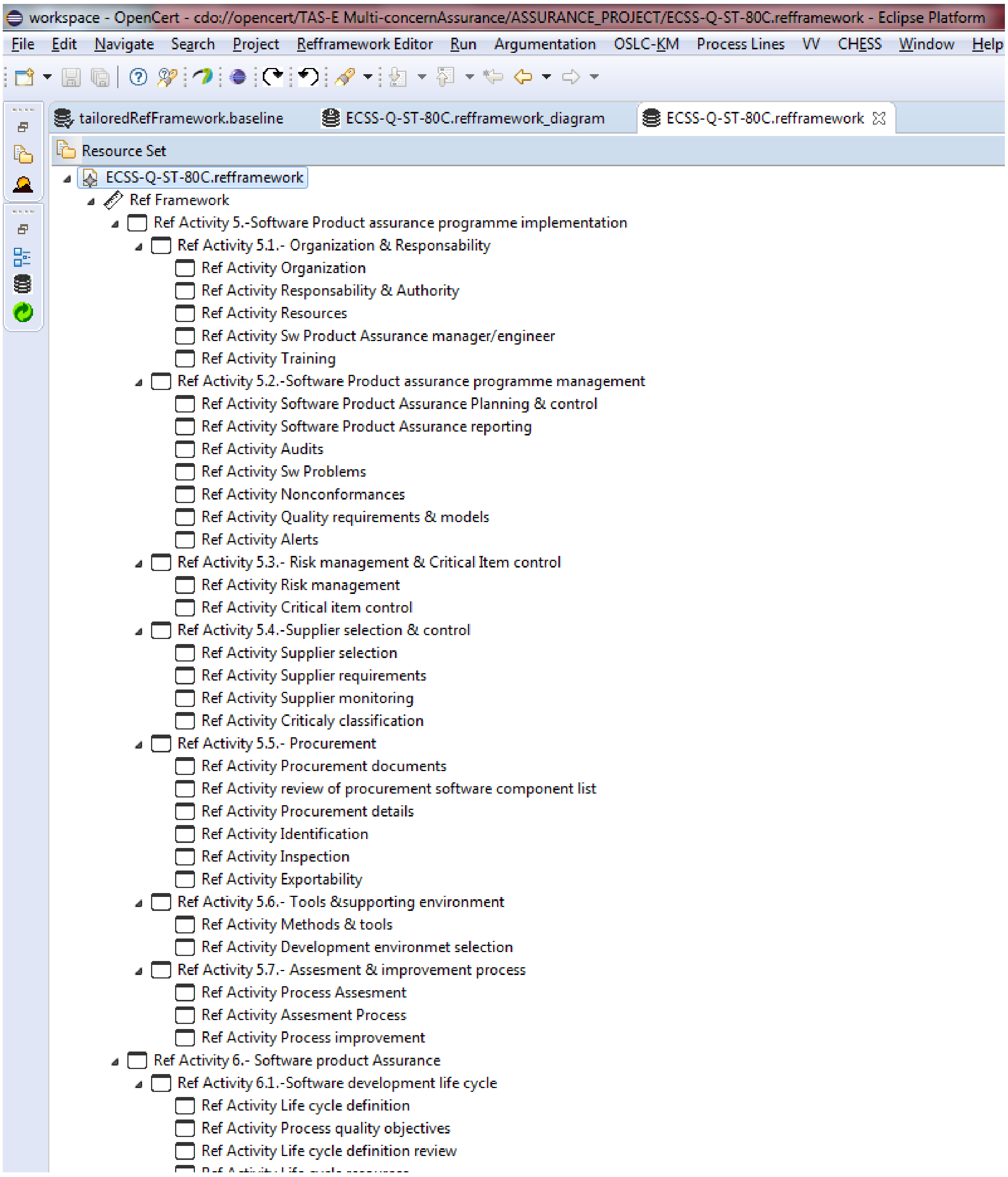

workspace - OpenCert - cdo://opencert/TAS-E Multi-concernAssurance/ASSURANCE_PROJECT/ECSS-Q-ST-80C.refframework_diagram - Eclipse Platform
File Edit Diagram Navigate Search Project Run Argumentation OSLC-KM Process Lines VV CHESS Window Help
Segoe UI 9 100%
tailoredRefFramework.baseline
ECSS-Q-ST-80C.refframework_diagram
review of procurement software component list
Identification
Inspection
Exportability
5.6.- Tools &supporting environment
Methods & tools
Development environmet selection
5.7.- Assesment & improvement process
Process Assesment
Assesment Process
Process improvement
6.- Software product Assurance
6.1.-Software development life cycle
Life cycle definition
Process quality objectives
Life cycle definition review
Life cycle resourc...
Software validation process schedule
6.2.- Requierements applicable to all software engineering processes
Documentation of processes
Sw dependability & safety
Handling of critical Sw
Sw configuration management
Process metrics
Verification
Reuse of existing Sw
Automatic code generation
6.3.-Requierements applicable to individual software engineering processes or activities
Sw related system requirements process
Sw requirements analisys
Sw architectural & Sw items design
Coding
Testing & Validation
Sw delivery & acceptance
Operations
Maintenance
SDP
Quality Engineer
PAF (Product Assurance Plan)
SCF
SCMF
SRS
SDP
Sw Technical responsi
SPAP (
Sw Engineer
MTG
MF
DJF
SVR
SRF
TS
OP
Testing Engineer
MF

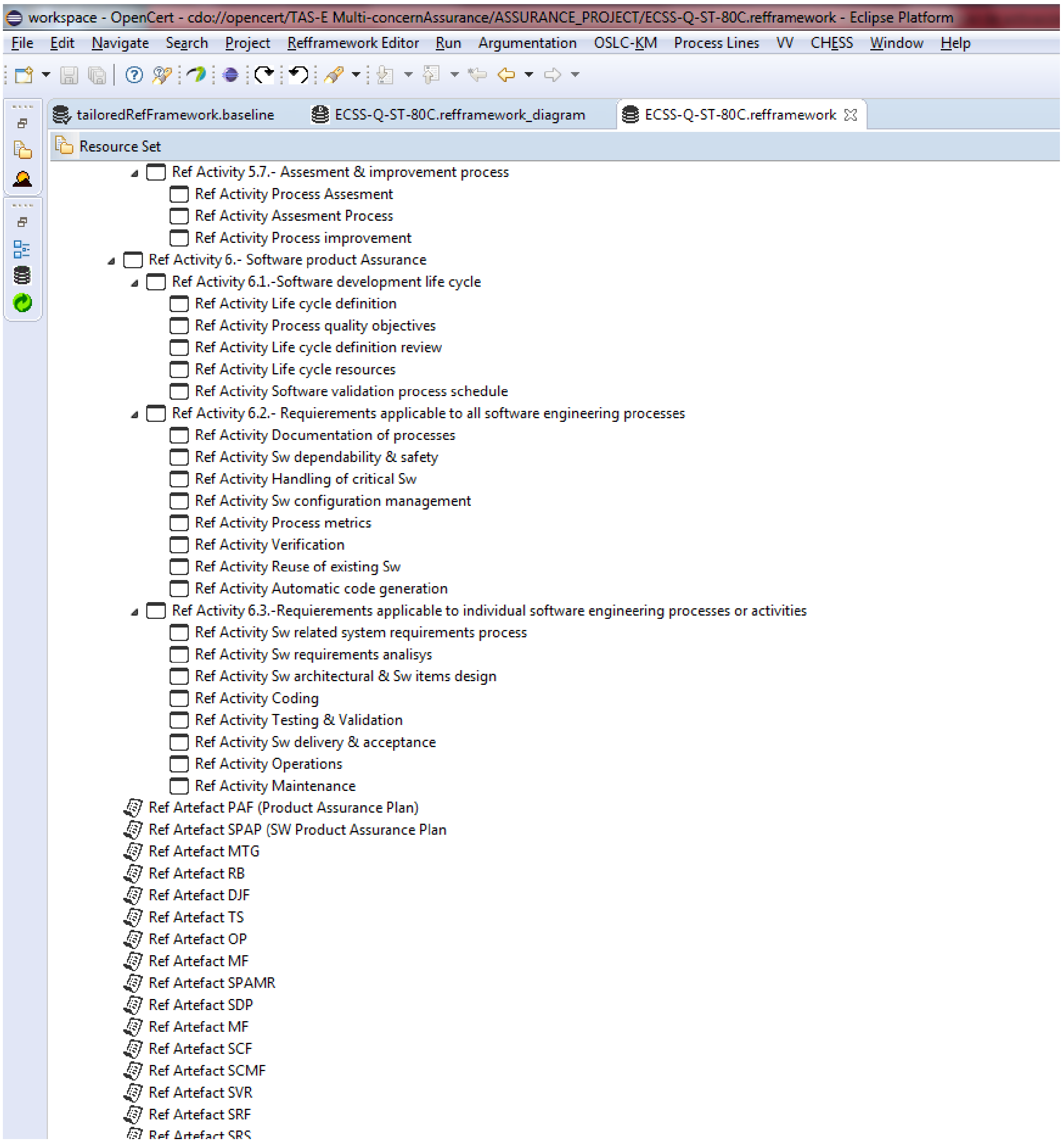
Figure 109. ECSS-Q-80C Assurance Architecture for TAS-E (2)

Here are attached the results of definition of activities, roles, artefacts and dependencies between modules:


workspace - OpenCert - cdo://opencert/TAS-E Multi-concernAssurance/ASSURANCE_PROJECT/ECSS-Q-ST-80C.refframework - Eclipse Platform
File Edit Navigate Search Project Refframework Editor Run Argumentation OSLC-KM Process Lines VV CHESS Window Help
tailoredRefFramework.baseline
ECSS-Q-ST-80C.refframework_diagram
ECSS-Q-ST-80C.refframework
Resource Set

- ECSS-Q-ST-80C.refframework
  - Ref Framework
    - Ref Activity 5.-Software Product assurance programme implementation
      - Ref Activity 5.1.- Organization & Responsability
        - Ref Activity Organization
        - Ref Activity Responsability & Authority
        - Ref Activity Resources
        - Ref Activity Sw Product Assurance manager/engineer
        - Ref Activity Training
      - Ref Activity 5.2.-Software Product assurance programme management
        - Ref Activity Software Product Assurance Planning & control
        - Ref Activity Software Product Assurance reporting
        - Ref Activity Audits
        - Ref Activity Sw Problems
        - Ref Activity Nonconformances
        - Ref Activity Quality requirements & models
        - Ref Activity Alerts
      - Ref Activity 5.3.- Risk management & Critical Item control
        - Ref Activity Risk management
        - Ref Activity Critical item control
      - Ref Activity 5.4.-Supplier selection & control
        - Ref Activity Supplier selection
        - Ref Activity Supplier requirements
        - Ref Activity Supplier monitoring
        - Ref Activity Criticaly classification
      - Ref Activity 5.5.- Procurement
        - Ref Activity Procurement documents
        - Ref Activity review of procurement software component list
        - Ref Activity Procurement details
        - Ref Activity Identification
        - Ref Activity Inspection
        - Ref Activity Exportability
      - Ref Activity 5.6.- Tools &supporting environment
        - Ref Activity Methods & tools
        - Ref Activity Development environmet selection
      - Ref Activity 5.7.- Assesment & improvement process
        - Ref Activity Process Assesment
        - Ref Activity Assesment Process
        - Ref Activity Process improvement
    - Ref Activity 6.- Software product Assurance
      - Ref Activity 6.1.-Software development life cycle
        - Ref Activity Life cycle definition
        - Ref Activity Process quality objectives
        - Ref Activity Life cycle definition review

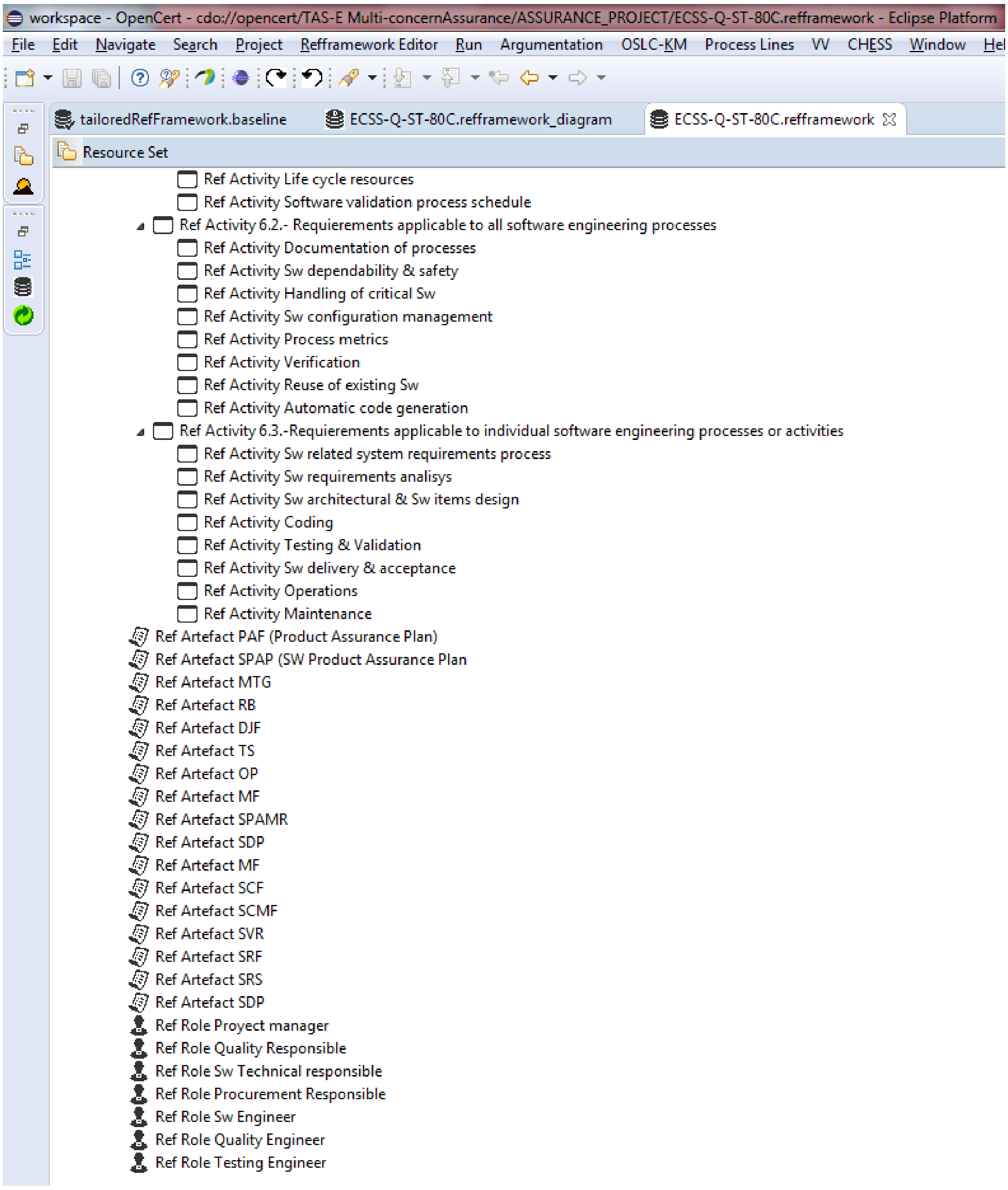
Figure 110. Refframework of activities (1)


workspace - OpenCert - cdo://opencert/TAS-E Multi-concernAssurance/ASSURANCE_PROJECT/ECSS-Q-ST-80C.refframework - Eclipse Platform

File Edit Navigate Search Project Refframework Editor Run Argumentation OSLC-KM Process Lines VV CHESS Window Help
tailoredRefFramework.baseline
ECSS-Q-ST-80C.refframework_diagram
ECSS-Q-ST-80C.refframework

Resource Set
- Ref Activity 5.7.- Assesment & improvement process
  - Ref Activity Process Assesment
  - Ref Activity Assesment Process
  - Ref Activity Process improvement
- Ref Activity 6.- Software product Assurance
  - Ref Activity 6.1.-Software development life cycle
    - Ref Activity Life cycle definition
    - Ref Activity Process quality objectives
    - Ref Activity Life cycle definition review
    - Ref Activity Life cycle resources
    - Ref Activity Software validation process schedule
  - Ref Activity 6.2.- Requierements applicable to all software engineering processes
    - Ref Activity Documentation of processes
    - Ref Activity Sw dependability & safety
    - Ref Activity Handling of critical Sw
    - Ref Activity Sw configuration management
    - Ref Activity Process metrics
    - Ref Activity Verification
    - Ref Activity Reuse of existing Sw
    - Ref Activity Automatic code generation
  - Ref Activity 6.3.-Requierements applicable to individual software engineering processes or activities
    - Ref Activity Sw related system requirements process
    - Ref Activity Sw requirements analisys
    - Ref Activity Sw architectural & Sw items design
    - Ref Activity Coding
    - Ref Activity Testing & Validation
    - Ref Activity Sw delivery & acceptance
    - Ref Activity Operations
    - Ref Activity Maintenance
- Ref Artefact PAF (Product Assurance Plan)
- Ref Artefact SPAP (SW Product Assurance Plan
- Ref Artefact MTG
- Ref Artefact RB
- Ref Artefact DJF
- Ref Artefact TS
- Ref Artefact OP
- Ref Artefact MF
- Ref Artefact SPAMR
- Ref Artefact SDP
- Ref Artefact MF
- Ref Artefact SCF
- Ref Artefact SCMF
- Ref Artefact SVR
- Ref Artefact SRF
- Ref Artefact SRS

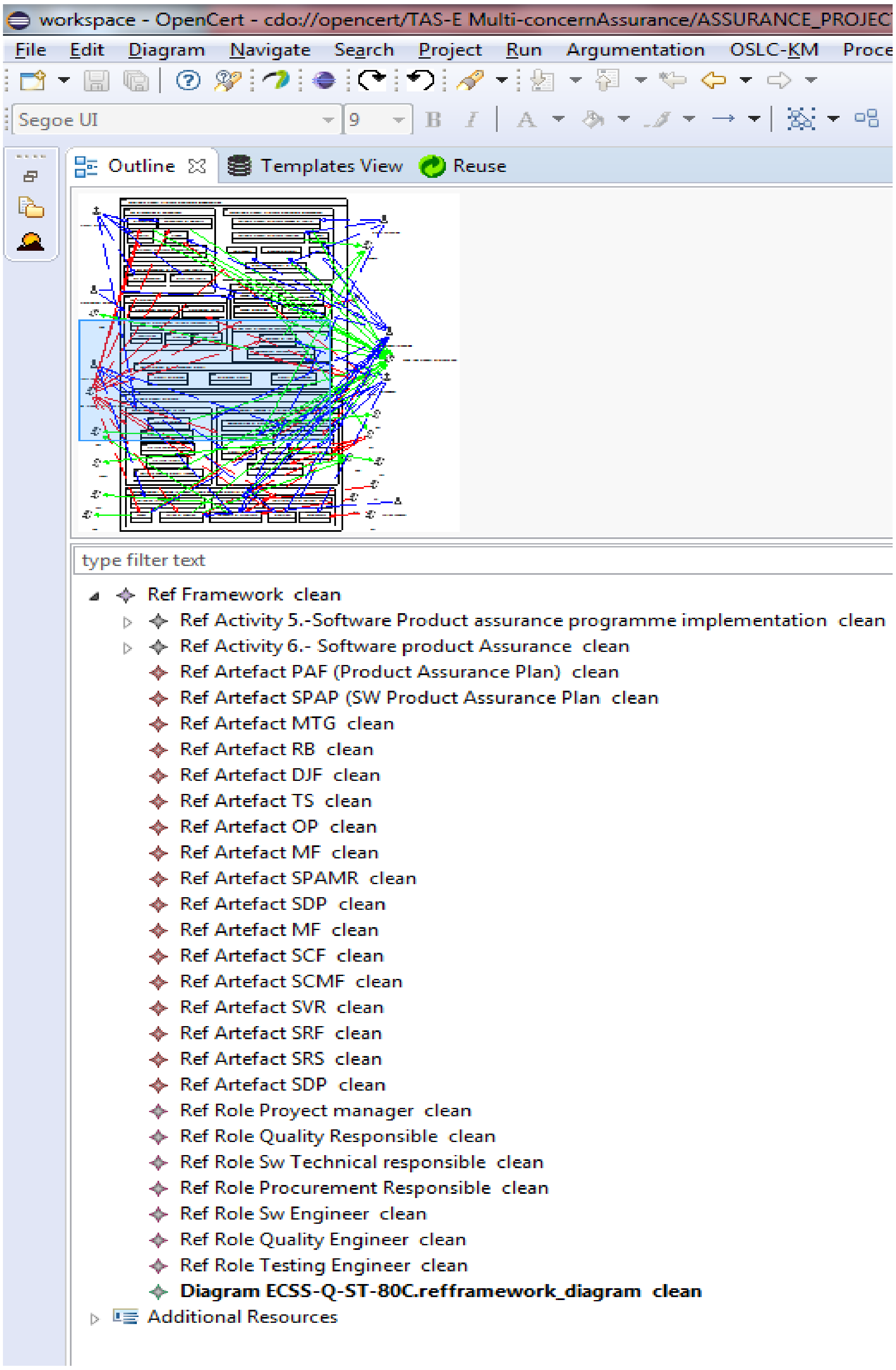
Figure 111. Refframework of activities (2)


workspace - OpenCert - cdo://opencert/TAS-E Multi-concernAssurance/ASSURANCE_PROJECT/ECSS-Q-ST-80C.refframework - Eclipse Platform
File Edit Navigate Search Project Refframework Editor Run Argumentation OSLC-KM Process Lines VV CHESS Window Hel
tailoredRefFramework.baseline
ECSS-Q-ST-80C.refframework_diagram
ECSS-Q-ST-80C.refframework

Resource Set
- Ref Activity Life cycle resources
- Ref Activity Software validation process schedule
- Ref Activity 6.2.- Requierements applicable to all software engineering processes
  - Ref Activity Documentation of processes
  - Ref Activity Sw dependability & safety
  - Ref Activity Handling of critical Sw
  - Ref Activity Sw configuration management
  - Ref Activity Process metrics
  - Ref Activity Verification
  - Ref Activity Reuse of existing Sw
  - Ref Activity Automatic code generation
- Ref Activity 6.3.-Requierements applicable to individual software engineering processes or activities
  - Ref Activity Sw related system requirements process
  - Ref Activity Sw requirements analisys
  - Ref Activity Sw architectural & Sw items design
  - Ref Activity Coding
  - Ref Activity Testing & Validation
  - Ref Activity Sw delivery & acceptance
  - Ref Activity Operations
  - Ref Activity Maintenance
- Ref Artefact PAF (Product Assurance Plan)
- Ref Artefact SPAP (SW Product Assurance Plan
- Ref Artefact MTG
- Ref Artefact RB
- Ref Artefact DJF
- Ref Artefact TS
- Ref Artefact OP
- Ref Artefact MF
- Ref Artefact SPAMR
- Ref Artefact SDP
- Ref Artefact MF
- Ref Artefact SCF
- Ref Artefact SCMF
- Ref Artefact SVR
- Ref Artefact SRF
- Ref Artefact SRS
- Ref Artefact SDP
- Ref Role Proyect manager
- Ref Role Quality Responsible
- Ref Role Sw Technical responsible
- Ref Role Procurement Responsible
- Ref Role Sw Engineer
- Ref Role Quality Engineer
- Ref Role Testing Engineer

Figure 112. Refframework of artefacts and roles (1)

We can outline the process by:


workspace - OpenCert - cdo://opencert/TAS-E Multi-concernAssurance/ASSURANCE_PROJEC
File Edit Diagram Navigate Search Project Run Argumentation OSLC-KM Proce
Segoe UI
9
Outline
Templates View
Reuse
type filter text

Ref Framework clean
- Ref Activity 5.-Software Product assurance programme implementation clean
- Ref Activity 6.- Software product Assurance clean
- Ref Artefact PAF (Product Assurance Plan) clean
- Ref Artefact SPAP (SW Product Assurance Plan clean
- Ref Artefact MTG clean
- Ref Artefact RB clean
- Ref Artefact DJF clean
- Ref Artefact TS clean
- Ref Artefact OP clean
- Ref Artefact MF clean
- Ref Artefact SPAMR clean
- Ref Artefact SDP clean
- Ref Artefact MF clean
- Ref Artefact SCF clean
- Ref Artefact SCMF clean
- Ref Artefact SVR clean
- Ref Artefact SRF clean
- Ref Artefact SRS clean
- Ref Artefact SDP clean
- Ref Role Proyect manager clean
- Ref Role Quality Responsible clean
- Ref Role Sw Technical responsible clean
- Ref Role Procurement Responsible clean
- Ref Role Sw Engineer clean
- Ref Role Quality Engineer clean
- Ref Role Testing Engineer clean
- **Diagram ECSS-Q-ST-80C.refframework_diagram clean**

Additional Resources

Figure 113. Summary of the Assurance Case Study

3.10.2.2.2. Allocate system goals to concerns

Allocation is important to permit detailed analysis of requirements. Once a set of requirements has been allocated to a component, the individual requirements can be further analysed to discover further requirements on how the component needs to interact with other components in order to satisfy the allocated requirements.

We can define a system as: “an interacting combination of elements to accomplish a defined objective. These include hardware, software, firmware, people, information, techniques, facilities, services, and other support elements” according to the International Council on Software and Systems Engineering (INCOSE).

We can distinguish between:

- System requirements are the requirements for the system as a whole. In a system containing software components, software requirements are derived from system requirements.
- Product requirement is a need or constraint on the software to be developed.
- Process requirement is essentially a constraint on the development of the software.

System requirements defines the high-level system requirements from the Space domain perspective. It includes representatives of the system users/customers. The document lists the system requirements along with background information about the overall objectives for the system, its target environment, and a statement of the constraints, assumptions, and non-functional requirements. It may include conceptual models designed to illustrate the system context, usage scenarios, and the principal domain entities, as well as workflows.

We can model the allocation of the system goals to concerns using OpenCert plugin:

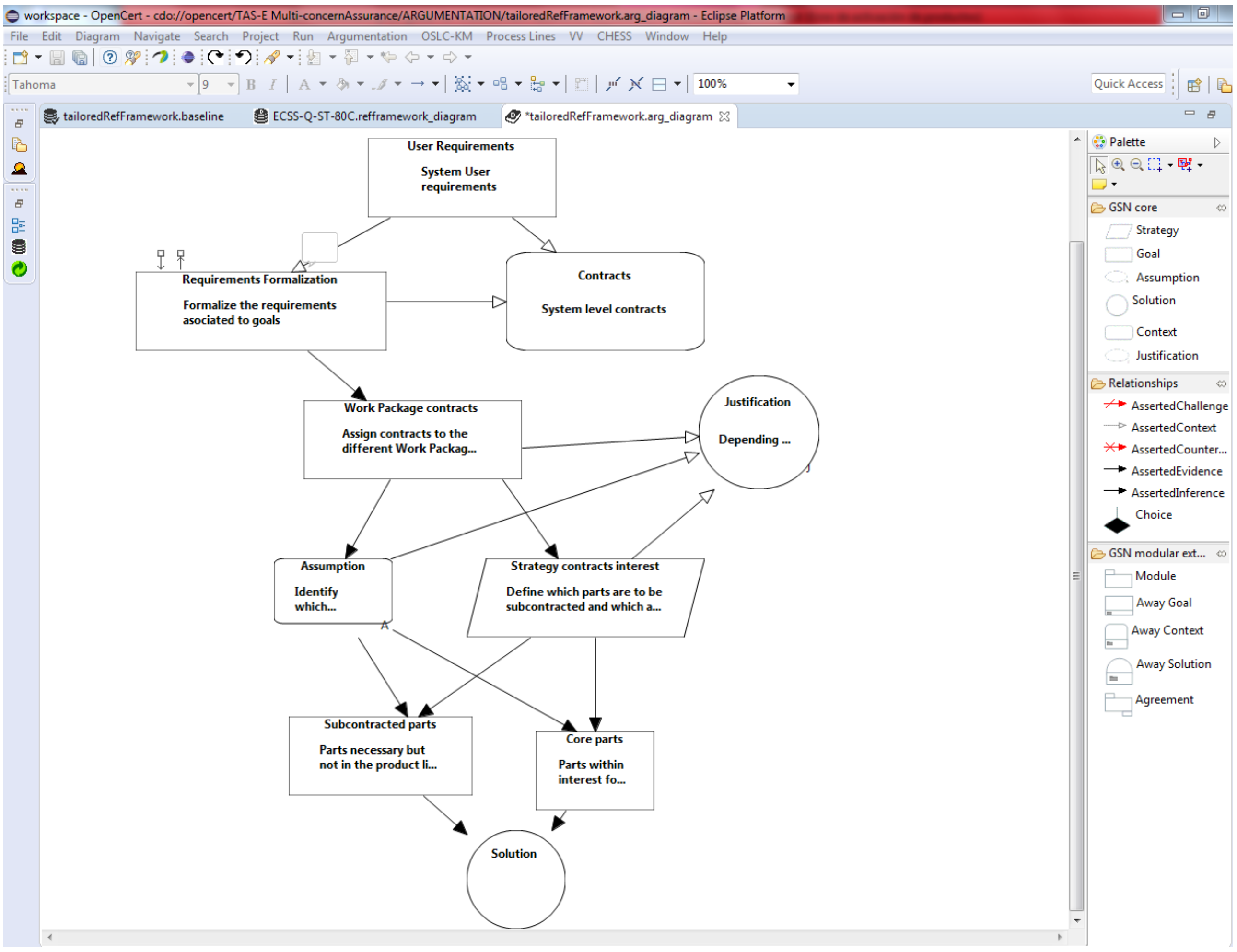

workspace - OpenCert - cdo://opencert/TAS-E Multi-concernAssurance/ARGUMENTATION/tailoredRefFramework.arg_diagram - Eclipse Platform
File Edit Diagram Navigate Search Project Run Argumentation OSLC-KM Process Lines VV CHESS Window Help
Tahoma
9
100%
Quick Access
tailoredRefFramework.baseline
ECSS-Q-ST-80C.refframework_diagram
*tailoredRefFramework.arg_diagram
User Requirements
System User requirements
Requirements Formalization
Formalize the requirements asociated to goals
Contracts
System level contracts
Work Package contracts
Assign contracts to the different Work Packag...
Justification
Depending ...
Assumption
Identify which...
Strategy contracts interest
Define which parts are to be subcontracted and which a...
Subcontracted parts
Parts necessary but not in the product li...
Core parts
Parts within interest fo...
Solution
Palette
GSN core
Strategy
Goal
Assumption
Solution
Context
Justification
Relationships
AssertedChallenge
AssertedContext
AssertedCounter...
AssertedEvidence
AssertedInference
Choice
GSN modular ext...
Module
Away Goal
Away Context
Away Solution
Agreement

Figure 114. Model of the system allocation requirements

3.10.2.2.3. Derive system requirements

Developers of systems with substantial software and non-software components separate the description of system requirements from the description of software requirements. At the point when system requirements are specified, the software requirements are derived from the system requirements, and then the requirements for the software components are specified.

Software requirements specification establishes the basis for agreement between customers and contractors or suppliers on what the software product is to do as well as what it is not expected to do.

Software requirements specification permits a rigorous assessment of requirements before design can begin and reduces later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules. Software requirements specification can also use a software requirements specification document as the basis for developing effective verification and validation plans.

Here is attached the model developed for TAS-E Usage Scenario with its particularities:

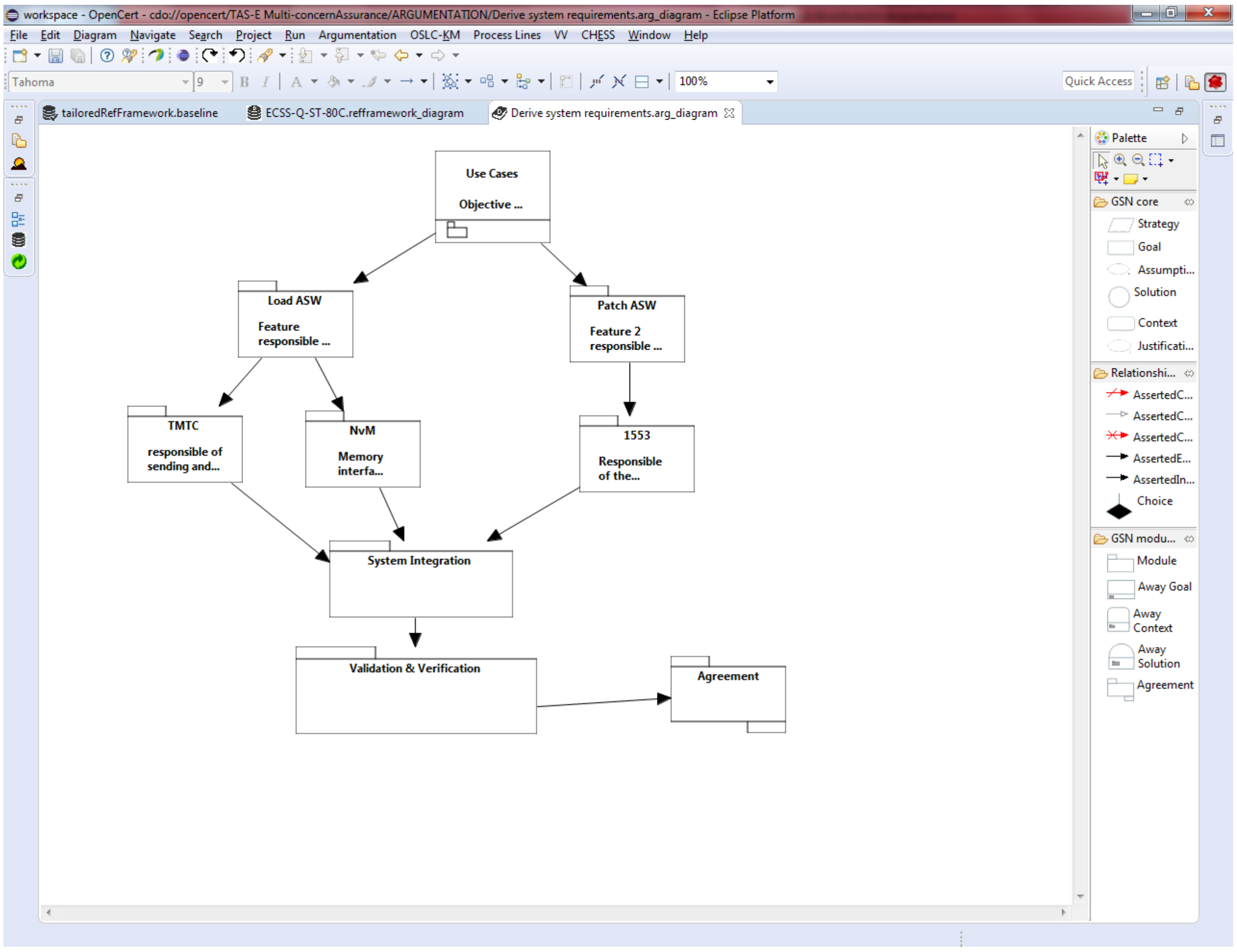

workspace - OpenCert - cdo://opencert/TAS-E Multi-concernAssurance/ARGUMENTATION/Derive system requirements.arg_diagram - Eclipse Platform
File Edit Diagram Navigate Search Project Run Argumentation OSLC-KM Process Lines VV CHESS Window Help
Tahoma 9 100% Quick Access
tailoredRefFramework.baseline
ECSS-Q-ST-80C.reframework_diagram
Derive system requirements.arg_diagram
Use Cases
Objective ...
Load ASW
Feature responsible ...
Patch ASW
Feature 2 responsible ...
TMTC
responsible of sending and...
NvM
Memory interfa...
1553
Responsible of the...
System Integration
Validation & Verification
Agreement
Palette
GSN core
Strategy
Goal
Assumpti...
Solution
Context
Justificati...
Relationshi...
AssertedC...
AssertedC...
AssertedC...
AssertedE...
AssertedIn...
Choice
GSN modu...
Module
Away Goal
Away Context
Away Solution
Agreement

Figure 115. Derive of the system requirements to SW requirements model

3.10.2.2.4. Analyse the interplay

Once the system requirements have passed to software requirements, the criticism of the software must be evaluated. For this business there are two analyses to be done:

- FMEA (Failure Mode Effects Analysis): is a step-by-step approach for identifying all possible failures in a design, a manufacturing or assembly process, or a product or service. "Failure modes" means the ways, or modes, in which something might fail. Failures are any errors or defects, especially ones that affect the customer, and can be potential or actual. Failures are prioritized according to how serious their consequences are, how frequently they occur and how easily they can be detected. The purpose of the FMEA is to take actions to eliminate or reduce failures, starting with the highest-priority ones

- SCAR (Static Code Analysis Review): also known as Source Code Analysis Review, is a white-box testing which refers to the running of Static Code Analysis that attempt to highlight possible vulnerabilities within 'static' (non-running) source code by using techniques such as Taint Analysis and Data Flow Analysis. SCAR is an automated process in which a machine, informed by what it knows about the language analysis (usually from the type system), analyses a program and tries to pick out things that could be incorrect, inefficient, poor style, or otherwise suboptimal.

Once the results of this analysis are available the software receives, based on them, a criticism level establish in the ECSS-Q-ST-80C:

Category	Definition
A	Software that if not executed, or if not correctly executed, or whose anomalous behaviour can cause or contribute to a system failure resulting in: ➔ Catastrophic consequences
B	Software that if not executed, or if not correctly executed, or whose anomalous behaviour can cause or contribute to a system failure resulting in: ➔ Critical consequences
C	Software that if not executed, or if not correctly executed, or whose anomalous behaviour can cause or contribute to a system failure resulting in: ➔ Major consequences
D	Software that if not executed, or if not correctly executed, or whose anomalous behaviour can cause or contribute to a system failure resulting in: ➔ Minor or Negligible consequences

Usage Scenario defined by TAS-E has been given category C.

3.10.2.2.5. Edit assurance case

Once the analysis of interplay is being done, there is one applicability matrix that represents a tailoring of the requirements of the ECSS-Q-ST-80C based on the software criticality categories defined above.

For each clause of the ECSS-Q-ST-80C and for each software criticality category, an indication is given whether that clause is applicable (Y), not applicable (N), or applicable under the conditions thereby specified to that software criticality category (for more information look at ECSS-Q-ST-80C Annex D.2).

According to the matrix the requirements must to be tailored. This classification is defined by the ECSS-E-ST-40C in the annex R.2.

Table 65. CS10-Multi-concern assurance: US1&US2-Assurance Project Management (Create Assurance Project)

Realisation Scenario	Assurance Project Management (Create Assurance Project)
Scope	Define an Architectural assurance model for software in Space Domain. It includes all the process from the Client/User desires down to the more basic requirements: organization, scheduling, programming, verification…
Tool Settings	Papyrus, OpenCert
Participants	• TAS-E • UC assessment: TEC, GMV
Activities realised	Model Standard ECSS-Q-ST-80C as assurance modelling architecture with Papyrus and OpenCert. Definition of system requirements allocation with OpenCert

	Definition of the derivative process from system requirements to software requirements with OpenCert
Usage Decisions	Modelling the software product assurance architecture and the up to down requirements and objects to take into account during the life cycle process.
Expected Results	Assurance case and system to software requirements process modelling.
Conclusions	

3.10.3. Coverage of AMASS Prototype P1 Architecture

Table 66 illustrates the implemented functionalities during this second iteration within the Case Study 10.

Table 66. AMASS Prototype P1 Coverage by CS10

STO	AMASS Functionality Groups	Tools
Architecture-Driven Assurance	System Component Specification	CHESS
	System Architecture Modelling for Assurance	CHESS
	Architectural Patterns for Assurance	-
	Contract-based Design for Assurance	CHESS + OCRA
	Activities supporting Assurance Case	V&V Failed, FMEA mentioned
Multi-Concern Assurance	Assurance Case Specification	OpenCert
	Dependability Assurance	OpenCert
	System Dependability Co-Analysis/Co-Assessment	OpenCert
	Contract-Based Multi-concern Assurance	-
Seamless Interoperability	Evidence Management	OpenCert
	Tool Integration Management	-
	Collaborative Work Management	-
	Tool Quality Assessment and Characterization	-
Cross Intra-Domain Reuse	Compliance Management	OpenCert
	Reuse Assistant	-
	Process-related reuse via management of variability at process level	-
	Process-related reuse via management of variability at product level	-
	Automatic generation of process-based arguments	-
	Automatic generation of product-based arguments	-

3.10.4. Conclusions

At this stage, the main benefits and potential improvements are identified in Table 67.

Table 67. Benefits and potential improvements for CS10

Artefact	Achievement/Benefits	Improvements/Recommendations
Requirements specification and formalization	• SW requirements specification • Requirements formalization • Requirements mapped formal properties	• Formal specification language (OCRA) not very intuitive, requires more training or examples (for example to formalize execution of SW functions).

Artefact	Achievement/Benefits	Improvements/Recommendations
Design of system/software architecture	• Class diagram • Blocks diagram (BDD) • Interface diagram, modelled using flow ports	• In BDD diagrams, do not know very precisely how to use other properties apart from flow ports. • Possibility to hide connector lines in the IBD for the sake of clarity.
Contracts & Contract refinement	• Contracts made based on Design Blocks formal properties	
V&V Tools	• Check on formal properties not working with current CS10 model.	• Include EByte array type to be used as the port type in the model.
Assurance case architecture	• Standard & argumentation base model	
Allocate & Derive requirements	• Template and valid model to derive requirements	
Analyse the interplay	• Explicit analysis of risks and security based in formal aspects described above.	

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

3.11.1. 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:

1. Ensure re-use of methods and components across different projects/missions.
2. Seamless integration of development tools to semi-automate evidence management.
3. Creation of a SW development cycle that guarantees compliance with ECSS standards required by ESA and compliance with customer requirements.

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

3.11.2. US1: Managing compliance with ECSS-E-ST-40C

3.11.2.1. STO2 Multi-concern Assurance

ConcertoFLA allows engineers to decorate component-based architectural models with dependability related information, execute FLA (Failure Logic Analysis) techniques, and get the results back-propagated onto the original model. In the use case we customized the CHESS methodology and ConcertoFLA in the context of the ECSS standards to enable architects and dependability experts to define a system and perform dependability-centred co-analysis for assuring the required non-functional properties of the system according to ECSS requirements. The result of the customization was accepted at AdaEurope-2018. Figure 124 summarizes the workflow. The execution of the workflow was exploited to illustrate the usage of ConcertoFLA in D4.7 [6] . Thus, the content is not reproduced in this deliverable.

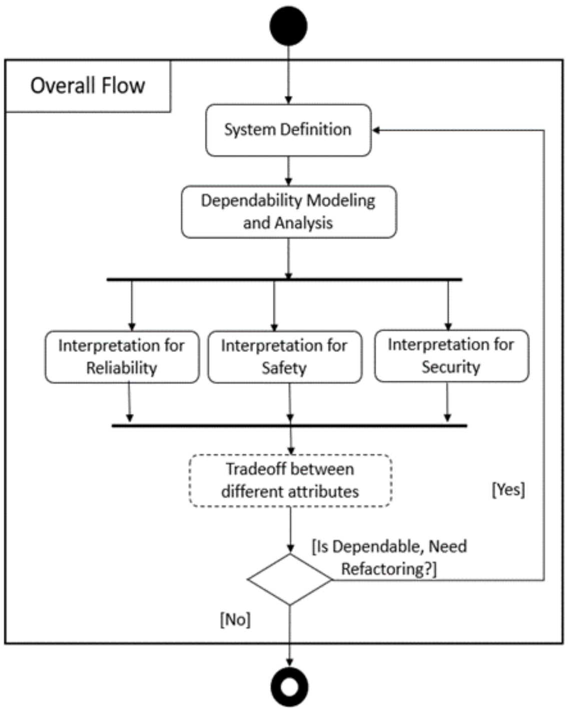

Overall Flow
System Definition
Dependability Modeling and Analysis
Interpretation for Reliability
Interpretation for Safety
Interpretation for Security
Tradeoff between different attributes
[Yes]
[Is Dependable, Need Refactoring?]
[No]

Figure 116. Dependability Co-Analysis via CHESS and ConcertoFLA

Table 68. CS11-Multi-concern Assurance: US1-FLA techniques in accordance with ECSS

Realisation Scenario	FLA techniques in accordance with ECSS
Scope	Multi-concern dependability assurance
Tool Settings	ConcertoFLA, CHESS
Participants	Tool Support: MDH User: OHB, MDH
Activities realised	Exploration of potential usefulness of ConcertoFLA in the space domain
Usage Decisions	The usage is restricted to a simplified and illustrative example
Expected Results	Mono-concern analysis
Conclusions	As expected, the results obtained are limited to a series of mono-concern analysis. Multi-concern remains fully manual. The automation of the mono-concern analysis seems promising and it will be further evaluated for the third release of the prototype.

3.11.3. US2: V&V integration of RapiCov

3.11.3.1. STO3 Seamless Interoperability

This scenario analyses the use of a commercial S/W tool for code-coverage in the ELECTRA AOCS Subsystem compared to the current practise of using open-source S/W.

Currently the ELECTRA AOCS project (from now on only stated as AOC) is using open-source gcov for code-coverage analysis on a commercial LEON2 emulator TSIM.

The commercial S/W tool selected was Rapita Systems Ltd RVS tool RapiCov.

RAPITA also has tools for Timing analysis RapiTime, scheduling/event tracking RapiTask and unit/system testing RapiTest. These was not integrated as the AOC in itself is embedded in the OBSW and as such runs within a single task and the timing analysis is done with ABSINT and is done seldom so any gain for AOC is minimal.

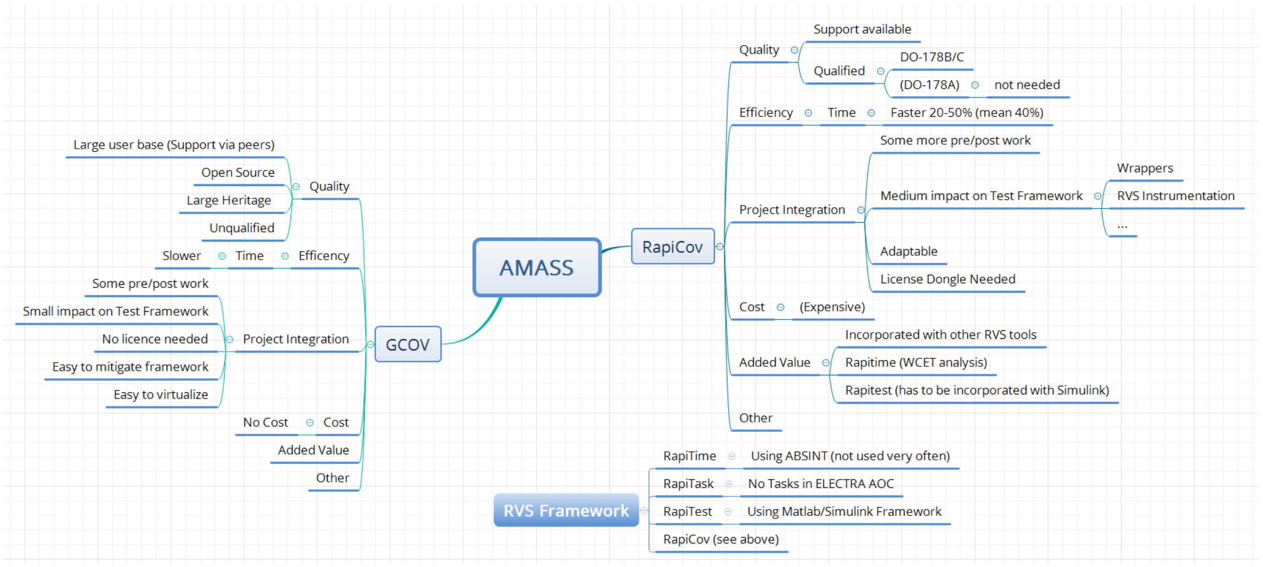


Figure 117. V&V tool integration of RapiCov

Test Approach

The Unit Tests are developed and executed within a CSU (Computer Software Unit) Test Framework (in Matlab/Simulink) as a test harness encompassing the CSU under test, the test harness may take source data from a file as input.

Unit Tests are open loop and are executed both in Simulink, using the CSU Model, and on a target processor emulator (TSIM) using the generated and compiled (CSU) code, this provides Equivalence Testing, where the generated code is tested against the corresponding CSU Model. The code generation and compilation is the same as will be used in flight.

Test Environment

Matlab/Simulink running on a win64 system is used for Unit Tests on model level. The Simulink environment consists of Matlab, Control System Toolbox, Simulink, Stateflow, Embedded Coder, Stateflow Coder, and Simulink Verification and Validation Toolbox. Unit tests are re-executed on TSIM.

Test Case Design

Unit Testing of CSUs will be done using a test harness that will provide the CSU with stimuli and will also read the output.

Table 69. CS11-Seamless Interoperability: US2-V&V tool integration for code coverage

Realisation Scenario	**V&V tool integration for code coverage**
Scope	Seamless V&V tool integration
Tool Settings	Rapita Systems Code Coverage
Participants	Tool Support: Rapita Systems User: OHB
Activities realised	1. Integrate RAPITA RVS Tool into AOC: Done Rapita Verification Suite (RVS) is a set of tools for performing on-target software verification of embedded systems. In order to provide this facility, RVS needs to be integrated with the target. Integration means: • Building the RVS instrument into the toolchain for building the target software. • Designing and implementing a mechanism for collecting verification data

	from the target. • Collecting and processing verification data into a form that can be processed by the RVS analysis tools. The RAPITA RVS tool RapiCov has been successfully integrated within our development environment. 2. Compare the two approaches: To be performed • Quality • Efficiency (time saving performing and maintaining etc.) • Added value (qualified etc.) • Cost • Project Integration
Usage Decisions	
Expected Results	Improve quality of the source code. Improve efficiency of the V&V activities.
Conclusions	

3.11.4. US3: Process-Related Reuse via Management of Process Lines

3.11.4.1. STO4 Cross Intra-domain reuse

ECSS-E-ST-40C targets software development. It is one of the series of ECSS standards intended to be applied together for the management, engineering and product assurance in space projects and applications. Similar to other standards, it represents the effect “standards for making standards”, the idea being that this permits suppliers to use their own standards, provided that they comply with the requirements of ECSS-E-40 or some tailoring of it defined by the customer. The tailoring rules are provided in a specific annex, Annex R (normative). Specifically, the tailoring is conducted based on the software criticality, which ranges from A to D. The integration, for instance, is composed of two tasks: Software integration test plan and Software integration test report. According to Annex R, the former is applicable (Y) for levels A-B, and is also applicable (Y) for level C except SUITP K.9 and K10; but it is not applicable for level D. The latter is applicable (Y) for levels A-C; but inapplicable (N) for level D. Thus, the support for valid tailoring is fundamental for process engineering.

This usage scenario is related to the process variability management. The software processes are modelled in EPF Composer. However, to be able to configure the process lines, the seamless integration with variability modelling and management solution, in particularly BVR tool is achieved. The generation of target configurations is performed with BVR VSpec, Resolution, and Realisation editors, as illustrated in Figure 118.

The constraints are specified in the VSpec editor; valid tailoring is guaranteed if the constraints are properly specified. It might be noted that the automatic generation of variability model (VSpec) would be supported in the next release (Prototype P2). The configured process models are automatically exported back to the EPF Composer. Consequently, the popup menu is appeared. The yes option loads the changes into EPF Composer.

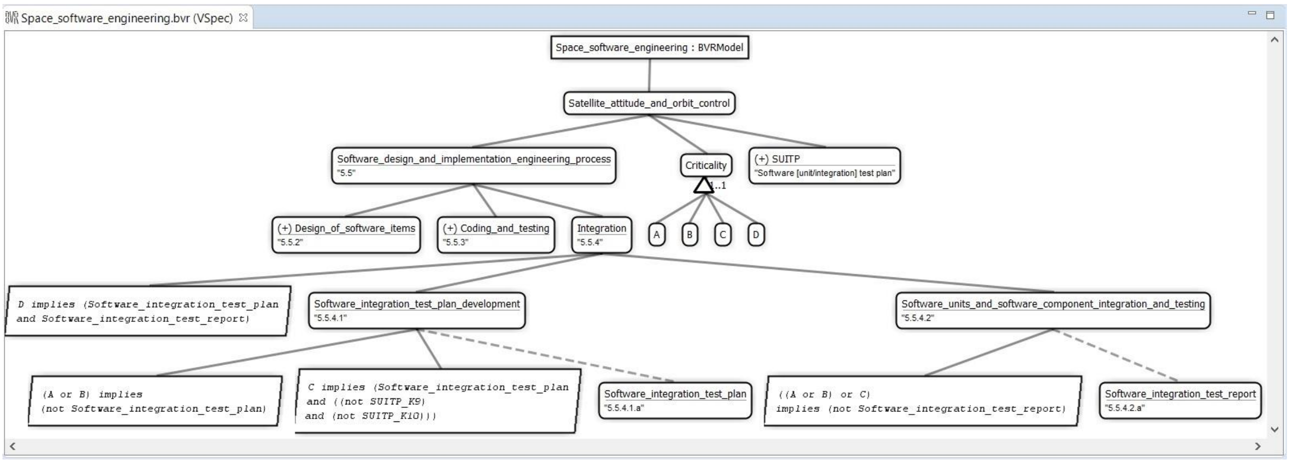

Space_software_engineering.bvr (VSpec)
Space_software_engineering : BVRModel
Satellite_attitude_and_orbit_control
Software_design_and_implementation_engineering_process "5.5"
Criticality
1..1
(+) SUITP "Software [unit/integration] test plan"
(+) Design_of_software_items "5.5.2"
(+) Coding_and_testing "5.5.3"
Integration "5.5.4"
A
B
C
D
D implies (Software_integration_test_plan and Software_integration_test_report)
Software_integration_test_plan_development "5.5.4.1"
Software_units_and_software_component_integration_and_testing "5.5.4.2"
(A or B) implies (not Software_integration_test_plan)
C implies (Software_integration_test_plan and ((not SUITP_K9) and (not SUITP_K10)))
Software_integration_test_plan "5.5.4.1.a"
((A or B) or C) implies (not Software_integration_test_report)
Software_integration_test_report "5.5.4.2.a"

Figure 118. VSpec for ECSS-E-ST-40C (Section 5)

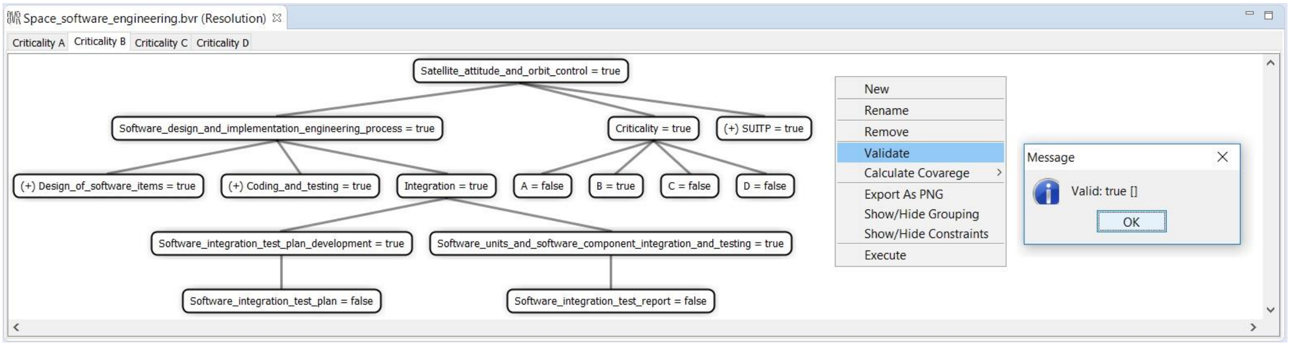

Space_software_engineering.bvr (Resolution)
Criticality A
Criticality B
Criticality C
Criticality D
Satellite_attitude_and_orbit_control = true
Software_design_and_implementation_engineering_process = true
Criticality = true
(+) SUITP = true
(+) Design_of_software_items = true
(+) Coding_and_testing = true
Integration = true
A = false
B = true
C = false
D = false
Software_integration_test_plan_development = true
Software_units_and_software_component_integration_and_testing = true
Software_integration_test_plan = false
Software_integration_test_report = false
New
Rename
Remove
Validate
Calculate Covarege
Export As PNG
Show/Hide Grouping
Show/Hide Constraints
Execute
Message
Valid: true []
OK

Figure 119. Configuration based on the software criticality

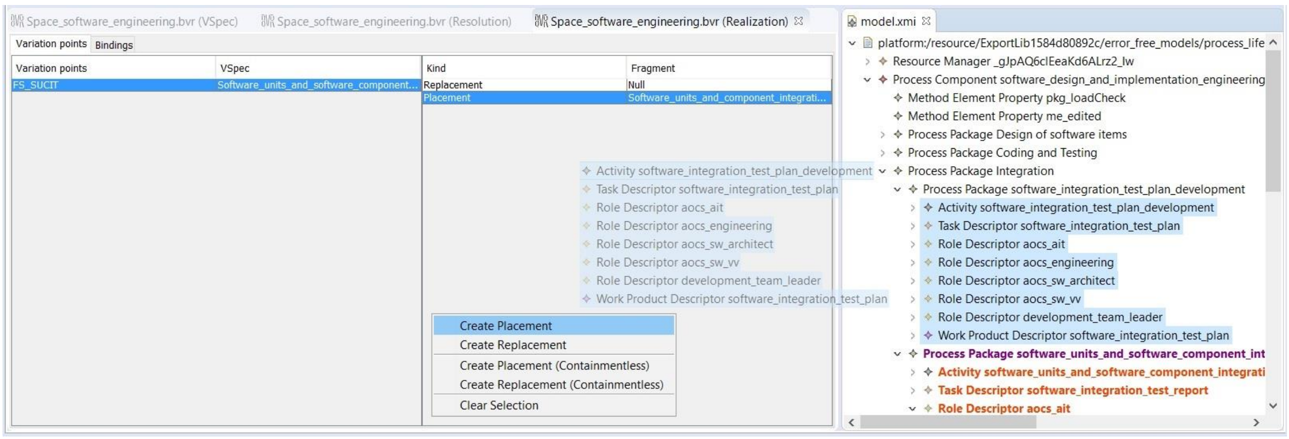

Space_software_engineering.bvr (VSpec)
Space_software_engineering.bvr (Resolution)
Space_software_engineering.bvr (Realization)
Variation points
Bindings
VSpec
Kind
Fragment
FS_SUCIT
Software_units_and_software_component...
Replacement
Null
Placement
Software_units_and_component_integrati...
Activity software_integration_test_plan_development
Task Descriptor software_integration_test_plan
Role Descriptor aocs_ait
Role Descriptor aocs_engineering
Role Descriptor aocs_sw_architect
Role Descriptor aocs_sw_vv
Role Descriptor development_team_leader
Work Product Descriptor software_integration_test_plan
Create Placement
Create Replacement
Create Placement (Containmentless)
Create Replacement (Containmentless)
Clear Selection
model.xmi
platform:/resource/ExportLib1584d80892c/error_free_models/process_life
Resource Manager _gJpAQ6clEeaKd6ALrz2_lw
Process Component software_design_and_implementation_engineering
Method Element Property pkg_loadCheck
Method Element Property me_edited
Process Package Design of software items
Process Package Coding and Testing
Process Package Integration
Process Package software_integration_test_plan_development
Activity software_integration_test_plan_development
Task Descriptor software_integration_test_plan
Role Descriptor aocs_ait
Role Descriptor aocs_engineering
Role Descriptor aocs_sw_architect
Role Descriptor aocs_sw_vv
Role Descriptor development_team_leader
Work Product Descriptor software_integration_test_plan
Process Package software_units_and_software_component_int
Activity software_units_and_software_component_integrati
Task Descriptor software_integration_test_report
Role Descriptor aocs_ait

Figure 120. Realisation of process integration

Table 70. CS11-Cross Intra-domain reuse: US3-Configuration of Process Lines

Realisation Scenario	Configuration of Process Lines
Scope	Process-related reuse
Tool Settings	EPF Composer BVR Tool
Participants	Tool Support: MDH User: OHB, MDH
Activities realised	• A software process with variability is modelled in EPF Composer. Software processes tend to be reused, modified and extended to individual projects.

	• The command "Process Lines -> Seamless Integration between EPF Composer and BVR Tool" is executed for importing the method library and resolving problems with the XMI files for variability management with the BVR tool. • The feature diagram associated to the process line is modelled in the VSpec editor, the process configurations are performed in the resolution editor, and the placement and replacement fragments are defined in the realisation editor. • Multiple resolutions might be defined for the process with variability. To generate the desired process, a specific configuration is executed. • The configured process models are automatically exported back to the EPF Composer.
Usage Decisions	The usage is restricted to a simplified and illustrative example
Expected Results	The achievement of the different configuration of the process model is based on the selection and composition of commonalities and variabilities.
Conclusions	The configuration of the process is achieved. A more complex evaluation is however expected to take place for the third release of the prototype.

3.11.5. US4: Product-Related Reuse via Management of Process Lines

3.11.5.1. STO4 Cross Intra-domain reuse

Another usage scenario is related to the product related reuse. It strives for building a component once and re-use it in different applications or products. The CHESS Modelling Language (CHESSML) is selected to model the systems. Similar to process lines, the integration with BVR tool is achieved for configuring the product lines. We have developed a small GEO product line for the attitude and orbit control, and electrical propulsion subsystems. Small GEO comes in two major configurations: (i) "FAST" with a combination of chemical and electrical propulsion, and (ii) "FLEX" based on only electrical propulsion for both orbit transfer and station-keeping.

Currently, the VSpec model is manually modelled, as shown in Figure 121. The solid lines indicate that the particular feature applies to all configurations, whereas the dashed lines for example "Chemical" propulsion represents a variation point. The automatic generation of variability model (VSpec) would be supported in the next release (prototype P2). In BVR, variability realisation is based on the placements and replacements within the fragment substitutions, as shown in Figure 124.

Therefore, the links between VSpec features and fragment substitutions needed to be established. The execution of variation point is based on the "true" option for the linked VSpec within the specific configuration, as shown in Figure 123.

Otherwise, the variation point will not be considered for tailoring purpose, as illustrated in Figure 122. So, the execution of variations can be managed for the different configurations. The configured product models are automatically exported back to the CHESS tool.

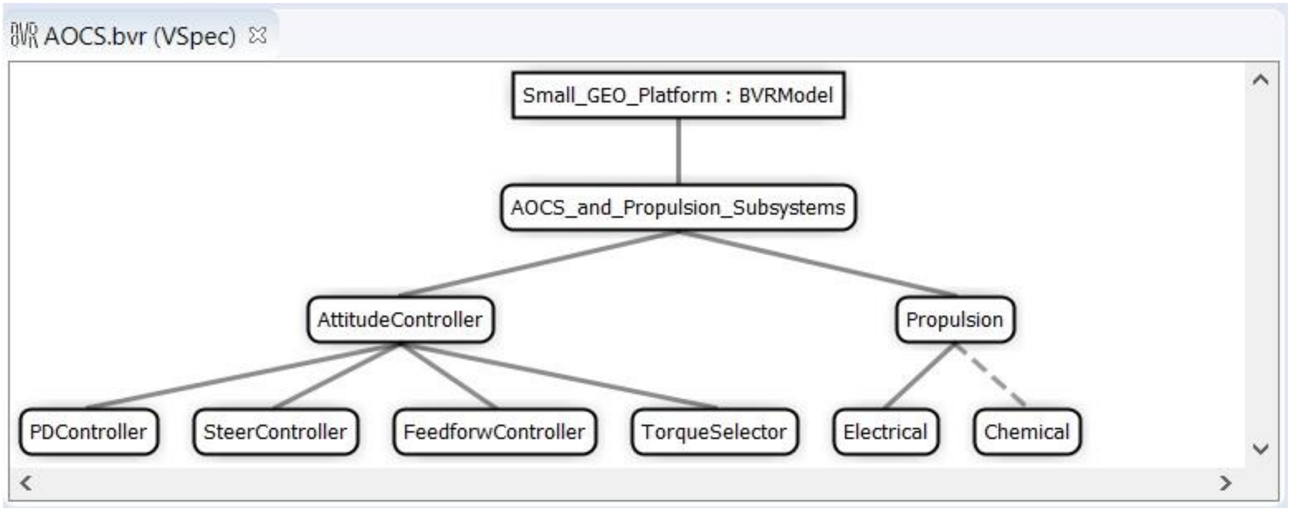


Figure 121. Small GEO VSpec

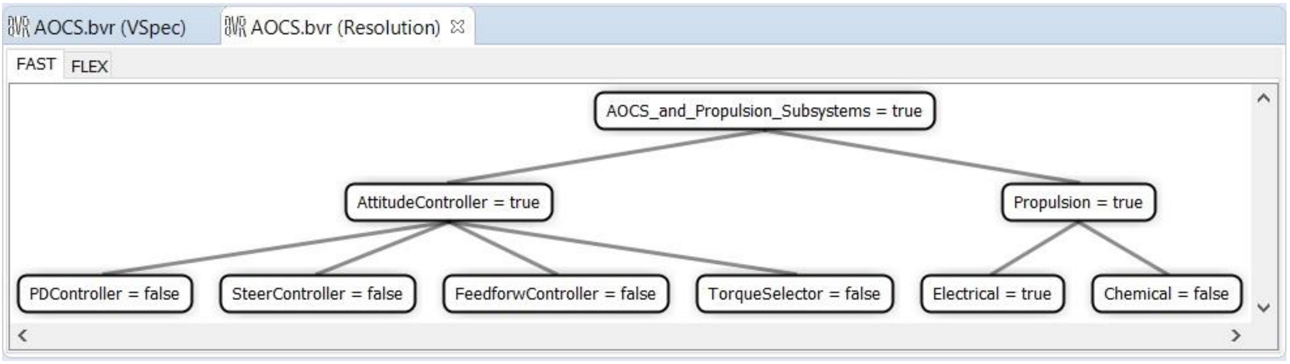


Figure 122. FAST configuration

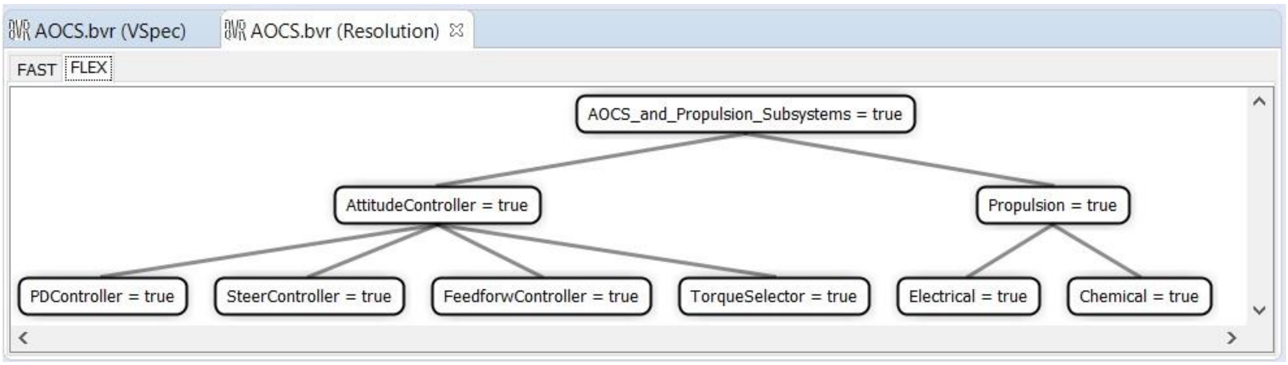
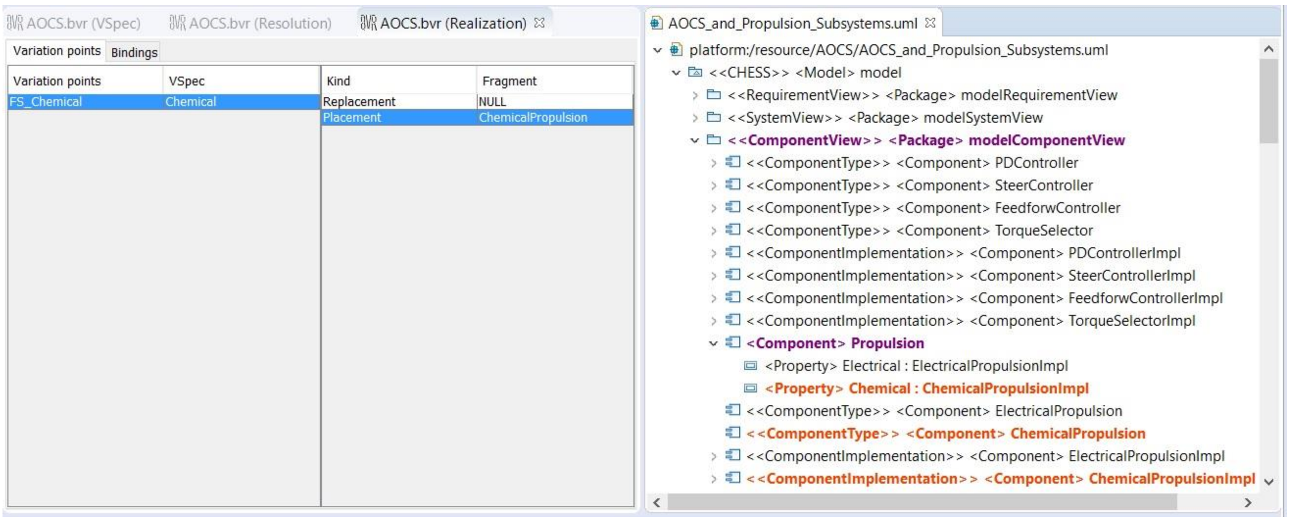


Figure 123. FLEX configuration



AOCS.bvr (VSpec) AOCS.bvr (Resolution) AOCS.bvr (Realization)

Variation points | Bindings

Variation points	VSpec
FS_Chemical	Chemical

Kind	Fragment
Replacement	NULL
Placement	ChemicalPropulsion

AOCS_and_Propulsion_Subsystems.uml

```
platform:/resource/AOCS/AOCS_and_Propulsion_Subsystems.uml
  <<CHESS>> <Model> model
    <<RequirementView>> <Package> modelRequirementView
    <<SystemView>> <Package> modelSystemView
    <<ComponentView>> <Package> modelComponentView
      <<ComponentType>> <Component> PDController
      <<ComponentType>> <Component> SteerController
      <<ComponentType>> <Component> FeedforwController
      <<ComponentType>> <Component> TorqueSelector
      <<ComponentImplementation>> <Component> PDControllerImpl
      <<ComponentImplementation>> <Component> SteerControllerImpl
      <<ComponentImplementation>> <Component> FeedforwControllerImpl
      <<ComponentImplementation>> <Component> TorqueSelectorImpl
      <Component> Propulsion
        <Property> Electrical : ElectricalPropulsionImpl
        <Property> Chemical : ChemicalPropulsionImpl
      <<ComponentType>> <Component> ElectricalPropulsion
      <<ComponentType>> <Component> ChemicalPropulsion
      <<ComponentImplementation>> <Component> ElectricalPropulsionImpl
      <<ComponentImplementation>> <Component> ChemicalPropulsionImpl
```

Figure 124. Realisation of chemical propulsion

Table 71. CS11-Cross Intra-domain reuse: US4-Configuration of Product Lines

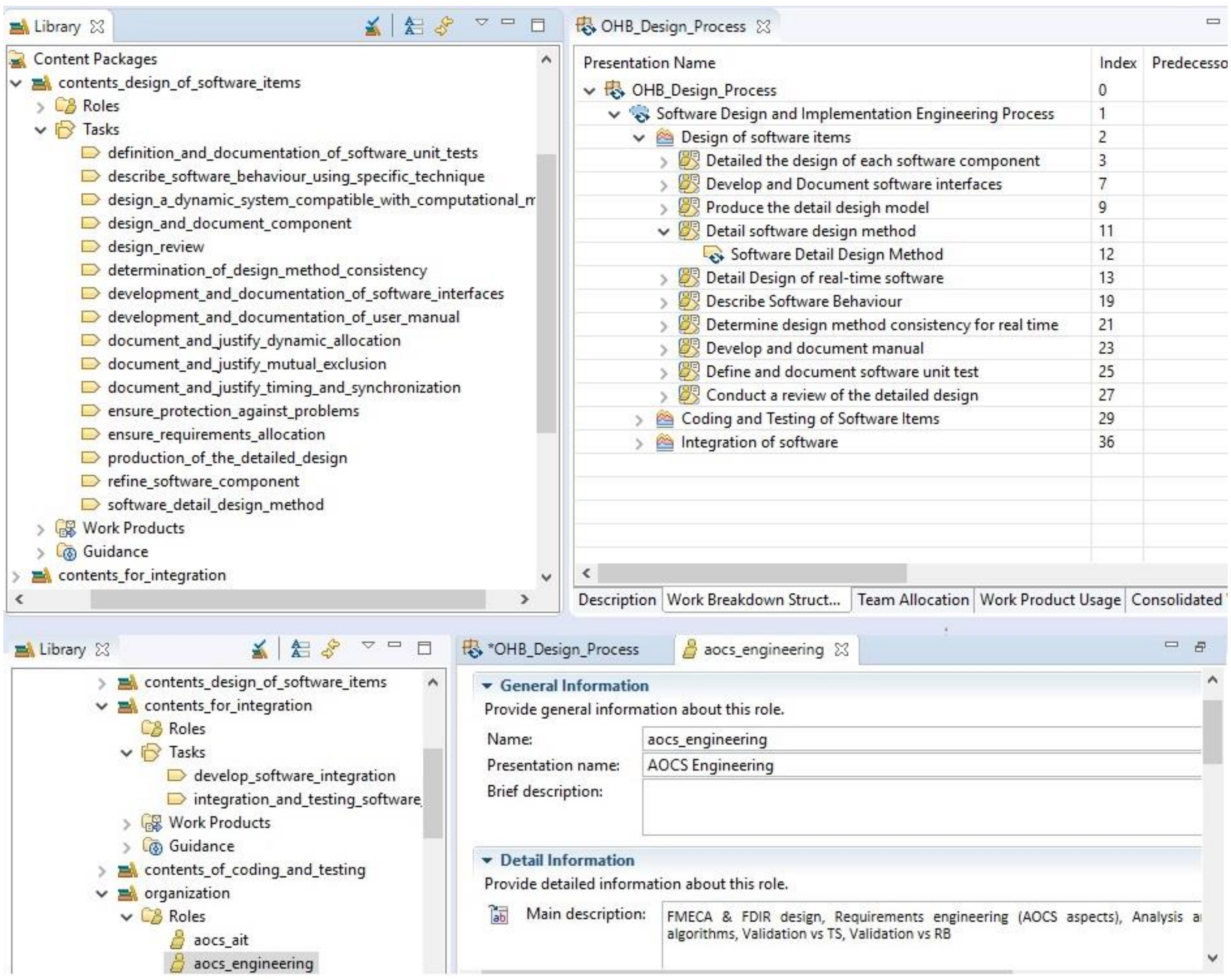
Realisation Scenario	**Configuration of Product Lines**
Scope	Product-related reuse
Tool Settings	CHESS tool BVR Tool
Participants	Tool Support: MDH User: OHB, MDH
Activities realised	• A software product with variability is modelled in CHESS. • The feature diagram associated to the product line is modelled in the VSpec editor, the product configurations are performed in the resolution editor, and the placement and replacement fragments are defined in the realisation editor. • Multiple resolutions might be defined for the product. To generate the desired product model, a specific configuration is executed. • The configured product models are automatically exported back to the CHESS tool.
Usage Decisions	The usage is restricted to a simplified and illustrative example
Expected Results	The achievement of the different configuration of the architectural specification is based on the selection and composition of commonalities and variabilities.
Conclusions	A new configuration of the architectural specification can be obtained as expected. A more complex evaluation is however expected to take place for the third release of the prototype.

3.11.6. US5: Compliance Management (generation of process-based arguments)

3.11.6.1. STO4 Cross Intra-domain reuse

This usage scenario focuses on the systematic reuse of process based engineering and arguments artefacts. The compliance management via the generation of process-based arguments argue/justify that the Software (SW) development process has been planned according to the standard with the aim to reduce effort and cost for creating arguments fragments/assurance activities. In addition, it focuses on the re-use of the assurance assets. Attitude and orbit control subsystem (AOCS) is used in a number of different telecommunication satellite platforms. The software associated with AOCS requires high level of assurance and provisions of evidence that SW fulfils the ECSS standard requirements. To do that some meaningful excerpts of ECSS-E-ST-40C standard that represent plans are identified and modelled in EPF Composer as reusable process content. For example, activities, tasks (e.g., software detailed design method), and a set of work products, roles (e.g., AOCS engineer and his responsibilities) and guidance (such as checklists, tool mentors, guidelines, and examples) are modelled in EPF Composer as shown in Figure 125.

Activities/steps that are involved in the generation of process-based arguments are presented in Table 72. In case the task (represents a commonality) is reused, its corresponding argument can be reused as it is and composed with other argumentation fragments (argumentation pattern). Figure 126 and Figure 127 show the generated process-based argumentation model and diagram in CDO repository. In the current prototype (P1), the basic functionality is performed. For the next release (prototype P2), advanced support for process-based argumentation generation would be provided.



Content Packages
- contents_design_of_software_items
  - Roles
  - Tasks
    - definition_and_documentation_of_software_unit_tests
    - describe_software_behaviour_using_specific_technique
    - design_a_dynamic_system_compatible_with_computational_m
    - design_and_document_component
    - design_review
    - determination_of_design_method_consistency
    - development_and_documentation_of_software_interfaces
    - development_and_documentation_of_user_manual
    - document_and_justify_dynamic_allocation
    - document_and_justify_mutual_exclusion
    - document_and_justify_timing_and_synchronization
    - ensure_protection_against_problems
    - ensure_requirements_allocation
    - production_of_the_detailed_design
    - refine_software_component
    - software_detail_design_method
  - Work Products
  - Guidance
- contents_for_integration

Presentation Name	Index	Predecesso
OHB_Design_Process	0	
Software Design and Implementation Engineering Process	1	
Design of software items	2	
Detailed the design of each software component	3	
Develop and Document software interfaces	7	
Produce the detail design model	9	
Detail software design method	11	
Software Detail Design Method	12	
Detail Design of real-time software	13	
Describe Software Behaviour	19	
Determine design method consistency for real time	21	
Develop and document manual	23	
Define and document software unit test	25	
Conduct a review of the detailed design	27	
Coding and Testing of Software Items	29	
Integration of software	36	

- contents_design_of_software_items
- contents_for_integration
  - Roles
  - Tasks
    - develop_software_integration
    - integration_and_testing_software
  - Work Products
  - Guidance
- contents_of_coding_and_testing
- organization
  - Roles
    - aocs_ait
    - aocs_engineering

General Information

Provide general information about this role.

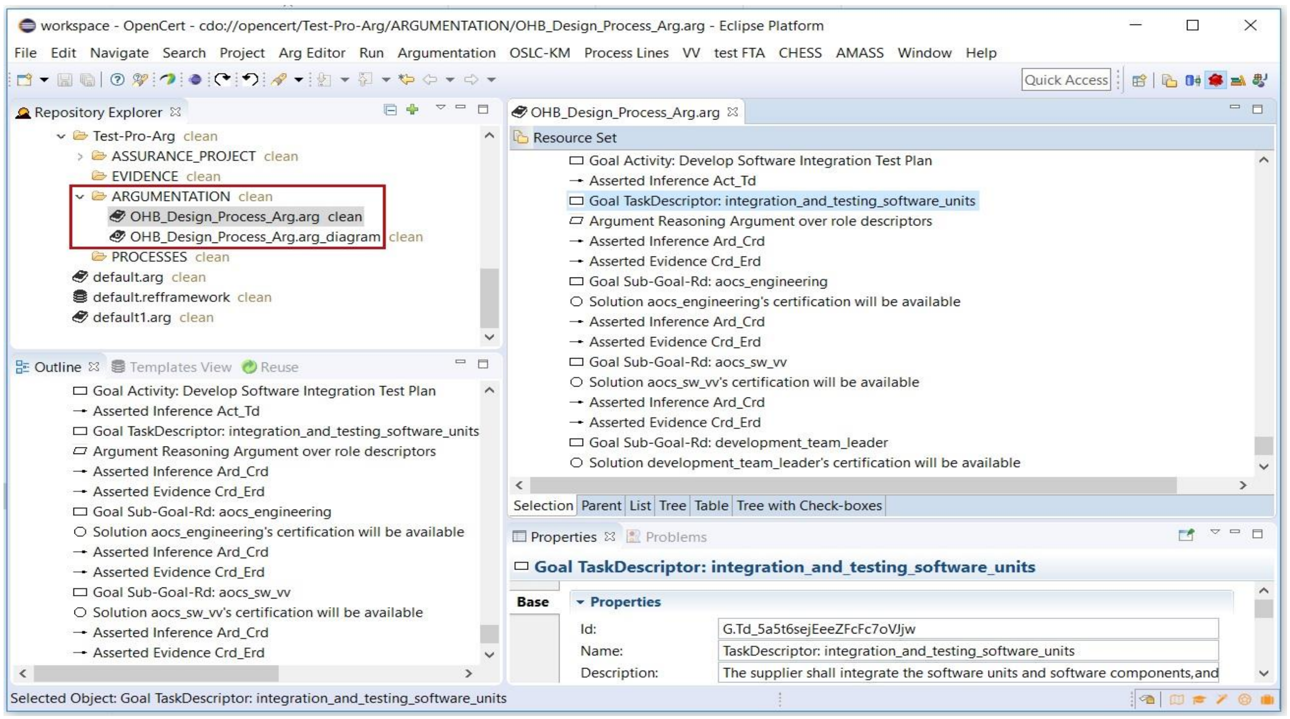
Name:	aocs_engineering
Presentation name:	AOCS Engineering
Brief description:	

Detail Information

Provide detailed information about this role.

Main description: FMECA & FDIR design, Requirements engineering (AOCS aspects), Analysis a algorithms, Validation vs TS, Validation vs RB

Figure 125. ECSS Process Fragments


workspace - OpenCert - cdo://opencert/Test-Pro-Arg/ARGUMENTATION/OHB_Design_Process_Arg.arg - Eclipse Platform

File Edit Navigate Search Project Arg Editor Run Argumentation OSLC-KM Process Lines VV test FTA CHESS AMASS Window Help

Repository Explorer
- Test-Pro-Arg clean
  - ASSURANCE_PROJECT clean
  - EVIDENCE clean
  - ARGUMENTATION clean
    - OHB_Design_Process_Arg.arg clean
    - OHB_Design_Process_Arg.arg_diagram clean
  - PROCESSES clean
- default.arg clean
- default.refframework clean
- default1.arg clean

Outline | Templates View | Reuse
- Goal Activity: Develop Software Integration Test Plan
- Asserted Inference Act_Td
- Goal TaskDescriptor: integration_and_testing_software_units
- Argument Reasoning Argument over role descriptors
- Asserted Inference Ard_Crd
- Asserted Evidence Crd_Erd
- Goal Sub-Goal-Rd: aocs_engineering
- Solution aocs_engineering's certification will be available
- Asserted Inference Ard_Crd
- Asserted Evidence Crd_Erd
- Goal Sub-Goal-Rd: aocs_sw_vv
- Solution aocs_sw_vv's certification will be available
- Asserted Inference Ard_Crd
- Asserted Evidence Crd_Erd

OHB_Design_Process_Arg.arg
Resource Set
- Goal Activity: Develop Software Integration Test Plan
- Asserted Inference Act_Td
- Goal TaskDescriptor: integration_and_testing_software_units
- Argument Reasoning Argument over role descriptors
- Asserted Inference Ard_Crd
- Asserted Evidence Crd_Erd
- Goal Sub-Goal-Rd: aocs_engineering
- Solution aocs_engineering's certification will be available
- Asserted Inference Ard_Crd
- Asserted Evidence Crd_Erd
- Goal Sub-Goal-Rd: aocs_sw_vv
- Solution aocs_sw_vv's certification will be available
- Asserted Inference Ard_Crd
- Asserted Evidence Crd_Erd
- Goal Sub-Goal-Rd: development_team_leader
- Solution development_team_leader's certification will be available

Selection | Parent | List | Tree | Table | Tree with Check-boxes

Properties | Problems

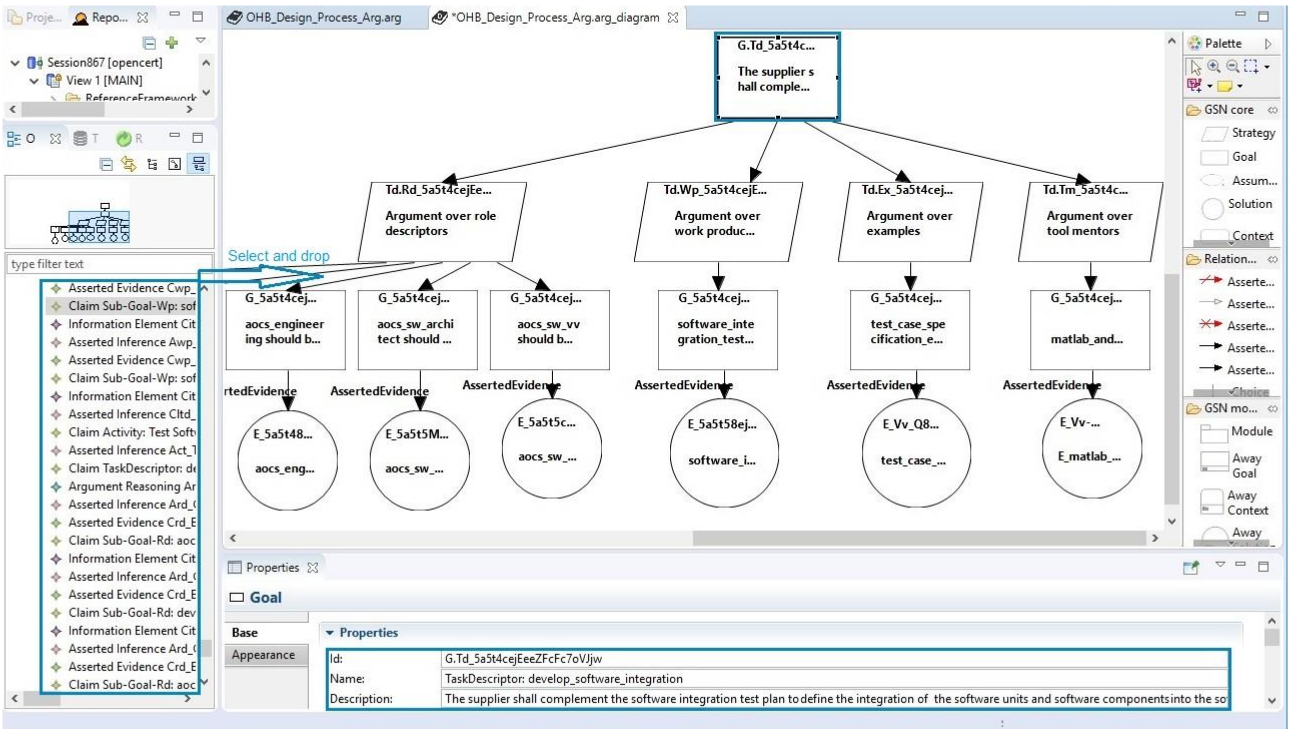
Goal TaskDescriptor: integration_and_testing_software_units

Base | Properties

Id:	G.Td_5a5t6sejEeeZFcFc7oVJjw
Name:	TaskDescriptor: integration_and_testing_software_units
Description:	The supplier shall integrate the software units and software components,and

Selected Object: Goal TaskDescriptor: integration_and_testing_software_units

Figure 126. Model Generated Argumentation



Asserted Evidence Cwp_
Claim Sub-Goal-Wp: sof
Information Element Cit
Asserted Inference Awp_
Asserted Evidence Cwp_
Claim Sub-Goal-Wp: sof
Information Element Cit
Asserted Inference Cltd_
Claim Activity: Test Soft
Asserted Inference Act_T
Claim TaskDescriptor: de
Argument Reasoning Ar
Asserted Inference Ard_(
Asserted Evidence Crd_E
Claim Sub-Goal-Rd: aoc
Information Element Cit
Asserted Inference Ard_(
Asserted Evidence Crd_E
Claim Sub-Goal-Rd: dev
Information Element Cit
Asserted Inference Ard_(
Asserted Evidence Crd_E
Claim Sub-Goal-Rd: aoc

Properties

Goal

Base | Appearance | Properties

Id:	G.Td_5a5t4cejEeeZFcFc7oVJjw
Name:	TaskDescriptor: develop_software_integration
Description:	The supplier shall complement the software integration test plan to define the integration of the software units and software components into the so

Figure 127. Process-based Argumentation Diagram

Table 72. CS11-Compliance Management: US5-Process-based Arguments

Realisation Scenario	**Process-based Arguments**
Scope	Aims to show that the system has been developed in compliance with the process defined in the standard and justify the safety-related decisions as well as reuse of process-based arguments fragments.
Tool Settings	EPF Composer

	OpenCert
Participants	Tool Support: MDH Tool User: OHB, MDH
Activities realised	• Identify and model some meaningful excerpts of ECSS-E-ST-40C that represent plans. • Export process model (i.e. a delivery process) in XML format by using “export one or more method plug-ins” option. • Select “EPF Composer -> Argumentation” option from OpenCert. • Browse exported XML file. • Generate process-based arguments. • Process-based safety arguments (model and diagram) are stored in the corresponding destination assurance case in the CDO repository.
Usage Decisions	The usage is restricted to a simplified and illustrative example
Expected Results	Automatic generation of process-based argument.
Conclusions	The generation took place as expected. The generated argument is stored in CDO and can be reused if the corresponding process element is reused. A more complex evaluation is however expected to take place for the third release of the prototype.

3.11.7. Coverage of AMASS Prototype P1 Architecture

Table 73 illustrates the implemented functionalities during this second iteration within the Case Study 11.

Table 73. AMASS Prototype P1 Coverage by CS11

STO	AMASS Functionalities	Tools
Architecture-Driven Assurance	System Component Specification	CHESS
	System Architecture Modelling for Assurance	CHESS with variability
	Architectural Patterns for Assurance	-
	Contract-based Design for Assurance	-
	Activities supporting Assurance Case	V&V integration of RapiCov
Multi-Concern Assurance	Assurance Case Specification	-
	Dependability Assurance	-
	System Dependability Co-Analysis/Co-Assessment	ConcertoFLA
	Contract-Based Multi-concern Assurance	-
Seamless Interoperability	Evidence Management	-
	Tool Integration Management	V&V integration of RapiCov
	Collaborative Work Management	-
	Tool Quality Assessment and Characterization	-
Cross Intra-Domain Reuse	Compliance Management	EPF-C
	Reuse Assistant	-
	Process-related reuse via management of variability at process level	EPF Composer BVR VSpec, Resolution, and Realisation editors
	Product-related reuse via management of variability at product level	EPF Composer BVR Tool, Small GEO Vspec
	Automatic generation of process-based arguments	OpenCert

	Automatic generation of product-based arguments	OpenCert

3.11.8. Conclusions

At this stage, the main benefits and potential improvements are identified in Table 74.

Table 74. Benefits and potential improvements for CS11

Artefact	Achievement/Benefits	Improvements/Recommendations
V&V Tools RapiCov	• Efficiency: 40% faster (mean value) • Quality: Support available and Qualified DO-178(A)B/C (A only needed for manned flight). • Cost: Original solution open source at no cost. RapiCov not open source. • Seamless integration: Small impact on environment, seamless integrated.	As long as the software criticality is category B, qualified tools are not required. When qualified tools are required it is recommended to switch to not only RapiCov but also other Rapi Tools to ensure added value to the tool chain.
ECSS compliance with EPF Composer	• Mapping of ECSS requirement to existing processes to analyse possible compliance gaps. • Ability to generate compliance metrics with argumentation.	• Easy to import standard requirements • Hard to maintain information in the tool.

Regarding the evaluation of the other functionalities:

- Process-related reuse via management of variability at process level
- Product-related reuse via management of variability at product level
- Automatic generation of process-based argument

it is premature to draw conclusions on benefits for possible adoption. An additional iteration of evaluation will be conducted on more significant scenarios, especially regarding the product line.

4. Coverage of the AMASS prototype P1 functionalities by the Case Studies

This section shows the coverage that has been achieved by the AMASS Case Studies in the specific STOs for the second iteration of the AMASS platform (Prototype P1).

It is worth mentioning that the AMASS P1 functionalities have broadly been tackled by the Case Studies. After high coverage has been achieved during the second iteration, the objectives for the third iteration will be twofold: on the one hand, the case studies will take profit of those functionalities which were not finalised by the time this deliverable has been submitted (e.g. *Architectural Patterns for Assurance*), and, on the other hand, the already existing functionalities will be further used by those Case Studies which have not deployed them yet.

4.1. Architecture-Driven Assurance (STO1)

Table 75. Coverage of Architecture-Driven Assurance (STO1) for AMASS Prototype P1

	Architecture-Driven Assurance (STO1)				
Case Study	System Component Specification	System Architecture Modelling for Assurance	Architectural Patterns for Assurance	Contract-based Design for Assurance	Activities supporting Assurance Case
CS1	MORETO	MORETO	MORETO	-	-
CS2	Enterprise Architecture	Enterprise Architecture	-	-	-
CS3	SAVONA/CHESS	SAVONA/CHESS	-	SAVONA/CHESS	SAVONA KM Functional Verification by Simulink and AMT 2.0 monitors (ongoing) Medini Analyze Safety V&V CHESS/SAVONA-Sabotage (ongoing)
CS4	CHESS	CHESS	-	CHESS/OCRA	CHESS, OCRA, xSAP, nuXmv
CS5	CHESS	CHESS	-	CHESS/OCRA	OCRA
CS6	Papyrus/SysML	-	-	Requirements formalisation (external)	Requirements early validation, Functional Early Verification, model-based safety analysis (external tools)
CS7	CHESS	CHESS	-	CHESS+OCRA	DIVINE, NuSMV, nuXmv, Looney, Acacia+ V&V Manager
CS8	Papyrus/SysML	Papyrus/SysML	-	-	-

CS9	CHESS	CHESS	-	CHESS+OCRA	Papyrus, OCRA and xSAP
CS10	CHESS	CHESS	-	CHESS+OCRA	V&V Failed, FMEA mentioned
CS11	CHESS	CHESS with variability	-	-	V&V integration of RapiCov

4.2. Multi-Concern Assurance (STO2)

Table 76. Coverage of Multi-concern Assurance (STO2) for AMASS Prototype P1

	Multi-concern Assurance (STO2)			
Case Study	Assurance Case Specification	Dependability Assurance Modelling	System Dependability Co-Analysis/Co-Assessment	Contract-based Multi-concern assurance
CS1	OpenCert (Safety and Security Assurance Case)	OpenCert	FMVEA	-
CS2	-	-	-	-
CS3	OpenCert	OpenCert (safety and security case)	FMVEA, EPF-C+BVR (ISO 26262 for functional safety and SAE J3061)	-
CS4	OpenCert	-	Concerto FLA	-
CS5	OpenCert	-	Papyrus SSE	-
CS6	OpenCert	-	-	-
CS7	-	-	EPF-C+ OpenCert	-
CS8	OpenCert	OpenCert	OpenCert	-
CS9	-	-	-	-
CS10	OpenCert	OpenCert	OpenCert	-
CS11	-	-	ConcertoFLA	-

4.3. Seamless Interoperability (STO3)

Table 77. Coverage of Seamless Interoperability (STO3) for AMASS Prototype P1

	Seamless Interoperability (STO3)			
Case Study	Evidence Management	Tool Integration Management	Collaborative Work Management	Tool Quality Assessment and Characterization
CS1	OpenCert	-	-	-
CS2	SVN	-	-	-
CS3	-	SQA, KM via OSLC	-	-

CS4	-	V&V Manager and OSLC Automation V&V Tool Integration	-	-
CS5	-	Generation of Frama-C asserted C code from B models, Atelier B formal IDE including target specific code generator, Frama-C, V&V Tool Integration	-	-
CS6	-	-	-	-
CS7	OpenCert	V&V Manager and OSLC Automation V&V Tool Integration	-	-
CS8	OpenCert	-	-	-
CS9	OpenCert	-	-	-
CS10	OpenCert	-	-	-
CS11	-	V&V integration of RapiCov	-	-

4.4. Cross Intra-Domain Reuse (STO4)

Table 78. Coverage of Cross Intra-Domain Reuse (STO4) for AMASS Prototype P1

	Cross Intra-Domain Reuse (STO4)					
Case Study	Compliance Management	Reuse Assistant	Process-related reuse via management of variability at process level	Product-related reuse via management of variability at product level	Automatic generation of process-based arguments	Automatic generation of product-based arguments
CS1	OpenCert	OpenCert	-	-	-	-
CS2	-	-	-	-	-	-
CS3	EPF-C Semantic modelling of ISO 26262	-	EPF-Composer and BVR: ISO 26262 for functional safety and SAE J3061	-	OpenCert	OpenCert
CS4	OpenCert	-	-	-	-	-
CS5	-	-	-	-	-	-
CS6	Modelling of CENELEC EN 50126, EN 50128, EN	-	-	-	OpenCert	-

	50129.					
CS7	OpenCert/Knowledge Manager (semantics mapping)	-	-	-	OpenCert	OpenCert
CS8	EPF-C. Import to OpenCert	-	-	-	-	-
CS9	OpenCert	-	-	-	-	-
CS10	OpenCert	-	-	-	-	-
CS11	EPF-C	-	EPF Composer BVR VSpec, Resolution, and Realisation editors	EPF Composer BVR Tool, Small GEO Vspec	OpenCert	OpenCert

5. Conclusions

This document presents the utilisation of the different AMASS functionalities addressing the usage scenarios proposed in D1.1 [1] and based on the data collection done in D1.2 [2]. Moreover, this deliverable elaborates upon the work done in D1.4 [4]. For the evaluation of the AMASS Prototype P1, some new features have been added to expand the usage of the AMASS platform.

The objective during this second iteration is twofold: on the one hand, the already existing Prototype P1 functionalities have been strengthen based on the feedback provided during the first iteration. On the other hand, new functionalities have been added. Special reference should be made of the action on making the Core Prototype much more stable.

Due to the tight timeframe between the finalisation of the Prototype P1 and the deadline for this deliverable, a proper measurement of the metrics has not been possible. However, it has to be mentioned that the work on this issue is ongoing. In brief, metrics will be provided when the final toolset is available as a result of the corresponding analysis. Therefore, benchmarking will be one of the main aspects for evolutions of this deliverable.

Meanwhile the AMASS Prototype P2 is being developed, industrial partners will continue the evaluation of the already available functionalities. The Benchmarking studio and metrics evaluation will be done in parallel as well, as it is specified in [3].

In brief, this deliverable offers an overview of the AMASS platform functionalities and the different approaches of the problems solved with them in terms of different application domains. All this is conducted from an industrial perspective. It is a keystone for the AMASS solutions , to get closer to the industrial sector. Thus, the feedback from the industrial partners will be considered as part of the AMASS ongoing activities to improve the quality of the final solution (P2).

Appendix A: Abbreviations and Definitions

ACC	Adaptive Cruise Control
ACSL	A C Specification Language
ADAS	Advanced Driver Assistance System
ADC	Automated Driving Function
AL	Assurance Level
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
ASIL	Automotive Safety Integrity Level
ASW	Application Software
BDD	Behaviour Driven development
BSW	Boot Software
BVR	Base Variability Resolution
BXML	XML-based format for B models
CA	Consortium Agreement
CAL	Cybersecurity Assurance Level
CDO	Connected Data Objects
CENELEC	Comité Européen de Normalisation Électrotechnique (European Committee for Electrotechnical Standardization)
CHESS	Composition with Guarantees for High-integrity Embedded Software Components Assembly
CNS/ATM	Communication Navigation Surveillance / Air Traffic Management
COPPILOT	System to Open and Close the Platform Screen Doors DPAS - Détecteur de Passage (Crossing Detection Equipment)
CPS	Cyber Physical System
CPU	Central Processing Unit
CS	Case Study
CSU	Computer Software Unit
CSV	Comma-Separated Values
DME	Distance Measuring Equipment
DRF	Détecteur de Roue Fer (Steel Wheel Presence Sensor)
DSP	Digital Signal processor
EA	Enterprise Architect
ECSS	European Cooperation for Space Standardization
EDSA	Embedded Device Security Assurance
EooC	Elements out of Context
EPF	Eclipse Process Framework
EPF-C	Eclipse Process Framework-Composer
ERTMS	European Rail Traffic Management System
ESA	European Space Agency
FLA	Failure Logic Analysis
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
FSA	Functional Safety Assessment
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
GSN	Goal Structuring Notation
GPP	General-purpose pre-processor
GUI	Graphical User Interface
HARA	Hazard Analysis and Risk Assessment
IACS	Industrial and Automation Control System
IBD	Internal Block Diagram
ICM	Instrument Control Module
IDE	Integrated Development Editor
IEC	International Electrotechnical Commission
INCOSE	International Council on Software and Systems Engineering
ISO	International Organization for Standardization
KM	Knowledge Manager
MMI	Multi-Modal Interactions
MPSoC	MicroProcesor System on Chip
NoC	Network-on-chip
NVD	National Vulnerability Database
OCRA	Othello Contracts Refinement Analysis
OEU	OLCI Electronic Unit
OLCI	Ocean & Land Colour Instrument
OSLC	Open Services for Lifecycle Collaboration
PSD	Platform Screen Door
PUS	Packet Utilization Standard
RFP	Request for Proposal
RTCA	Radio Technical Commission for Aeronautics
RTU	Remote Terminal Unit
SAE	Society of Automotive Engineers
SCADA	Supervisory Control And Data Acquisition
SCAR	Shite Compared to autorun
SIL	Safety Integrity Level
SiSoPL	Security-informed Safety-oriented Process Lines
SL	Security Level
SoC	System-On-Chip
SoS	System of Systems
SQA	System Quality Anayzer
SSDP	Scalable Sensor Data Processor Breadboard
SSE	Safety and Security Engineering
STO	Scientific and Technical Objective
SVN	Subversion
SW	Software
SWD	Software Design
SWV	Software Verification & Validation
SysML	System Modelling Language
TARA	Threat Analysis and Risk Assessment
TC	TeleComand
TM	Telemetry
TRL	Technology Readiness Levels
TSIM	Simulator tool
UL	Underwriters Laboratories
UML	Unified Modelling Language
US	Usage Scenario

V&V	Verification and validation
VAM	Video Acquisition Module
WEFACT	Workflow Engine for Analysis, Certification and Test
XML	eXtensible Markup Language
xSAP	eXtended Safety Assessment Platform

Appendix B: MORETO

The Model-based Security Requirements Management Tool (MORETO) is a SysML-based tool for security analysis, requirements, and architecture design. The toolbox starts with four diagrams for the modelling process and security analysis process (i.e. Network Topology Diagram, Internal Block Diagram, Dataflow Diagram, and Requirements Diagram for IEC 62443-4-2) as is shown in Figure 128.

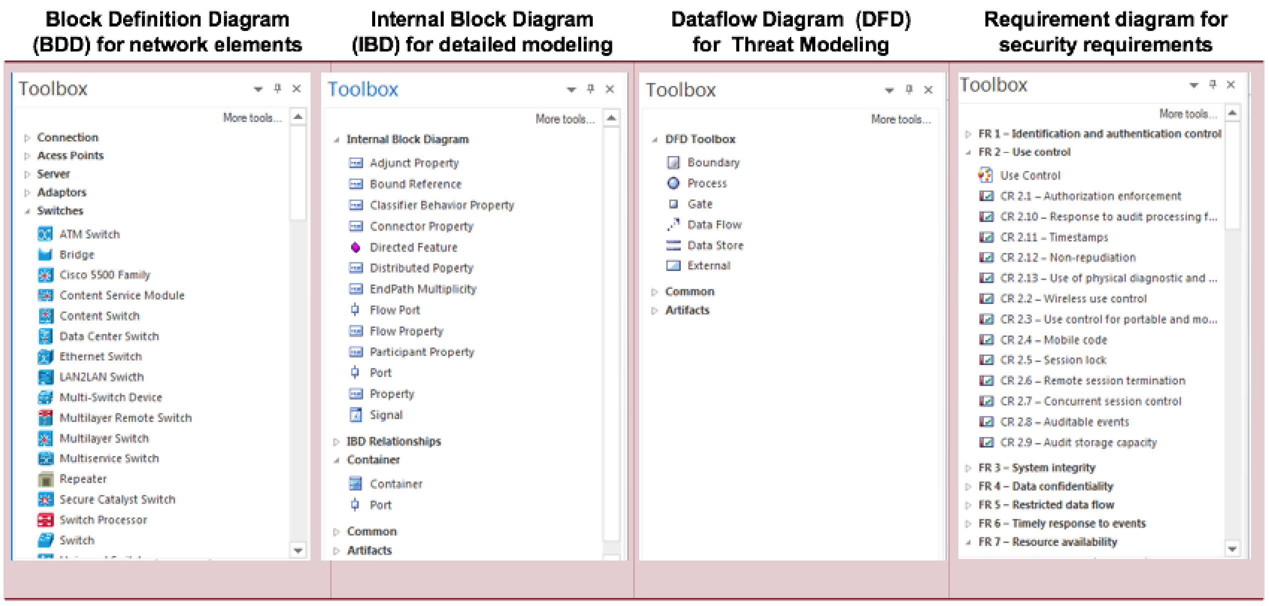


Figure 128. MORETO four diagrams

Recently, two additional diagrams have been integrated with MORETO to expand the functionality of the MORETO toolbox (i.e. RTU-Remote Terminal Unit Diagram, and IEEE 1686 Requirements Diagram) (see Figure 129).

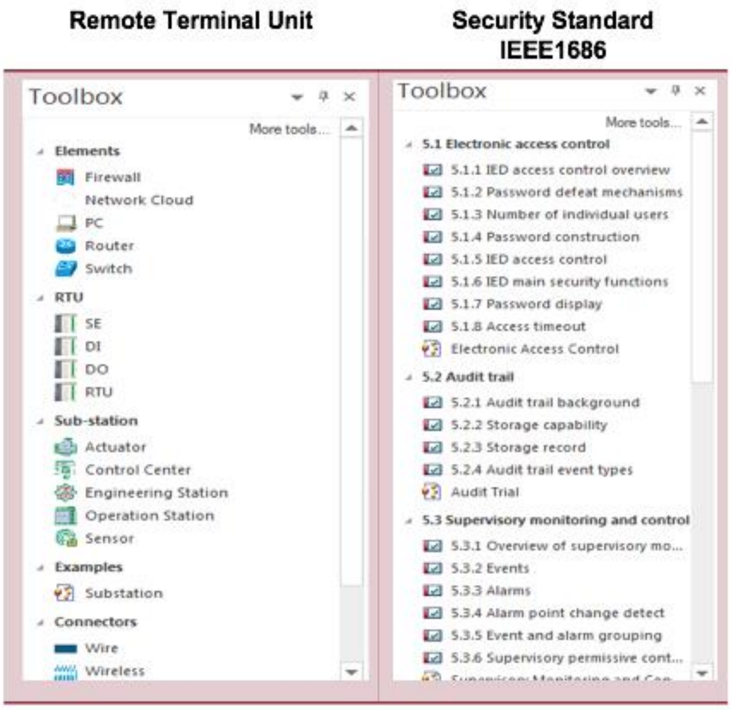

Remote Terminal Unit
Security Standard IEEE1686
Toolbox
More tools...
Elements
Firewall
Network Cloud
PC
Router
Switch
RTU
SE
DI
DO
RTU
Sub-station
Actuator
Control Center
Engineering Station
Operation Station
Sensor
Examples
Substation
Connectors
Wire
Wireless
Toolbox
More tools...
5.1 Electronic access control
5.1.1 IED access control overview
5.1.2 Password defeat mechanisms
5.1.3 Number of individual users
5.1.4 Password construction
5.1.5 IED access control
5.1.6 IED main security functions
5.1.7 Password display
5.1.8 Access timeout
Electronic Access Control
5.2 Audit trail
5.2.1 Audit trail background
5.2.2 Storage capability
5.2.3 Storage record
5.2.4 Audit trail event types
Audit Trial
5.3 Supervisory monitoring and control
5.3.1 Overview of supervisory mo...
5.3.2 Events
5.3.3 Alarms
5.3.4 Alarm point change detect
5.3.5 Event and alarm grouping
5.3.6 Supervisory permissive cont...

Figure 129. MORETO new Diagrams

Figure 130 shows the workflow model of all MORETO working phases. MORETO has six different stages that can be followed by the user to define his/her system modelling components, scan the security vulnerabilities in the system, and cover the detected security weaknesses by suitable security standards.

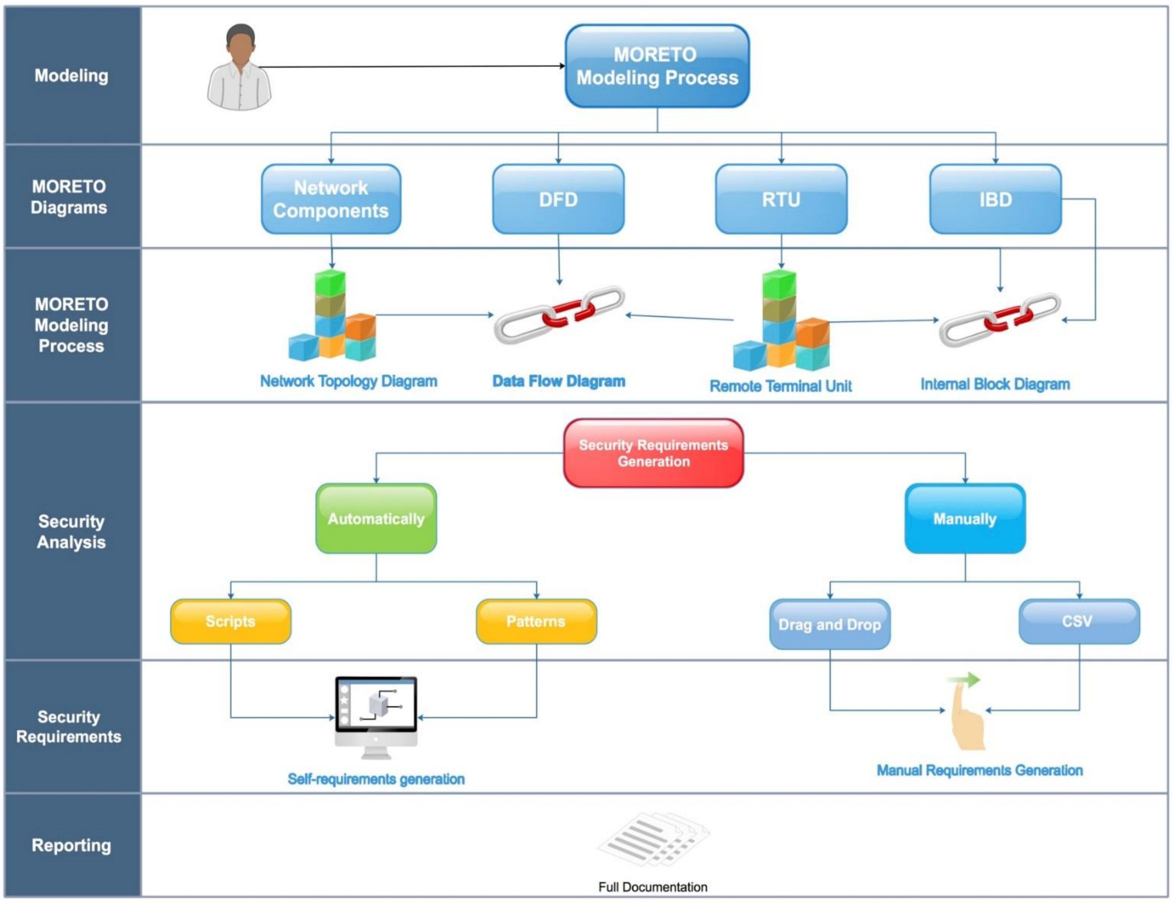


Figure 130. Workflow of the MORETO Toolbox

The user chooses the appropriate diagram which fits into his/her requirements for describing the system components. The security analysis process constitutes an integral part of MORETO, which is able to scan all elements of the user's model and detect security flaws. Afterwards, MORETO covers these flaws with proper security requirements. Finally, MORETO is collecting all the details of the security analysis and requirements in a neat report.

For the 3rd iteration (P2) the use of the FMVEA tool is planned. An integrative approach could be used to determine the security level by means of the FMVEA and use this result in MORETO for selecting the appropriate security requirements. The FMVEA shall be contained and described in D4.6 *Prototype for multi-concern assurance (b)*.

MORETO Layers

The modelling process in MORETO can be done in three different layers. This feature gives the MORETO user the flexibility to describe his/her system components in detail in a traceable way to follow the system components in a simple way.

Top Layer

The top layer design reflects the external view of the modelling process. In other words, MORETO tries to divide the modelling process into sub-layers of which each one has more details than other layers. The external layer defines a general schematic form of network parts or components in the real world, such as an industrial apparatus. Figure 131 depicts a network topology which was created by MORETO toolbox. MORETO gives the user different network elements to construct his/her needs.

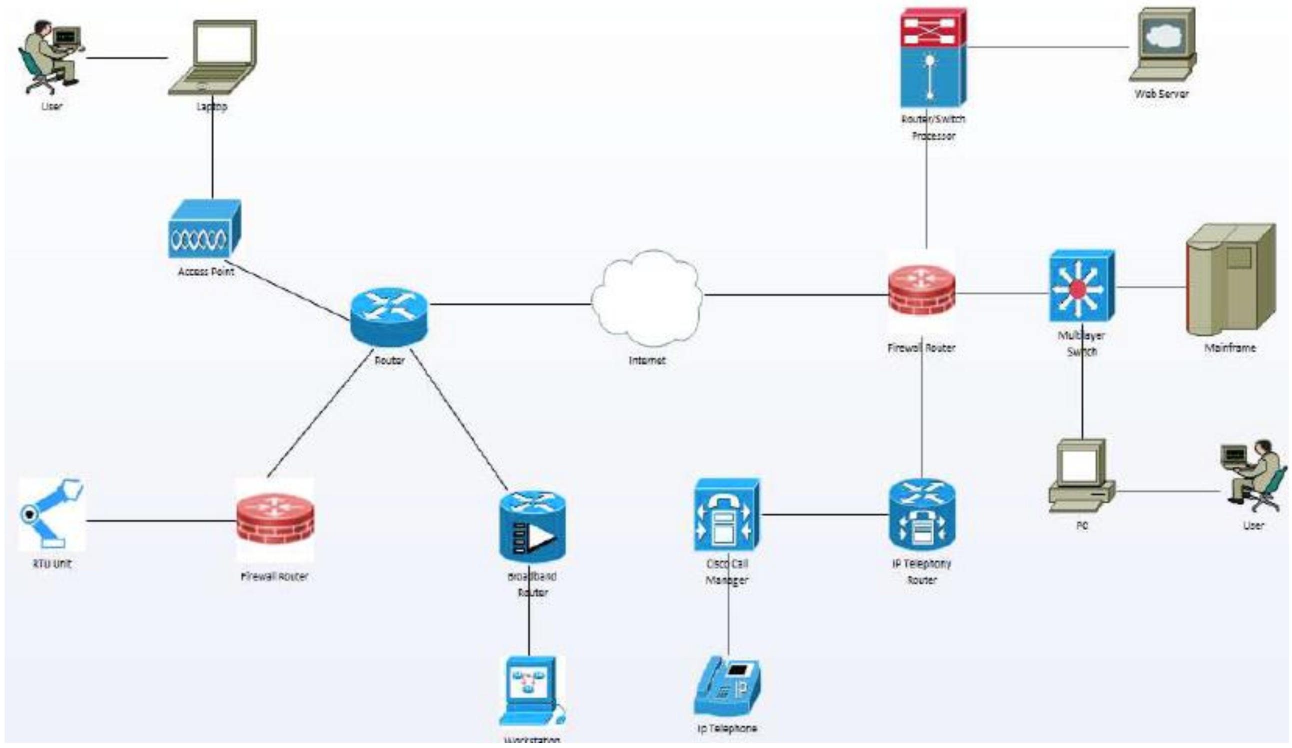


Figure 131. MORETO External Layer

<u>Intermediate Layer</u>

The user defines the interactions between different components of the network topology using internal block diagram. This layer defines the internal components of each unit which is provided in the external layer. In the example, the internal design of the RTU unit was designed. Enterprise Architect (EA) provides a unique service which makes the navigation process between different layers possible. By double clicking on the RTU unit, EA would generate internal connections between the RTU unit on the external level with the intermediate layer. Figure 132 shows the internal design of the RTU unit.

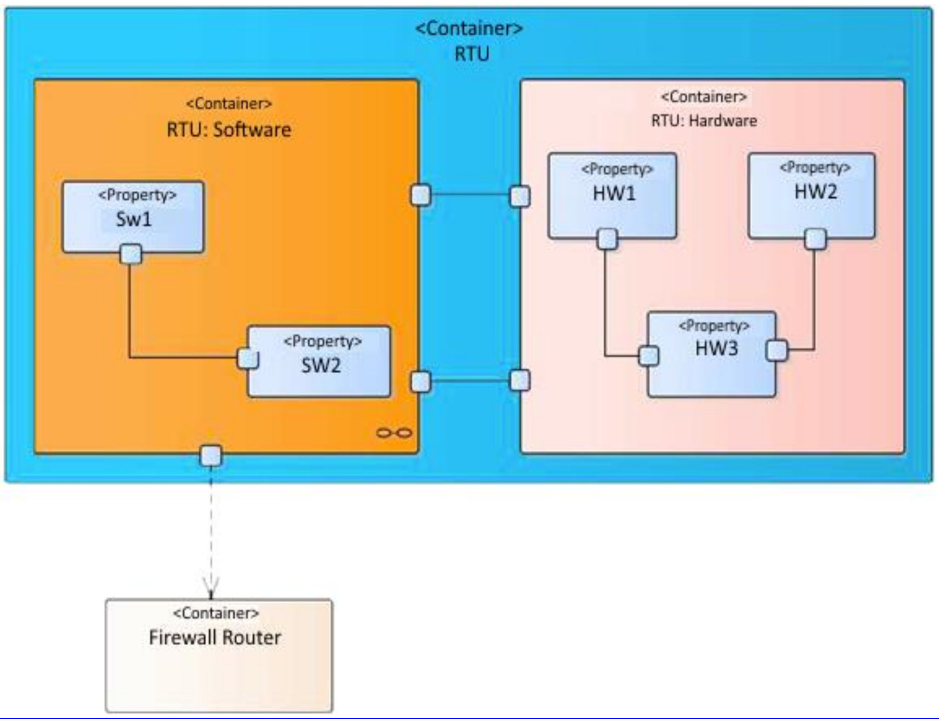


Figure 132. MORETO Intermediate Layer

As defined before, the intermediate layer aims to define the internal connections and components of a particular external unit. However, there are some details not defined yet in this layer, so MORETO gives the user another area to add new details of the internal design of components by defining another sub-layer, which is called internal layer. For example, the above figure describes a simple graphical notation about the internal design of the RTU unit.

Internal Layer

This layer defines the internal structure or decomposition of a block of a system into its sub parts or subsystems. The internal layer defines more details about the internal components and connections of a particular component. In this example, the software component was chosen in order to define more details about its internal structure. Figure 133 depicts the internal connections and interactions between components of the SW unit.

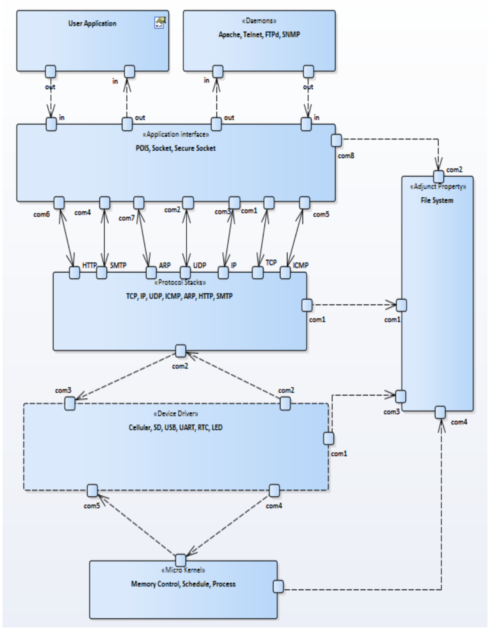


Figure 133. MORETO Internal Design

Security Analysis in MORETO

Security requirements generation features one of the unique services provided by MORETO toolbox to cover security gaps in the user's model. The security generation process could be manual or automatic. The following Figure 134 shows a graphical representation of the generation process of security requirements.

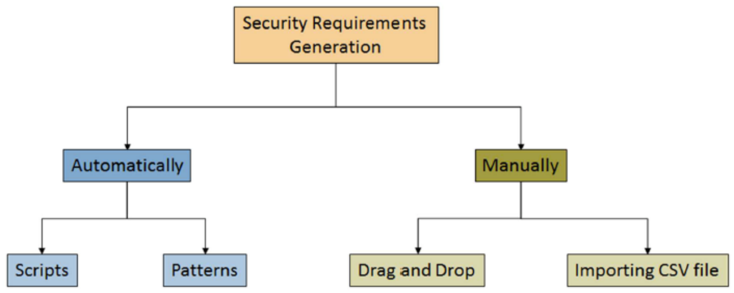


Figure 134. MORETO Hierarchy for generation of Security requirements

Manual Mode

The user uses this mode to create or generate his/her security requirements. This process can be done either by drag and drop or by importing a CSV file.

Drag and Drop

This feature is standard of system modelling software, so the user could specify the security requirements from the toolbox and then drag and drop into the workspace. Figure 135 shows a simple example about how to create security requirements manually. The user would navigate into the security requirements on the left-hand-side and then drag and drop the needed security requirements into the workspace.

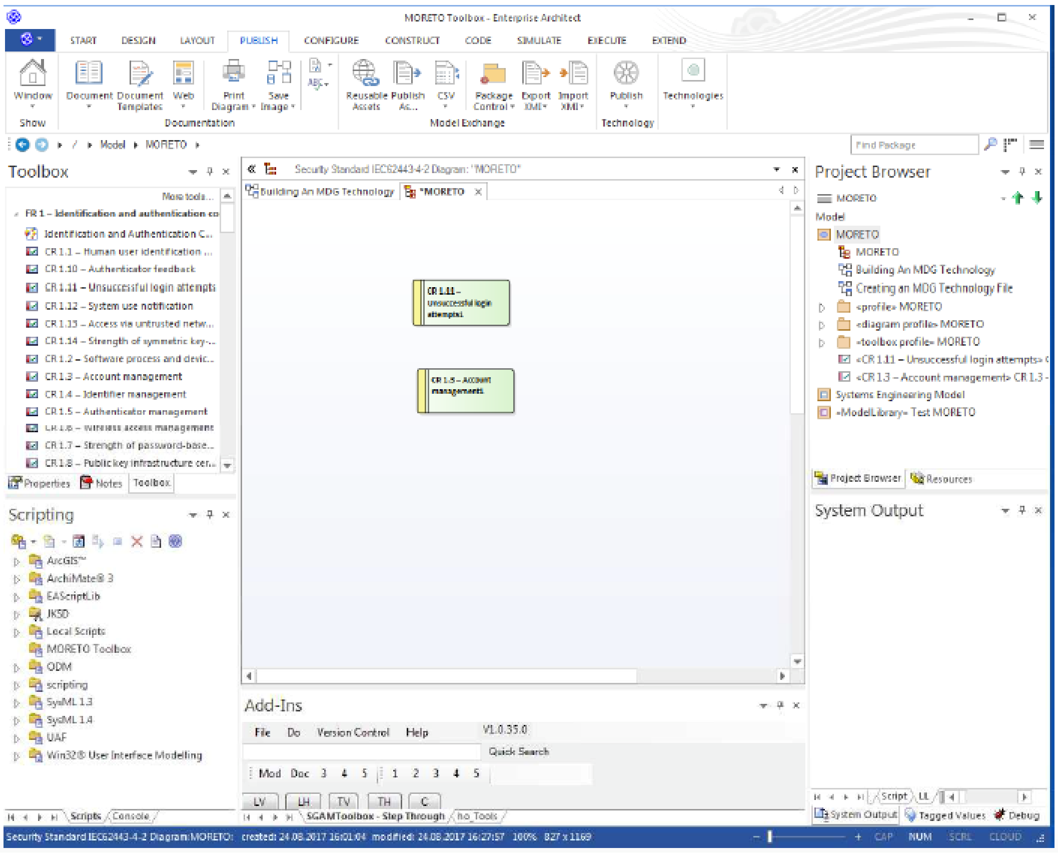

MORETO Toolbox - Enterprise Architect
START
DESIGN
LAYOUT
PUBLISH
CONFIGURE
CONSTRUCT
CODE
SIMULATE
EXECUTE
EXTEND
Toolbox
FR 1 – Identification and authentication co
Identification and Authentication C...
CR 1.1 – Human user identification ...
CR 1.10 – Authenticator feedback
CR 1.11 – Unsuccessful login attempts
CR 1.12 – System use notification
CR 1.13 – Access via untrusted netw...
CR 1.14 – Strength of symmetric key-...
CR 1.2 – Software process and devic...
CR 1.3 – Account management
CR 1.4 – Identifier management
CR 1.5 – Authenticator management
CR 1.6 – Wireless access management
CR 1.7 – Strength of password-base...
CR 1.8 – Public key infrastructure cer...
Scripting
Security Standard IEC62443-4-2 Diagram: "MORETO"
CR 1.11 – Unsuccessful login attempts
CR 1.3 – Account management
Project Browser
MORETO
Model
Building An MDG Technology
Creating an MDG Technology File
«profile» MORETO
«diagram profile» MORETO
«toolbox profile» MORETO
«CR 1.11 – Unsuccessful login attempts»
«CR 1.3 – Account management» CR 1.3
Systems Engineering Model
«ModelLibrary» Test MORETO
System Output
Add-Ins

Figure 135. Drag-and-Drop in MORETO

Importing CSV Files

Importing CSV files is another way to generate security requirements quickly without any efforts from the user side. This feature makes the generating process of security requirements much more comfortable than creating these security requirements one-by-one. For example, if the user has security requirements stored on CSV of excel sheet and he/she wants to transfer all of these requirements into the Enterprise Architect (EA) environment. Figure 136 shows how the user can import CSV contents into EA.

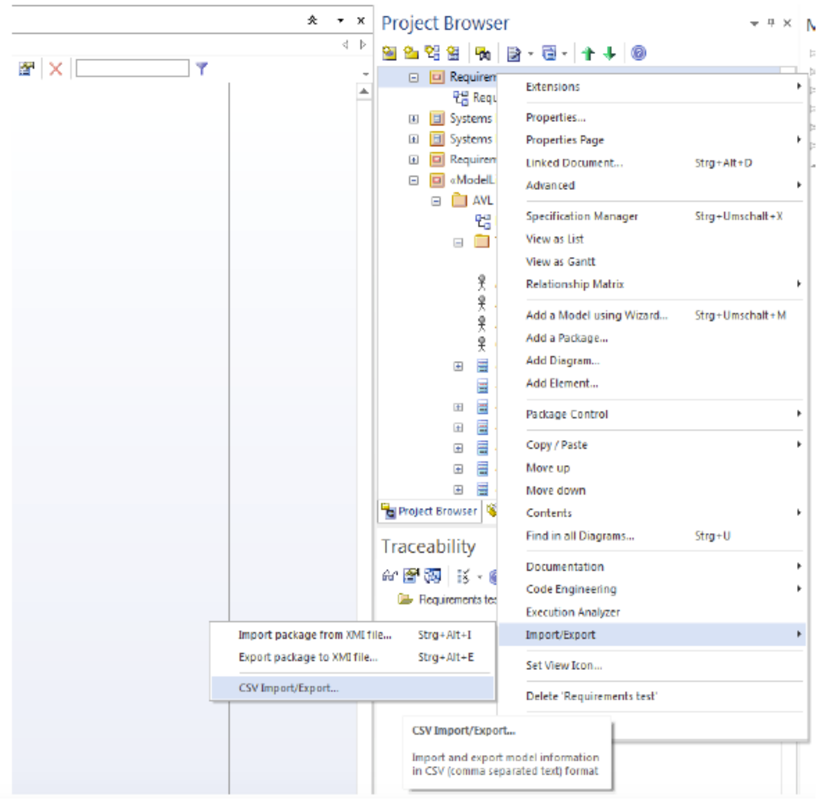


Figure 136. Importing CSV files in MORETO

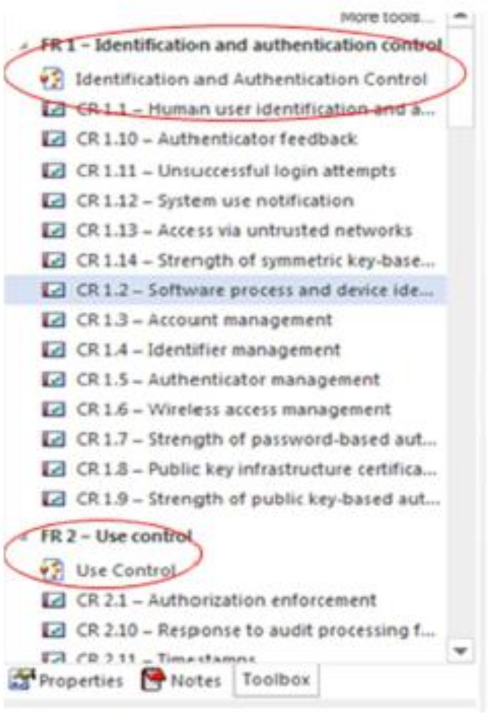
By this way, the EA converts all the contents of CSV into a list of requirements and integrates it with its environment. Now, the user can deal with the imported data as security requirements and choose which of these imported data can be used to be as security needs for his model.

Automatic Mode

As is depicted in Figure 134, there are two different ways for generating security requirements automatically: either by using scripts or patterns.

<u>Patterns</u>

The pattern is a pre-defined template implemented by MORETO developers team. This feature can generate a list of security requirements with their connection paths without assistance from the users. Figure 137 shows the security requirements toolbox with pattern icons for each classification of these requirements.



More tools...

FR 1 – Identification and authentication control

Identification and Authentication Control

CR 1.1 – Human user identification and a...

CR 1.10 – Authenticator feedback

CR 1.11 – Unsuccessful login attempts

CR 1.12 – System use notification

CR 1.13 – Access via untrusted networks

CR 1.14 – Strength of symmetric key-base...

CR 1.2 – Software process and device ide...

CR 1.3 – Account management

CR 1.4 – Identifier management

CR 1.5 – Authenticator management

CR 1.6 – Wireless access management

CR 1.7 – Strength of password-based aut...

CR 1.8 – Public key infrastructure certifica...

CR 1.9 – Strength of public key-based aut...

FR 2 – Use control

Use Control

CR 2.1 – Authorization enforcement

CR 2.10 – Response to audit processing f...

CR 2.11 – Timestamps

Properties Notes Toolbox

Figure 137. Pattern for automatic generation of security requirements in MORETO

As described above in the drag and drop section, the user can drag one of these patterns into the workspace to generate a list of integrated security requirements. Figure 138 depicts the automatic generation process of security requirements.

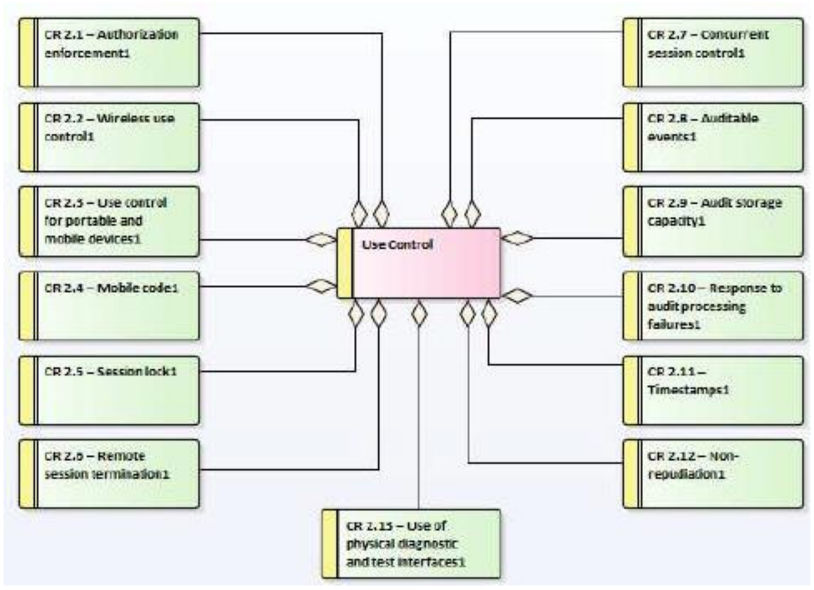

CR 2.1 – Authorization enforcement1
CR 2.2 – Wireless use control1
CR 2.3 – Use control for portable and mobile devices1
CR 2.4 – Mobile code1
CR 2.5 – Session lock1
CR 2.6 – Remote session termination1
Use Control
CR 2.13 – Use of physical diagnostic and test interfaces1
CR 2.7 – Concurrent session control1
CR 2.8 – Auditable events1
CR 2.9 – Audit storage capacity1
CR 2.10 – Response to audit processing failures1
CR 2.11 – Timestamps1
CR 2.12 – Non-repudiation1

Figure 138. Security requirements pattern in MORETO

Scripts (Automation)

The automation in MORETO considers the unique feature which is provided my MORETO toolbox, which is able to scan the whole system components of the user and detect the security gaps. Afterwards, MORETO covers the security gaps with proper security standards. By scripts feature, MORETO is able to generate a list of security requirements automatically on behalf of the user. Figure 139 shows an example of using the scripts feature to generate security requirements.

To automatically generate a list of security requirements for a particular model, the user can fire this service by choosing “Security Requirement Generation” from the extension is as follows.

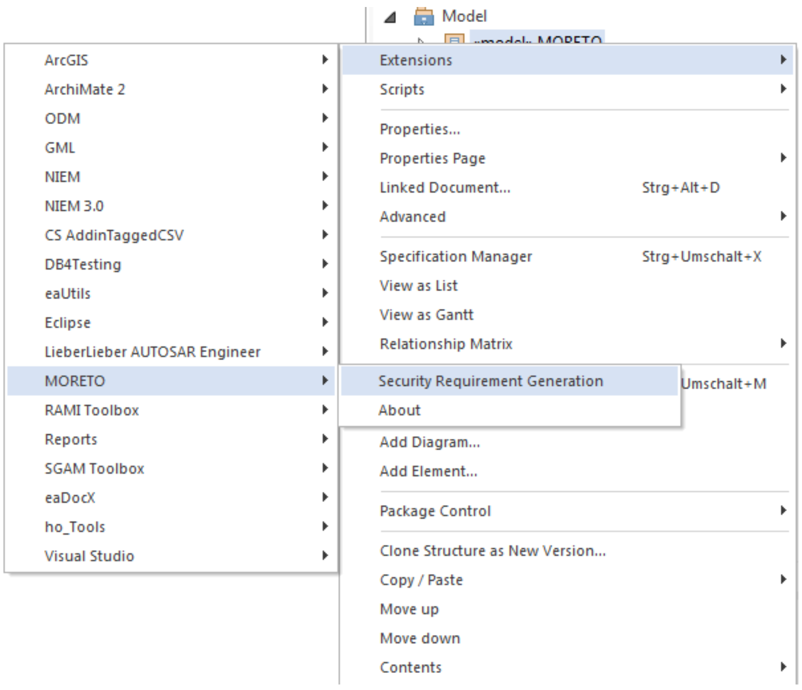


Figure 139. Automatic generation of security requirements in MORETO

Afterwards, MORETO starts to scan and analyse security issues of the system components and generate a list of security requirements based on the detected security gaps.

References

[1] AMASS D1.1 Case studies description and business impact, November 2016
[2] AMASS D1.2 Case Study Data Collection, March 2017
[3] AMASS D1.3 Evaluation Framework and Quality Metrics, September 2017
[4] AMASS D1.4 AMASS Demonstrators (a), April 2017
[5] AMASS D4.3 Design of the AMASS tools and methods for multi-concern assurance (b), April 2018
[6] AMASS D4.7 Methodological guide for multi-concern assurance (a), December 2017
[7] PolarSys: CHESS project. https://www.polarsys.org/projects/polarsys.chess
[8] PolarSys: OpenCert project. https://www.polarsys.org/projects/polarsys.opencert
[9] Eclipse Process Framework Project (EPF) https://eclipse.org/epf/
[10] OPENCOSS project http://www.opencoss-project.eu/
[11] CHESS project http://www.chess-project.org/
[12] SafeCer project https://artemis-ia.eu/project/30-psafecer.html