ECSEL Research and Innovation actions (RIA)

AMASS

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

Prototype for Cross-Domain and Intra-Domain Reuse (c) D6.6

Work Package:	WP6 Cross-Domain and Intra-Domain Reuse
Dissemination level:	PU = Public
Status:	Final
Date:	31 October 2018
Responsible partner:	Borja López (TRC)
Contact information:	borja.lopez@reusecompany.com
Document reference:	AMASS_D6.6_WP6_TRC_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
Borja López, Luis M. Alonso	The REUSE Company (TRC)
Ángel López, Huáscar Espinoza, Alejandra Ruiz	Tecnalia Research & Innovation (TEC)
Barbara Gallina, Inmaculada Ayala, Faiz Ul Muram, Muhammad Atif Javed, Irfan Sljivo, Julieth Castellanos	Maelardalens Hoegskola (MDH)
Stefano Puri	Intecs (INT)
Jose María Álvarez, Roy Mendieta, Miguel Rozalen, Fabio di Ninno	Universidad Carlos III de Madrid (UC3)

Reviewers

Names	Organisation
Helmut Martin (Peer-reviewer)	Virtual Vehicle (VIF)
Morayo Adedjouma (Peer-reviewer)	Commissariat a L'energie Atomique et aux Energies Alternatives (CEA)
Cristina Martinez (Quality Manager)	Tecnalia Research & Innovation (TEC)
Garazi Juez (TC review)	Tecnalia Research & Innovation (TEC)
Stefano Puri (TC review)	Intecs (INT)

TABLE OF CONTENTS

List of Figures

List of Tables

Executive Summary

The deliverable D6.6 – Implementation for Cross Domain and Intra Domain Reuse (c), is the third output of the task T6.3 (Implementation for Cross-Domain and Intra-Domain Reuse). Based on the results of tasks T2.2 (AMASS Reference Tool Architecture and Integration) and T6.2 (Conceptual Approach for Cross-Domain and Intra-Domain Reuse), task T6.3 develops a prototype-tooling framework to support cross and intra-domain reuse, as well as compliance management. D6.6 is the evolution of D6.5, which described the second prototype.

This deliverable reports the final status of the tooling framework for the AMASS platform by describing the supported WP6-related functionality and the details about its implementation.

T6.3 progresses iteratively and incrementally, in close connection with the conceptual task (T6.2) and the other technical WPs (WP2 to WP5). The implementation follows the requirements of the case studies, which must benchmark the prototypes. This iterative process helps benchmarking the prototype implementation, so that it makes easier to check the continuous refinement done during the different AMASS prototypes.

The WP6-related part of the final iteration (P2) of the AMASS platform extends the initial implementation of the basic building blocks for this prototype, which has been a consolidation and integration of results from previous projects. More concretely, the developed tools in the WP6-related part of P2 support the following functional areas:

- Capture, retrieve and share information from standards.
- Define compliance and equivalence mappings.
- Generate argumentation fragments based on development processes.
- Manage assurance projects.
- Monitor progress status of assurance project.
- Reuse discovery and reuse assistance.
- Variability management at assurance case/process/product level.
- Semi-automatic generation of product/process arguments.

This document presents in detail the pieces of functionality implemented in the AMASS platform for the areas above, their software architecture, the technology used, and source code references.

D6.6 relates to other AMASS outcomes referred in this deliverable:

- D6.5, where the main differences are described in the chapter Appendix A: Document changes with respect to D6.5.
- Installable AMASS platform tools for the Prototype P2.
- User manual and installation instructions [18].
- Source code [17].

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 [1] and SafeCer [2] 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 (based on OSLC (Open Services for Lifecycle Collaboration) specifications [12]).

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. AMASS openness is based on both standard OSLC APIs with external tools (e.g. engineering tools including V&V tools) and on open-source release of the AMASS building blocks.

c) The **Open AMASS Community**, which will manage the project outcomes, for maintenance, evolution and industrialisation. 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. As Eclipse Foundation is part of the AMASS consortium, the Polarsys/Eclipse community (www.polarsys.org) is going to host the AMASS Open Tool Platform.

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.

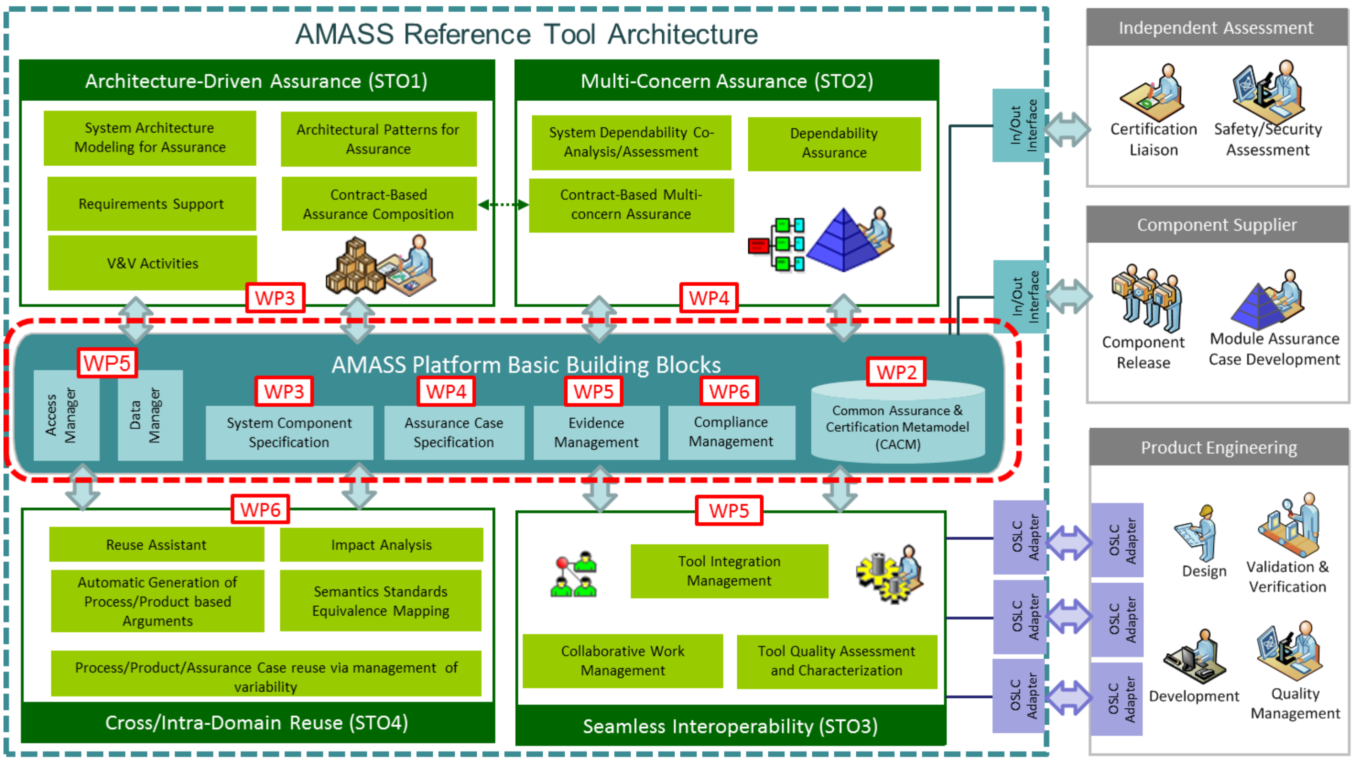

AMASS Reference Tool Architecture
Architecture-Driven Assurance (STO1)
System Architecture Modeling for Assurance
Architectural Patterns for Assurance
Requirements Support
Contract-Based Assurance Composition
V&V Activities
WP3
Multi-Concern Assurance (STO2)
System Dependability Co-Analysis/Assessment
Dependability Assurance
Contract-Based Multi-concern Assurance
WP4
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
Reuse Assistant
Impact Analysis
Automatic Generation of Process/Product based Arguments
Semantics Standards Equivalence Mapping
Process/Product/Assurance Case reuse via management of variability
Cross/Intra-Domain Reuse (STO4)
WP5
Tool Integration Management
Collaborative Work Management
Tool Quality Assessment and Characterization
Seamless Interoperability (STO3)
In/Out Interface
In/Out Interface
OSLC Adapter
OSLC Adapter
OSLC Adapter
OSLC Adapter
OSLC Adapter
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 in AMASS Prototype P2

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

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

AMASS has planned three prototype iterations:

1. During the **first prototyping** iteration (Prototype Core), the AMASS Platform Basic Building Blocks (see Figure 1), were aligned, merged and consolidated at Technology Readiness Levels TRL4[1]. Concerning this first prototype, the basic building block assigned to WP6 was Compliance Management.
2. During the **second prototyping** iteration (Prototype P1), the AMASS-specific Building Blocks were developed and benchmarked at TRL4; this comprises the blue basic building blocks as well as the green building blocks in Figure 1. Regarding WP6, in this second prototype the specific building blocks included a (cross/intra-domain) reuse assistant potentially using semantics standards equivalence mappings and a toolset for product/process/assurance case line specification.
3. Finally, at the **third prototyping** iteration (Prototype P2), all AMASS building blocks are integrated in a comprehensive toolset operating at TRL5. WP6 functionalities developed during the second prototype iteration are enhanced and fully integrated with functionalities of other technical work packages.

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. The case studies are described in D1.1 [20].

As part of the Prototype P2, WP6 is responsible for driving the work resulting on intra-domain and cross-domain reuse, in order to design and implement the building blocks for "**Reuse Assistant**", "**Impact Analysis**", "**Automatic Generation of Process/Product based Arguments**", "**Semantics Standards Equivalence Mapping**" and "**Product/Process/Assurance Case reuse via management of variability**" (cf. Figure 1).

This deliverable reports the **tool development** results of the building blocks commented above. It presents in detail the design of the functionality implemented in the AMASS platform tools, the software architecture, the technology used, and source code references. The design is based on the investigated state-of-the-art and state-of-practice approaches. Their gaps are identified to come up with a way forward, enabling the formulation of requirements to achieve the reuse-oriented vision of AMASS. This activity will serve to ensure both, the innovation of the project and future feasibility of exploitation of results.

Other important documents related to D6.6 are:

- Installable AMASS Platform tools for the Prototype P2.
- User manuals and Installation instructions [18].
- Source code.

[1] In the context of AMASS, the EU H2020 definition of TRL is used, see http://ec.europa.eu/research/participants/data/ref/h2020/other/wp/2016_2017/annexes/h2020-wp1617-annex-g-trl_en.pdf

2. Implemented Functionality

2.1 Scope

As stated in Section 1, the building blocks assigned to WP6 in this prototype are "**Reuse Assistant**", "**Semantics Standards Equivalence Mapping**" and "**Product/Process/Assurance Case Line Specification**". Also, some improvements have been implemented regarding the "**Compliance Management**" and "**Assurance Project Lifecycle Management**" blocks (see Figure 2).

These blocks are highlighted with a red circle in Figure 2, showing the general functional overview of the AMASS platform (from deliverable D3.6 [14]).

Furthermore, WP6 is aimed at providing the framework for cross-domain and intra-domain reuse.

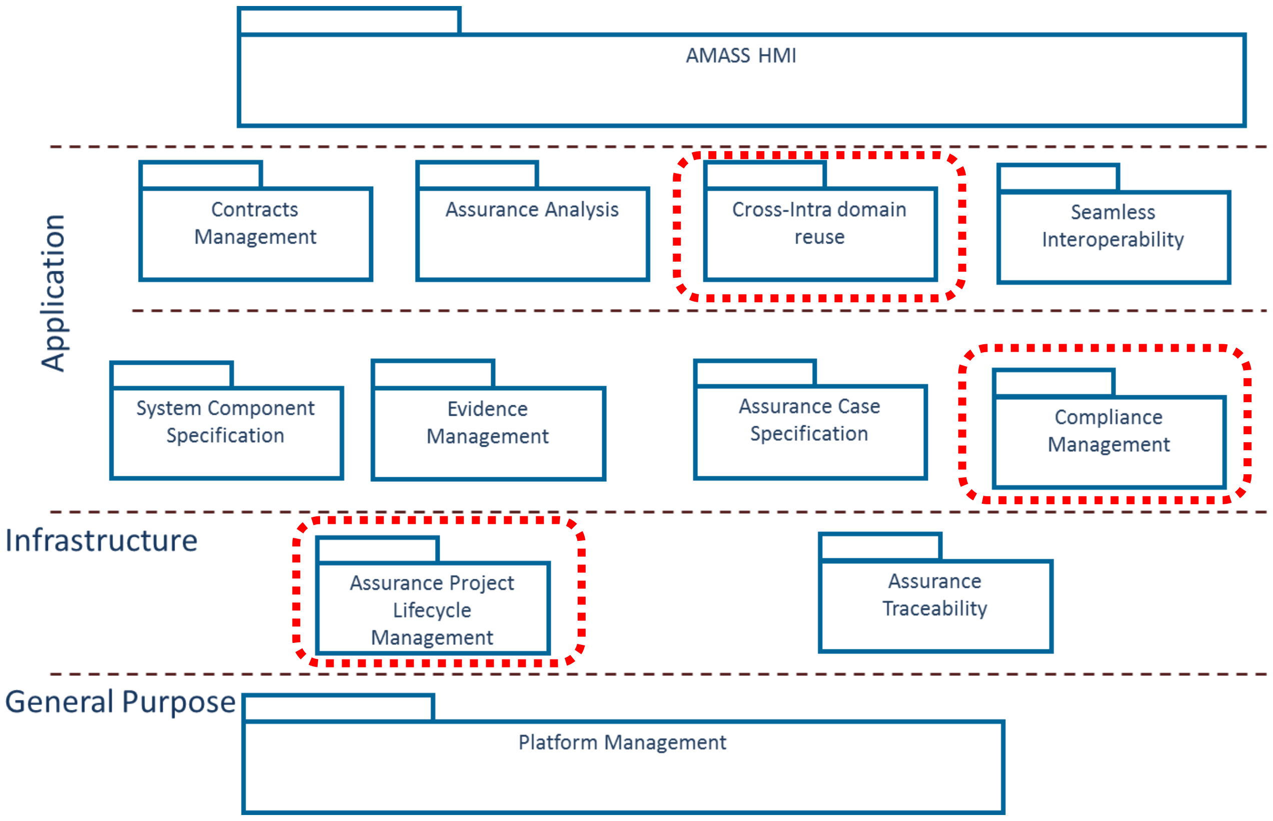


Figure 2. Functional decomposition for the AMASS platform

The Compliance Management building block has been improved in this third prototype to allow monitoring the compliance progress using filtering by criticality levels (e.g., Safety Integrity Level, Automotive Safety Integrity Level, Development Assurance Level). The compliance is also performed by the generation of a compliance report in a web client application.

The AMASS platform supports the Compliance Management functionalities with two toolsets:

- Tools from the OpenCert project[2].
- Tools from the EPF (Eclipse Process Framework) project[3].

[2] Further information about the OPENCOSS toolset can be found at www.opencoss-project.eu and https://www.polarsys.org/projects/polarsys.opencert

[3] Further information about the EPF toolset can be found at http://www.eclipse.org/epf/

The following section details both the satisfied requirements and the deployed components to show the implementation scope of the WP6-related part of the Prototype P2.

2.2 Implemented Requirements

From the requirements point of view, D6.6 has focused on a set of AMASS requirements as defined in deliverable D2.1 [13]. The requirements listed in Table 1 were implemented in the first iteration of the WP6-related part of the AMASS platform (Prototype Core). The column "Related requirement" refers to the list of items gathered in the deliverable D2.1.

Table 1. Requirements implemented in the first prototype of the AMASS platform (Core Prototype)

Function name	Related Requirement	Description
Modelling of standards	WP6_CM_001	The AMASS tools shall be able to model a set of industrial standards (including the parts, activities, requirements, work products, and criticality levels from the standards).
Tailoring of standards models to specific projects	WP6_CM_002	The AMASS tools shall enable the tailoring of standards models to a specific project (e.g., by establishing the parts of the standard that apply to a given assurance project).
Compliance Monitoring	WP6_CM_005	The AMASS tools shall support web-based monitoring of compliance status to be filtered by any custom criteria.
Process Compliance (informal) management	WP6_CM_008	The AMASS tools shall enable users to visualise process compliance. This means showing the links between the requirements and the applicant's evidence (during the planning as well as execution phase). This visualisation could be done via compliance maps (matrix) or via arguments aimed at justifying the satisfaction of the requirements coming from the standards.

In addition to that, the Prototype P1 focused on a second set of AMASS requirements as defined in deliverable D2.1 [13].

Table 2. Requirements implemented in the second prototype of the AMASS platform (Prototype P1)

Function name	Related Requirement	Description
Intra-Domain, Intra standard, Reuse Assistance	WP6_RA_001	The AMASS tools shall enable partial reuse of compliance artefacts when transiting from one project to another (different criticality level, etc.). The commonality that characterises the different projects should be recognised and proposed as reusable process structure.
Intra-Domain, Cross standards, Reuse Assistance	WP6_RA_002	The AMASS tools shall enable partial reuse of compliance artefacts when transiting from one project to another (different/same criticality level, if applicable, but different standards (e.g., Automotive SPICE, ISO 26262)). The commonality that characterises the different projects should be recognised and proposed as a reusable process structure.

Function name	Related Requirement	Description
Intra-Domain, Cross versions, Reuse Assistance	WP6_RA_003	The AMASS tools shall enable partial reuse of compliance artefacts when transiting from one project to another (different/same criticality level, if applicable, but different standards or different versions of a standard (e.g., ISO 26262-2011, ISO 26262-2018)). The commonality that characterises the different projects should be recognized and proposed as reusable process structure.
Cross-Domain Reuse Assistance	WP6_RA_004	The AMASS tools shall enable partial reuse of compliance artefacts when transiting from one project to another belonging to different domains (e.g., from automotive to avionics). The commonality that characterises the different projects should be recognised and proposed as reusable process structure.
Intra-Domain, Intra standard, Different Stakeholders, Reuse/Integration Assistance	WP6_RA_005	The AMASS tools shall enable partial reuse of compliance artefacts during the integration (manufacturer/supplier). Assumed process requirements vs. actual process requirements.
The AMASS tools must support variability management at process level	WP6_PPA_001	The AMASS tools shall enable the specification/systematisation of variability at the process level.
The AMASS tools must support specification of variability at the component level	WP6_PPA_004	The system shall enable users to specify what varies (and what remains unchanged) from one component and its evolved version at component level.
The AMASS tools must support variability management at the assurance case level	WP6_PPA_005	The system shall enable users to specify what varies (and what remains unchanged) from one component and its evolved version at component level.
Semi-automatic generation of product arguments	WP6_PPA_002	Assurance case arguments. This could be done by enabling semi-automatic generation of product-based arguments-fragments.
Semi-automatic generation of process arguments	WP6_PPA_003	The system shall reduce efforts of manual creation of process-based assurance case arguments. This could be done by enabling semi-automatic generation of process-based arguments-fragments.
Process Compliance (formal) management	WP6_CM_009	The AMASS tools shall enable users to formally check process compliance.

Finally, this last phase (Prototype P2) focuses on the last iteration of the AMASS requirements as defined in deliverable D2.1 [13]. The requirements listed in Table 3 have been implemented during the third iteration of the WP6-related part of the AMASS platform. The column "Related requirement" refers to the list of items gathered in the deliverable D2.1.

Table 3. Requirements implemented in the third prototype of the AMASS platform (Prototype P2)

Function name	Related Requirement	Description
Triggering compliance Checking	WP6_CM_004	The AMASS tools shall provide the functionality for automatically triggering the requirements for (re)checking the compliance of safety processes against rules – especially, when there are changes in the standards/ regulations.
Process Compliance (formal) management)	WP6_CM_009	The AMASS tools shall enable users to formally check process compliance.
Reusable off the shelf components	WP6_RA_006	The AMASS tool shall provide the capability for reuse of pre-developed components and their accompanying artefacts.
Semantics-based mapping of standards	WP6_SEM_001	The AMASS tools shall enable the mapping of standards based on their semantics.
The AMASS tools must support variability management at the assurance case level	WP6_PPA_005	The system shall enable users to specify what varies (and what remains unchanged) from one component and its evolved version at component level.

Each requirement, together with the implementation done so far that implements the requirement, is shortly outlined in the following sections.

2.2.1 Modelling of standards

This chapter describes the implementation of the requirement **WP6_CM_001,** described in Table 1.

This feature is supported by both the OpenCert [26] and EPF toolsets [27]. Figure 3 shows examples of models for these two toolsets.

- OpenCert uses Reference Framework Editor (as part of the OpenCert tools) to model standards (IEC 61508, ISO 26262, DO-178C, EN 50126, and the like) and regulations (either as additional requirements or model elements in a given model representing a standard or a new Reference Framework). Each Reference Framework model can be also mapped to others Reference Framework models by using the concept of Equivalence Map (diamond form).
- EPF can be used to model company-specific processes (e.g., the process at Alstom or Thales to develop safety-critical systems).

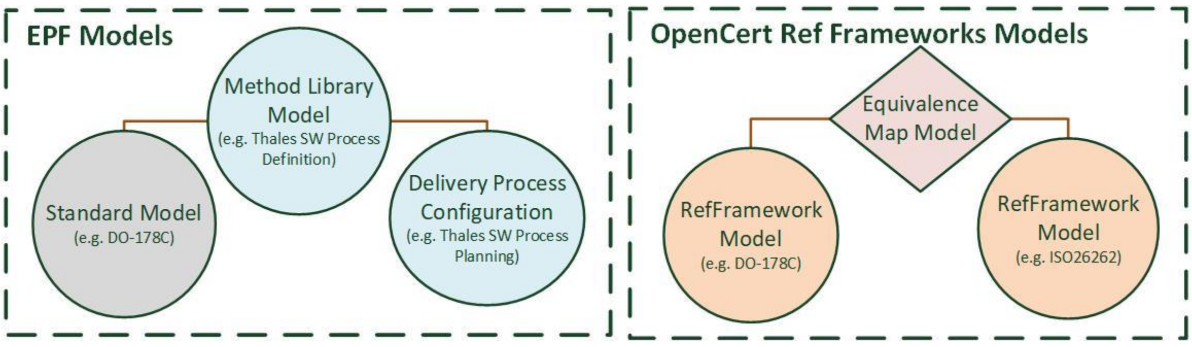
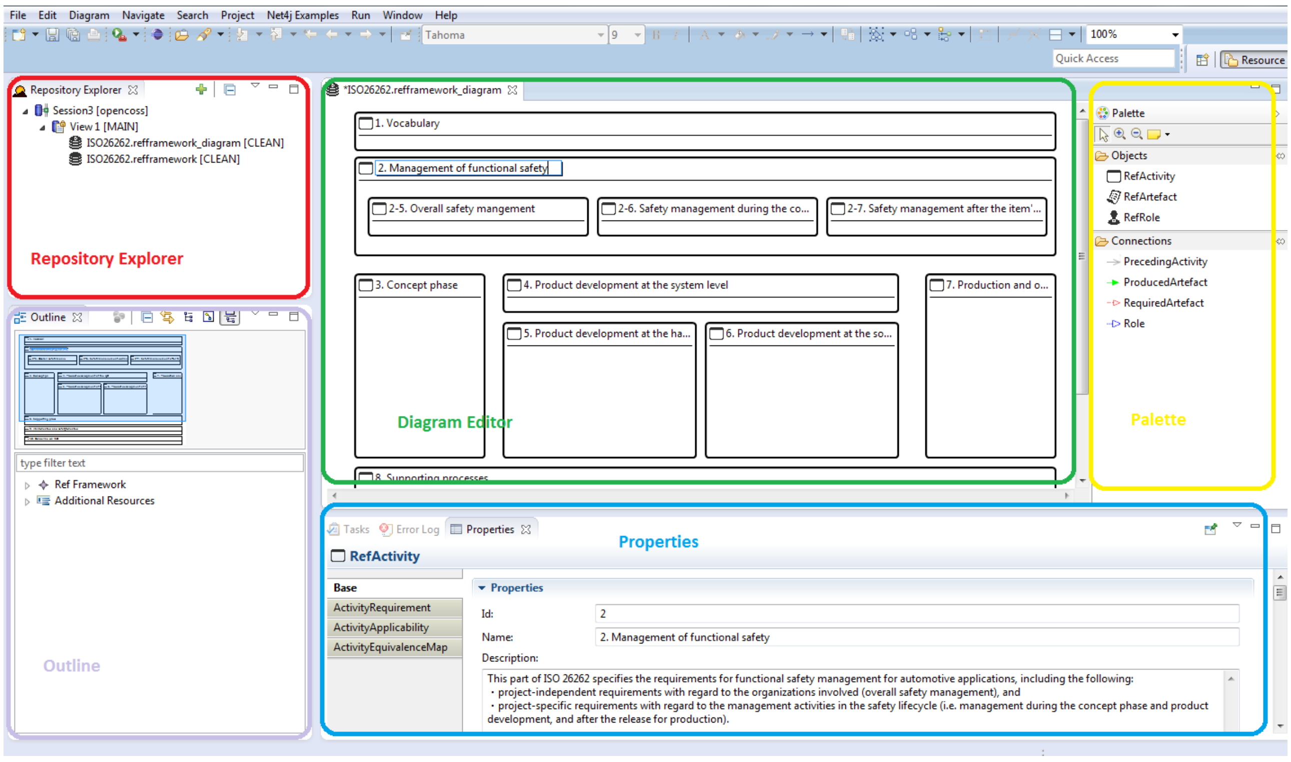

EPF Models
Method Library Model (e.g. Thales SW Process Definition)
Standard Model (e.g. DO-178C)
Delivery Process Configuration (e.g. Thales SW Process Planning)
OpenCert Ref Frameworks Models
Equivalence Map Model
RefFramework Model (e.g. DO-178C)
RefFramework Model (e.g. ISO26262)

Figure 3. Standard and Process Modelling

3.1.2.1 Reference Framework Editor

The Reference Framework Editor is composed of five views (see Figure 4):

1. The **Repository Explorer** view shows the contents of the repository.
2. The **Outline** view shows the elements of the model and permits its edition.
3. The **Diagram Editor** permits the graphical modelling of a subset of concepts of the Reference Framework. The Diagram Editor can be replaced by the **Tree View Editor**, which is opened by double clicking on the file ".refframework" in the Repository Explorer.
4. The **Palette** view is a toolbox with the concepts of the model and the connections between them to add to the diagram.
5. The **Properties** view is used to edit the properties of the element of the selected model.


File Edit Diagram Navigate Search Project Net4j Examples Run Window Help
Repository Explorer
Session3 [opencoss]
View 1 [MAIN]
ISO26262.refframework_diagram [CLEAN]
ISO26262.refframework [CLEAN]
Outline
type filter text
Ref Framework
Additional Resources
*ISO26262.refframework_diagram
1. Vocabulary
2. Management of functional safety
2-5. Overall safety mangement
2-6. Safety management during the co...
2-7. Safety management after the item'...
3. Concept phase
4. Product development at the system level
5. Product development at the ha...
6. Product development at the so...
7. Production and o...
8. Supporting processes
Diagram Editor
Palette
Objects
RefActivity
RefArtefact
RefRole
Connections
PrecedingActivity
ProducedArtefact
RequiredArtefact
Role
Tasks
Error Log
Properties
RefActivity
Base
ActivityRequirement
ActivityApplicability
ActivityEquivalenceMap

Properties

Id: 2

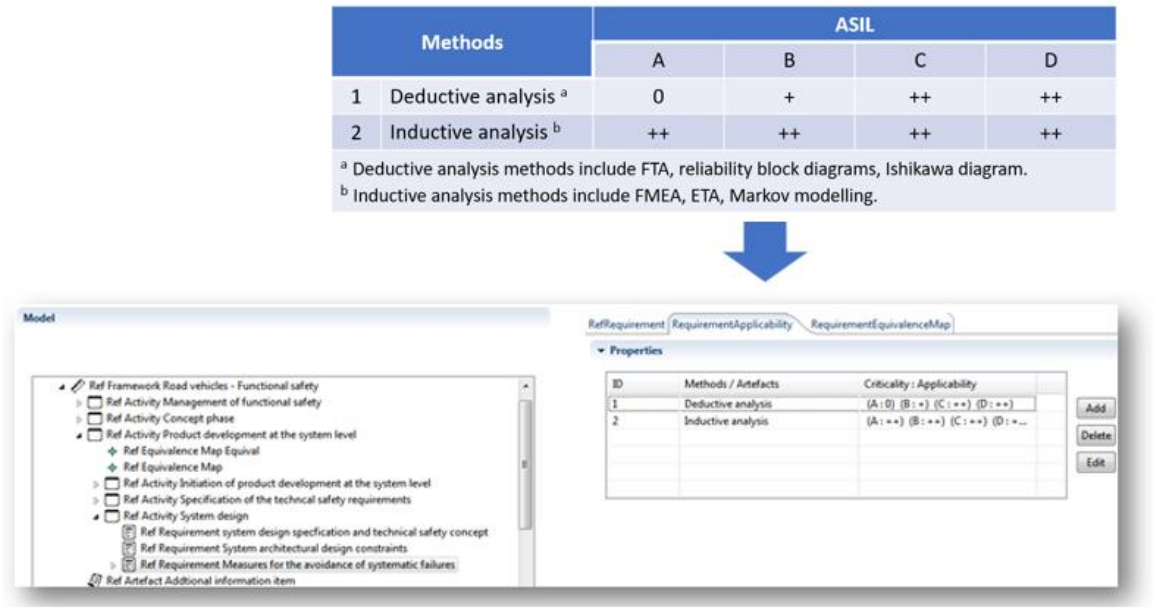
Name: 2. Management of functional safety

Description:

This part of ISO 26262 specifies the requirements for functional safety management for automotive applications, including the following:
• project-independent requirements with regard to the organizations involved (overall safety management), and
• project-specific requirements with regard to the management activities in the safety lifecycle (i.e. management during the concept phase and product development, and after the release for production).

Figure 4. Reference Framework Editor

In addition, the recommendation or applicability tables (that is, the SIL levels) from industry standards can be specified using the Tree View editor, by associating the Tables to specific Requirements or Activities. Figure 5 illustrates an example for ISO 26262 recommendation table. A similar approach can be used for other standards such as DO-178C objective tables.



Methods		ASIL			
		A	B	C	D
1	Deductive analysis [a]	0	+	++	++
2	Inductive analysis [b]	++	++	++	++

[a] Deductive analysis methods include FTA, reliability block diagrams, Ishikawa diagram.
[b] Inductive analysis methods include FMEA, ETA, Markov modelling.

Model

- Ref Framework Road vehicles - Functional safety
  - Ref Activity Management of functional safety
  - Ref Activity Concept phase
  - Ref Activity Product development at the system level
    - Ref Equivalence Map Equival
    - Ref Equivalence Map
    - Ref Activity Initiation of product development at the system level
    - Ref Activity Specification of the technical safety requirements
    - Ref Activity System design
      - Ref Requirement system design specfication and technical safety concept
      - Ref Requirement System architectural design constraints
      - Ref Requirement Measures for the avoidance of systematic failures
  - Ref Artefact Addtional information item

RefRequirement | RequirementApplicability | RequirementEquivalenceMap

Properties

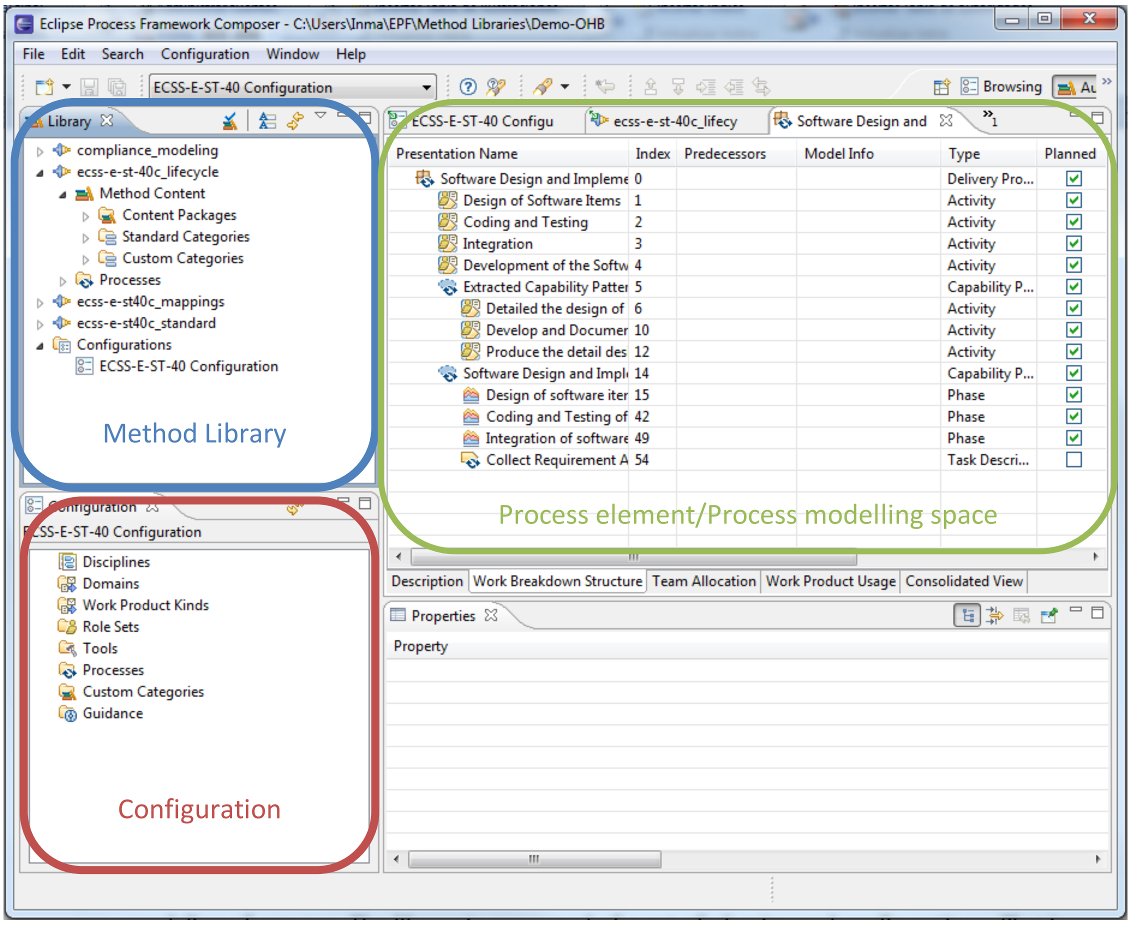
ID	Methods / Artefacts	Criticality : Applicability
1	Deductive analysis	(A : 0) (B : +) (C : ++) (D : ++)
2	Inductive analysis	(A : ++) (B : ++) (C : ++) (D : +...

Figure 5. ISO 26262: Recommendation Table associated to a Requirement

3.1.2.2 Standards modelling with EPF

The EPF composer is a tool for the modelling of engineering process based on the SPEM 2.0 (Software & Systems Process Engineering Metamodel) OMG (Object Management Group) standard [25]. The functionality of the EPF composer is organised in two views, the Authoring perspective (opened by default in the EPF Composer) and the Browsing perspective. The goal of the Authoring perspective is to provide functionality to formally model process element and processes, while the goal of the Browsing perspective is to present the contents modelled of the Authoring perspective. So, most of the work of the user will take place in this last perspective.

Figure 6 shows a screenshot of this modelling perspective, which is composed of three parts: the Method library (left top of the workbench), the Configuration (left bottom of the workbench) and the Process element/process modelling space (right part of the workbench) that, in this case, is showing the modelling of a delivery process.



- compliance_modeling
- ecss-e-st-40c_lifecycle
  - Method Content
    - Content Packages
    - Standard Categories
    - Custom Categories
  - Processes
- ecss-e-st40c_mappings
- ecss-e-st40c_standard
- Configurations
  - ECSS-E-ST-40 Configuration

Configuration: ECSS-E-ST-40 Configuration

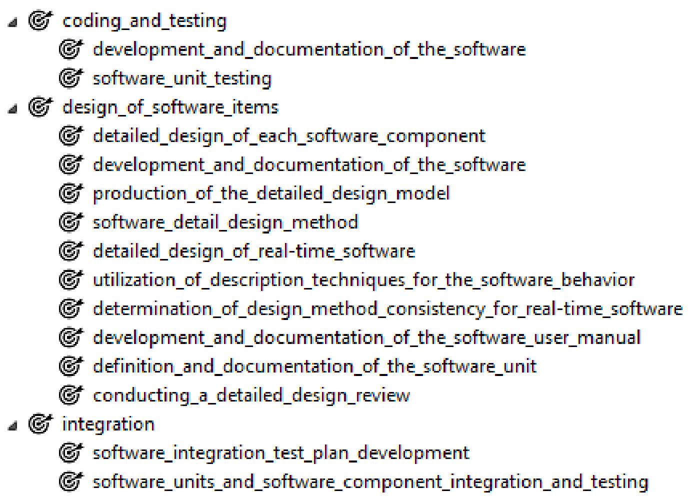
- Disciplines
- Domains
- Work Product Kinds
- Role Sets
- Tools
- Processes
- Custom Categories
- Guidance

Presentation Name	Index	Predecessors	Model Info	Type	Planned
Software Design and Impleme	0			Delivery Pro...	✓
Design of Software Items	1			Activity	✓
Coding and Testing	2			Activity	✓
Integration	3			Activity	✓
Development of the Softw	4			Activity	✓
Extracted Capability Patter	5			Capability P...	✓
Detailed the design of	6			Activity	✓
Develop and Documer	10			Activity	✓
Produce the detail des	12			Activity	✓
Software Design and Impl	14			Capability P...	✓
Design of software iter	15			Phase	✓
Coding and Testing of	42			Phase	✓
Integration of software	49			Phase	✓
Collect Requirement A	54			Task Descri...	☐

Figure 6. Authoring perspective of the EPF composer

Standards can be modelled in the EPF composer tool [23] following an approach similar to those described in [22] for the IBM Rational Method Composer. In the work presented in [22], standards are modelled using a new user defined type named *Requirement*. However, EPF does not support the definition of a new user defined type. In order to overcome this limitation, we have customized the guidance “Practice” with an icon and variability relationships making possible the application of the mentioned work. The EPF composer supports variability mechanisms that are at disposal in SPEM 2.0. These variability mechanisms focus on set semantic relationships between process elements of the same type. These semantics relationships make possible to define a new process element as a variation of an existing one.

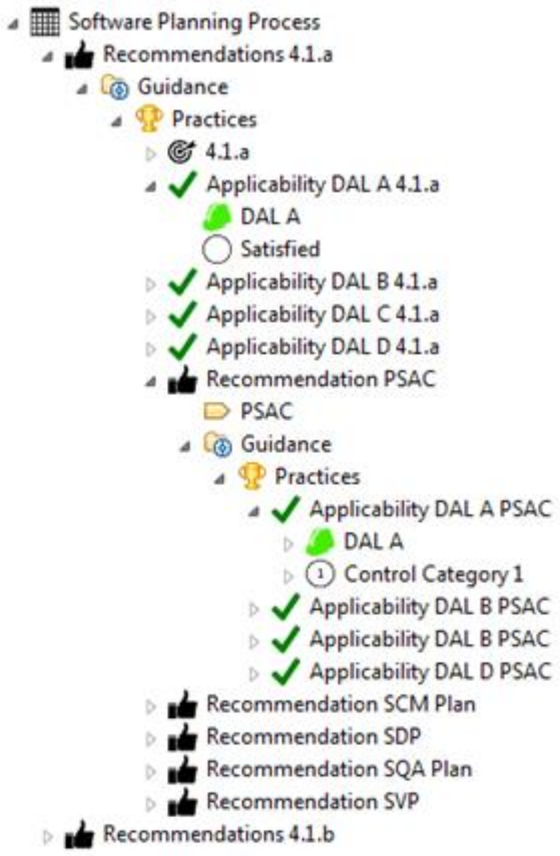
Practices in EPF represent a proven way or strategy of doing work to achieve a goal that has a positive impact on work product or process quality. They are usually used to group process elements that belong to some practice like risk management, software quality verification or component-based development just to mention a few. Therefore, in our view, practice semantics and use are aligned with the semantics of requirements. Then, standards are modelled in the Authoring perspective of the EPF composer as a collection of nested Requirements (i.e. customised practices) (see Figure 7).



coding_and_testing
development_and_documentation_of_the_software
software_unit_testing
design_of_software_items
detailed_design_of_each_software_component
development_and_documentation_of_the_software
production_of_the_detailed_design_model
software_detail_design_method
detailed_design_of_real-time_software
utilization_of_description_techniques_for_the_software_behavior
determination_of_design_method_consistency_for_real-time_software
development_and_documentation_of_the_software_user_manual
definition_and_documentation_of_the_software_unit
conducting_a_detailed_design_review
integration
software_integration_test_plan_development
software_units_and_software_component_integration_and_testing

Figure 7. Standard modelled in the EPF composer

The EPF composer also supports the modelling of the recommendation tables by means of customized practices. In this case, five customized practices were created to represent tables, criticality levels, recommendation levels and concepts to make possible to associate these three concepts. Recommendation tables are modelled as a composition of customized practices of different kinds. Figure 8 shows the result of modelling of a recommendation table from RTCA DO-178C.



Software Planning Process
Recommendations 4.1.a
Guidance
Practices
4.1.a
Applicability DAL A 4.1.a
DAL A
Satisfied
Applicability DAL B 4.1.a
Applicability DAL C 4.1.a
Applicability DAL D 4.1.a
Recommendation PSAC
PSAC
Guidance
Practices
Applicability DAL A PSAC
DAL A
Control Category 1
Applicability DAL B PSAC
Applicability DAL B PSAC
Applicability DAL D PSAC
Recommendation SCM Plan
Recommendation SDP
Recommendation SQA Plan
Recommendation SVP
Recommendations 4.1.b

Figure 8. Recommendation table modelled in the EPF composer

The EPF composer allows import and export projects (known as plug-ins in EPF) in the library space. This allows using process and process elements defined in the imported project in other projects of the library space. In order to facilitate the modelling of standard information in the EPF composer, we have grouped all the modelling concepts that we have developed for the modelling of standard information in a plug-in that can be imported in any method library.

2.2.2 Tailoring of Standards models to specific projects

This chapter describes the implementation for requirement **WP6_CM_002,** described in Table 1.

The information managed by the AMASS tools can be organised in two types: project-independent information that can be used by various projects (e.g., models of generic processes and standards) and project-specific information (e.g., evidence and argumentation models). The main element of project-specific information is the so-called Assurance Project. One important part of an Assurance Project is the Baseline model. A Baseline model represents what is planned to comply with (regarding a Reference Framework model) a specific Assurance Project. Baseline models can be tailored from Reference Framework models.

As shown in Figure 9, Baseline models can be instantiated from the Reference Framework models. It is possible to import Baseline models based on the Standard processes modelled in EPF.

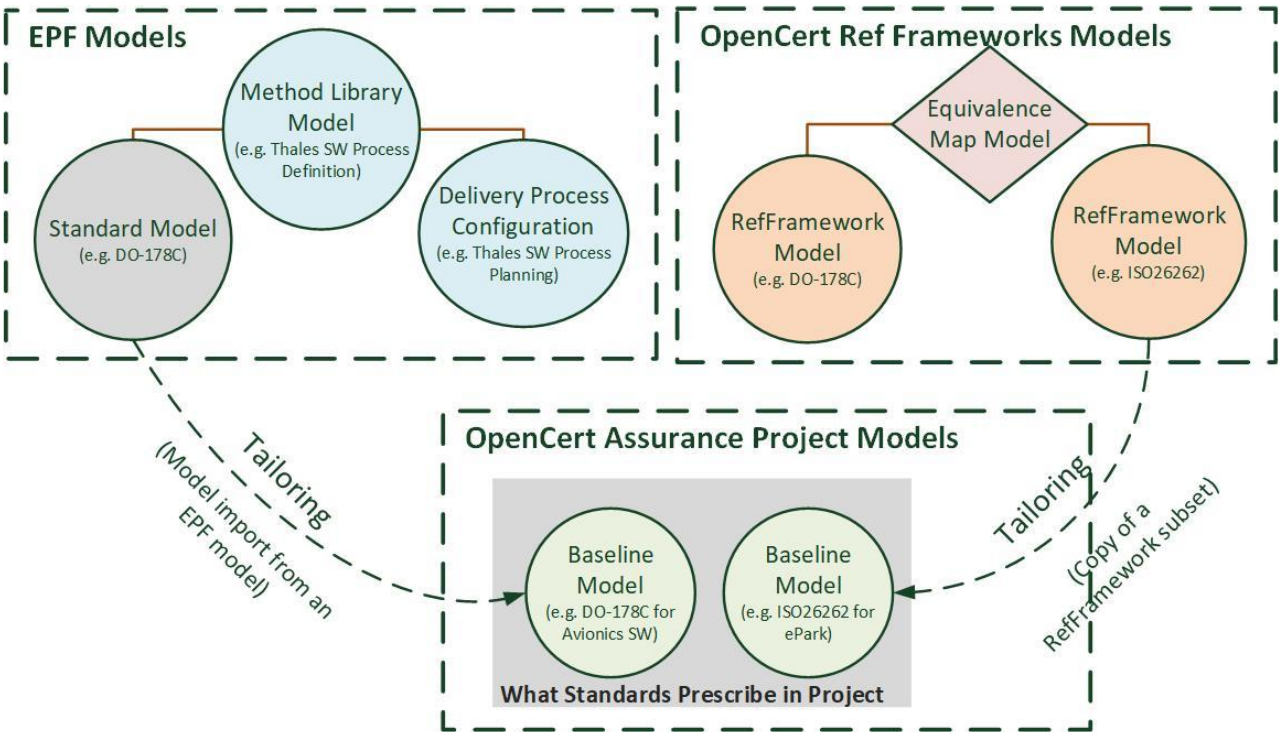
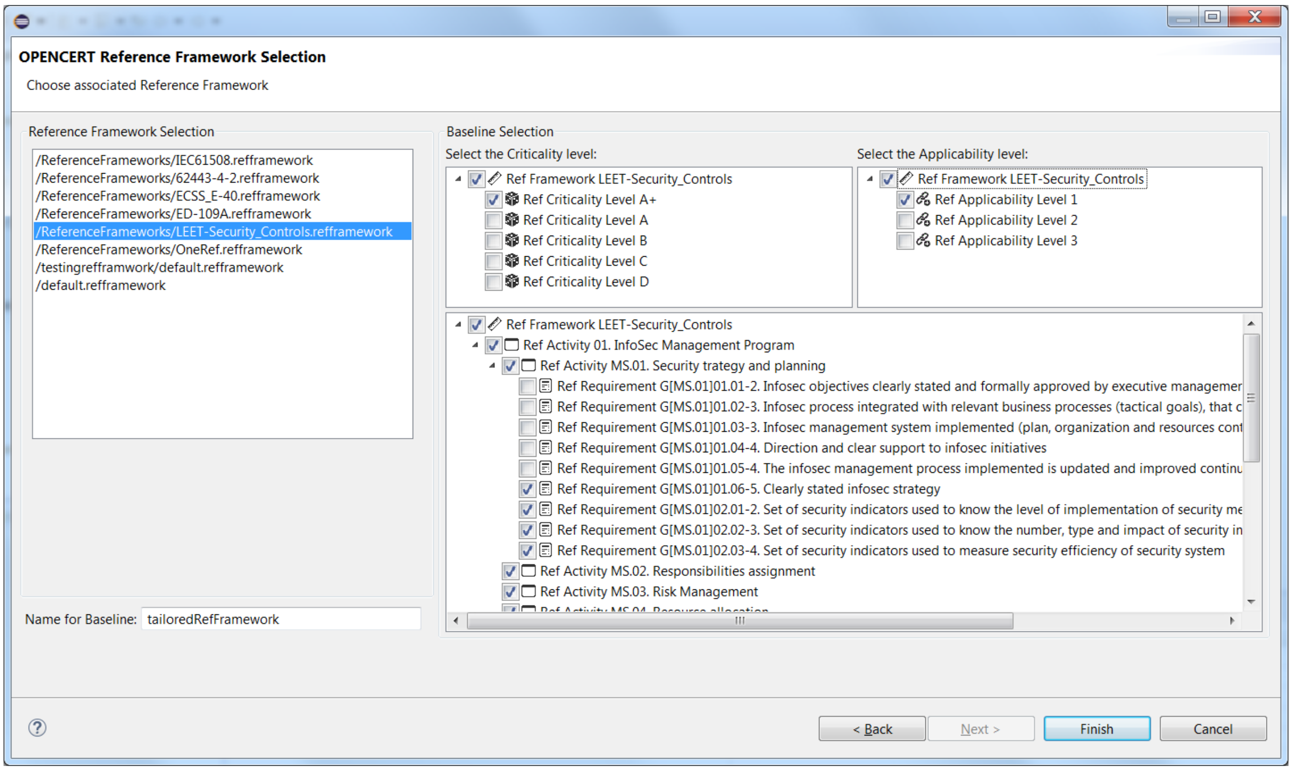

EPF Models
Method Library Model (e.g. Thales SW Process Definition)
Standard Model (e.g. DO-178C)
Delivery Process Configuration (e.g. Thales SW Process Planning)
OpenCert Ref Frameworks Models
Equivalence Map Model
RefFramework Model (e.g. DO-178C)
RefFramework Model (e.g. ISO26262)
Tailoring (Model import from an EPF model)
Tailoring (Copy of a RefFramework subset)
OpenCert Assurance Project Models
Baseline Model (e.g. DO-178C for Avionics SW)
Baseline Model (e.g. ISO26262 for ePark)
What Standards Prescribe in Project

Figure 9. Tailoring of Baseline Models from Standard Models

Each baseline model results from importing (copying) a Reference Framework model and selecting the subset of activities, artefacts and the like that apply in a given Assurance Project. Figure 10 shows the selection step before creating a baseline model upon a reference framework model (the latter displayed in a tree view).


OPENCERT Reference Framework Selection
Choose associated Reference Framework

Reference Framework Selection
/ReferenceFrameworks/IEC61508.refframework
/ReferenceFrameworks/62443-4-2.refframework
/ReferenceFrameworks/ECSS_E-40.refframework
/ReferenceFrameworks/ED-109A.refframework
/ReferenceFrameworks/LEET-Security_Controls.refframework
/ReferenceFrameworks/OneRef.refframework
/testingrefframwork/default.refframework
/default.refframework

Name for Baseline: tailoredRefFramework

Baseline Selection
Select the Criticality level:
- [✓] Ref Framework LEET-Security_Controls
  - [✓] Ref Criticality Level A+
  - [ ] Ref Criticality Level A
  - [ ] Ref Criticality Level B
  - [ ] Ref Criticality Level C
  - [ ] Ref Criticality Level D

Select the Applicability level:
- [✓] Ref Framework LEET-Security_Controls
  - [✓] Ref Applicability Level 1
  - [ ] Ref Applicability Level 2
  - [ ] Ref Applicability Level 3

- [✓] Ref Framework LEET-Security_Controls
  - [✓] Ref Activity 01. InfoSec Management Program
    - [✓] Ref Activity MS.01. Security trategy and planning
      - [ ] Ref Requirement G[MS.01]01.01-2. Infosec objectives clearly stated and formally approved by executive managemer
      - [ ] Ref Requirement G[MS.01]01.02-3. Infosec process integrated with relevant business processes (tactical goals), that c
      - [ ] Ref Requirement G[MS.01]01.03-3. Infosec management system implemented (plan, organization and resources cont
      - [ ] Ref Requirement G[MS.01]01.04-4. Direction and clear support to infosec initiatives
      - [ ] Ref Requirement G[MS.01]01.05-4. The infosec management process implemented is updated and improved continu
      - [✓] Ref Requirement G[MS.01]01.06-5. Clearly stated infosec strategy
      - [✓] Ref Requirement G[MS.01]02.01-2. Set of security indicators used to know the level of implementation of security me
      - [✓] Ref Requirement G[MS.01]02.02-3. Set of security indicators used to know the number, type and impact of security in
      - [✓] Ref Requirement G[MS.01]02.03-4. Set of security indicators used to measure security efficiency of security system
    - [✓] Ref Activity MS.02. Responsibilities assignment
    - [✓] Ref Activity MS.03. Risk Management

< Back Next > Finish Cancel

Figure 10. Baseline tailoring

We have implemented a filter so that the elements affected by the criticality and the applicability levels of the standard are selected accordingly. Figure 11 shows an example of baseline model automatically generated from the reference framework model.

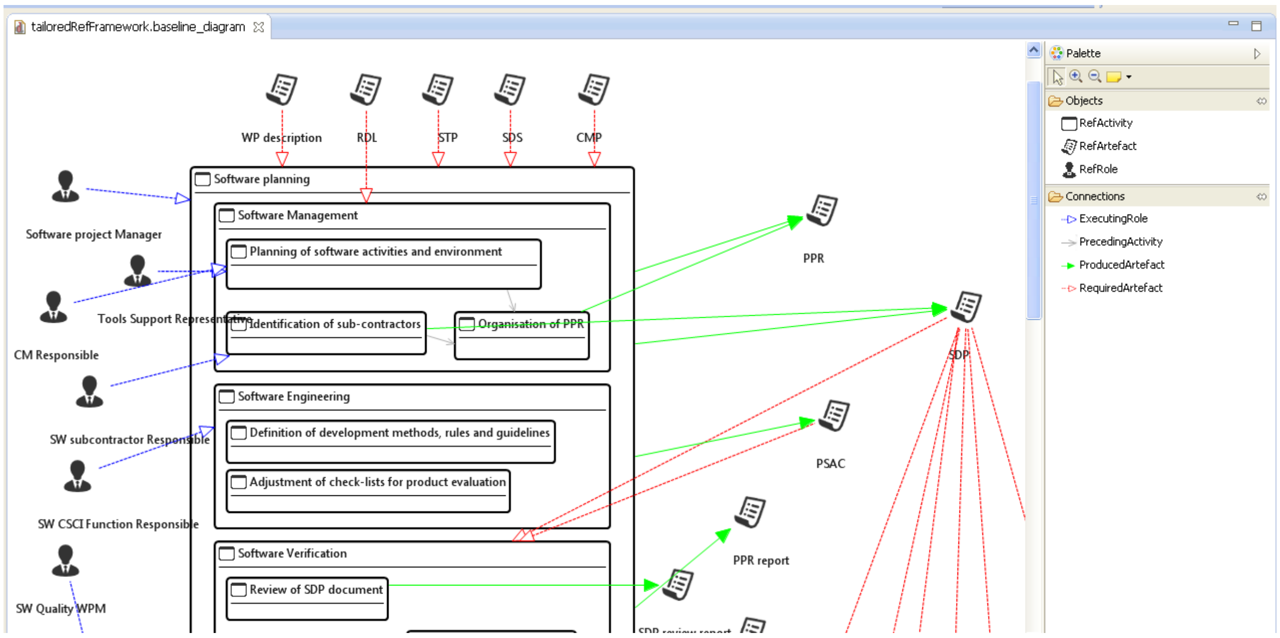


Figure 11. Baseline graphical editor

Transformation of Standard Requirements to Baseline Model: Model-Driven Engineering principles and techniques can facilitate and improve assurance of safety-critical systems by providing support for the transformation of standard requirements modelled (through the customization of the guidance “Practice”) in EPF Composer to Baseline models in OpenCert. The mapping between Unified Method Architecture

(UMA) metamodel [27] and Baseline metamodel is achieved by using Epsilon Transformation Language (ETL) [8]. For that, a plugin has implemented that transform the standard requirements to Baseline models. The main mapping rules for baseline generation are as follows:

- Content Package of EPF Composer is mapped into Base Framework
- Requirements (modelled as practices) are mapped into Base Requirements
- Id, name and description of requirements are mapped into Id, name and description

2.2.3 Process Compliance (informal) management

This chapter describes the implementation for requirement **WP6_CM_008,** described in Table 1.

As described in Section 2.2.2, users can maintain the lifecycle of projects by creating Assurance Projects. Figure 12 illustrates the elements of an Assurance Project. An Assurance Project has three main elements:

1. **Baseline Configuration.** A Baseline Configuration has a set of Baseline Models. A Baseline model represents what is prescribed in a specific assurance project.
2. **Permissions Configuration**. This functionality has been revised for Prototype P2. It supports profile creation to enable restricted access to AMASS functionality and data.
3. **Assurance Assets Package**. This is a pointer to project-specific Artefacts models, Argumentation models, Process models and System models. These four models represent what has been done in a specific assurance project. System Component models are managed by the CHESS toolset (see deliverable D3.4 [15]). The implementation plan for this point has been revised for P2 due to CDO technical problems during P1. Dependencies with external projects have prevented us to make the link for this prototype. We expect to achieve the problems in the last prototype.

The mapping of Assurance Asset Package elements with Baseline Models is specified using the concept of Compliance Map.

Additionally, Evidence and Process models can be created using EPF process planning models (model transformation arrows in Figure 12). This model transformation helps users get a first version of their evidence and execute process models to demonstrate compliance with standards.

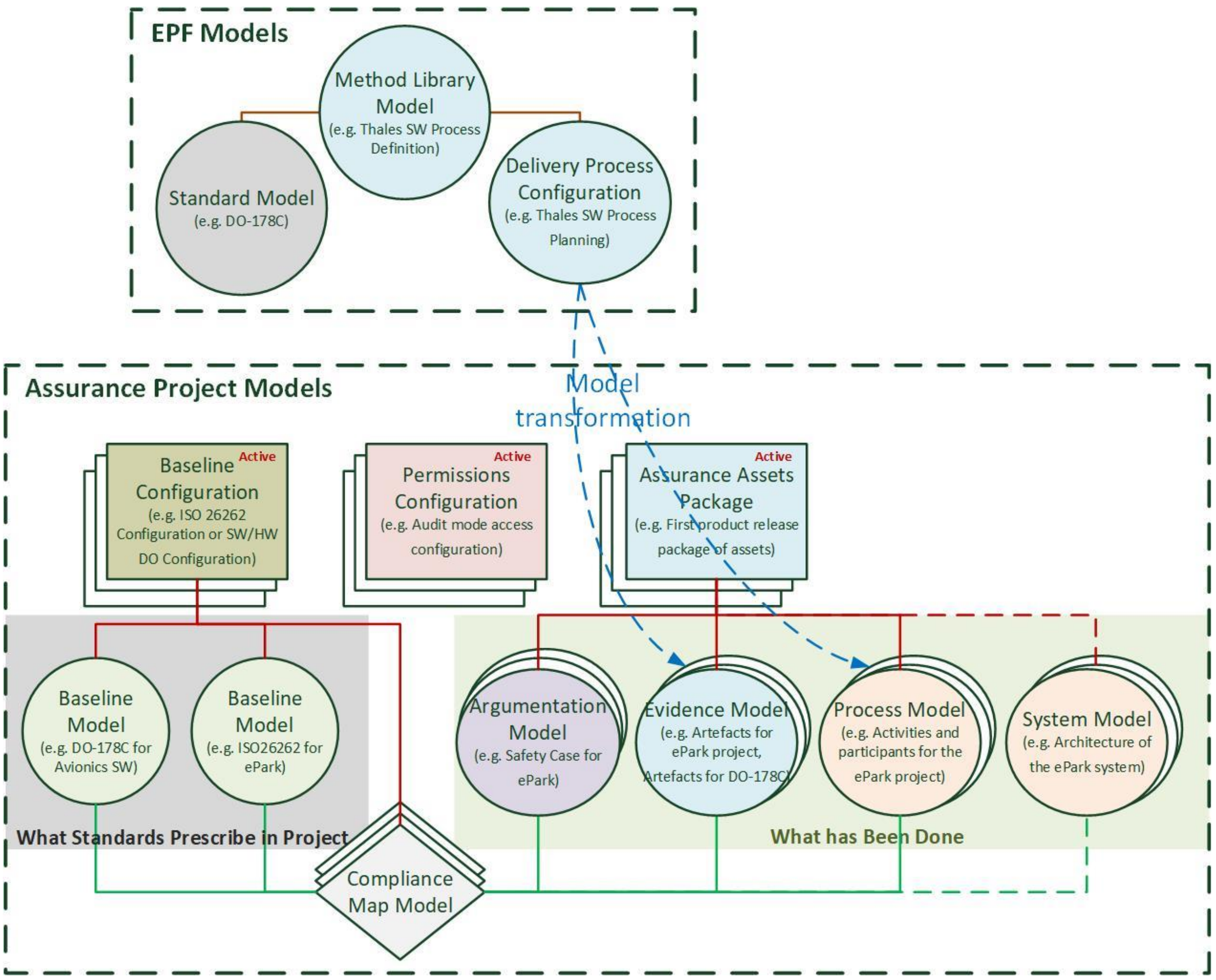
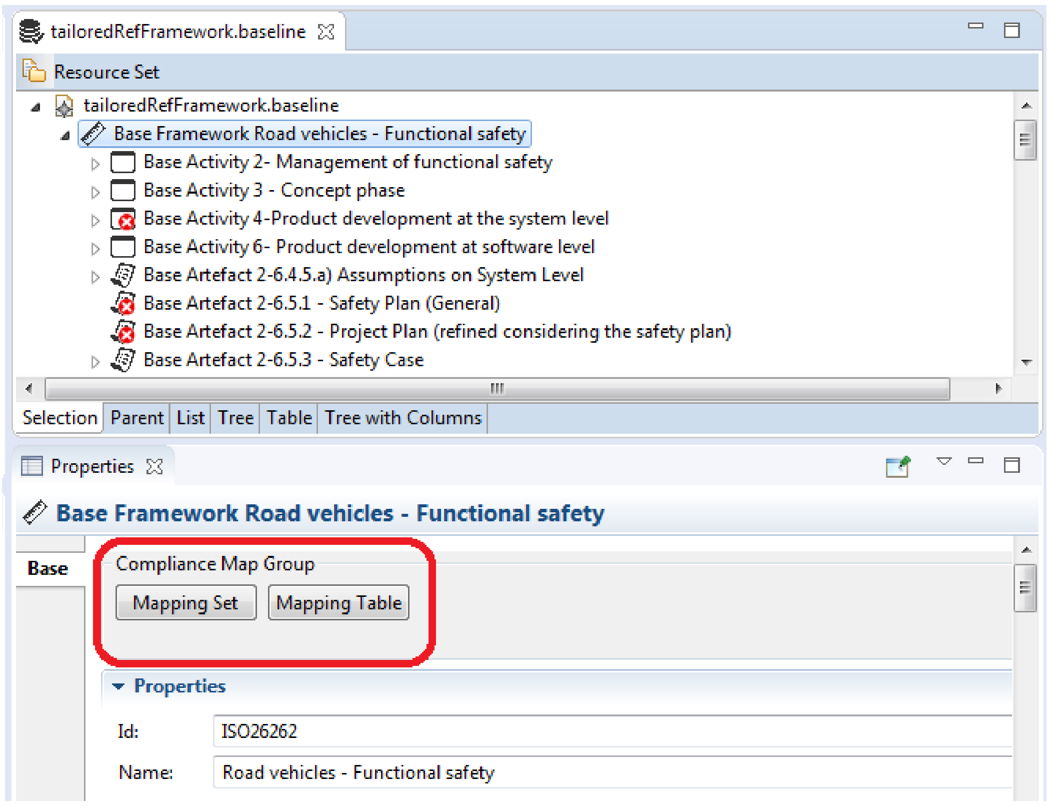

EPF Models
Method Library Model (e.g. Thales SW Process Definition)
Standard Model (e.g. DO-178C)
Delivery Process Configuration (e.g. Thales SW Process Planning)
Model transformation
Assurance Project Models
Baseline Configuration (e.g. ISO 26262 Configuration or SW/HW DO Configuration)
Active
Permissions Configuration (e.g. Audit mode access configuration)
Active
Assurance Assets Package (e.g. First product release package of assets)
Active
Baseline Model (e.g. DO-178C for Avionics SW)
Baseline Model (e.g. ISO26262 for ePark)
What Standards Prescribe in Project
Argumentation Model (e.g. Safety Case for ePark)
Evidence Model (e.g. Artefacts for ePark project, Artefacts for DO-178C)
Process Model (e.g. Activities and participants for the ePark project)
System Model (e.g. Architecture of the ePark system)
What has Been Done
Compliance Map Model

Figure 12. Assurance Project and System Component Specification structure

3.1.2.3 Compliance modelling with OpenCert

Compliance maps can be edited in two different views that can be opened from the Baseline models, as shown in Figure 13. “Mapping Set” allows users to edit each of the compliance maps by using a tree view. “Mapping Table” shows a summary of the compliance maps and their status in the form of a table.



tailoredRefFramework.baseline

Resource Set

- tailoredRefFramework.baseline
  - Base Framework Road vehicles - Functional safety
    - Base Activity 2- Management of functional safety
    - Base Activity 3 - Concept phase
    - Base Activity 4-Product development at the system level
    - Base Activity 6- Product development at software level
    - Base Artefact 2-6.4.5.a) Assumptions on System Level
    - Base Artefact 2-6.5.1 - Safety Plan (General)
    - Base Artefact 2-6.5.2 - Project Plan (refined considering the safety plan)
    - Base Artefact 2-6.5.3 - Safety Case

Selection | Parent | List | Tree | Table | Tree with Columns

Properties

Base Framework Road vehicles - Functional safety

Base

Compliance Map Group

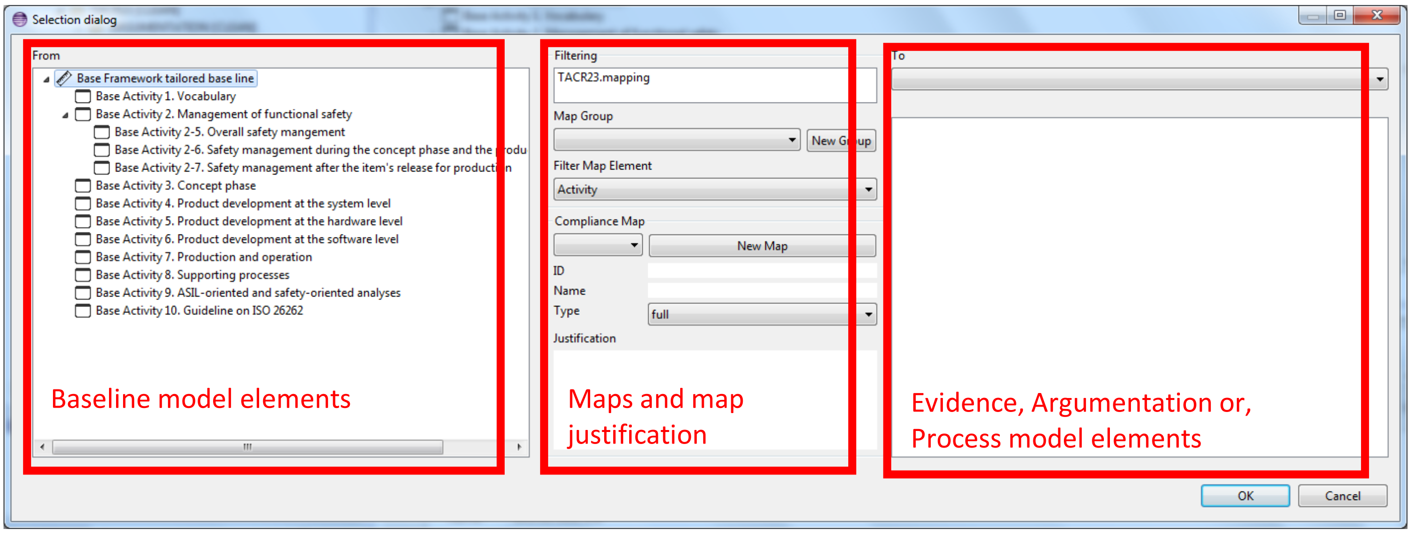
Mapping Set | Mapping Table

Properties

Id:	ISO26262
Name:	Road vehicles - Functional safety

Figure 13. How to create Compliance Maps

Figure 14 shows the Compliance Set view.



Selection dialog

From

- Base Framework tailored base line
  - Base Activity 1. Vocabulary
  - Base Activity 2. Management of functional safety
    - Base Activity 2-5. Overall safety mangement
    - Base Activity 2-6. Safety management during the concept phase and the rodu
    - Base Activity 2-7. Safety management after the item's release for producti n
  - Base Activity 3. Concept phase
  - Base Activity 4. Product development at the system level
  - Base Activity 5. Product development at the hardware level
  - Base Activity 6. Product development at the software level
  - Base Activity 7. Production and operation
  - Base Activity 8. Supporting processes
  - Base Activity 9. ASIL-oriented and safety-oriented analyses
  - Base Activity 10. Guideline on ISO 26262

Filtering: TACR23.mapping

Map Group

Filter Map Element: Activity

Compliance Map

ID	
Name	
Type	full

Justification

To

New Gr up
New Map
OK
Cancel
Baseline model elements
Maps and map justification
Evidence, Argumentation or, Process model elements

Figure 14. Compliance Set view

The Compliance Set view form is organised in three zones:

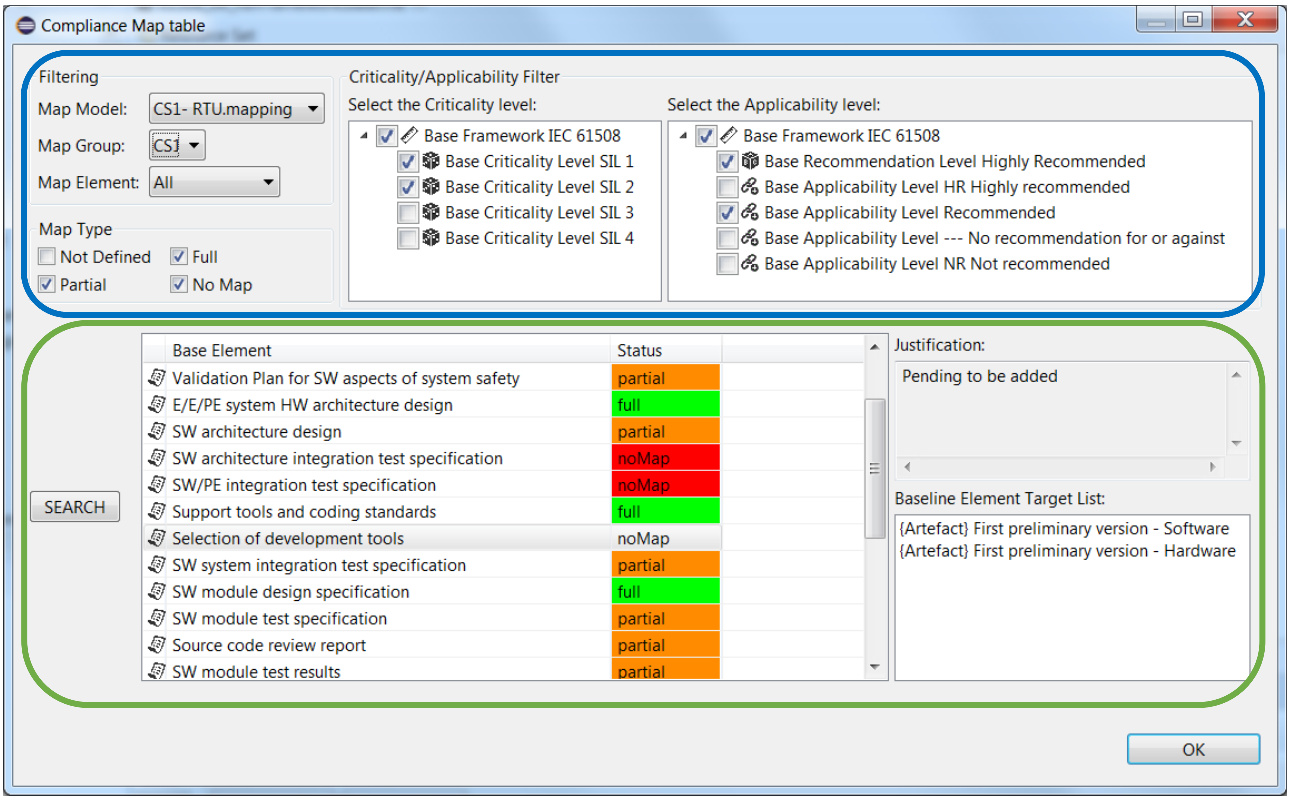
- The *left zone* shows the baseline model elements.
- The *middle zone* allows to make different filters and to add the type of mapping (Full, Partial, No Map) and any mapping justification.
- The *right zone* shows the list of target models and their model elements (evidence, argumentation or process).

It is possible to create compliance maps for activities, artefacts, requirements, roles and techniques, with the following allowed maps:

- BaseArtefact -> Artefact
- BaseRequirement -> Artefact , Claim or Activity
- BaseActivity -> Activity
- BaseRole -> Participant
- BaseTechnique -> Technique

Figure 15 shows the Compliance Table view. The Compliance Map window is organised in two zones:

- The upper part (blue circle) has controls to allow filtering. It is possible to filter by criticality level, applicability level, map model, a map group of the selected map model, a type of element that could be mapped and the mapping type. The *"Not Defined"* option is used to include in the table all the elements of the baseline that have not mapping established; in this way, it is possible to see the compliance Gap. It is necessary to click the Search button to begin the search process based in the filter options selected that will fill the table.
- The lower part (green circle) shows three controls:
  - The compliance mapping table that shows all the baseline elements that accomplish with the searching criteria selected by the user. By default, all the baseline elements of a compliance map are shown in the table.
  - A text box that shows the compliance map justification introduced for the base element selected in the bottom left table with a simple left click.
  - A list that shows all the target elements of the base element selected in the table with a simple left click.


Compliance Map table

Filtering

Map Model: CS1- RTU.mapping

Map Group: CS1

Map Element: All

Map Type

☐ Not Defined ☑ Full

☑ Partial ☑ No Map

Criticality/Applicability Filter

Select the Criticality level:

- ☑ Base Framework IEC 61508
  - ☑ Base Criticality Level SIL 1
  - ☑ Base Criticality Level SIL 2
  - ☐ Base Criticality Level SIL 3
  - ☐ Base Criticality Level SIL 4

Select the Applicability level:

- ☑ Base Framework IEC 61508
  - ☑ Base Recommendation Level Highly Recommended
  - ☐ Base Applicability Level HR Highly recommended
  - ☑ Base Applicability Level Recommended
  - ☐ Base Applicability Level --- No recommendation for or against
  - ☐ Base Applicability Level NR Not recommended

Base Element	Status
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

Justification:

Pending to be added

Baseline Element Target List:

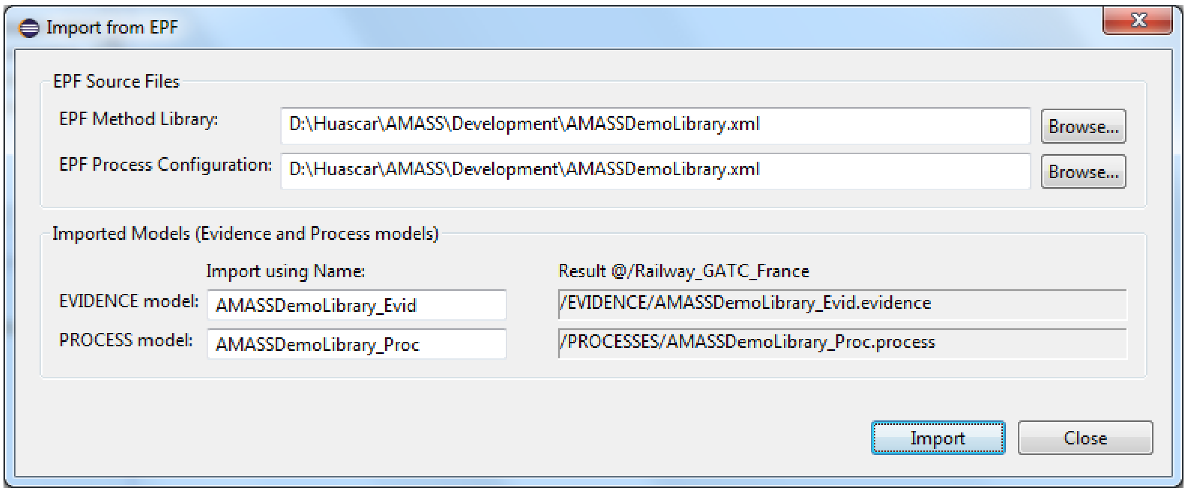
{Artefact} First preliminary version - Software
{Artefact} First preliminary version - Hardware

Figure 15. Mapping Table view

Importing EPF process models into OpenCert for compliance management

It is also possible to export the EPF planned process models and import them in OpenCert. As abovementioned, EPF can be used for modelling the definition and planning of processes. OpenCert also allows users to model processes but looking at post-planning phases. We can get benefit of the EPF process information to create a first view (which can evolve during an assurance project) of process models in OpenCert by importing EPF information into OpenCert. Specifically, the transformation process takes a delivery process modelled in EPF and generates an evidence model and a process model in OpenCert.

Figure 16 shows the Importing View implemented in OpenCert.


Import from EPF

EPF Source Files

EPF Method Library: D:\Huascar\AMASS\Development\AMASSDemoLibrary.xml

EPF Process Configuration: D:\Huascar\AMASS\Development\AMASSDemoLibrary.xml

Imported Models (Evidence and Process models)

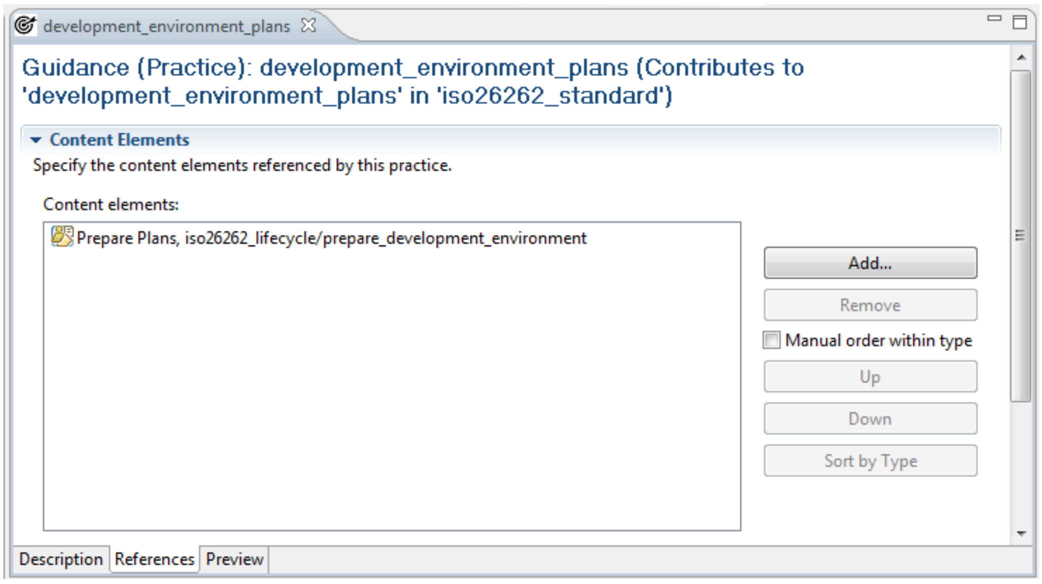
	Import using Name:	Result @/Railway_GATC_France
EVIDENCE model:	AMASSDemoLibrary_Evid	/EVIDENCE/AMASSDemoLibrary_Evid.evidence
PROCESS model:	AMASSDemoLibrary_Proc	/PROCESSES/AMASSDemoLibrary_Proc.process

Figure 16. Import View in OpenCert for EPF: Result of the import operation

3.1.2.4 Compliance mappings in EPF

The compliance of a process with a specific standard can be modelled in the EPF composer as proposed in [22]. The use of EPF for compliance modelling allows planning how our process will address standard requirements.

As a part of the approach presented in [22], it is necessary to define a new method plug-in in the EPF composer that will contain just the requirements of the standard mapped to elements of a process. This procedure makes possible to re-use standards and processes for different compliance mappings. Compliance mappings are modelled in the references tab of the requirements (see Figure 17). In this tab, we can use as evidence for compliance process activities, a portion of a process (i.e. a capability of a pattern) and guidance elements like guidelines, tools or practices. Activities include actions, roles and work products involved in the activity. EPF includes filters and patterns to make the selection of evidence for compliance easier.


development_environment_plans

Guidance (Practice): development_environment_plans (Contributes to 'development_environment_plans' in 'iso26262_standard')

Content Elements

Specify the content elements referenced by this practice.

Content elements:

Prepare Plans, iso26262_lifecycle/prepare_development_environment

Figure 17. Modelling of compliance mappings in the EPF composer

This modelling solution supports three types of compliance: full compliance, partial compliance and no-compliance. The full compliance is modelled providing at least one evidence for requirement like the case of "Document Architecture Alternative" (see Figure 18). The partial compliance is modelled by decomposing requirements in sub-requirements and providing evidence just for the part of the requirements that is accomplished. This is illustrated by the requirement "*Document sys requirements*": the sub-requirement "*Detail a use case*" is fulfilled by the process with the activity "*Detail use case scenarios*", while the sub-requirement "*Identify and outline requirements*" is not fulfilled. Therefore, "*Document sys requirements*" is partially satisfied by the process. Finally, "*Development Environment Multi Part*" does not have any associated evidence, so the process does not address this requirement.

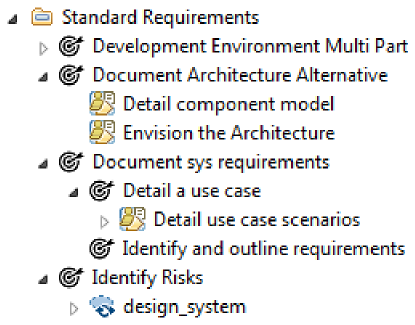

Standard Requirements
Development Environment Multi Part
Document Architecture Alternative
Detail component model
Envision the Architecture
Document sys requirements
Detail a use case
Detail use case scenarios
Identify and outline requirements
Identify Risks
design_system

Figure 18. Mapped Requirements in the EPF composer

The transformation of standard's requirements modelled in EPF Composer (compliant with UMA metamodel) to baseline models (compliant with CACM metamodel) is supported (see Figure 19). The generated models will be stored in selected assurance case project in the CDO repository.

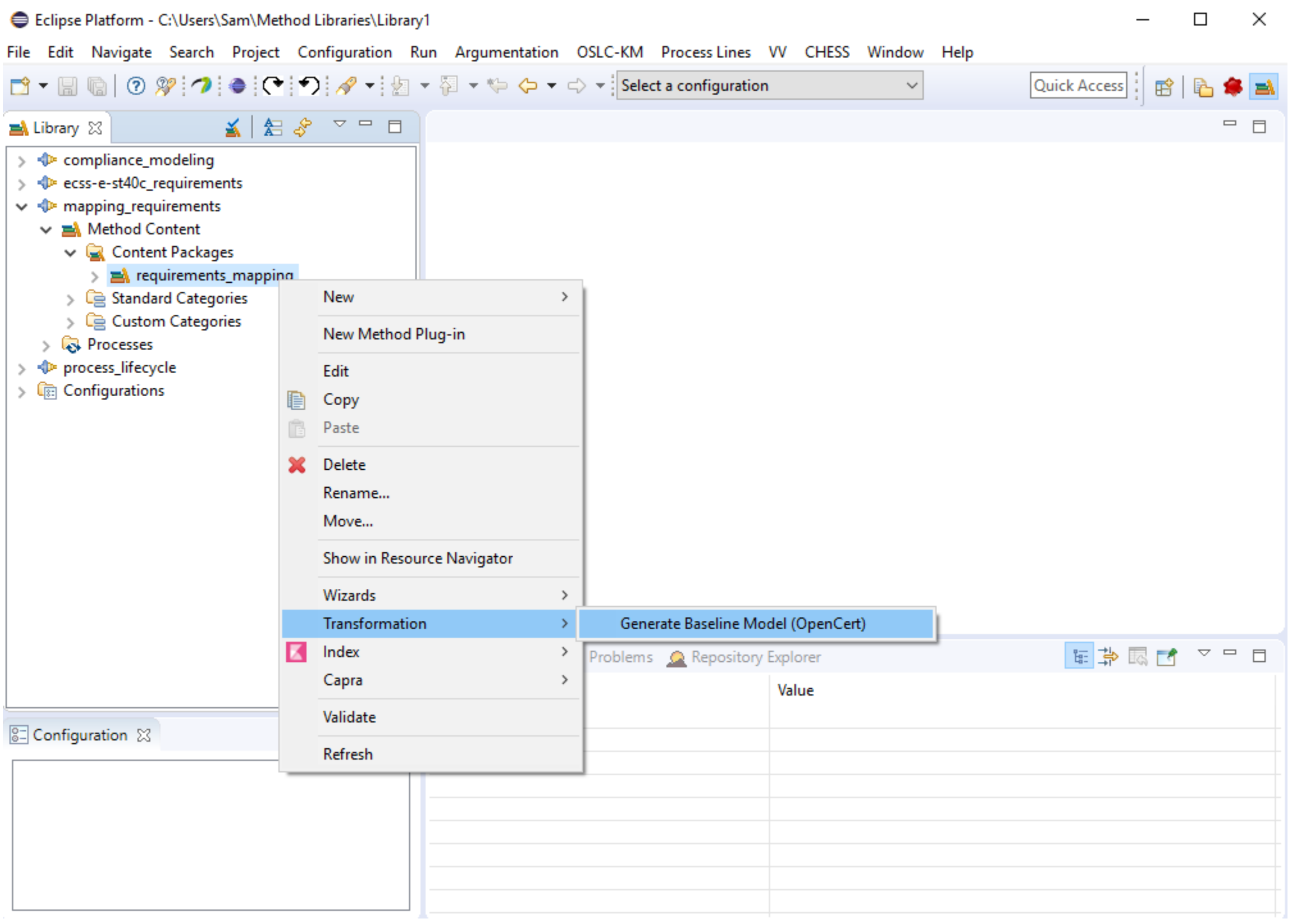


Figure 19. Generating baseline model from EPF mapping requirements

2.2.4 Solution for Process Compliance Checking

This chapter describes the implementation for the requirements **WP6_CM_009** described in the Table 3.

The compliance checking vision is supported by EPF composer [27] and Regorous [33]. To be able to check compliance, Regorous requires three models, namely the execution semantics of the process, the compliance effects annotations, and the rule set. To provide the process descriptions required by Regorous, the creation of three plugins, as done by the IBM approach for mapping standards requirements [34], is the methodology adopted. The three plugins can be seen in Figure 20, and are explained below.

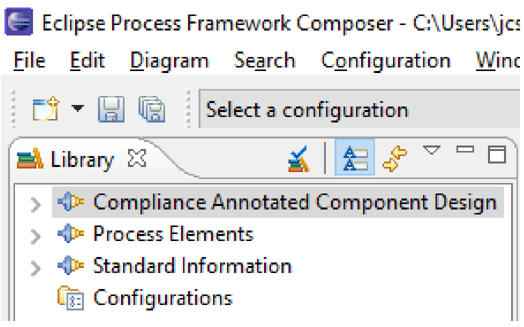

Eclipse Process Framework Composer - C:\Users\jcs
File Edit Diagram Search Configuration Winc
Select a configuration
Library
Compliance Annotated Component Design
Process Elements
Standard Information
Configurations

Figure 20. Plugins required for compliance checking.

1. **<u>Plugin for capturing standard's requirements</u>**: In the method authoring of EPF composer, we capture the standard's requirements using custom categories. The root of the custom categories is the name of the standard. The novelty added to this plugin is that the rule set and the rules are added by using a

customized reusable asset for the former and customized concepts for the latter. The rules are associated to the corresponding standard's requirements.

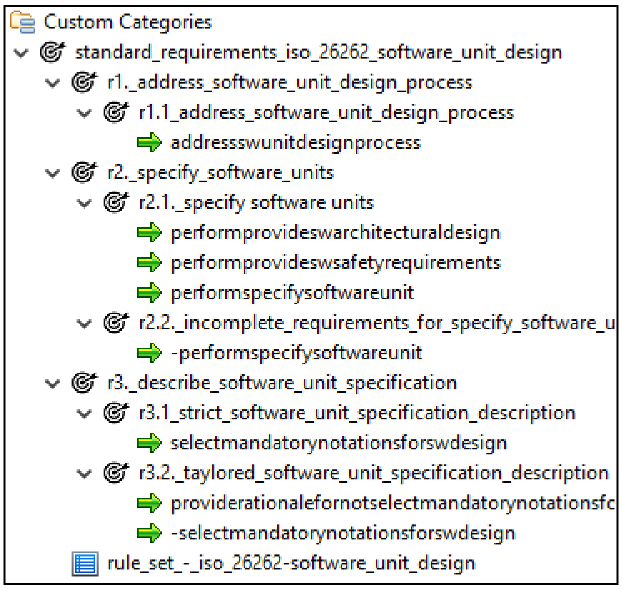

Custom Categories
standard_requirements_iso_26262_software_unit_design
r1._address_software_unit_design_process
r1.1_address_software_unit_design_process
addresswunitdesignprocess
r2._specify_software_units
r2.1._specify software units
performprovideswarchitecturaldesign
performprovideswsafetyrequirements
performspecifysoftwareunit
r2.2._incomplete_requirements_for_specify_software_u
-performspecifysoftwareunit
r3._describe_software_unit_specification
r3.1_strict_software_unit_specification_description
selectmandatorynotationsforswdesign
r3.2._taylored_software_unit_specification_description
providerationalefornotselectmandatorynotationsfc
-selectmandatorynotationsforswdesign
rule_set_-_iso_26262-software_unit_design

Figure 21. Custom Categories

The rule is written in the main description field of the compliance effect, as presented in Figure 22.

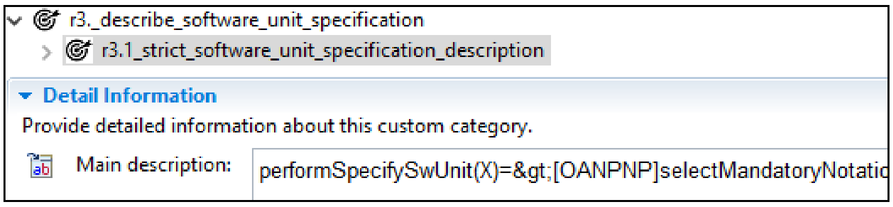

r3._describe_software_unit_specification
r3.1_strict_software_unit_specification_description
Detail Information
Provide detailed information about this custom category.
Main description: performSpecifySwUnit(X)=&gt;[OANPNP]selectMandatoryNotatic

Figure 22. Rule specification

The rule set is defined in a customized reusable asset, which contains the superiority relations between rules, as presented in Figure 23.

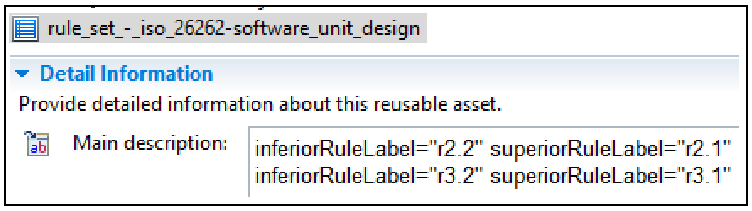

rule_set_-_iso_26262-software_unit_design
Detail Information
Provide detailed information about this reusable asset.
Main description: inferiorRuleLabel="r2.2" superiorRuleLabel="r2.1"
inferiorRuleLabel="r3.2" superiorRuleLabel="r3.1"

Figure 23. Rule set specification

2. **Plugin for capturing process elements**: In the method authoring of EPF composer, we capture the process elements required to support the software process modeling by using the current modeling capabilities of EPF composer (see Figure 24).

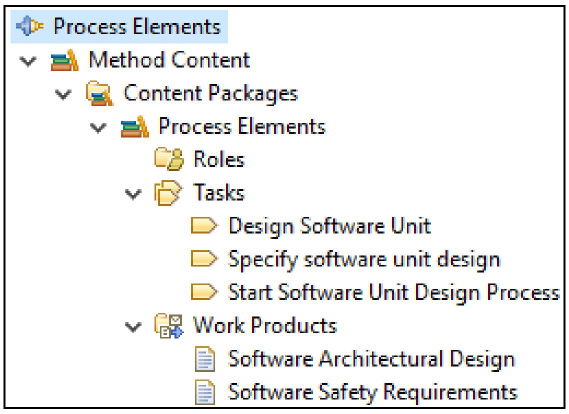

Process Elements
Method Content
Content Packages
Process Elements
Roles
Tasks
Design Software Unit
Specify software unit design
Start Software Unit Design Process
Work Products
Software Architectural Design
Software Safety Requirements

Figure 24. Process Elements Plugin

3. **Plugin for annotating process description**: This plugin is used to match standard's requirements with processes. This plugin contains an extended copy of the tasks defined in the previous plugin (by using *contributes* to the original ones) in the method authoring of EPF composer (see the tasks that are surrounded by a red box in Figure 25).

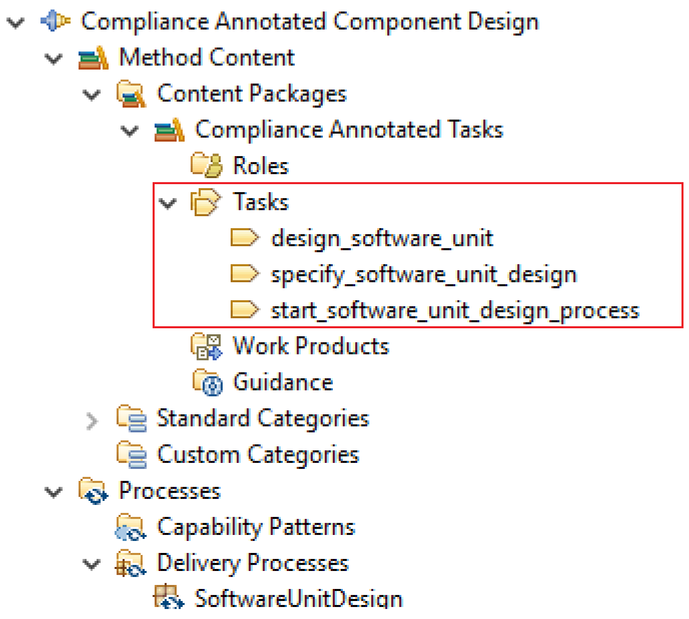

Compliance Annotated Component Design
Method Content
Content Packages
Compliance Annotated Tasks
Roles
Tasks
design_software_unit
specify_software_unit_design
start_software_unit_design_process
Work Products
Guidance
Standard Categories
Custom Categories
Processes
Capability Patterns
Delivery Processes
SoftwareUnitDesign

Figure 25. Annotated process description plugin

Then, tasks are annotated with compliance effects. To annotate the task, we double click on the task to open the task descriptor. Then we click on the tab Guidance and add the compliance effect annotations as presented in Figure 26.

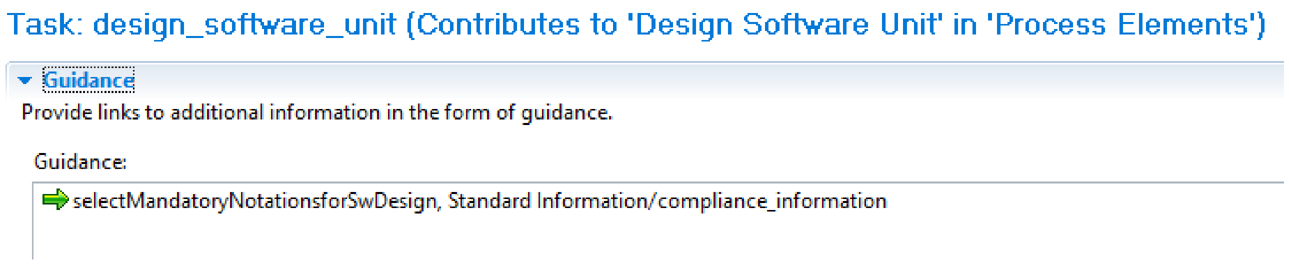
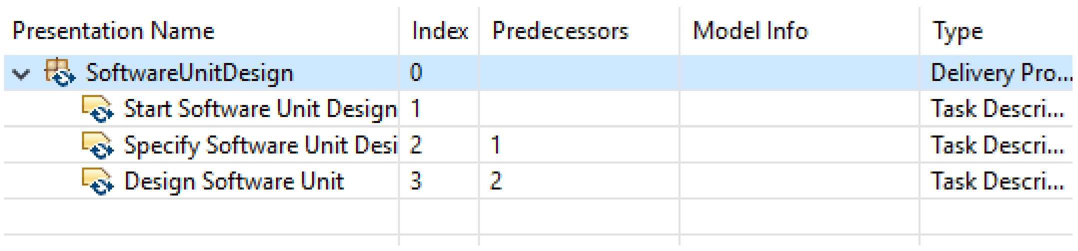

Task: design_software_unit (Contributes to 'Design Software Unit' in 'Process Elements')
Guidance
Provide links to additional information in the form of guidance.
Guidance:
selectMandatoryNotationsforSwDesign, Standard Information/compliance_information

Figure 26. Annotating a task with compliance effects

Once the tasks are annotated, we model the delivery process, by inserting the annotated tasks in the work breakdown structure, as presented in Figure 27.



Presentation Name	Index	Predecessors	Model Info	Type
SoftwareUnitDesign	0			Delivery Pro...
Start Software Unit Design	1			Task Descri...
Specify Software Unit Desi	2	1		Task Descri...
Design Software Unit	3	2		Task Descri...

Figure 27. The work breakdown structure of the delivery process

Then, an activity diagram is created by using the proprietary activity diagram provided by EPF composer (see Figure 28).

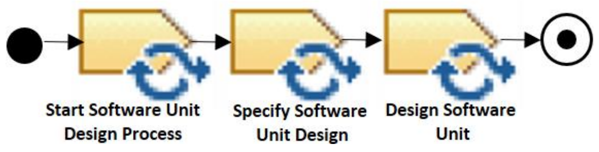

Start Software Unit Design Process
Specify Software Unit Design
Design Software Unit

Figure 28. Activity diagram of the delivery process

Once the plugins are modelled, we export them by using the function “export XML file” provided by EPF composer. From the exported plugins, two files are extracted:

- First, an XMI file (usually called diagram), which describes the activity diagram, is selected for transferring the process description required by Regorous. The elements of interest are an *Activity* that provides the name of the process, an *initial node* and a *final node* that represent the start and end event respectively, one *Activity parameter node* for every task and one *control flow* for every sequence. Other process elements of interest could be a *decision*, *merge*, *and fork* and *join nodes* for modeling *exclusive* and *parallel gateway* respectively, which can be useful for complex processes.
- Second, an XML file, which provides the compliance annotated process information, is also extracted. In The activity name corresponds to the process name. Tasks have associated concepts that correspond to the compliance effects. We can also create the rule set since every concept is described with the actual rule and the reusable asset with the superiority relation.

These two files are transformed to the three models required by Regorous. First, the rule set can be obtained from the *Delivery Process* created in the plugin for the annotated process description (provided by EPF Composer). The corresponding mapping descriptions are listed below.

- Reusable Asset, a type of content element, is transformed into the rule set. The attributes transferred are *name*, *presentationName* and *briefDescription*.
- Concept, a type of content element, is transformed into the Terms. The attribute transferred is *name*.
- Each content category that contains a rule in the field brief description is transformed into a rule. The attributes transferred are *name*, *presentationName*, and *briefDescription*.

Regorous also requires the representation of the process in a canonical form (CPF). Within AMASS, such representation is given via EPF Composer-supported representation, which is based on UML 2.0 Diagram Interchange Specification. Below, the mapping between CPF and UML is given.

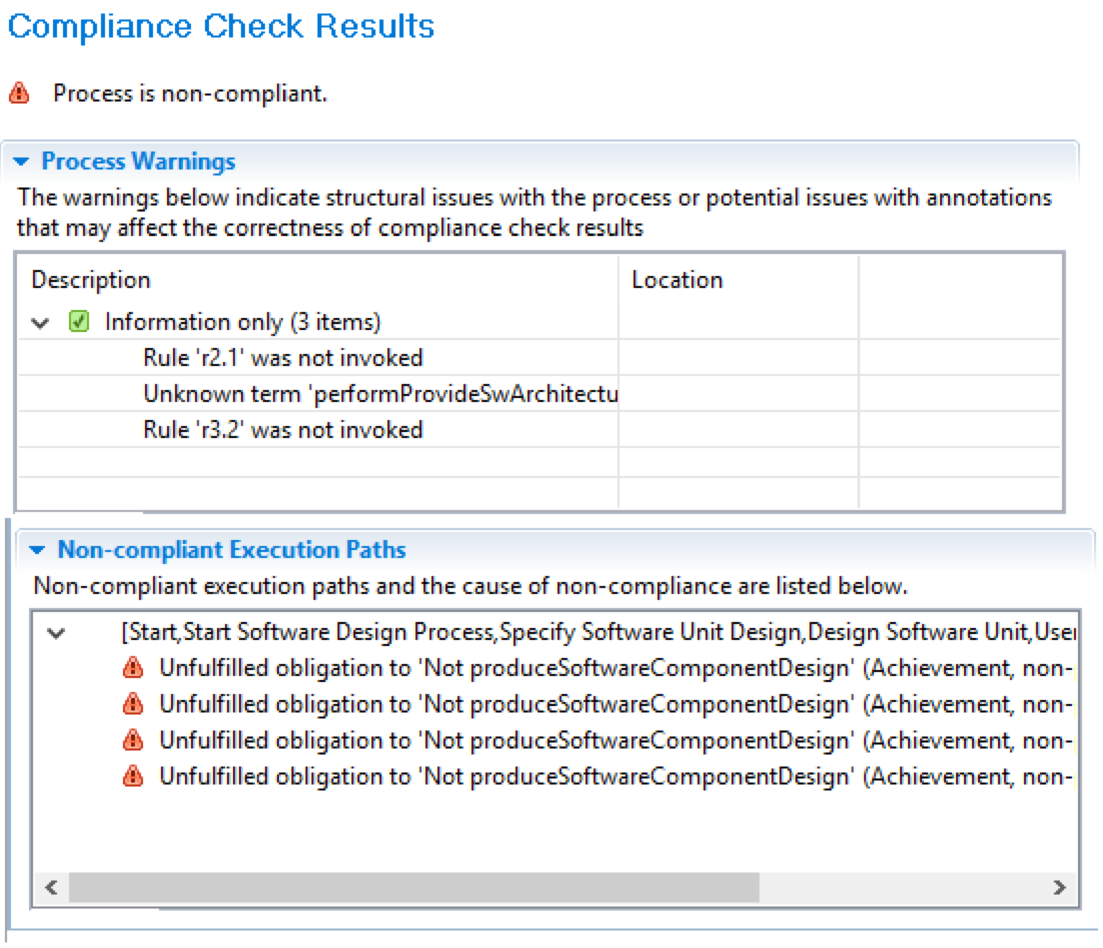
- Activity information is transformed to a canonical process in CPF. The attribute transferred is *id*.
- The Initial Node becomes a node with type start event in CPF.
- Each Activity Parameter Node becomes a task type in the CPF. Attributes transferred are id and name.
- Each Control Flow becomes an edge in CPF. Attributes transferred are id, name, source and target.

- The Activity Final Node becomes an end event type in CPF.
- The Decision Node becomes an XORSplitType in CPF.
- The Merge Node becomes an XORJoinType in CPF.
- The Fork Node becomes an ANDSplitType in CPF.
- The Join Node becomes an ANDJoinType in CPF.

Finally, the compliance effects annotations require a structure that complies with the Regorous schema called Compliance Check Annotations. This information can be retrieved from EPF Composer taking into account that the process elements can be extracted from the process structure (described with UML elements) and the compliance effects annotations can be extracted from the delivery process (described with UMA elements). The corresponding matching elements description is listed below.

- A reusable asset becomes a ruleSetList. The attribute transferred is the *name*.
- Each edge becomes a special element in the compliance annotations file called conditions. The attribute transferred is the *id*.
- The node becomes a Task Effects. The attribute transferred is the *id*. Then, the id is also used to search for the concepts that should be converted into the compliance effects in the delivery process file.
- Every concept associated to the task is transferred to the Effect. The attribute transferred is the *name*.

The generated files are provided as input to Regorous, which is able to check compliance and generate a compliance report as is shown in Figure 29.


Compliance Check Results

⚠ Process is non-compliant.

Process Warnings

The warnings below indicate structural issues with the process or potential issues with annotations that may affect the correctness of compliance check results

Description	Location	
✓ Information only (3 items)		
Rule 'r2.1' was not invoked		
Unknown term 'performProvideSwArchitectu		
Rule 'r3.2' was not invoked		

Non-compliant Execution Paths

Non-compliant execution paths and the cause of non-compliance are listed below.

```
[Start,Start Software Design Process,Specify Software Unit Design,Design Software Unit,Use
  ⚠ Unfulfilled obligation to 'Not produceSoftwareComponentDesign' (Achievement, non-
  ⚠ Unfulfilled obligation to 'Not produceSoftwareComponentDesign' (Achievement, non-
  ⚠ Unfulfilled obligation to 'Not produceSoftwareComponentDesign' (Achievement, non-
  ⚠ Unfulfilled obligation to 'Not produceSoftwareComponentDesign' (Achievement, non-
```

Figure 29. Compliance checking result provided by Regorous

2.2.5 Compliance Monitoring

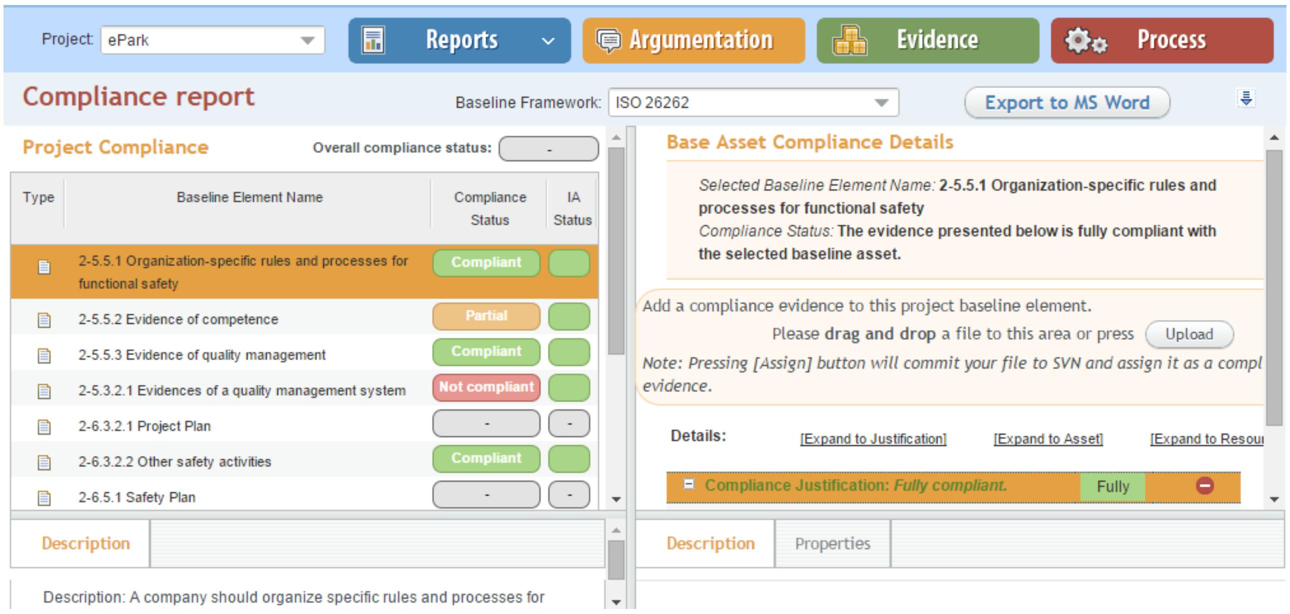
This chapter describes the implementation for requirement **WP6_CM_005,** described in Table 1.

Compliance report provides extensive functionality that helps the AMASS platform users to assess the current compliance of their project to the selected safety standard (i.e., baseline). Two modes of the report can be distinguished:

- An **interactive mode**, where the user can actively browse the report, select the specific baseline items, view their properties, their compliance mapping and the associated evidence, and add or remove the evidence resources mapped to the specific baseline element.
- A **printer** friendly **report** - which is a textual output presenting all the information of the current compliance of the selected project.

Figure 30 shows an example of compliance report in interactive mode. The "**Project Compliance**" table, which is placed in the left, presents base artefacts and base activities of the selected standard. The most important column is the "**Compliance Status**" one, which presents the overall compliance status of a project to the specific standard item. This column can be sorted by value, thus allowing user to assess the project compliance at one glance. In case base activities or base artefacts are defined to have a parent-child hierarchy, this relation is presented accordingly in a tree structure of the table.

In addition, users can look at the list of Argumentation, Evidence and Process model elements in the respective menu options at the top right corner of the web view.



Compliance report

Baseline Framework: ISO 26262

Project Compliance

Overall compliance status: -

Type	Baseline Element Name	Compliance Status	IA Status
🗎	2-5.5.1 Organization-specific rules and processes for functional safety	Compliant	🟩
🗎	2-5.5.2 Evidence of competence	Partial	🟩
🗎	2-5.5.3 Evidence of quality management	Compliant	🟩
🗎	2-5.3.2.1 Evidences of a quality management system	Not compliant	🟩
🗎	2-6.3.2.1 Project Plan	-	-
🗎	2-6.3.2.2 Other safety activities	Compliant	🟩
🗎	2-6.5.1 Safety Plan	-	-

Description: A company should organize specific rules and processes for

Base Asset Compliance Details

Selected Baseline Element Name: **2-5.5.1 Organization-specific rules and processes for functional safety**
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
Note: Pressing [Assign] button will commit your file to SVN and assign it as a compl evidence.

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

Compliance Justification: *Fully compliant.* Fully

Figure 30. Compliance report in web client

2.2.6 Reuse Assistance

This chapter describes the implementation for requirements **WP6_RA_001**, **WP6_RA_002, WP6_RA_003, WP6_RA_004, and WP6_RA_005,** described in the Table 2.

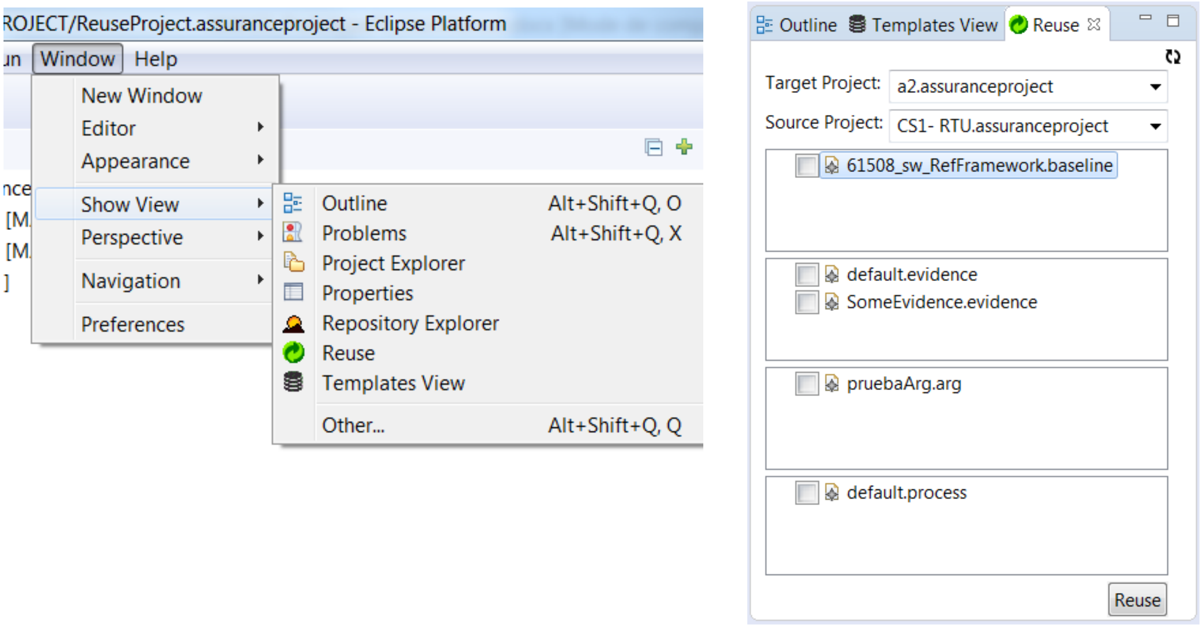
The reuse assistance functionality concerns intra and cross-domain reuse of assurance and certification assets. The Reuse Assistant functionality includes cross-system reuse and cross-standard reuse, as described in the following subsections.

3.1.2.5 Cross-systems reuse (intra standard or intra domain product upgrade)

This functionality implies reuse of assurance assets when a product or system evolves in terms of functionality or technology, e.g. product upgrade. Product upgrade corresponds to a development scenario in which an already-assessed system is modified and thus a new assessment (e.g., re-certification) is required. For example, a new system can be developed based on an existing one. Such a new system can include, for instance, some new components. We assume that the reusable assurance assets were compliant with the same standards we target in the new scenario.

We have implemented this functionality by using a specific view called "Reuse" in OpenCert. It is used to reuse models from one source assurance project to a target assurance project. This view is particularly useful to reuse a subset of model elements, which can be selected manually by the user.

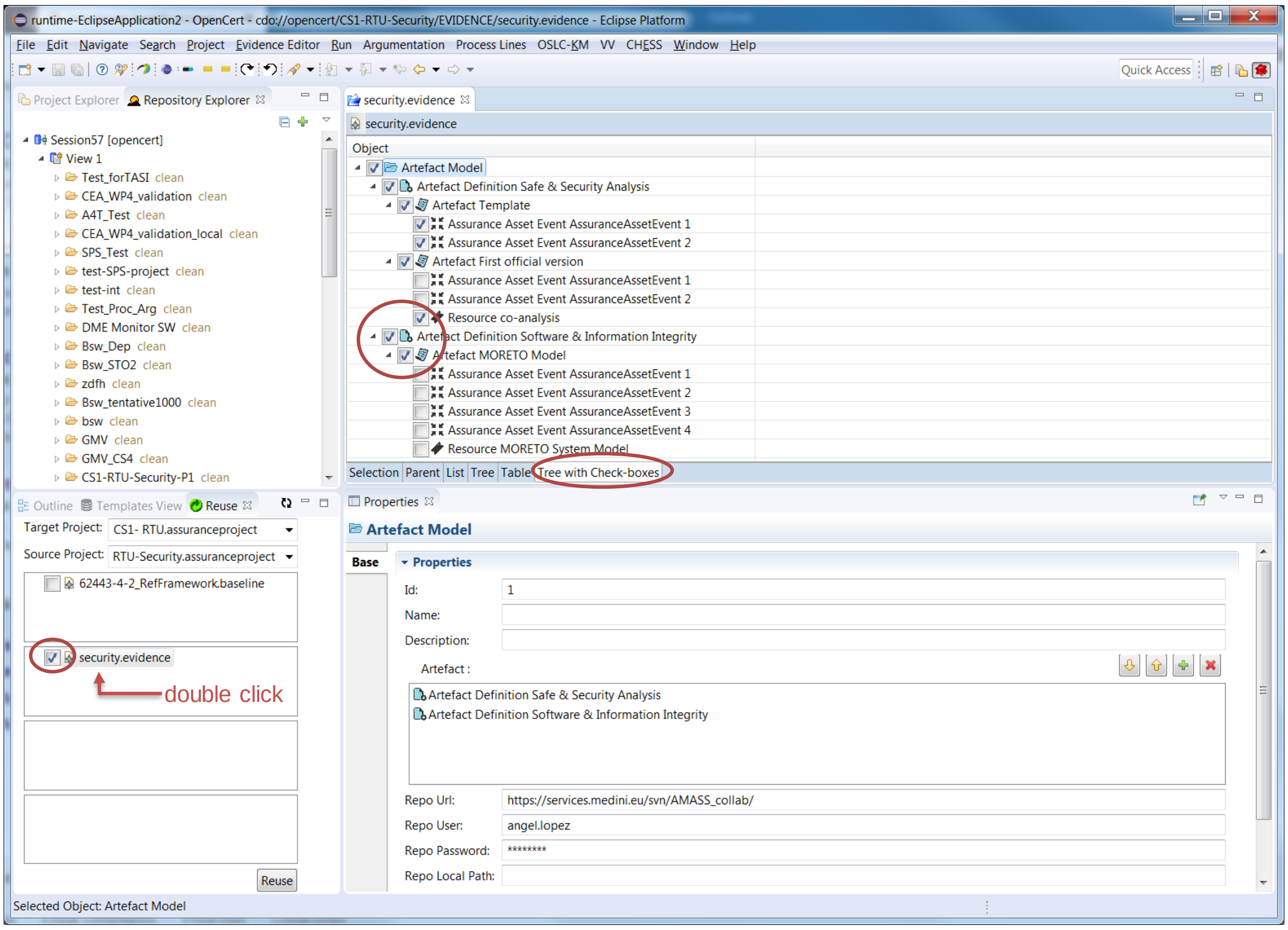
Figure 31 and Figure 32 show how to open the Reuse View and how to select model elements to be reused. The Reuse view is also integrated in the set of OpenCert views and it will be already opened when the OpenCert perspective is activated.



Target Project: a2.assuranceproject

Source Project: CS1- RTU.assuranceproject

Figure 31. Cross-project Reuse View


runtime-EclipseApplication2 - OpenCert - cdo://opencert/CS1-RTU-Security/EVIDENCE/security.evidence - Eclipse Platform
File Edit Navigate Search Project Evidence Editor Run Argumentation Process Lines OSLC-KM VV CHESS Window Help
Quick Access
Project Explorer
Repository Explorer
security.evidence
Selection Parent List Tree Table Tree with Check-boxes
Outline Templates View Reuse
Properties
Base
Reuse
double click

Session57 [opencert]
View 1
Test_forTASI clean
CEA_WP4_validation clean
A4T_Test clean
CEA_WP4_validation_local clean
SPS_Test clean
test-SPS-project clean
test-int clean
Test_Proc_Arg clean
DME Monitor SW clean
Bsw_Dep clean
Bsw_STO2 clean
zdfh clean
Bsw_tentative1000 clean
bsw clean
GMV clean
GMV_CS4 clean
CS1-RTU-Security-P1 clean

security.evidence
Object
- [✓] Artefact Model
  - [✓] Artefact Definition Safe & Security Analysis
    - [✓] Artefact Template
      - [✓] Assurance Asset Event AssuranceAssetEvent 1
      - [✓] Assurance Asset Event AssuranceAssetEvent 2
    - [✓] Artefact First official version
      - [ ] Assurance Asset Event AssuranceAssetEvent 1
      - [ ] Assurance Asset Event AssuranceAssetEvent 2
      - [✓] Resource co-analysis
  - [✓] Artefact Definition Software & Information Integrity
    - [✓] Artefact MORETO Model
      - [ ] Assurance Asset Event AssuranceAssetEvent 1
      - [ ] Assurance Asset Event AssuranceAssetEvent 2
      - [ ] Assurance Asset Event AssuranceAssetEvent 3
      - [ ] Assurance Asset Event AssuranceAssetEvent 4
      - [ ] Resource MORETO System Model

Target Project: CS1- RTU.assuranceproject
Source Project: RTU-Security.assuranceproject
[ ] 62443-4-2_RefFramework.baseline
[✓] security.evidence

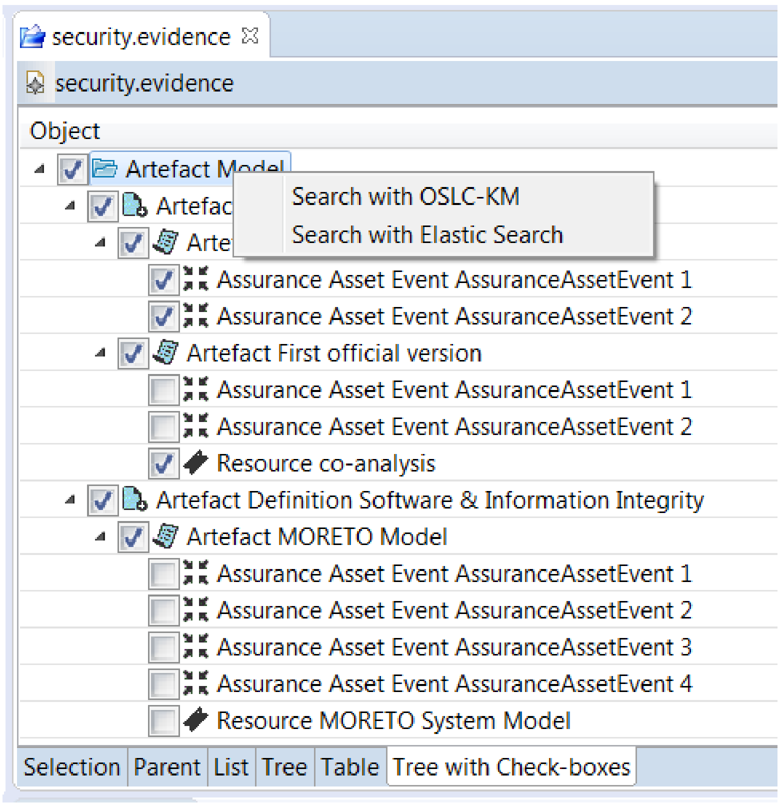
Artefact Model
Properties
Id: 1
Name:
Description:
Artefact :
Artefact Definition Safe & Security Analysis
Artefact Definition Software & Information Integrity
Repo Url: https://services.medini.eu/svn/AMASS_collab/
Repo User: angel.lopez
Repo Password: ********
Repo Local Path:

Selected Object: Artefact Model

Figure 32. Using the Reuse View

The AMASS Prototype P2 version allows users through a context menu (see Figure 33) search a specific subset of an evidence model elements using two reuse discovery approaches. The first one is related to OSLC-KM module (see chapter Reuse Discovery based on OSLC-KM) and the second one related to Elastic Search (see Reuse Discovery based on ElasticSearch).

The user can search reusable assets in the selected evidence model according a text and/or a selected critically level and/or a selected applicability (see Figure 34). Those criticality and applicability levels are specified by the standard that the source project is compliant with. If the user activates the bottom option, the elements to reuse will be select automatically according the results of the search, modifying the previous subset of model elements selected.



security.evidence

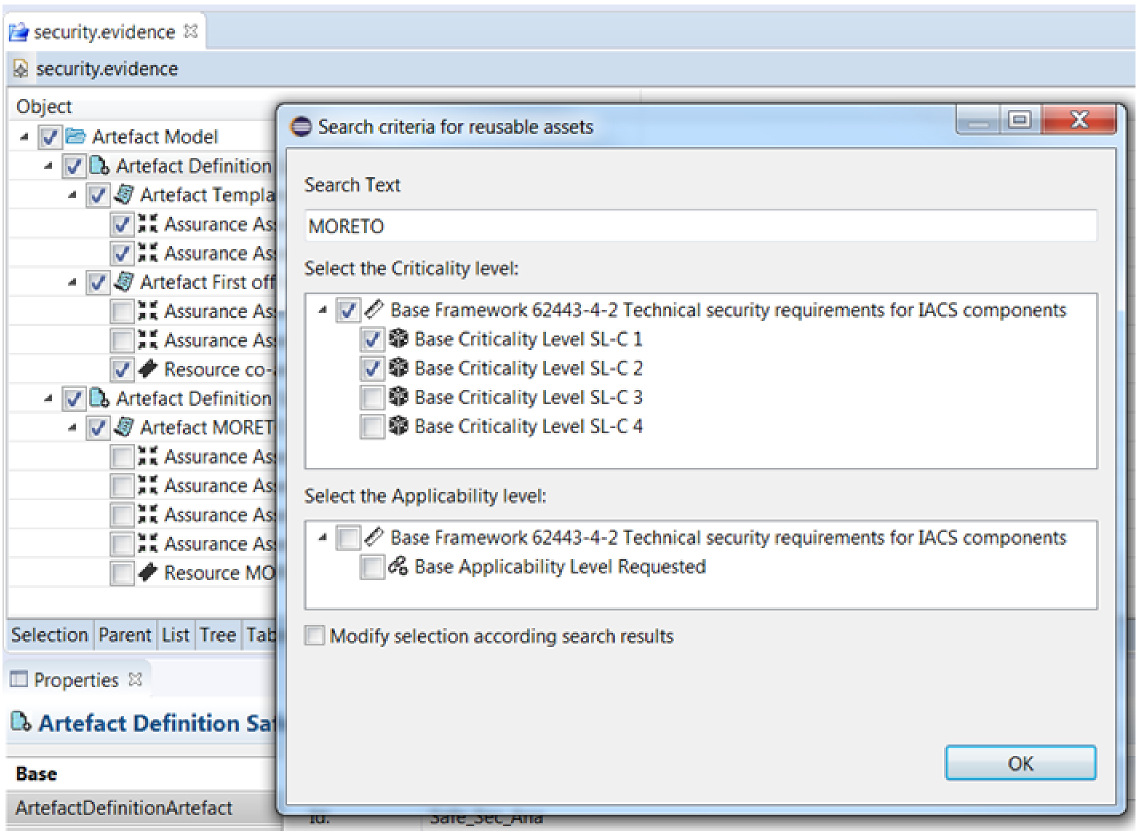
security.evidence

Object

- ✓ Artefact Model
  - ✓ Artefac
    - ✓ Arte
      - ✓ Assurance Asset Event AssuranceAssetEvent 1
      - ✓ Assurance Asset Event AssuranceAssetEvent 2
    - ✓ Artefact First official version
      - Assurance Asset Event AssuranceAssetEvent 1
      - Assurance Asset Event AssuranceAssetEvent 2
      - ✓ Resource co-analysis
  - ✓ Artefact Definition Software & Information Integrity
    - ✓ Artefact MORETO Model
      - Assurance Asset Event AssuranceAssetEvent 1
      - Assurance Asset Event AssuranceAssetEvent 2
      - Assurance Asset Event AssuranceAssetEvent 3
      - Assurance Asset Event AssuranceAssetEvent 4
      - Resource MORETO System Model

Search with OSLC-KM
Search with Elastic Search
Selection | Parent | List | Tree | Table | Tree with Check-boxes

Figure 33. Context menu to choose a search technology



Search criteria for reusable assets

Search Text

MORETO

Select the Criticality level:

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

Select the Applicability level:

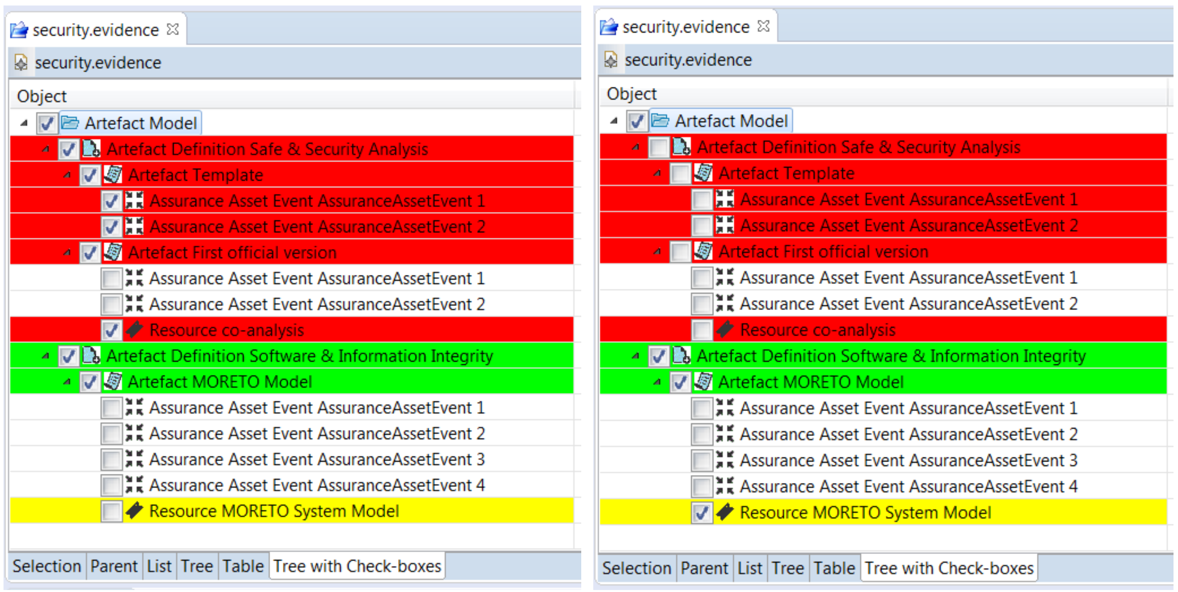
- Base Framework 62443-4-2 Technical security requirements for IACS components
  - Base Applicability Level Requested

☐ Modify selection according search results

Figure 34. Window to introduce the search parameters

The results are shown to the user with a colour code (see Figure 35).

- In green. The previously selected asset, by the user or by a previous search, is a good candidate for reusing according the introduced searching criteria.
- In red. The previously selected asset, by the user asset or by a previous search, does not comply with the introduced searching criteria and therefore should be unselected to avoid its reuse.
- In yellow. The previously not selected asset, by the user asset or by a previous search, is a good candidate for reusing and therefore should be selected to be reused.



```
security.evidence
security.evidence
Object
  ✓ Artefact Model
    ✓ Artefact Definition Safe & Security Analysis
      ✓ Artefact Template
          ✓ Assurance Asset Event AssuranceAssetEvent 1
          ✓ Assurance Asset Event AssuranceAssetEvent 2
      ✓ Artefact First official version
          ☐ Assurance Asset Event AssuranceAssetEvent 1
          ☐ Assurance Asset Event AssuranceAssetEvent 2
          ✓ Resource co-analysis
    ✓ Artefact Definition Software & Information Integrity
      ✓ Artefact MORETO Model
          ☐ Assurance Asset Event AssuranceAssetEvent 1
          ☐ Assurance Asset Event AssuranceAssetEvent 2
          ☐ Assurance Asset Event AssuranceAssetEvent 3
          ☐ Assurance Asset Event AssuranceAssetEvent 4
          ☐ Resource MORETO System Model
Selection | Parent | List | Tree | Table | Tree with Check-boxes
```

```
security.evidence
security.evidence
Object
  ✓ Artefact Model
    ☐ Artefact Definition Safe & Security Analysis
      ☐ Artefact Template
          ☐ Assurance Asset Event AssuranceAssetEvent 1
          ☐ Assurance Asset Event AssuranceAssetEvent 2
      ☐ Artefact First official version
          ☐ Assurance Asset Event AssuranceAssetEvent 1
          ☐ Assurance Asset Event AssuranceAssetEvent 2
          ☐ Resource co-analysis
    ✓ Artefact Definition Software & Information Integrity
      ✓ Artefact MORETO Model
          ☐ Assurance Asset Event AssuranceAssetEvent 1
          ☐ Assurance Asset Event AssuranceAssetEvent 2
          ☐ Assurance Asset Event AssuranceAssetEvent 3
          ☐ Assurance Asset Event AssuranceAssetEvent 4
          ✓ Resource MORETO System Model
Selection | Parent | List | Tree | Table | Tree with Check-boxes
```

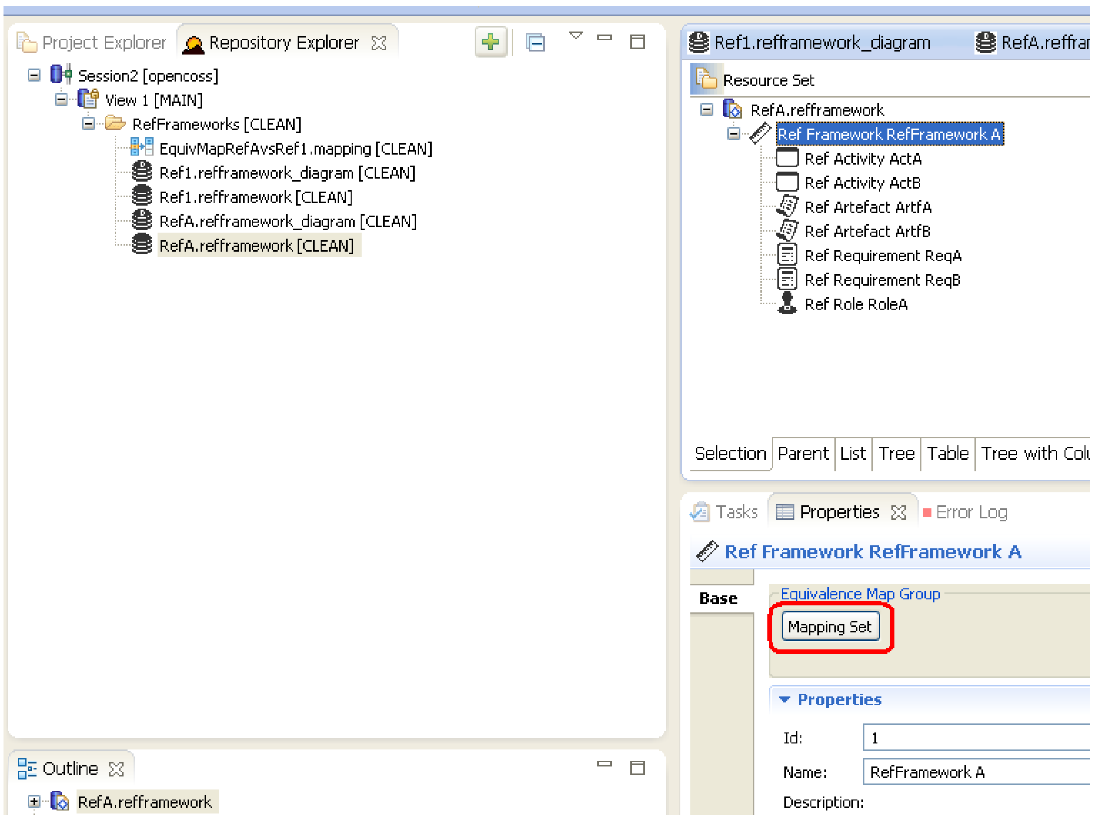
Figure 35. Results of a search (right one with automatic selection option active)

3.1.2.6 Cross-standard reuse (cross-concern or cross-domain)

This functionality is related to the reuse of assurance assets from a project that was completed in compliance with a different dependability concern (e.g., security-compliant assurance project reused from a safety-compliant assurance project) or different domain (e.g. avionics-compliant assurance project reused from an automotive compliant assurance project). The second standard could correspond to a new standard, a new version of a standard, or a different interpretation of a standard (e.g., by a different certification authority).

For cross-standard reuse, AMASS enables reuse of assurance assets of one assurance project in another project, when they relate to different industry domains, dependability concerns or industry standards in general. To perform cross-standard reuse, an **Equivalence Map** model must be created between the source and the target standard models. The **Reuse Assistant** provides information on the reuse opportunities as result of the equivalence relationships. Once the actor selects the assurance assets to be reused, the reuse operation itself can be executed. A module for compliance gap analysis allows AMASS users to look at the reuse post-conditions identified in the equivalence map model.

To create Equivalence Maps, we have developed a tailored functionality in OpenCert. To open it, we provide a menu called "Mapping Set" on the properties form of the reference framework using the tree view editor (see Figure 36).



Project Explorer | Repository Explorer

- Session2 [opencoss]
  - View 1 [MAIN]
    - RefFrameworks [CLEAN]
      - EquivMapRefAvsRef1.mapping [CLEAN]
      - Ref1.refframework_diagram [CLEAN]
      - Ref1.refframework [CLEAN]
      - RefA.refframework_diagram [CLEAN]
      - RefA.refframework [CLEAN]

Outline: RefA.refframework

Ref1.refframework_diagram | RefA.reffrar

Resource Set

- RefA.refframework
  - Ref Framework RefFramework A
    - Ref Activity ActA
    - Ref Activity ActB
    - Ref Artefact ArtfA
    - Ref Artefact ArtfB
    - Ref Requirement ReqA
    - Ref Requirement ReqB
    - Ref Role RoleA

Selection | Parent | List | Tree | Table | Tree with Colu

Tasks | Properties | Error Log

Ref Framework RefFramework A

Base

Equivalence Map Group: Mapping Set

Properties

Id: 1

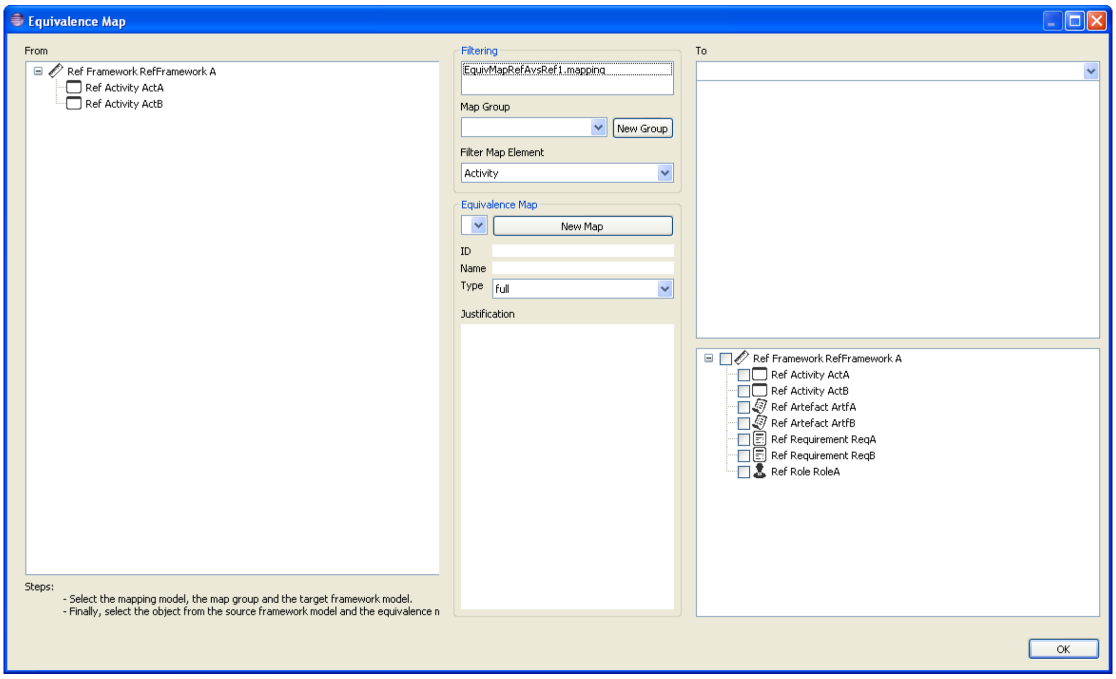
Name: RefFramework A

Description:

Figure 36. How to create Equivalence Map.

Figure 37 shows the form for Equivalence Map. The Equivalence Map form is organised in three zones:

- The *left zone* shows the actual reference framework, and it loads the type of elements for which we want to make the equivalence maps.
- The *middle zone* allows to make different filters like:
  - *Filter Mapping Model* lists all the mapping models stored in the database. It will be necessary to select one of them and one group model.
  - *Filter Map Element*. It is possible to create equivalence maps for activities, artefacts, requirements, roles and techniques.
  - *Filter Equivalence Map*. This filter allows making different equivalence maps for the same refframework element.
  - The user must also introduce the mapping information in the middle part; this information consists ofthe ID, the name, the type and a justification text.
- The *right zone* shows two lists and a combo box.
  - *The combo box* shows all the database refframeworks to select the reference framework that will be the target of the equivalence map to create.
  - The *upper list* loads the elements, according to the filter selected, of the refframework chosen in the combo box that will be the target of the equivalence map to create.
  - The *lower list* displays the full content (not filtered) of the source refframework that will be postConditions in case of reusing. The postConditions are mandatory extra activities, not included in the standard, that must be performed in case of reusing the target element from one assurance project based in the target refframework in another assurance project based in the source refframework using the cross-domain functionality.



Filtering: EquivMapRefAvsRef1.mapping
Filter Map Element: Activity
Type: full

Steps:
- Select the mapping model, the map group and the target framework model.
- Finally, select the object from the source framework model and the equivalence n

Figure 37. Equivalence Map form.

For the reuse assistant, it is mandatory that the target assurance project is based on a refframework with equivalence maps associated to the refframework, in which the source assurance project is based. Figure 38 shows how to open the reuse assistant for cross-domain reuse. We focus on the evidence reuse for this version of AMASS prototype.

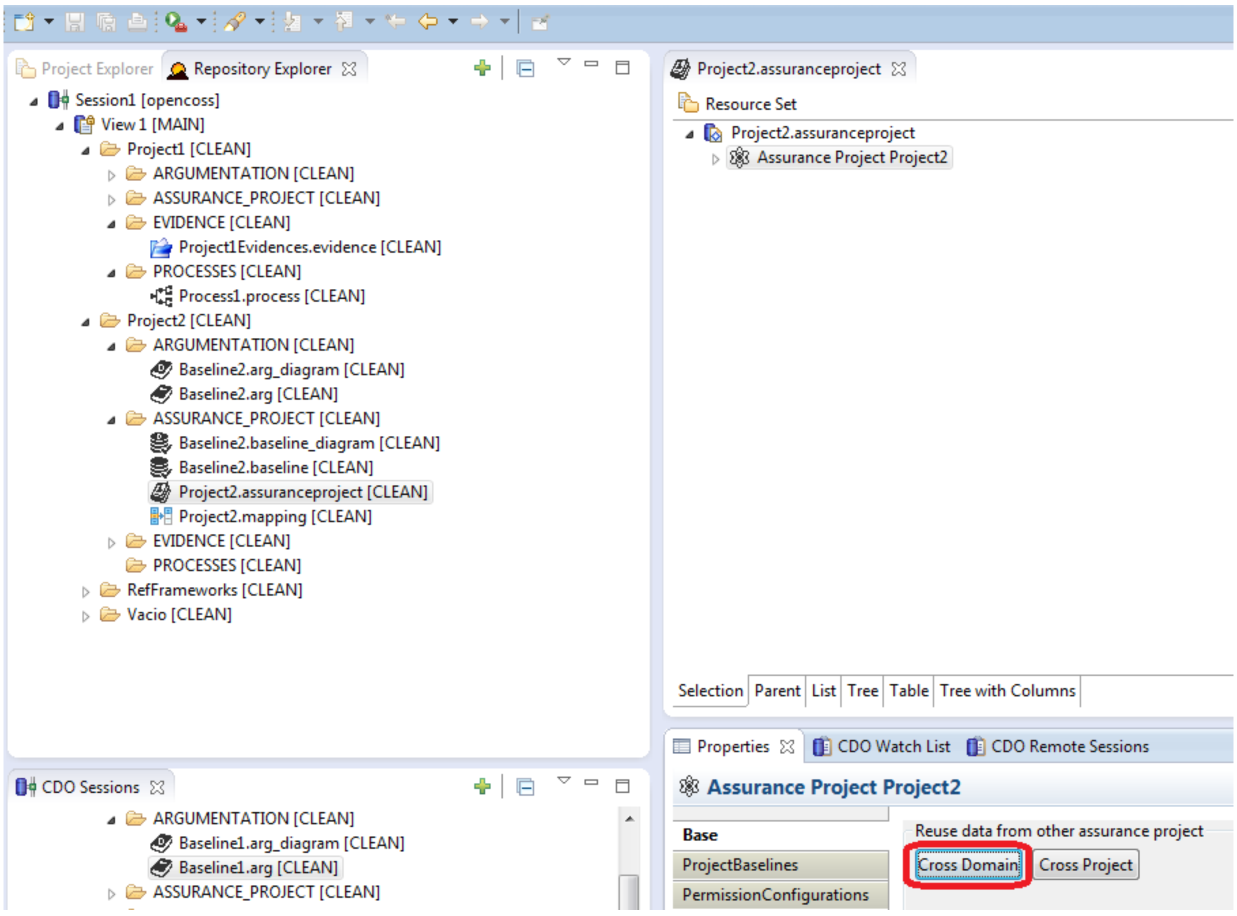
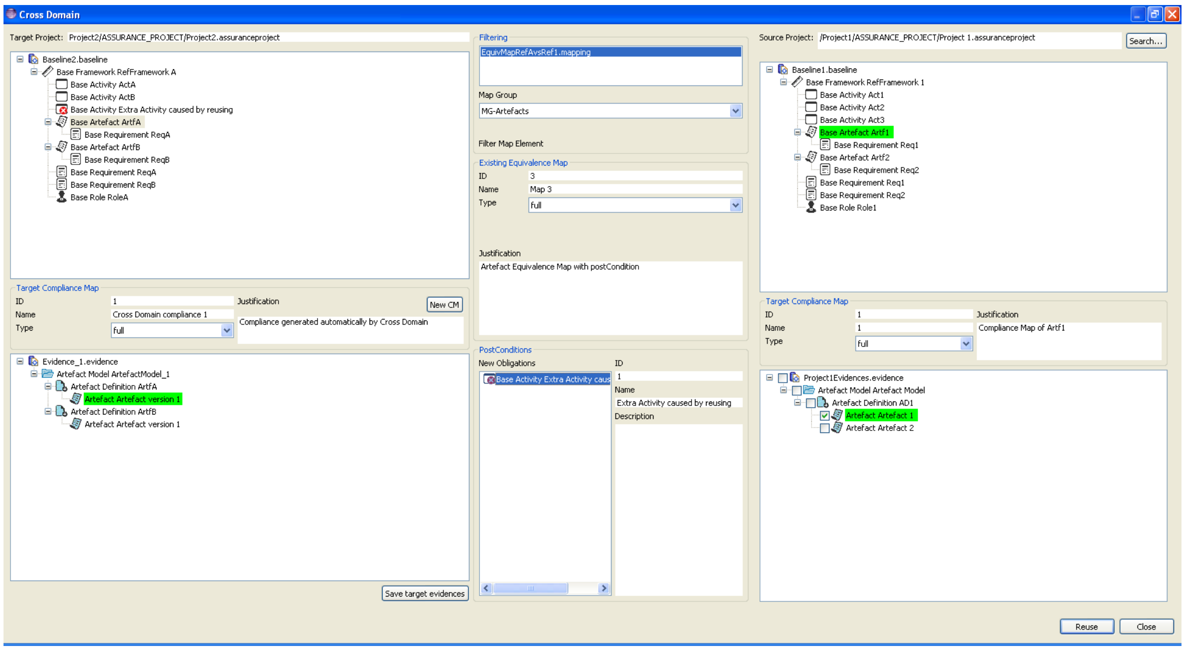


Figure 38. Cross-Domain button

Figure 39 shows the cross-domain assistant form. It is organised in three zones:

- The *left zone* shows information about the target project. In the top part, the URL of the target assurance project can be found. Below, a tree shows the contents of the target baseline, where we can select one of the items in order to display its compliance map information in the tree at the bottom.
- The *middle zone* displays equivalence map information. It includes controls to select the equivalence mapping model and the equivalence map group and displays the equivalence map details of the target baseline element selected and its postconditions in a list (to see the ID, Name and description one postcondition must be selected).
- The *right zone* presents information about the source project. In the top part, the URL of the source assurance project can be found. Below, a tree shows the contents of the source baseline, where we can select one of the items to display its compliance map information and the contents of the source evidence model in the tree at the bottom.



Target Project: Project2/ASSURANCE_PROJECT/Project2.assuranceproject

```
Baseline2.baseline
  Base Framework RefFramework A
    Base Activity ActA
    Base Activity ActB
    Base Activity Extra Activity caused by reusing
    Base Artefact ArtfA
      Base Requirement ReqA
    Base Artefact ArtfB
      Base Requirement ReqB
    Base Requirement ReqA
    Base Requirement ReqB
    Base Role RoleA
```

Target Compliance Map
ID: 1
Name: Cross Domain compliance 1
Type: full
Justification: Compliance generated automatically by Cross Domain

```
Evidence_1.evidence
  Artefact Model ArtefactModel_1
    Artefact Definition ArtfA
      Artefact Artefact version 1
    Artefact Definition ArtfB
      Artefact Artefact version 1
```

Filtering
EquivMapRefAvsRef1.mapping
Map Group: MG-Artefacts
Filter Map Element

Existing Equivalence Map
ID: 3
Name: Map 3
Type: full
Justification: Artefact Equivalence Map with postCondition

PostConditions
New Obligations: Base Activity Extra Activity caus
ID: 1
Name: Extra Activity caused by reusing
Description:

Source Project: /Project1/ASSURANCE_PROJECT/Project 1.assuranceproject

```
Baseline1.baseline
  Base Framework RefFramework 1
    Base Activity Act1
    Base Activity Act2
    Base Activity Act3
    Base Artefact Artf1
      Base Requirement Req1
    Base Artefact Artf2
      Base Requirement Req2
    Base Requirement Req1
    Base Requirement Req2
    Base Role Role1
```

Target Compliance Map
ID: 1
Name: 1
Type: full
Justification: Compliance Map of Artf1

```
[ ] Project1Evidences.evidence
  [ ] Artefact Model Artefact Model
    [ ] Artefact Definition AD1
      [✓] Artefact Artefact 1
      [ ] Artefact Artefact 2
```

Figure 39. Cross-domain window

The user must follow the following steps:

- Choose the source project of reuse using the “Search button”, so that the source baseline and evidence model tree will be loaded.
- Select the equivalence model and the equivalence group.
- Select the target base element that will receive the evidences to be reused, so that its compliance and equivalence map information will be loaded (highlighting in green its target elements in the trees).
- Finally, select the target Artefact and press the “Reuse” button to start the copy of the checked source Artefacts to the target selected Artefact (only one can be selected).

The source repository configuration information inside the Artefact Model Object, the Resource objects of the checked source Artefacts and the repository files related to these resources will be copied to the target evidence model. Additionally, the postconditions will be selected in the target baseline model.

2.2.7 Product/Process/Assurance Case Line Specification

To manage families/lines, it is necessary to have at disposal modelling means for systematising commonalities and variabilities. These means might be provided either as a specific solution targeting a single type of family (e.g., a process line) or as an orthogonal solution applicable to any type of family

3.1.2.7 Variability management support at process level

This chapter describes the implementation for requirement **WP6_PPA_001,** described in the table above.

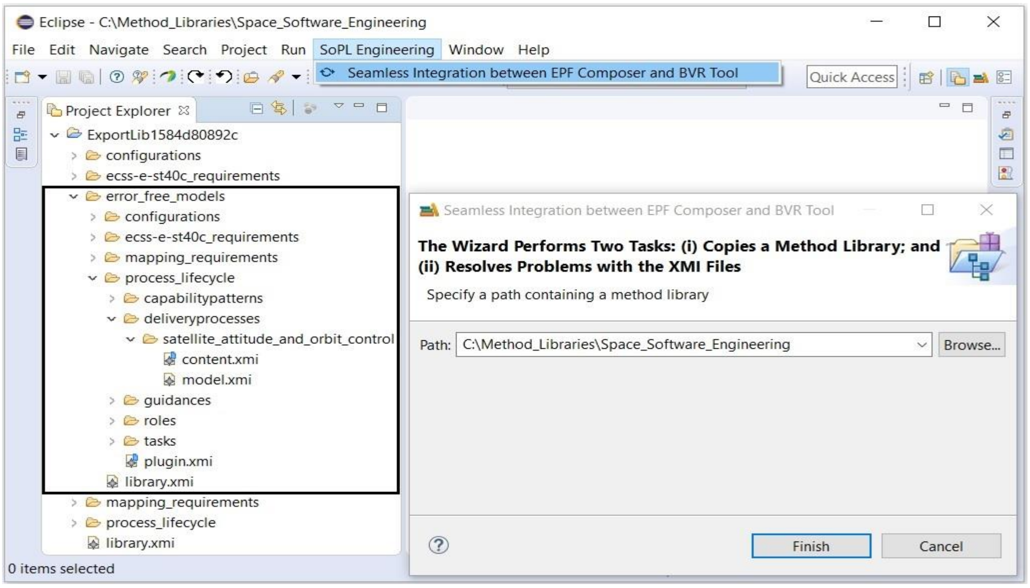
The safety-oriented processes tend to be reused, modified and extended to individual projects in a similar manner to the product lines. However, to be able to establish the Safety-oriented Process Lines (SoPLs), the seamless integration between process engineering and variability management is required. This might be done in two possible ways: either the support for variability modelling and management is incorporated and implemented in a process engineering framework, or otherwise the integration with variability management solution needs to be achieved. This chapter focuses on the seamless integration between Eclipse Process Framework (EPF) Composer and Base Variability Resolution (BVR) Tool. The former supports the major parts of the OMG's Software & Systems Process Engineering Metamodel (SPEM) Version 2.0, while the latter is a simplification and enhancement of the OMG's revised submission of Common Variability Language (CVL). EPF Composer and BVR Tool are implemented as Eclipse plugins, which are licensed under the Eclipse Public License (EPL) Version 1.0. In a tool paper, the integration between EPF Composer and BVR Tool is discussed [35].

EPF Composer is the only available implementation of OMG's SPEM 2.0, but the migration of EPF Composer to newer versions of technologies was never performed. Accordingly, we evolved the EPF Composer from Eclipse Galileo 3.5.2 to Eclipse Neon 4.6.3 after 11 years. This is done for integration in the AMASS platform. This contribution is acknowledged by the IBM, and therefore committer status is assigned for the project. The migration is performed in four steps [36]:

1. Compatible versions of required software's are installed from the Neon software repository and then deprecations in the source code are analysed and fixed.
2. Scheduling conflicts are resolved for the persistence of method elements (i.e., method configurations, method plugins, method content descriptions and processes) in their own folders and XMI files.
3. Appearance and height problems are resolved for the combo box which supports users in selecting the currently used method configuration, the blank views are removed from the authoring and browsing perspectives, and problems with the rich text editor are resolved for enabling users to format and style text.
4. Incompatible bundles are removed from the feature plugins, replacing bundles are added and other missing dependencies for the bundles are resolved for exporting the application. As per recommendation, the EPF Composer might be launched as a standalone application, but also in the Eclipse Integrated Development Environment (IDE).

BVR defines variability orthogonally for any MOF-compliant model, but the integration with EPF Composer brings additional challenges. EPF Composer is based on the UMA metamodel. It persists the method library contents in their own folders and files. The XMI files produced by EPF Composer are neither directly opened nor mapped at the realization editor. Therefore, the problems in XMI files have to be resolved for variability management with the BVR Tool. EPF Composer is based on the UMA metamodel. It persists the method library contents in their own folders and XMI files, in particular, method plugins, processes, content descriptions and configurations. In case of a new plugin, a plugin.xmi file is created in the new plugin directory and the reference of plugin is added to the library.xmi file. When a new capability pattern or delivery process is created, the model.xmi and content.xmi files are created in a new directory, and the reference of new process is added to the plugin.xmi file. Similarly, moving a content element to another plugin changes plugin.xmi in both plugins. The configuration file is used to specify the working set: It records the references to included content packages and processes.

It is decided to copy the method library before resolving the problems in XMI files for two reasons. Firstly, the library might be keep running in EPF Composer. Secondly, reverting back is just required for configured processes. The packages and resource factories are registered for the UMA metamodel. Otherwise, the package and class not found exceptions would be raised. We have identified that the hypertext references (hrefs) in XMI files are based on the globally unique identifiers (GUIDs) for example uma://_ErexoKA4EeaPp8nsuu2eew. This is the case with multiple UMA metamodel elements, such as tasks, roles, work products, tool mentors and method packages. As a result, the malformed URL exceptions are produced. The platform specific paths or otherwise Uniform Resource Identifiers (URIs) should be used instead. The support for the identification and resolution of problems with hrefs has been implemented.


Eclipse - C:\Method_Libraries\Space_Software_Engineering
File Edit Navigate Search Project Run SoPL Engineering Window Help
Seamless Integration between EPF Composer and BVR Tool
Quick Access
Project Explorer
ExportLib1584d80892c
configurations
ecss-e-st40c_requirements
error_free_models
configurations
ecss-e-st40c_requirements
mapping_requirements
process_lifecycle
capabilitypatterns
deliveryprocesses
satellite_attitude_and_orbit_control
content.xmi
model.xmi
guidances
roles
tasks
plugin.xmi
library.xmi
mapping_requirements
process_lifecycle
library.xmi
0 items selected
Seamless Integration between EPF Composer and BVR Tool
Finish
Cancel

The Wizard Performs Two Tasks: (i) Copies a Method Library; and (ii) Resolves Problems with the XMI Files

Specify a path containing a method library

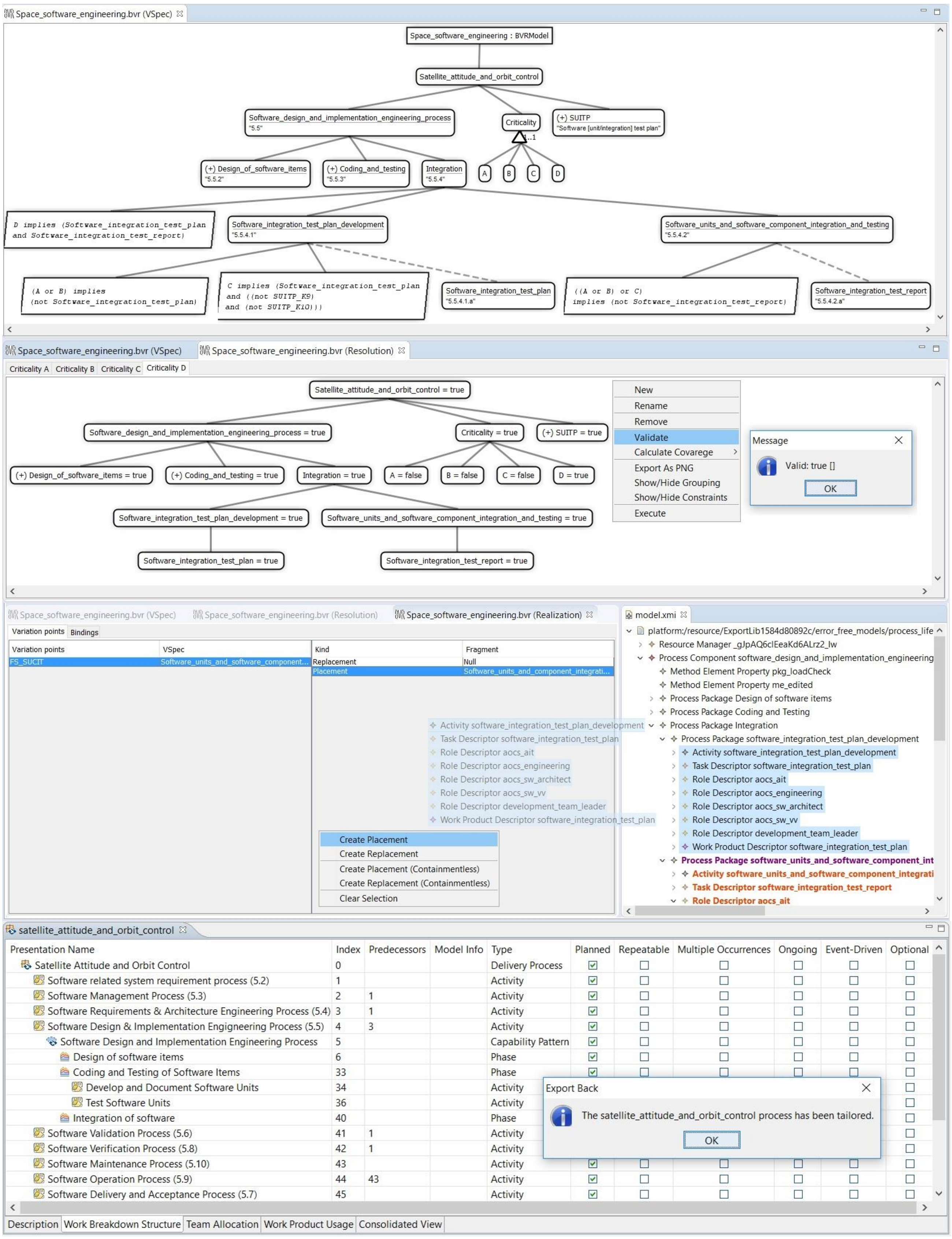
Path: C:\Method_Libraries\Space_Software_Engineering Browse...

Figure 40. The achievement of error free models

To create the placement and replacement fragments, the model elements are dragged and dropped to the realisation editor, and then create placement or create replacement option is selected from the context menu. This, however, produces the illegal operation exception for UMA compliant models. The analysis reveals that multiple metamodel elements have associated description implementations, such as deliverables and break down elements. Their naming structure "parent name,parent GUID" is not allowed. Accordingly, we performed temporary adaptations for supporting placements and replacements in the realization editor. The problems are resolved for all XMI files; the method library, configurations, plugins, processes and content descriptions might be considered for variability management with the BVR Tool. The visual support for highlighting objects placements in red while replacements in blue colours, as well as retrieving selections are supported for UMA compliant models.

We have implemented a dialogue wizard to support the mapping of target configurations at the realization editor, as shown in Figure 40 the recent/default path choice is automatically filled in the path text box otherwise the path containing a specific method library might be browsed. The dialogue wizard performs two tasks: (i) imports the contents of the method library in the target directory; and (ii) resolves problems with the XMI files. The error free models are made available in the project folder. All the model files can be opened, for example, method configurations, method plugins, method content descriptions and processes. The generation of target configurations for a software process modelled in EPF Composer is performed with VSpec, Resolution, and Realisation editors, as illustrated in Figure 41.

The generated process models are automatically exported back to the EPF Composer. If EPF Composer is running, the dialogue window pops up to inform that "the files have been changed on the file system. Do you want to load the changes?" The selection of yes option further allows the user to save the copy of previous model. Pressing the finish button loads the derived process model in EPF Composer. It might be noted that the changes for resolving problems in XMI files and supporting the communication with realization editor had been reverted back in exported models.


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 "5.5.4.1a"
((A or B) or C) implies (not Software_integration_test_report)
Software_integration_test_report "5.5.4.2a"
Space_software_engineering.bvr (VSpec)
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 = false
C = false
D = true
Software_integration_test_plan_development = true
Software_units_and_software_component_integration_and_testing = true
Software_integration_test_plan = true
Software_integration_test_report = true
New
Rename
Remove
Validate
Calculate Covarage
Export As PNG
Show/Hide Grouping
Show/Hide Constraints
Execute
Message
OK
Space_software_engineering.bvr (Realization)
model.xmi
Variation points
Bindings
Create Placement
Create Replacement
Create Placement (Containmentless)
Create Replacement (Containmentless)
Clear Selection
satellite_attitude_and_orbit_control
Export Back
OK
Description
Work Breakdown Structure
Team Allocation
Work Product Usage
Consolidated View

Valid: true []

Variation points	VSpec
FS_SUCIT	Software_units_and_software_component...

Kind	Fragment
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

- 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

Presentation Name	Index	Predecessors	Model Info	Type	Planned	Repeatable	Multiple Occurrences	Ongoing	Event-Driven	Optional
Satellite Attitude and Orbit Control	0			Delivery Process	☑	☐	☐	☐	☐	☐
Software related system requirement process (5.2)	1			Activity	☑	☐	☐	☐	☐	☐
Software Management Process (5.3)	2	1		Activity	☑	☐	☐	☐	☐	☐
Software Requirements & Architecture Engineering Process (5.4)	3	1		Activity	☑	☐	☐	☐	☐	☐
Software Design & Implementation Engigneering Process (5.5)	4	3		Activity	☑	☐	☐	☐	☐	☐
Software Design and Implementation Engineering Process	5			Capability Pattern	☑	☐	☐	☐	☐	☐
Design of software items	6			Phase	☑	☐	☐	☐	☐	☐
Coding and Testing of Software Items	33			Phase	☑	☐	☐	☐	☐	☐
Develop and Document Software Units	34			Activity						☐
Test Software Units	36			Activity						☐
Integration of software	40			Phase						☐
Software Validation Process (5.6)	41	1		Activity						☐
Software Verification Process (5.8)	42	1		Activity						☐
Software Maintenance Process (5.10)	43			Activity	☑	☐	☐	☐	☐	☐
Software Operation Process (5.9)	44	43		Activity	☑	☐	☐	☐	☐	☐
Software Delivery and Acceptance Process (5.7)	45			Activity	☑	☐	☐	☐	☐	☐

The satellite_attitude_and_orbit_control process has been tailored.

Figure 41. ECSS-E-ST-40C compliant SoPL

3.1.2.8 Specification of variability at component level

This section describes the implementation for requirement **WP6_PPA_004,** described in the Table 2. The integration between CHESS and BVR tools represents a feasible and technically advantageous solution for variability management at product level. Similar to process lines, the generation of target product is performed with three editors: VSpec, Resolution, and Realization.

The CHESS Tool is built on top of Eclipse Papyrus. Similar to Papyrus, the CHESS model is stored in .di, .notation and .uml files. In order to visualise the diagrams, the .di file is opened with the Papyrus editor, in which the dragged model variants from model explorer or palette are dropped. As a consequence, the style information is recorded in the .notation file. The model variants, however, are stored in .uml file. The placements and replacements would have been defined for the variations in .uml file; the interactions of CHESSML compliant models with the BVR Tool are supported.

When the model variants are removed from the .uml file, the dangling stereotypes problem is caused. In particular, the repair stereotypes dialog pops up after the removal of model variants. To resolve this problem, the stereotypes applied at the placement variants are retrieved and deleted. This is done before the removal of model variants. Besides the dangling stereotypes, the orphan views appear in the diagram editors. In this context, the implemented command for clean diagrams needs to be executed; the style information of orphan views is removed from the .notation file. This is done for both opened and closed diagrams. The replacements in executed fragments have also been recorded. Specifically, the replacement variants are tracked within the .uml file and dropped at the diagram editors. After that, the Arrange All command is executed for the editors. At the opening of EPF Composer and CHESS Tool, the dialogue window pops up to inform that “the files have been changed on the file system. Do you want to load the changes?” Pressing the “Yes” button loads the tailored model.

3.1.2.9 Variability management support at assurance case level

This chapter focuses on the implementation of requirement **WP6_PPA_005,** described in Table 3.

To support the variability management at assurance case level, the OpenCert Tool and BVR Tool are utilised. In a similar manner to the process and product lines, the generation of target model is performed with three editors: VSpec, Resolution, and Realization. Since the BVR Tool defines variability orthogonally for any Meta-Object Facility (MOF)-compliant model (representing the Base model), the interactions of CACM compliant models with the BVR Tool are supported to map the elements of a target configuration and variability abstractions in BVR. The argumentation is stored in .arg and .arg_diagram files. The placements and replacements are defined for the variations in .arg file. For making specification intuitive and visual, the placements and replacements are highlighted in red and blue colours, respectively. The execution of configuration/resolution generates the tailored model. Accordingly, the command for propagation of injected model elements into diagram is executed.

3.1.2.10 Variability Management and Change Impact Analysis in Multiple Lines

The process, product and assurance case variability might be specified in the combined or otherwise individual models. In the combined models, the individual branches might be taken into consideration for the process and product variability. The constraints are enforced over the model elements, for which their names are considered. It is therefore important to avoid duplicates; the occurrences can also be defined. However, the BVR Tool does not support variability management and impact analysis in multiple lines.

The idea with the individual models is separation of concerns, so that the process, product and assurance case engineers work on their respective models. The interactions between process, product and assurance case models have been supported; the logical operators such as implication, alternative, negation might be used in the cross-cutting constraints. It is a meaningful way to enforce the process-product-assurance case dependencies. There is also a need to consider the occurrence specifications between the variability models of a project. The presence of elements mentioned within the constraint is first checked in the current model. If the elements are not detected, the search is extended to other models in a project. In case the elements are detected in another model, the dialogue window pops up to inform the existence in specific model. The user, however, needs to authenticate the enforcement of cross-cutting constraints.

The resolution editor is used for specification, validation and execution of process, product and assurance case configurations. The resolutions are automatically generated from the VSpec model in which the varying choices needed to be included or excluded. It is possible to define multiple resolutions for the processes, products and assurance cases with variabilities. The constraints are used to specify the dependencies between choices. Therefore, there is a need to pair the process, product and assurance case resolutions to perform error checking and validation. The command for pairing of resolutions is incorporated; valid impact analysis and change propagation are guaranteed if the cross-cutting constraints are properly specified.

In BVR, the resolution execution was supported for a single base model. To be able to support the variant management and change impact analysis in integrated lines, the execution of two or more base models is needed, for which the source code is altered. A dialogue wizard is used to inform the possible candidates to the user, in particular, the models for which realisation fragments have been specified. The checklist selection is supported for the base models. Besides that, the user either selects the execution of cross-cutting dependencies for the purpose of impact propagation, or otherwise the whole joined resolution is executed. Accordingly, the back-propagation of tailored models is performed. The support for saving the copy of previous models is also incorporated.

3.1.2.11 Semi-automatic generation of product arguments

This section describes the implementation for requirement **WP6_PPA_002**, described in Table 2.

The generation of product-based arguments functionality uses the information specified in a CHESS model to generate a set of argument-fragments for each of the components from the specified model. The argument-fragments are created on the connected CDO [11] repository in the assurance case selected by the user. The generated .arg and .arg_diagram files for each component are available after generation in the corresponding "ARGUMENTATION" folder. Each argument-fragment contains information about the contracts of the corresponding component. If a contact is validated (has the status set to "validated") the clarification of the contracts as well as supporting evidence is added to the argument. If the contract is not validated, then a challenge is added to the argument, implying that the satisfaction of the contract is challenged by the refinement results. The generator uses traceability enabled by Capra to get the traces between contracts and supporting evidence, but it also generates the traces between components, contracts and formal properties with the corresponding automatically generated argumentation elements such as claims, contexts, evidence, etc.

3.1.2.12 Semi-automatic generation of process arguments

This chapter describes the implementation for requirement **WP6_PPA_003**, described in the Table 2.

2.2.7.1.1 Detecting fallacies in process models

The process-based argumentations cannot ensure that the evidences are sufficient to support the claim. In order to prevent or detect fallacies in the process-based argumentations, *fallacy detection* plugin is developed that validates the process models and prevents the occurrence of fallacy, specifically, "*Omission of Key Evidence"* in process-based arguments. Omission of key evidence fallacies within the context of process argumentation are the flaws or defects in which arguments can fail to provide sufficient evidence. For example, evidences about staff competency or skills to support the process claim about designer who is responsible for the design task, which deals with the production of design-related work products are not provided. Furthermore, the tool and its related tool mentor qualification is missing in order to ensure that the evidence generated by that tool is trustworthy and valid.

Detect Fallacies plugin enables the process engineers and/or safety engineers to detect whether the process models contains all the essential information for supporting the key evidence(s). In case of omission of crucial information, recommendations are provided to aid process engineers or safety engineers in resolving deviations. Based on the recommendation(s), engineers are expected to modify the process model. The operation will only be applicable by clicking on those elements of type ProcessComponent (Capability Pattern or Delivery Process) since these elements are the ones that have all the information of process (see Figure 42).

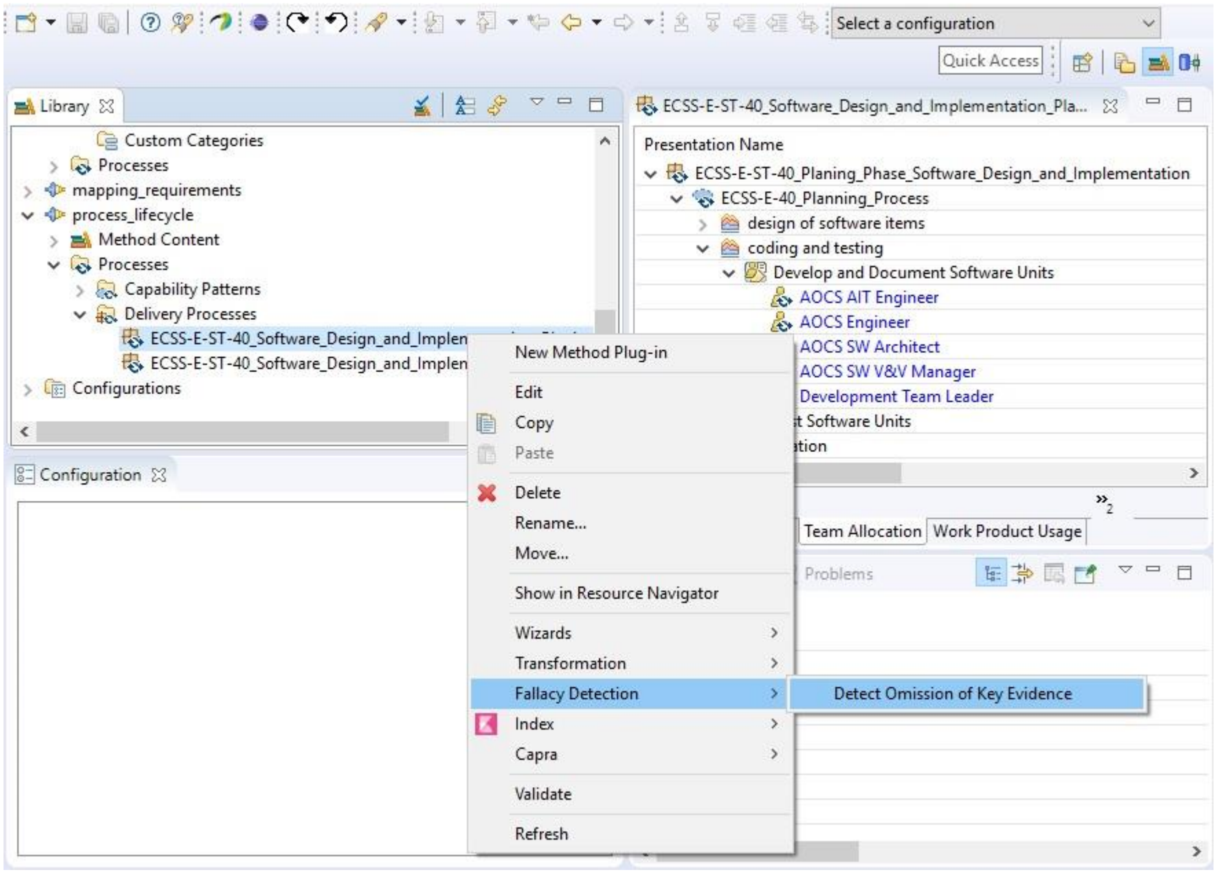
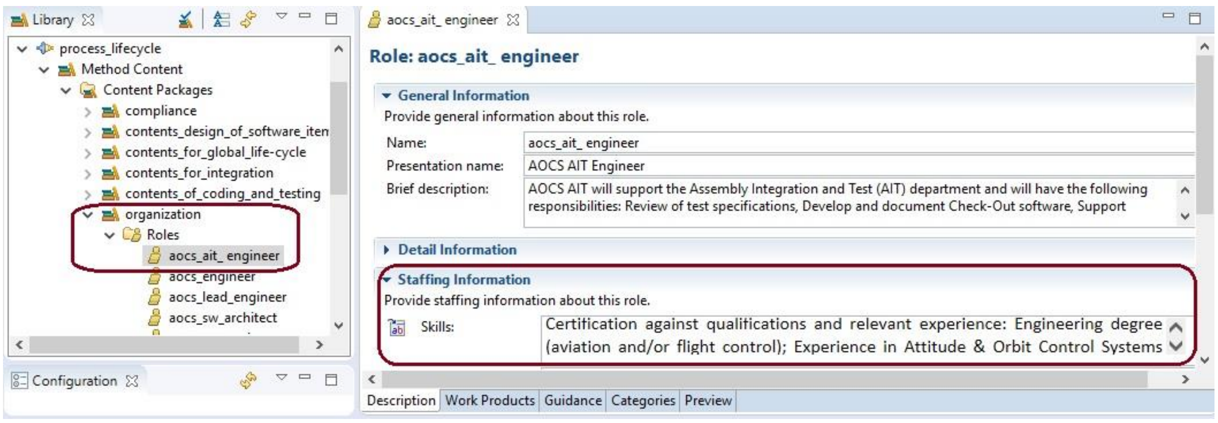


Figure 42. Detecting fallacies in process models menu

To detect fallacies in process models and to perform the generation of fallacy free process-based argumentation, a safety process in EPF Composer is modelled according to the best practices as well as according to the standard(s). Either, the requirements and associated process life cycle can be modelled by following the IBM approach as discussed in AMASS D6.3 [30] and AMASS User Manual [18]; or the standard requirements can automatically be imported into the EPF Composer as described in AMASS D1.5 [70]. Certifications against the required standard's requirements for the roles in *process_lifecycle* can be described in *Skills* (Staffing Information) field, as shown in Figure 43. Certifications or rationales against required tool qualifications can be defined in *Key considerations* (Detail Information) field, as shown in Figure 44. If the evidence details are omitted or the rationale is not provided; it means that process contains the omission of key evidence fallacies.



Role: aocs_ait_ engineer

General Information
Provide general information about this role.

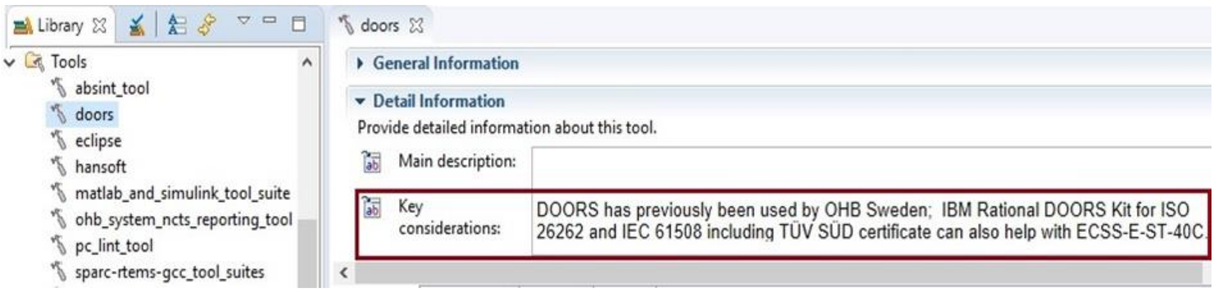
Name: aocs_ait_ engineer
Presentation name: AOCS AIT Engineer
Brief description: AOCS AIT will support the Assembly Integration and Test (AIT) department and will have the following responsibilities: Review of test specifications, Develop and document Check-Out software, Support

Detail Information

Staffing Information
Provide staffing information about this role.

Skills: Certification against qualifications and relevant experience: Engineering degree (aviation and/or flight control); Experience in Attitude & Orbit Control Systems

Figure 43. Modeling of evidence associated to Role


Library
Tools
absint_tool
doors
eclipse
hansoft
matlab_and_simulink_tool_suite
ohb_system_ncts_reporting_tool
pc_lint_tool
sparc-rtems-gcc_tool_suites
doors

General Information

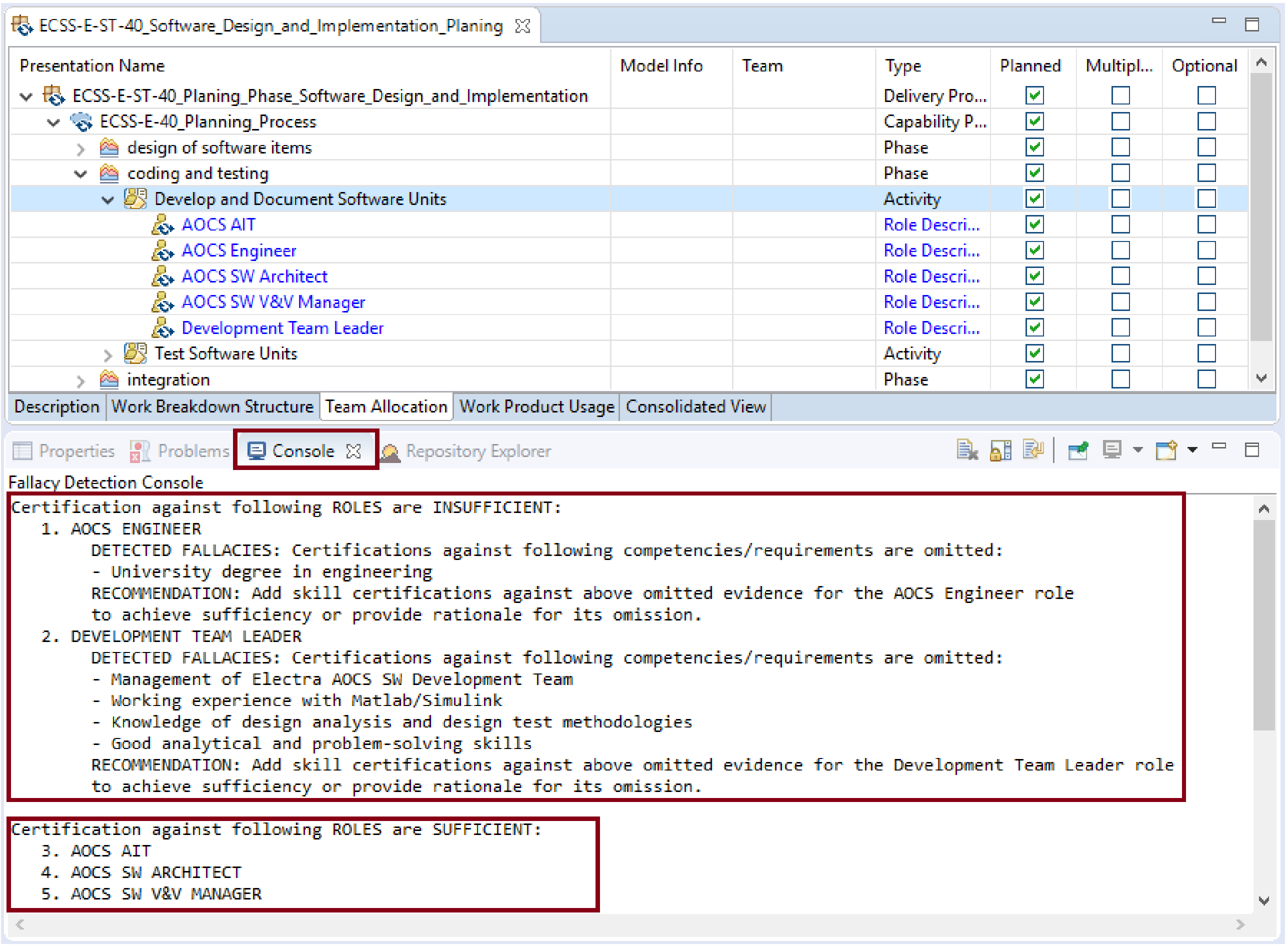
Detail Information
Provide detailed information about this tool.

Main description:

Key considerations: DOORS has previously been used by OHB Sweden; IBM Rational DOORS Kit for ISO 26262 and IEC 61508 including TÜV SÜD certificate can also help with ECSS-E-ST-40C

Figure 44. Modeling of evidence associated to Tool Qualification

The fallacy detecting functionality provides validation results including the list of elements containing sufficient and insufficient information (i.e., detected fallacies). In addition, the appropriate recommendations to resolve the particular deviations will be presented. These results are printed on the console (see Figure 45) as well as the validation reports, two new ".txt" files are created in the selected folder (see Figure 46). The information related to the *Roles* is stored under the file called "Staffing Plan Report.txt", and the information regarding the *Tools* is stored in the "Tool Qualification Plan Report.txt" file. Moreover, this information is sorted in the following way: the elements with *insufficient evidence* appear in the first place, and then the elements having sufficient evidences are placed below.

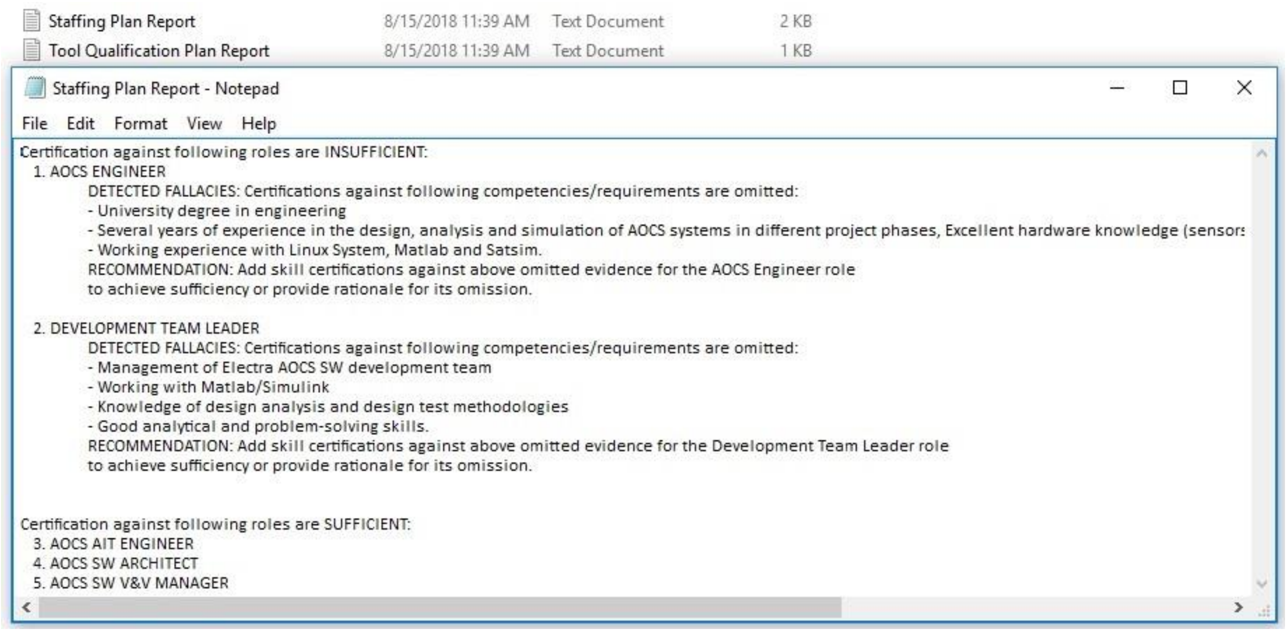


Presentation Name	Model Info	Team	Type	Planned	Multipl...	Optional
ECSS-E-ST-40_Planing_Phase_Software_Design_and_Implementation			Delivery Pro...	✓	☐	☐
ECSS-E-40_Planning_Process			Capability P...	✓	☐	☐
design of software items			Phase	✓	☐	☐
coding and testing			Phase	✓	☐	☐
Develop and Document Software Units			Activity	✓	☐	☐
AOCS AIT			Role Descri...	✓	☐	☐
AOCS Engineer			Role Descri...	✓	☐	☐
AOCS SW Architect			Role Descri...	✓	☐	☐
AOCS SW V&V Manager			Role Descri...	✓	☐	☐
Development Team Leader			Role Descri...	✓	☐	☐
Test Software Units			Activity	✓	☐	☐
integration			Phase	✓	☐	☐

Fallacy Detection Console

```
Certification against following ROLES are INSUFFICIENT:
   1. AOCS ENGINEER
        DETECTED FALLACIES: Certifications against following competencies/requirements are omitted:
        - University degree in engineering
        RECOMMENDATION: Add skill certifications against above omitted evidence for the AOCS Engineer role
        to achieve sufficiency or provide rationale for its omission.
   2. DEVELOPMENT TEAM LEADER
        DETECTED FALLACIES: Certifications against following competencies/requirements are omitted:
        - Management of Electra AOCS SW Development Team
        - Working experience with Matlab/Simulink
        - Knowledge of design analysis and design test methodologies
        - Good analytical and problem-solving skills
        RECOMMENDATION: Add skill certifications against above omitted evidence for the Development Team Leader role
        to achieve sufficiency or provide rationale for its omission.

Certification against following ROLES are SUFFICIENT:
   3. AOCS AIT
   4. AOCS SW ARCHITECT
   5. AOCS SW V&V MANAGER
```

Figure 45. Validation results printed on the console



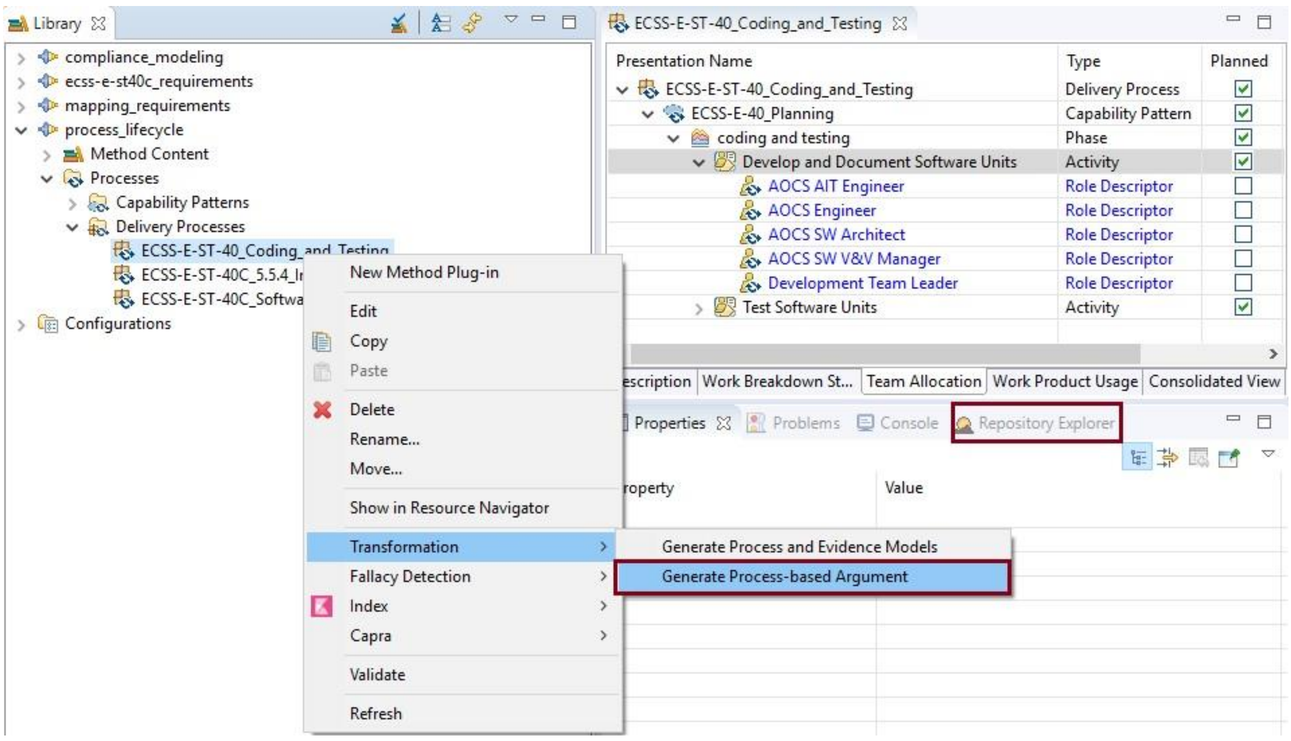
Staffing Plan Report	8/15/2018 11:39 AM	Text Document	2 KB
Tool Qualification Plan Report	8/15/2018 11:39 AM	Text Document	1 KB

```
Certification against following roles are INSUFFICIENT:
  1. AOCS ENGINEER
        DETECTED FALLACIES: Certifications against following competencies/requirements are omitted:
        - University degree in engineering
        - Several years of experience in the design, analysis and simulation of AOCS systems in different project phases, Excellent hardware knowledge (sensor:
        - Working experience with Linux System, Matlab and Satsim.
        RECOMMENDATION: Add skill certifications against above omitted evidence for the AOCS Engineer role
        to achieve sufficiency or provide rationale for its omission.

  2. DEVELOPMENT TEAM LEADER
        DETECTED FALLACIES: Certifications against following competencies/requirements are omitted:
        - Management of Electra AOCS SW development team
        - Working with Matlab/Simulink
        - Knowledge of design analysis and design test methodologies
        - Good analytical and problem-solving skills.
        RECOMMENDATION: Add skill certifications against above omitted evidence for the Development Team Leader role
        to achieve sufficiency or provide rationale for its omission.


Certification against following roles are SUFFICIENT:
  3. AOCS AIT ENGINEER
  4. AOCS SW ARCHITECT
  5. AOCS SW V&V MANAGER
```

Figure 46. Generated results reports

2.2.7.1.2 Semi-automatic generation of process arguments representing plans

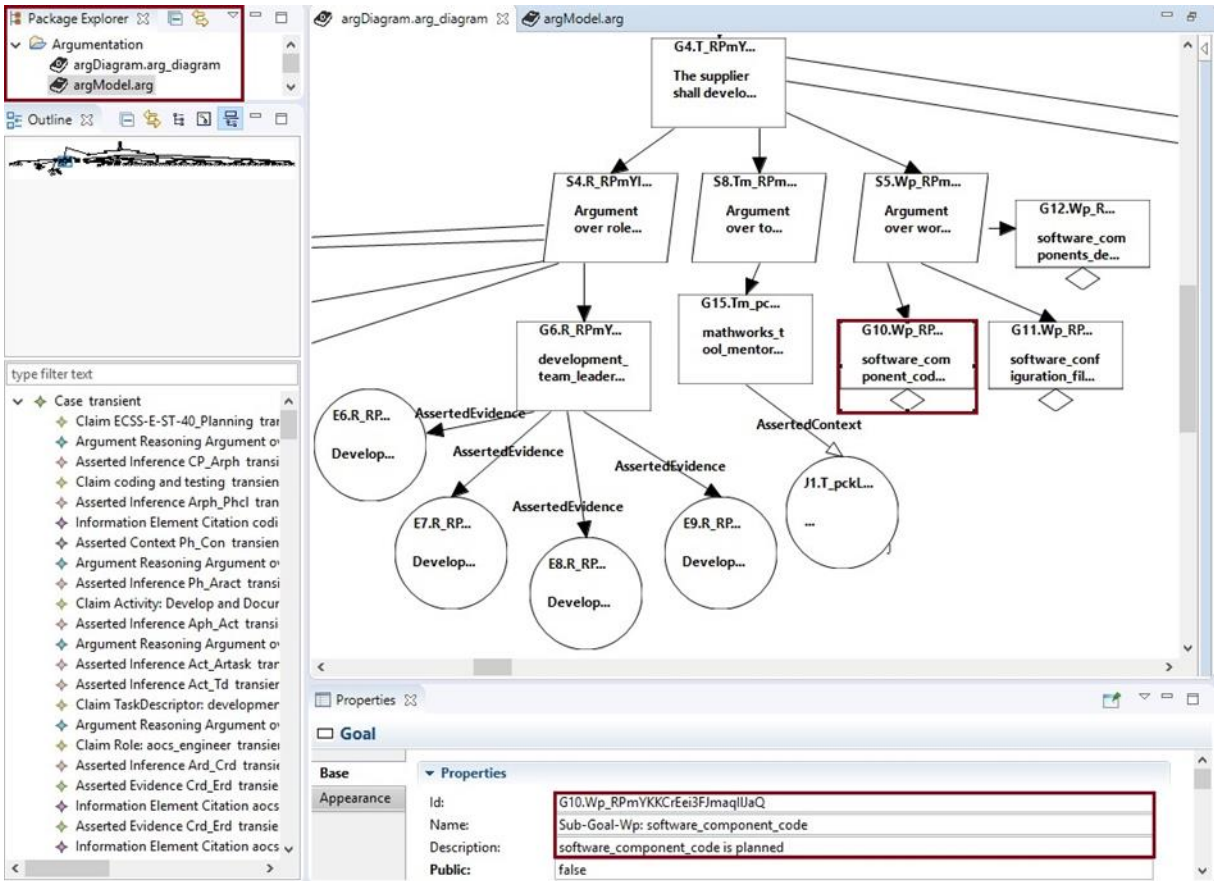
The main goal of this functionality is to generate fallacy *free process-based argumentation model* (with diagram). A specific process plan has derived from the family of processes (managed by EPF Composer), which is automatically transformed into safety argument fragments using Epsilon Transformation Language (ETL) [8]. In Prototype P2, the generation of process-based arguments functionality is invoked from a right-click menu of *ProcessComponent* (*Capability Pattern* or *Delivery* Process) modelled in EPF Composer, in particular, by selecting "Transformation -> Generate Process-based Argument" option, as shown in Figure 47.



Presentation Name	Type	Planned
ECSS-E-ST-40_Coding_and_Testing	Delivery Process	✓
ECSS-E-40_Planning	Capability Pattern	✓
coding and testing	Phase	✓
Develop and Document Software Units	Activity	✓
AOCS AIT Engineer	Role Descriptor	☐
AOCS Engineer	Role Descriptor	☐
AOCS SW Architect	Role Descriptor	☐
AOCS SW V&V Manager	Role Descriptor	☐
Development Team Leader	Role Descriptor	☐
Test Software Units	Activity	✓

Figure 47. Generating process-based argumentation menu (representing plan)

To perform the generation of process-based arguments, the mapping between process elements (SPEM/UMA) and argumentation elements (SACM/CACM) has implemented in EPF Composer. The mapping between meta-models has also been extended. In particular, the mapping of elements is focused on the *Work Breakdown Structure* of processes in EPF Composer, such as Capability Pattern, Phase, Activity, Task Descriptor, Role Descriptor, Work Product Descriptor, Tool, and Tool Mentors. Since the generation of process-based argumentation focused on the planning phase of the processes, the evidences of some elements such as Work Products, Checklist, Guideline, and Example are not available in those stages of the projects. These elements have mapped into undeveloped Claim, which requires further development (see Figure 48). The evidences associated to elements have mapped into InformationElementCitation typed "solutions" showing that the particular goals have been achieved. The rationale related to element (e.g., tool qualification is not needed since source code is fully tested) is mapped into InformationElementCitation typed "justification". The "purpose" attribute of the process is used to show that the process is compliant with the standard, which is mapped to InformationElementCitation "context". The detailed mapping between process model, argumentation mode is provided in D6.3 [30]. The generated process-based arguments are visualized in Assurance Case Editor in OpenCert under the "ARGUMENTATION" folder. The generated elements would be visible in the editing window of the Argumentation Diagram (.arg_diagram) by using drag and drop functionality. Furthermore, the argumentation model and diagram will be stored locally in a new project into the current Workspace under the name "Argumentation".



type filter text

- Case transient
  - Claim ECSS-E-ST-40_Planning trar
  - Argument Reasoning Argument o
  - Asserted Inference CP_Arph transi
  - Claim coding and testing transien
  - Asserted Inference Arph_Phcl tran
  - Information Element Citation codi
  - Asserted Context Ph_Con transien
  - Argument Reasoning Argument o
  - Asserted Inference Ph_Aract transi
  - Claim Activity: Develop and Docur
  - Asserted Inference Aph_Act transi
  - Argument Reasoning Argument o
  - Asserted Inference Act_Artask trar
  - Asserted Inference Act_Td transier
  - Claim TaskDescriptor: developmer
  - Argument Reasoning Argument o
  - Claim Role: aocs_engineer transie
  - Asserted Inference Ard_Crd transie
  - Asserted Evidence Crd_Erd transie
  - Information Element Citation aocs
  - Asserted Evidence Crd_Erd transie
  - Information Element Citation aocs

Id:	G10.Wp_RPmYKKCrEei3FJmaqlUaQ
Name:	Sub-Goal-Wp: software_component_code
Description:	software_component_code is planned
Public:	false

Figure 48. Locally generated argumentation model and diagram

2.2.8 Reuse Discovery

This chapter describes the implementation for requirements **WP6_RA_002** and **WP6_RA_006,** described in Table 2. This requirement has been developed in two different complementary implementations, so in the following chapter both implementations will be described.

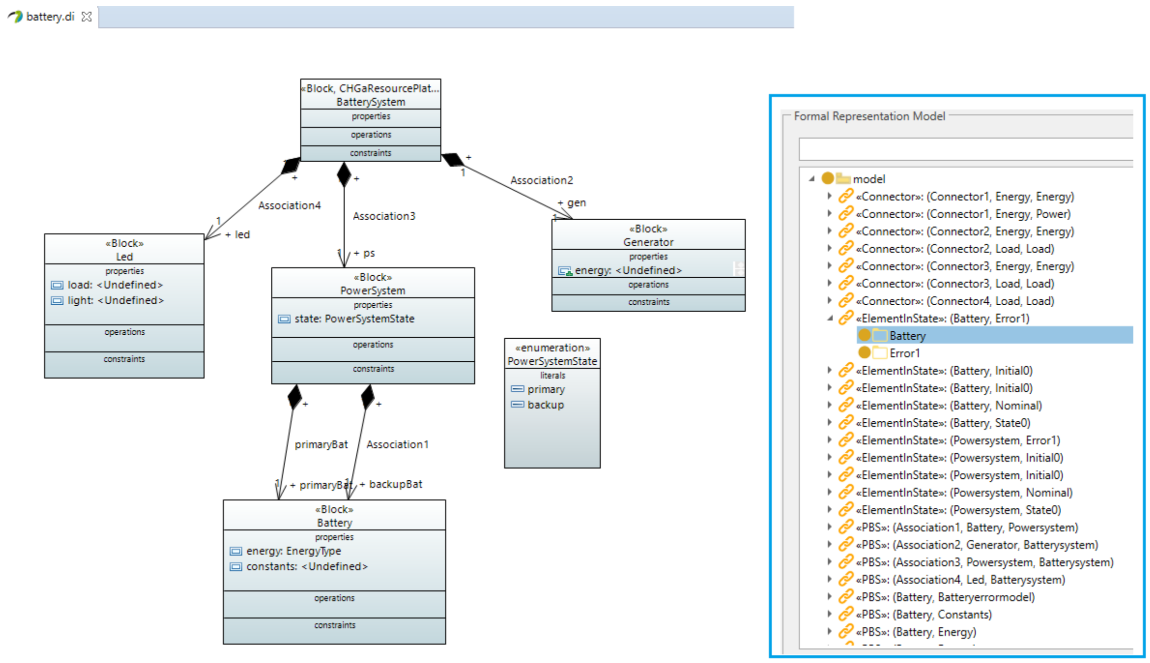
3.1.2.13 Reuse Discovery based on OSLC-KM

The implementation of the indexing and retrieval algorithms for Reuse Discovery, based on OSLC-KM, has been an iterative process within the AMASS platform. For P1, the development and integration focused on physical SysML Papyrus Models (see chapter 2.2.8.1.1), and for P2 the focus was set on the integration with the CDO Repository to index Requirements and Evidence Models to be reused later on in the Reuse Assistant Component (see chapter 2.2.8.1.2).

2.2.8.1.1 Indexing and Retrieval of SysML Physical models

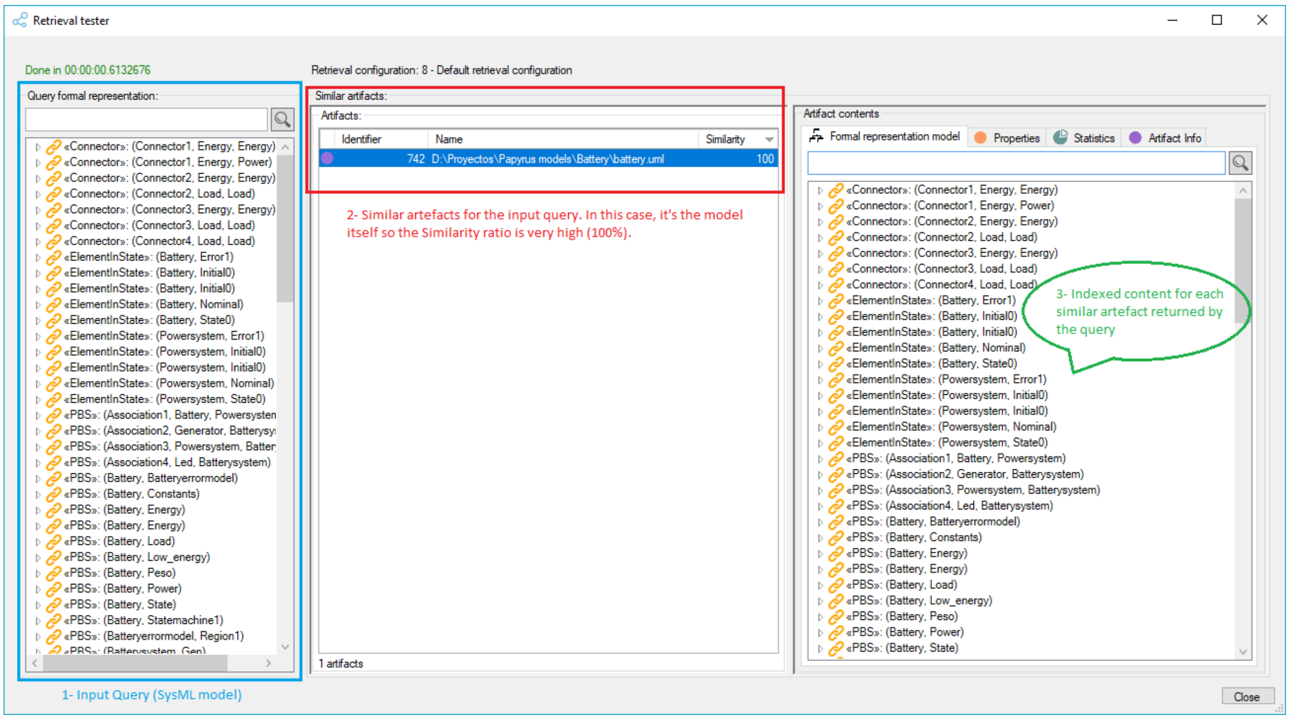
The main goal of this functionality is to allow the user to search for similar artefacts (e.g. Papyrus SysML models) within a set of indexed artefacts. This functionality was implemented for Prototype P1, and it's a very basic connector for indexing and searching for files, based on the OSLC-KM standard [28] which allows exchanging artefacts content from different and heterogeneous sources. For the Prototype P2, this will be the starting point for the further integration with the CDO.

The core functionality is developed in KM, as part of the TRC toolset. It allows indexing the content of a SysML model in a SKB (*System Knowledge Base*), as shown in Figure 49, so that this SKB can receive input queries in order to get similar artefacts by a given input artefact (see Figure 54).



model
- «Connector»: (Connector1, Energy, Energy)
- «Connector»: (Connector1, Energy, Power)
- «Connector»: (Connector2, Energy, Energy)
- «Connector»: (Connector2, Load, Load)
- «Connector»: (Connector3, Energy, Energy)
- «Connector»: (Connector3, Load, Load)
- «Connector»: (Connector4, Load, Load)
- «ElementInState»: (Battery, Error1)
  - Battery
  - Error1
- «ElementInState»: (Battery, Initial0)
- «ElementInState»: (Battery, Initial0)
- «ElementInState»: (Battery, Nominal)
- «ElementInState»: (Battery, State0)
- «ElementInState»: (Powersystem, Error1)
- «ElementInState»: (Powersystem, Initial0)
- «ElementInState»: (Powersystem, Initial0)
- «ElementInState»: (Powersystem, Nominal)
- «ElementInState»: (Powersystem, State0)
- «PBS»: (Association1, Battery, Powersystem)
- «PBS»: (Association2, Generator, Batterysystem)
- «PBS»: (Association3, Powersystem, Batterysystem)
- «PBS»: (Association4, Led, Batterysystem)
- «PBS»: (Battery, Batteryerrormodel)
- «PBS»: (Battery, Constants)
- «PBS»: (Battery, Energy)

Figure 49. SysML content transformed into RSHP metamodel


Retrieval tester

Done in 00:00:00.6132676

Retrieval configuration: 8 - Default retrieval configuration

Query formal representation:
- «Connector»: (Connector1, Energy, Energy)
- «Connector»: (Connector1, Energy, Power)
- «Connector»: (Connector2, Energy, Energy)
- «Connector»: (Connector2, Load, Load)
- «Connector»: (Connector3, Energy, Energy)
- «Connector»: (Connector3, Load, Load)
- «Connector»: (Connector4, Load, Load)
- «ElementInState»: (Battery, Error1)
- «ElementInState»: (Battery, Initial0)
- «ElementInState»: (Battery, Initial0)
- «ElementInState»: (Battery, Nominal)
- «ElementInState»: (Battery, State0)
- «ElementInState»: (Powersystem, Error1)
- «ElementInState»: (Powersystem, Initial0)
- «ElementInState»: (Powersystem, Initial0)
- «ElementInState»: (Powersystem, Nominal)
- «ElementInState»: (Powersystem, State0)
- «PBS»: (Association1, Battery, Powersysten
- «PBS»: (Association2, Generator, Batterysy
- «PBS»: (Association3, Powersystem, Batter
- «PBS»: (Association4, Led, Batterysystem)
- «PBS»: (Battery, Batteryerrormodel)
- «PBS»: (Battery, Constants)
- «PBS»: (Battery, Energy)
- «PBS»: (Battery, Energy)
- «PBS»: (Battery, Load)
- «PBS»: (Battery, Low_energy)
- «PBS»: (Battery, Peso)
- «PBS»: (Battery, Power)
- «PBS»: (Battery, State)
- «PBS»: (Battery, Statemachine1)
- «PBS»: (Batteryerrormodel, Region1)
- «PBS»: (Batterysystem, Gen)

1- Input Query (SysML model)

Similar artifacts:

Artifacts:

Identifier	Name	Similarity
742	D:\Proyectos\Papyrus models\Battery\battery.uml	100

2- Similar artefacts for the input query. In this case, it's the model itself so the Similarity ratio is very high (100%).

1 artifacts

Artifact contents

Formal representation model | Properties | Statistics | Artifact Info

- «Connector»: (Connector1, Energy, Energy)
- «Connector»: (Connector1, Energy, Power)
- «Connector»: (Connector2, Energy, Energy)
- «Connector»: (Connector2, Load, Load)
- «Connector»: (Connector3, Energy, Energy)
- «Connector»: (Connector3, Load, Load)
- «Connector»: (Connector4, Load, Load)
- «ElementInState»: (Battery, Error1)
- «ElementInState»: (Battery, Initial0)
- «ElementInState»: (Battery, Initial0)
- «ElementInState»: (Battery, Nominal)
- «ElementInState»: (Battery, State0)
- «ElementInState»: (Powersystem, Error1)
- «ElementInState»: (Powersystem, Initial0)
- «ElementInState»: (Powersystem, Initial0)
- «ElementInState»: (Powersystem, Nominal)
- «ElementInState»: (Powersystem, State0)
- «PBS»: (Association1, Battery, Powersystem)
- «PBS»: (Association2, Generator, Batterysystem)
- «PBS»: (Association3, Powersystem, Batterysystem)
- «PBS»: (Association4, Led, Batterysystem)
- «PBS»: (Battery, Batteryerrormodel)
- «PBS»: (Battery, Constants)
- «PBS»: (Battery, Energy)
- «PBS»: (Battery, Energy)
- «PBS»: (Battery, Load)
- «PBS»: (Battery, Low_energy)
- «PBS»: (Battery, Peso)
- «PBS»: (Battery, Power)
- «PBS»: (Battery, State)

3- Indexed content for each similar artefact returned by the query

Close

Figure 50. SysML model content as a query to search for similar artefacts

It was also integrated within the AMASS platform as a technological demonstrator of the OSLC-KM capabilities in order to integrate this core functionality in P1 (see Figure 51). So, this functionality is available from P1.

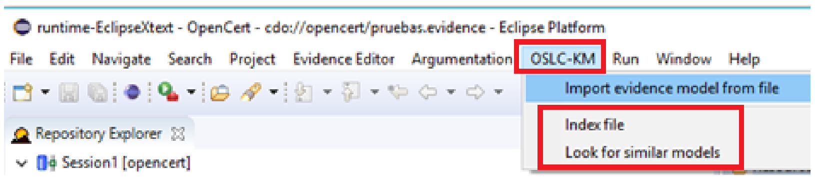

runtime-EclipseXtext - OpenCert - cdo://opencert/pruebas.evidence - Eclipse Platform
File Edit Navigate Search Project Evidence Editor Argumentation OSLC-KM Run Window Help
Import evidence model from file
Index file
Look for similar models
Repository Explorer
Session1 [opencert]

Figure 51. Reuse Discovery operations integrated in AMASS

The previous integration (P1) included two basic operations:

1. **Index file**: By selecting local files, it is possible to create a wide SKB to perform searches.
2. **Look for similar models**: By selecting a local file as an input, it is possible to query the SKB in order to get similar models. This search is based on the RSHP model [29], and its goal is to look for similar artefacts by artefact content (relations between elements and meta-properties).

Both operations will ask the user for a Papyrus model to work with. So far, this implementation for P1 is limited to work only with Papyrus files but not yet with other kind of SysML models (MagicDraw, Rhapsody, etc.). A further integration for P2, described in the next chapter, allows the user to select items from the CDO repository.

2.2.8.1.2 Indexing and Retrieval of CDO objects

The functionality developed for Prototype P2 includes the integration with the CDO database. Thus, the OSLC-KM based functionality for indexing and retrieval has been extended, so that is possible to navigate through the CDO database (manually, project by project) to index the different projects that we have. After this, a “Search” functionality is available in the Reuse Assistant component. This way, it’s possible to query for Artefacts that contain requirements of a given “SIL” level, or simply query by free text.

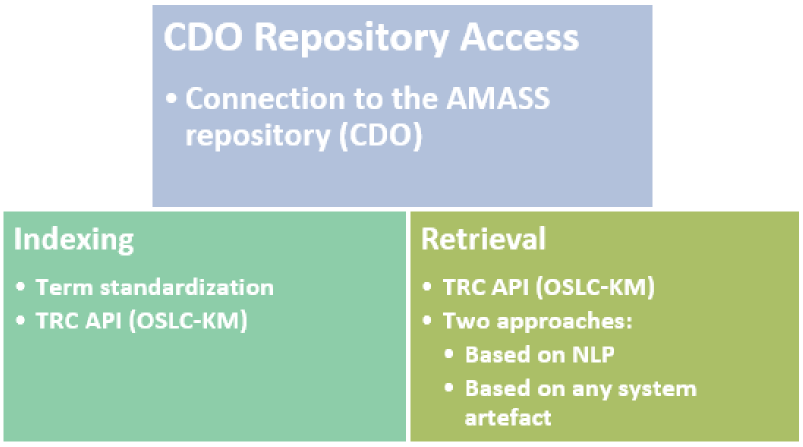


Figure 52. Operations integrated in the CDO Repository

Regarding the indexing process, we index two types of items:

- Assurance Projects: the process creates a mapping between the SIL Level of the requirements inside, and the Target Artefacts pointed by the Compliance Maps.
- Evidence Models: the process indexes all the active Artefact Models inside, together with the SIL Level mapping calculated by means of indexing the assurance project. This way is possible to reuse Artefacts via OSLC-KM later.

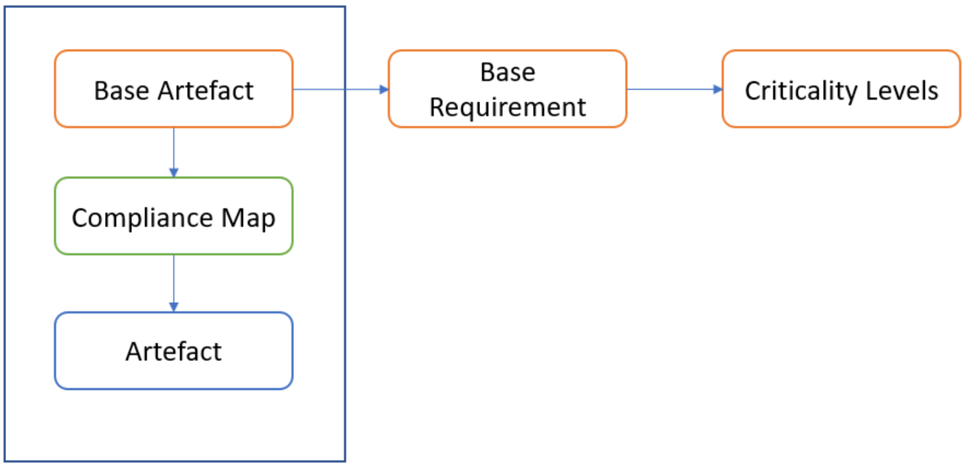


Figure 53. CDO metamodel to index

As an example, from the following project in the CDO repository, the Prototype P2 allows indexing the content via OSLC-KM through the top menu:

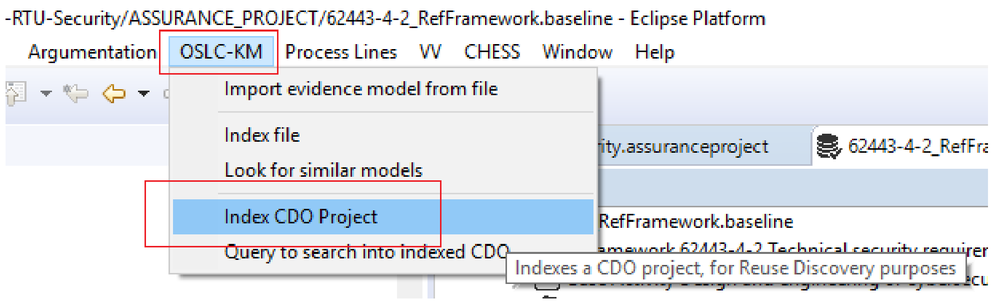
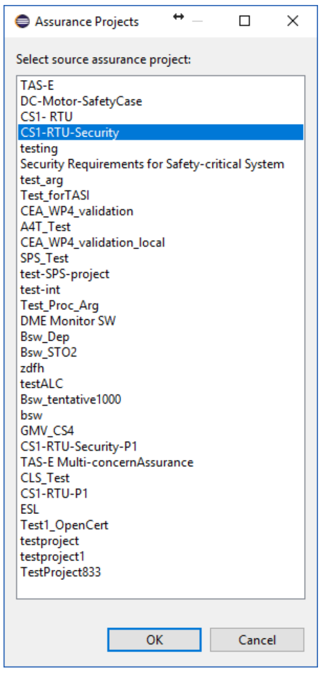

-RTU-Security/ASSURANCE_PROJECT/62443-4-2_RefFramework.baseline - Eclipse Platform
Argumentation
OSLC-KM
Process Lines
VV
CHESS
Window
Help
Import evidence model from file
Index file
Look for similar models
Index CDO Project
Query to search into indexed CDO
Indexes a CDO project, for Reuse Discovery purposes
ity.assuranceproject
62443-4-2_RefFra
RefFramework.baseline

Figure 54. Actions within the AMASS Platform


Assurance Projects

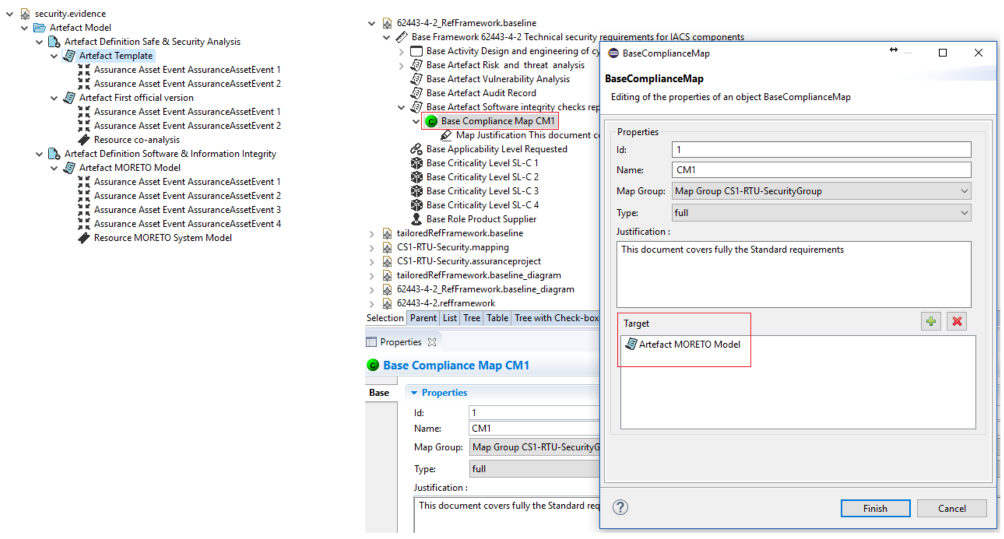
Select source assurance project:

TAS-E
DC-Motor-SafetyCase
CS1- RTU
CS1-RTU-Security
testing
Security Requirements for Safety-critical System
test_arg
Test_forTASI
CEA_WP4_validation
A4T_Test
CEA_WP4_validation_local
SPS_Test
test-SPS-project
test-int
Test_Proc_Arg
DME Monitor SW
Bsw_Dep
Bsw_STO2
zdfh
testALC
Bsw_tentative1000
bsw
GMV_CS4
CS1-RTU-Security-P1
TAS-E Multi-concernAssurance
CLS_Test
CS1-RTU-P1
ESL
Test1_OpenCert
testproject
testproject1
TestProject833

OK Cancel

Figure 55. Project Folder selection

Once the project is selected, the content of the project in the CDO (see image) will be indexed in the OSLC-KM representation schema.


security.evidence
- Artefact Model
  - Artefact Definition Safe & Security Analysis
    - Artefact Template
      - Assurance Asset Event AssuranceAssetEvent 1
      - Assurance Asset Event AssuranceAssetEvent 2
    - Artefact First official version
      - Assurance Asset Event AssuranceAssetEvent 1
      - Assurance Asset Event AssuranceAssetEvent 2
      - Resource co-analysis
  - Artefact Definition Software & Information Integrity
    - Artefact MORETO Model
      - Assurance Asset Event AssuranceAssetEvent 1
      - Assurance Asset Event AssuranceAssetEvent 2
      - Assurance Asset Event AssuranceAssetEvent 3
      - Assurance Asset Event AssuranceAssetEvent 4
      - Resource MORETO System Model

62443-4-2_RefFramework.baseline
- Base Framework 62443-4-2 Technical security requirements for IACS components
  - Base Activity Design and engineering of cy
  - Base Artefact Risk and threat analysis
  - Base Artefact Vulnerability Analysis
  - Base Artefact Audit Record
  - Base Artefact Software integrity checks rep
    - Base Compliance Map CM1
      - Map Justification This document c
  - Base Applicability Level Requested
  - Base Criticality Level SL-C 1
  - Base Criticality Level SL-C 2
  - Base Criticality Level SL-C 3
  - Base Criticality Level SL-C 4
  - Base Role Product Supplier

tailoredRefFramework.baseline
CS1-RTU-Security.mapping
CS1-RTU-Security.assuranceproject
tailoredRefFramework.baseline_diagram
62443-4-2_RefFramework.baseline_diagram
62443-4-2.refframework

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

Properties

Base Compliance Map CM1

Base | Properties

Id: 1
Name: CM1
Map Group: Map Group CS1-RTU-SecurityG
Type: full
Justification : This document covers fully the Standard req

BaseComplianceMap

BaseComplianceMap

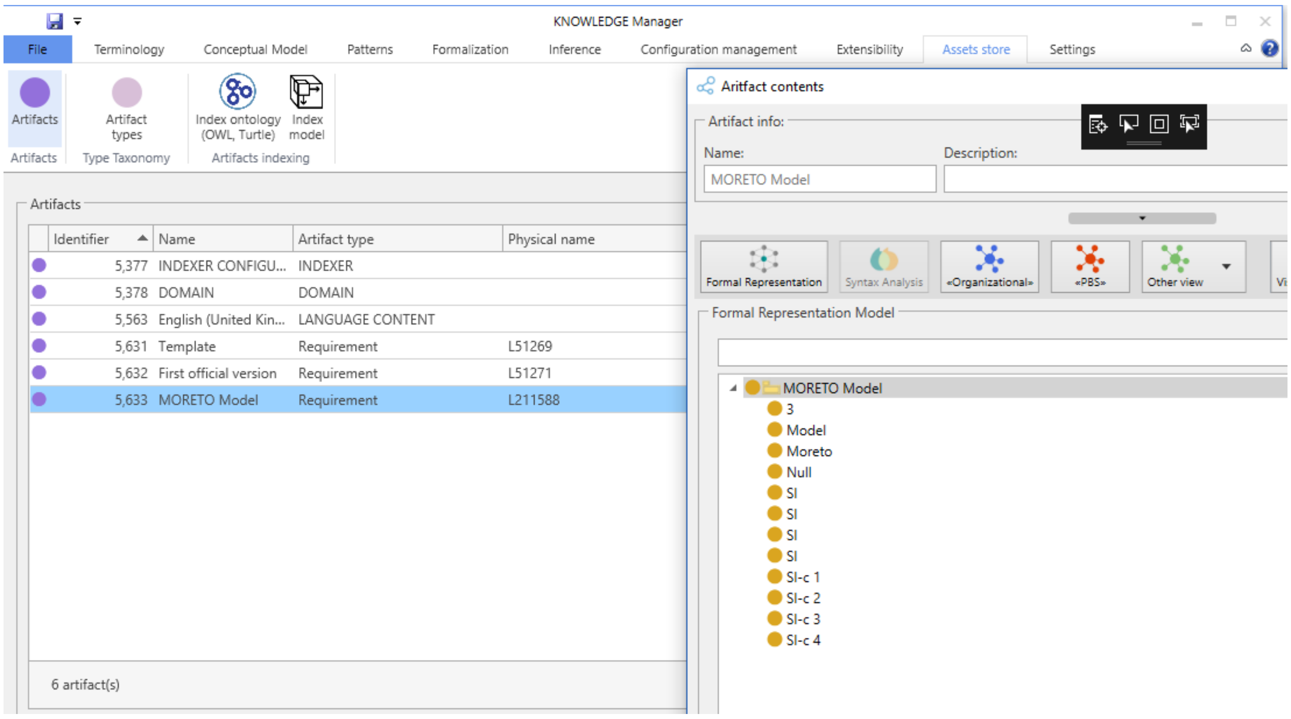
Editing of the properties of an object BaseComplianceMap

Properties

Id: 1
Name: CM1
Map Group: Map Group CS1-RTU-SecurityGroup
Type: full
Justification : This document covers fully the Standard requirements

Target
Artefact MORETO Model

Figure 56. Content of the selected project



Artifacts

	Identifier	Name	Artifact type	Physical name
●	5,377	INDEXER CONFIGU...	INDEXER	
●	5,378	DOMAIN	DOMAIN	
●	5,563	English (United Kin...	LANGUAGE CONTENT	
●	5,631	Template	Requirement	L51269
●	5,632	First official version	Requirement	L51271
●	5,633	MORETO Model	Requirement	L211588

6 artifact(s)

Artifact contents

Artifact info:

Name: MORETO Model

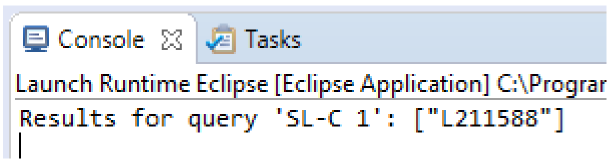
Description:

Formal Representation Model

- MORETO Model
  - 3
  - Model
  - Moreto
  - Null
  - SI
  - SI
  - SI
  - SI
  - SI-c 1
  - SI-c 2
  - SI-c 3
  - SI-c 4

Figure 57. Project indexed in OSLC-KM

On the other hand, the OSLC-KM service also provides with an API function to search by text. We can query SIL levels to check that the service retrieves indexed artefacts. This functionality will be integrated in the Reuse Assistant component.



```
Launch Runtime Eclipse [Eclipse Application] C:\Prograr
Results for query 'SL-C 1': ["L211588"]
```

Figure 58. Query results

3.1.2.14 Reuse Discovery based on ElasticSearch

With P1 a generic indexing feature based on Elasticsearch has been implemented in a first version. Indexing is possible at EMF (Eclipse Modelling Framework) level, independent on what kind of EMF storage system is used (CDO, XMI files, or any other). Objects and their attributes are indexed into any Elasticsearch server and can be visualized using Dashboard facilities as Kibana. The indexing is not automated yet but has to be triggered manually on an object, a resource or a whole object tree. The conceptual approach, challenges and way forward were described in detail in D6.3 [30].

With P2, an additional high-level "Search and Query API" was added to support value added features as reuse assistant or impact analysis. The API is again on rather high EMF level, that means any query can be made on high abstraction level (aka object level), is transformed into low level Elasticsearch REST calls and the reply (basically hits) is transferred back to EMF Object level. That includes resolution strategies and object identification for CDO but also file based (workspace) EMF storages.

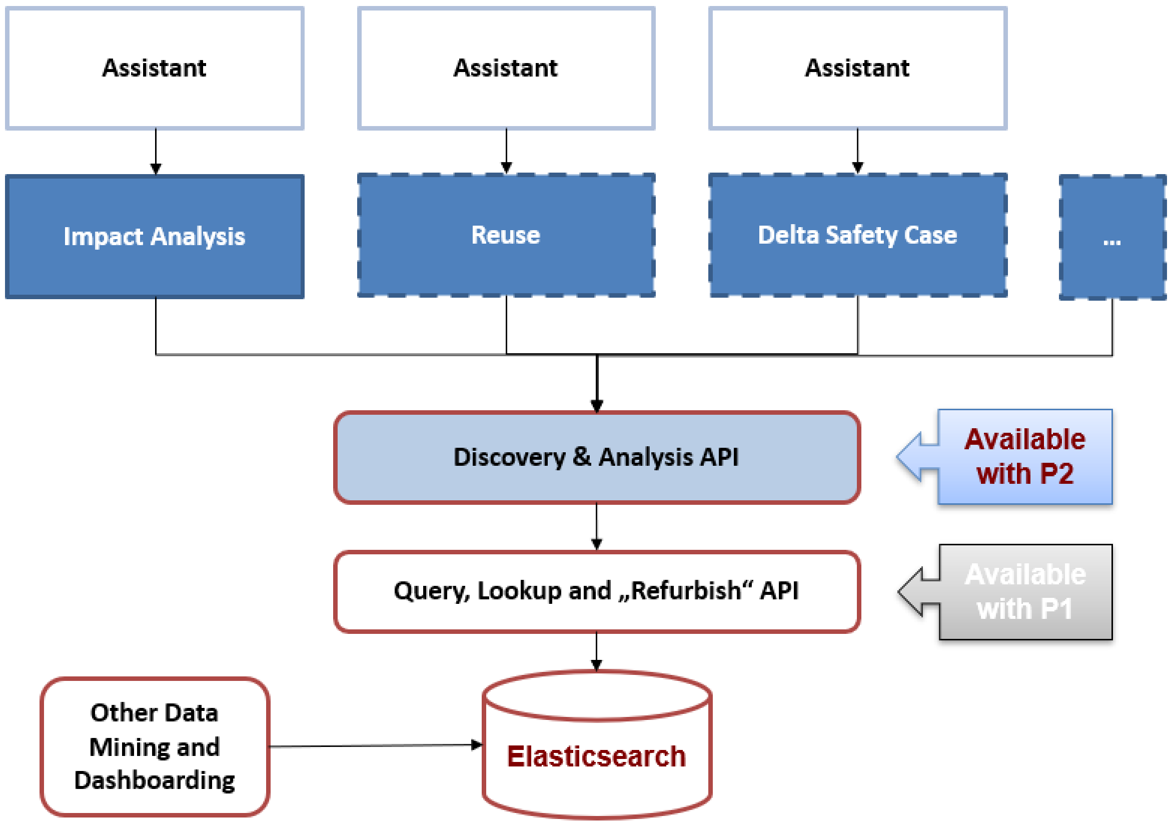
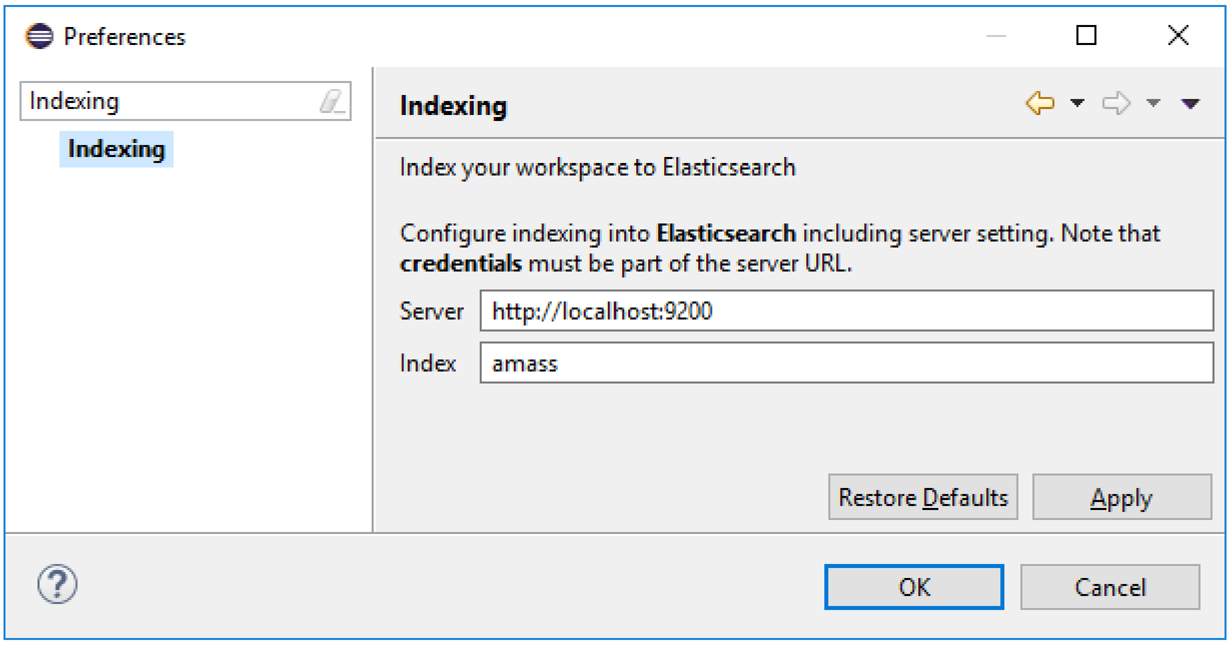


Figure 59. Big Picture of the Reuse Discovery functionality with ElasticSearch

The component requires a running Elasticsearch server. That server may run on any node in the IP network. Note that originally an Elasticsearch server could be embedded into any Java runtime environment (e.g. Eclipse Runtime) but that option was removed by the Elasticsearch development team later on. Anyway, installation and setup is rather easy and straight on all major operating systems.

ZIP packages are available at https://www.elastic.co/products/elasticsearch and can be downloaded freely. They just need to be extracted to an arbitrary location and the server can be started by using MS CMD or UNIX batch scripts. P1 was basically built on and thus supporting Elasticsearch version 5.0.0. For P2 the code was ported to Elasticsearch 5.6.x which should be also the minimum at the server side. Although the REST API is rather stable over time, there are a few obstacles that may occur when using an older version. Note that upcoming version 7.x is not supported due to a major change in the handling of indexes and types. Version 6.x on the other hand should be supported because it offers backward compatibility.

Once the server is up and running, the AMASS client can be configured in the Eclipse preferences. By default, Elasticsearch is started on port 9200 which would lead to “http://localhost:9200” as the default server URL - assuming Elasticsearch is running on local machine. An index name must be specified as well but an arbitrary alphanumeric name can be chosen here.



Index your workspace to Elasticsearch

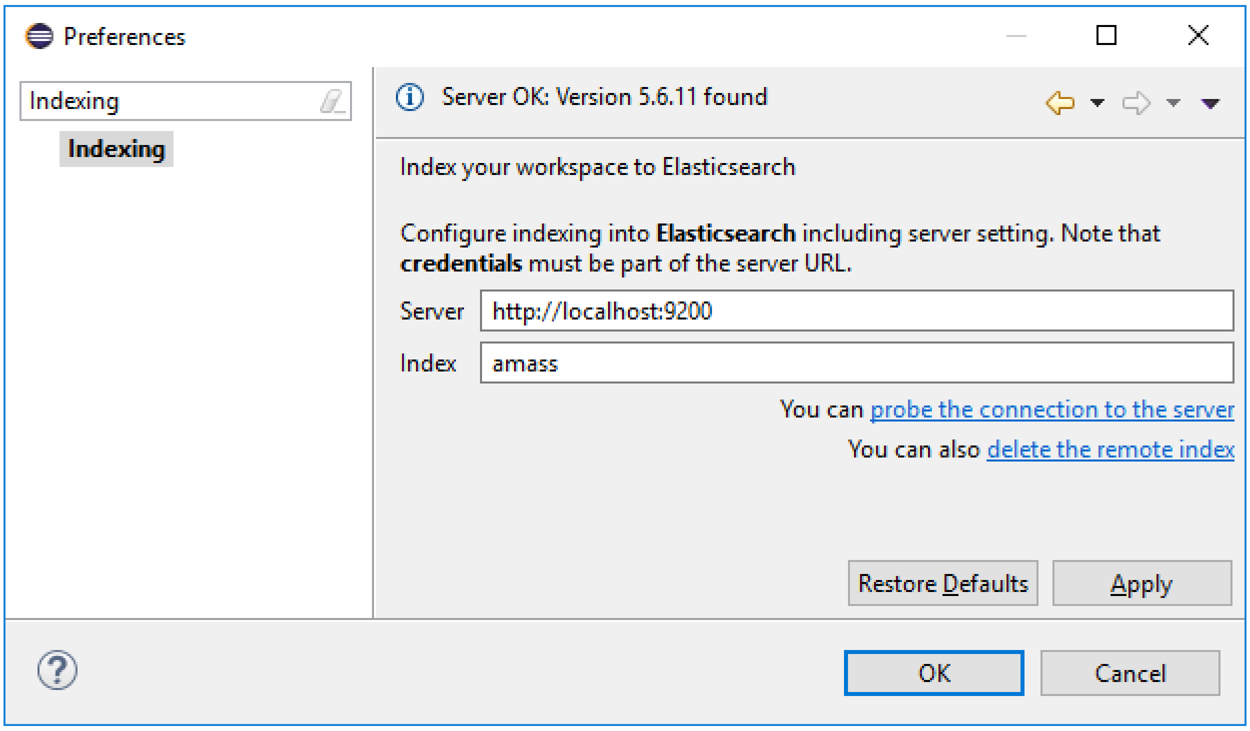
Configure indexing into **Elasticsearch** including server setting. Note that **credentials** must be part of the server URL.

Server: http://localhost:9200

Index: amass

Figure 60. ElasticSearch Preferences in P1

With P2 the preferences were improved to allow to ping the server but also to clean the remote index (this was only possible using the Kibana dashboard in P1).



Server OK: Version 5.6.11 found

Index your workspace to Elasticsearch

Configure indexing into **Elasticsearch** including server setting. Note that **credentials** must be part of the server URL.

Server: http://localhost:9200

Index: amass

You can probe the connection to the server

You can also delete the remote index

Figure 61. ElasticSearch Preferences improved in P2

An additional toolbar shows the status of the Elasticsearch configuration and connection at any time to the user. There are two "bulbs" to reflect the status, one showing the server connection and details, the second the status of the index. Both should be green in best case. Note that the search index is created on first usage, so it will switch from yellow to green at that time.

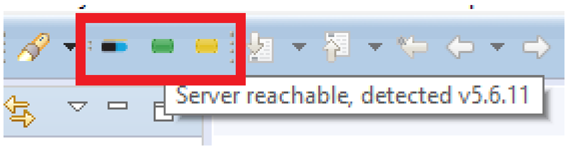


Figure 62. ElasticSearch status icons

2.3 Installation and User Manuals

The steps necessary to install the AMASS Prototype P2 are exhaustively described in the AMASS User Manual [18] and will not be repeated here. That document contains all required steps and document references to set up the tools. A pre-packaged distribution will be supplied in this iteration of the AMASS platform.

In summary, the AMASS User Manual itself is the user guide of the AMASS tool prototype implementation. The users can find the installation instructions, the tool environment description, and the functionalities described in this document.

3. Implementation Description

3.1 Implemented Modules

3.1.1 Compliance Management

We have decomposed the Compliance Management module into three components: *Standards Editor*, *Process Editor* and *Compliance Editor* (see Figure 63). The *Standards Editor* tool component includes services to capture the knowledge from the standards, while the *Process Editor* tool captures information of the life-cycles that are described in the process. The focus of the third component, the *Compliance Editor* tool, is to map the information from the standards (i.e. obligations) with the information managed in the context of a given assurance project (i.e. accomplishment). We use two toolsets for compliance management: OpenCert is used for modelling standards and processes, as well as compliance maps between those models and the assurance and certification assets created in specific projects; and EPF (Eclipse Process Framework) is used for modelling processes and for modelling standards as well.

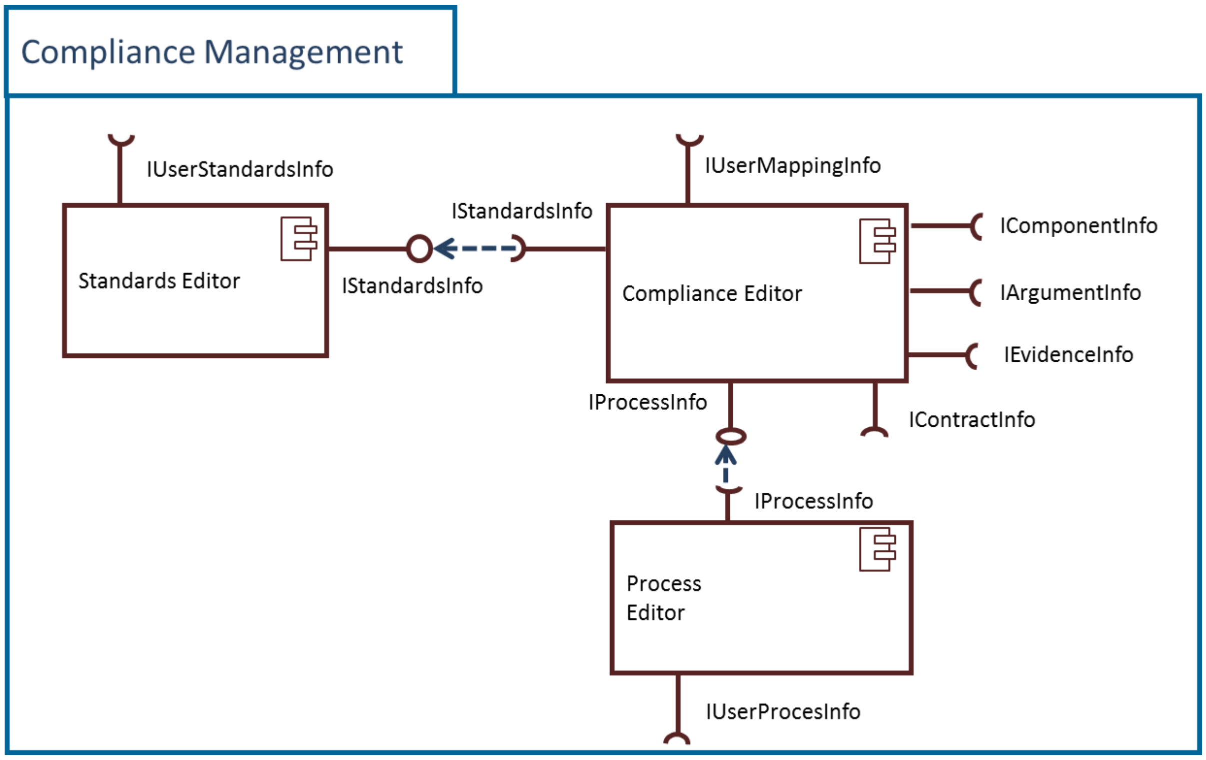


Figure 63. Tool components for Compliance Management

3.1.2 Management of families/lines

To manage families/lines, it is necessary to have at disposal modelling means for systematising commonalities and variabilities. These means might be provided as either a specific solution targeting a single type of family (e.g., a process line) or as an orthogonal solution applicable to any type of family

3.1.2.1 Semi-automatic generation of process-based arguments

This section provides the big picture of the solution for the automatic generation of process-based arguments within AMASS (see Figure 64). The solution embraces both phases of the *certification liaison process* (which is explicitly defined within DO-178C and implicitly in place in all certification/qualification frameworks): the planning and the execution phase.

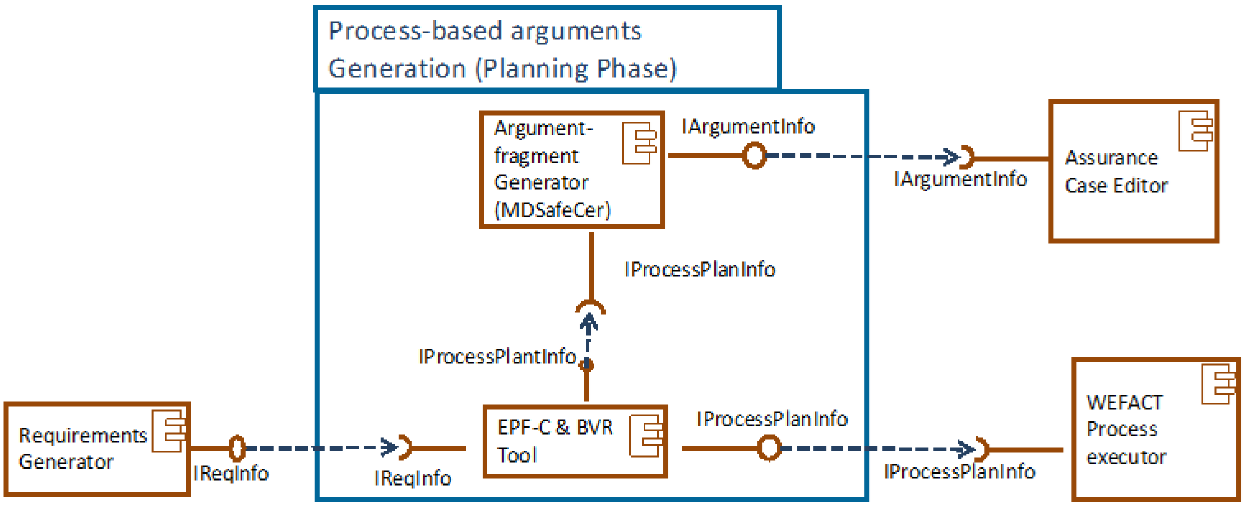


Figure 64. Tool components for Automatic generation of process-based arguments

3.1.2.2 Semi-automatic generation of process-based arguments representing plans

This section provides the big picture of the solution for the generation of *fallacy free process-based arguments* during the planning phase within AMASS (see Figure 65). The solution embraces, at the planning phase, a specific process plan is derived from the family of processes (managed by EPF-C & BVR Tool), then an argument is automatically generated by following MDSafeCer [32] principles and visualized via the Assurance Case Editor. Furthermore, the detect fallacies solution validates the process models (managed by EPF Composer), and prevents the occurrence of fallacy, specifically, omission of key evidence in process-based arguments.

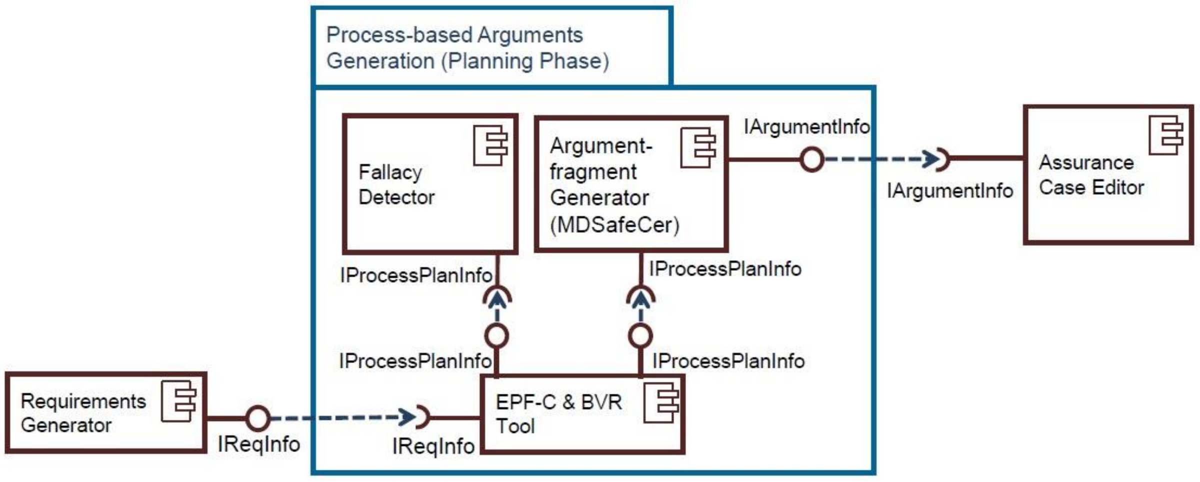


Figure 65. Tool components for generation of fallacy free process-based arguments

3.1.2.3 Semi-automatic generation of product-based arguments

This section provides the big picture of the solution for the automatic generation of product-based arguments (see Figure 66). The generator uses a pre-existing argument pattern for the generation. The generated argument-fragments include only those contracts whose assumptions are validated, hence only those artefacts related to the validated contracts.

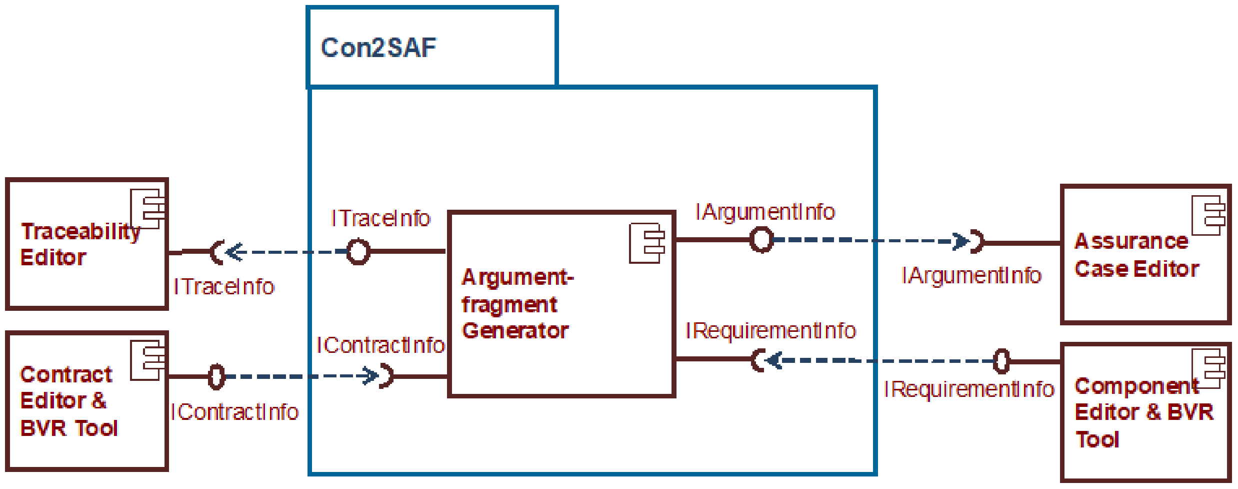


Figure 66. Tool components for Automatic generation of product-based arguments

3.1.3 Cross/Intra Domain Reuse

The Cross/Intra domain reuse module is composed of two main components: *Reuse Assistance* and *Line Specification* (see Figure 67). The *Reuse Assistance* tool focuses in intra and cross-domain reuse of assurance and certification assets. The *Line Specification* tool is a composite component constituted of a seamless integrator and a set of editors enabling the management of variability. More specifically, the Line Specification can get in input models regarding Product/Process/Assurance Cases and via the Seamless Integrator these models can be cleaned if necessary and used to feed the Variability Editor, the Resolution Editor and the Realization Editor, which can be used to change the models in accordance to the Line constraints.

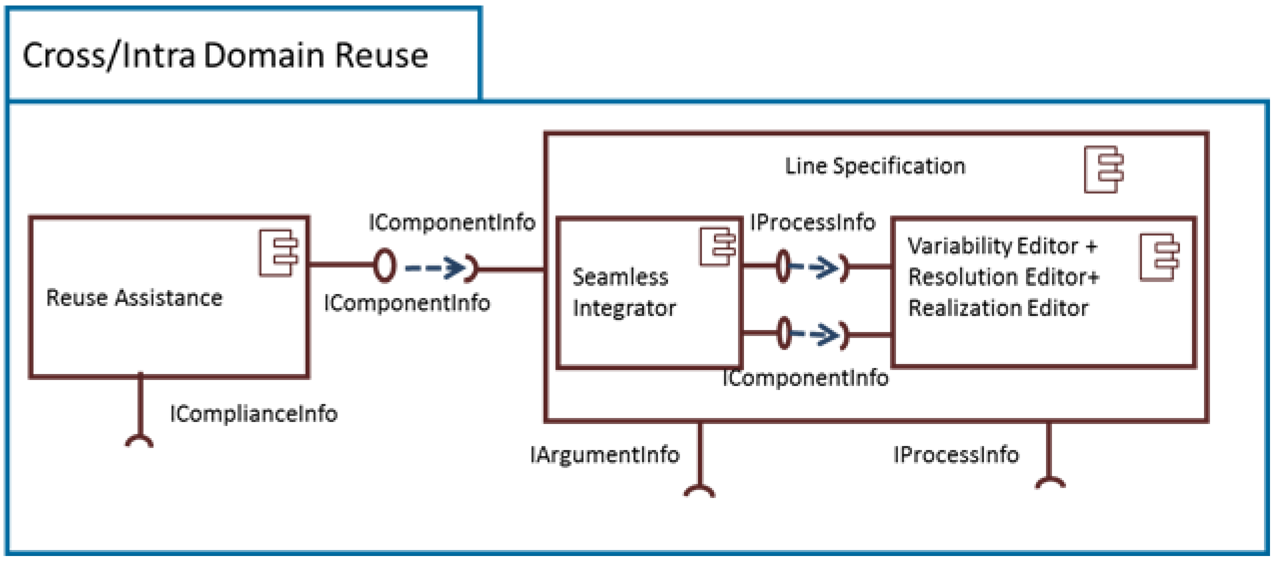


Figure 67. Tool components for Cross/Intra Domain Reuse

3.2 Source Code Description for the AMASS Tool Platform

The source code of the WP6-related components in AMASS prototype P2 can be found in the source code GIT repository [17].

The code for the Compliance Management module in Prototype Core was stored together with the code of the other building blocks in the SVN repository under "tag" to distinguish the state of the code at the time of the integrated release.

The necessary plugins for the Standards, Projects, Compliance and Processes Management Specification are:

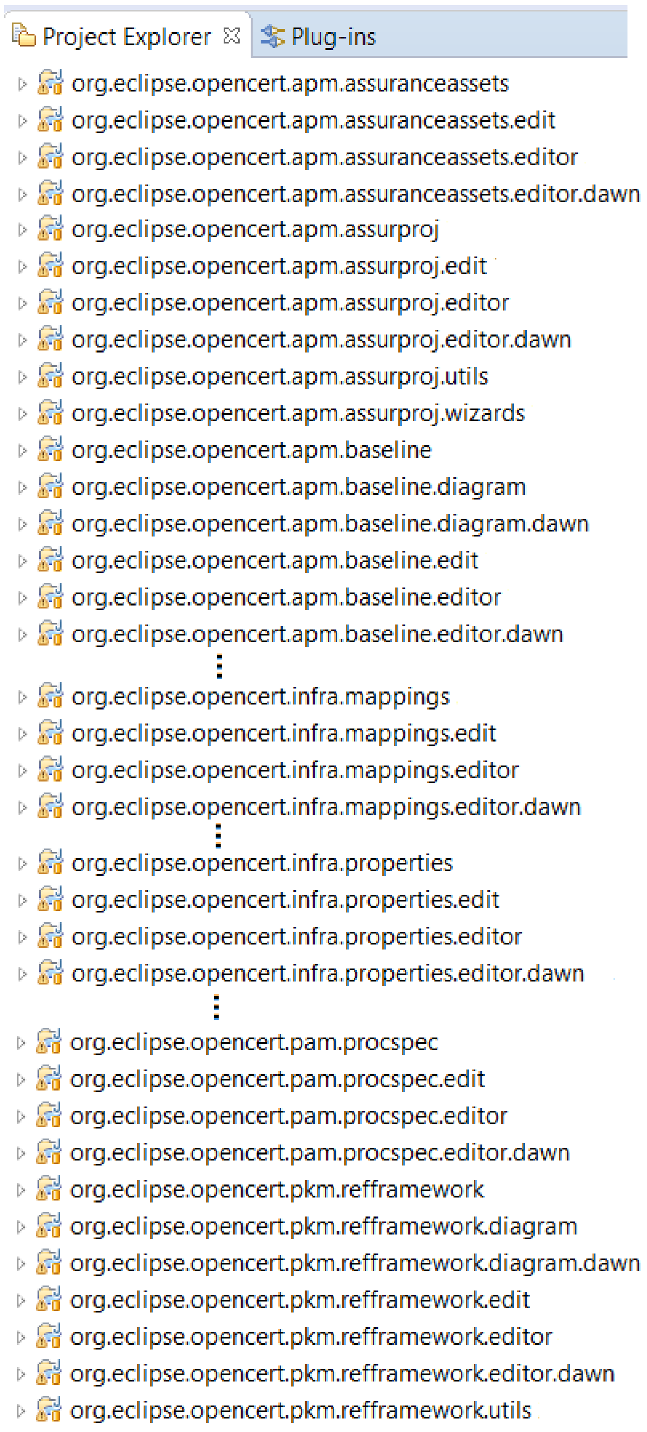
- **org.eclipse.opencert.apm.assuranceassets**
  In this plugin, the Assurance Assets metamodel is defined and stored, and the Java implementation classes for this model are generated.
- **org.eclipse.opencert.apm.assuranceassets.edit**
  The edit plugin includes adapters that provide a structured view and perform command-based edition of the Assurance Assets model objects.
- **org.eclipse.opencert.apm.assuranceassets.editor**
  This plugin provides the user interface to view instances of the model using several common viewers, and to add, remove, cut, copy and paste model objects, or modify the objects in a standard property sheet. This editor saves the generated data in a local file.
- **org.eclipse.opencert.apm.assuranceassets.editor.dawn**
  This plugin is an extension of the previous one. It aims to communicate with the CDO Server to store the generated model in a database instead of a local file.
- **org.eclipse.opencert.apm.assurproj**
  In this plugin, the Assurance Project metamodel is defined and stored, and the Java implementation classes for this model are generated.
- **org.eclipse.opencert.apm.assurproj.edit**
  The edit plugin includes adapters that provide a structured view and perform command-based edition of the Assurance Project model objects. This plugin also includes the import functionality from EPF process models into OpenCert Evidence and Process models.
- **org.eclipse.opencert.apm.assurproj.editor**
  This plugin provides the user interface to view instances of the model using several common viewers, and to add, remove, cut, copy and paste model objects, or modify the objects in a standard property sheet. This editor saves the generated data in a local file.
- **org.eclipse.opencert.apm.assurproj.editor.dawn**
  This plugin is an extension of the previous one. It aims to communicate with the CDO Server to store the generated model in a database instead of a local file.
- **org.eclipse.opencert.apm.assurproj.reuse**
  This plugin includes the reuse view that offers the reuse assistance functionalities.
- **org.eclipse.opencert.apm.assurproj.utils**
  This plugin provides additional features for to the standard CheckboxTreeViewer.
- **org.eclipse.opencert.apm.assurproj.wizards**
  This plugin provides a wizard to facilitate to the user the assurance project creation process and another wizard for adding a new baseline to an existing assurance project or updating an existing baseline of a project.
- **org.eclipse.opencert.pkm.baseline**
  In this plugin, the Baseline definition metamodel is defined and stored, and the Java implementation classes for this model are generated.
- **org.eclipse.opencert.pkm.baseline.edit**
  This plugin contains a provider to display Baseline definition models in a user interface. This plugin also contains the window to view the existing compliance maps of a project and the window to create or update one project's compliance maps.
- **org. eclipse.opencert.pkm.baseline.editor**

This plugin provides the user interface to view instances of the model in a tree-based way using several common viewers, and to add, remove, cut, copy and paste model objects, or modify the objects in a standard property sheet. This editor saves the generated data in a local file.

- **org.eclipse.opencert.pkm.baseline.editor.dawn**
  This plugin is an extension of the previous one. It aims to communicate with the CDO Server to store the generated model in a database instead of a local file.
- **org.eclipse.opencert.pkm.baseline.diagram**
  This plugin provides the user interface to view instances of the model in a graphical way using several common viewers, and to add, remove, cut, copy and paste model objects, or modify the objects in a standard property sheet. This editor saves the generated data in a local file.
- **org.eclipse.opencert.pkm.baseline.diagram.dawn**
  This plugin is an extension of the previous one. It aims to communicate with the CDO Server to store the generated diagram model and the standard definition model in a database instead of a local file.
- **org.eclipse.opencert.infra.mappings**
  In this plugin, the Mapping metamodel is defined and stored, and the Java implementation classes for this model are generated.
- **org.eclipse.opencert.infra.mappings.edit**
  This plugin contains a provider to display Mapping models in a user interface.
- **org.eclipse.opencert.infra.mappings.editor**
  This plugin provides the user interface to view instances of the model in a tree based way using several common viewers, and to add, remove, cut, copy and paste model objects, or modify the objects in a standard property sheet. This editor saves the generated data in a local file.
- **org.eclipse.opencert.infra.mappings.editor.dawn**
  This plugin is an extension of the previous one. It aims to communicate with the CDO Server to store the generated Mapping model in a database instead of a local file.
- **org.eclipse.opencert.infra.properties**
  In this plugin, the Property metamodel is defined and stored, and the Java implementation classes for this model are generated.
- **org.eclipse.opencert.infra.properties.edit**
  This plugin contains a provider to display Property models in a user interface.
- **org.eclipse.opencert.infra.properties.editor**
  This plugin provides the user interface to view instances of the model in a tree-based way using several common viewers, and to add, remove, cut, copy and paste model objects, or modify the objects in a standard property sheet. This editor saves the generated data in a local file.
- **org.eclipse.opencert.infra.properties.editor.dawn**
  This plugin is an extension of the previous one. It aims to communicate with the CDO Server to store the generated Property model in a database instead of a local file.
- **org.eclipse.opencert.pam.procspec**
  In this plugin, the Process definition metamodel is defined and stored, and the Java implementation classes for this model are generated.
- **org.eclipse.opencert.pam.procspec.edit**
  This plugin contains a provider to display Process definition models in a user interface.
- **org.eclipse.opencert.pam.procspec.editor**
  This plugin provides the user interface to view instances of the model in a tree-based way using several common viewers, and to add, remove, cut, copy and paste model objects, or modify the objects in a standard property sheet. This editor saves the generated data in a local file.

- **org.eclipse.opencert.pam.procspec.editor.dawn**
  This plugin is an extension of the previous one. It aims to communicate with the CDO Server to store the generated process model in a database instead of a local file.
- **org.eclipse.opencert.pkm.refframework**
  In this plugin, the Standard definition metamodel is defined and stored, and the Java implementation classes for this model are generated.
- **org.eclipse.opencert.pkm.refframework.edit**
  This plugin contains a provider to display standard definition models in a user interface.
- **org.eclipse.opencert.pkm.refframework.editor**
  This plugin provides the user interface to view instances of the model in a tree-based way using several common viewers, and to add, remove, cut, copy and paste model objects, or modify the objects in a standard property sheet. This editor saves the generated data in a local file.
- **org.eclipse.opencert.pkm.refframework.editor.dawn**
  This plugin is an extension of the previous one. It aims to communicate with the CDO Server to store the generated model in a database instead of a local file.
- **org.eclipse.opencert.pkm.refframework.diagram**
  This plugin provides the user interface to view instances of the model in a graphical way using several common viewers, and to add, remove, cut, copy and paste model objects, or modify the objects in a standard property sheet. This editor saves the generated data in a local file.
- **org.eclipse.opencert.pkm.refframework.diagram.dawn**
  This plugin is an extension of the previous one. It aims to communicate with the CDO Server to store the generated diagram model and the standard definition model in a database instead of a local file.
- **org.eclipse.opencert.chessdiagram.adapter**
  This plugin adopts CHESS editor to interact with the BVR tool bundle. *BVREnabledEditor* is expected to interact with the CHESSBVREditor in order to highlight/select modelling elements to be placed/replaced as well as to exportTailoredProducts.
- **org.eclipse.opencert.lines (*)**
  This plugin provides support for process, product and assurance case variability management, and integration of process-product-assurance case lines for the purpose of change impact analysis. The integration between EPF Composer, CHESS Tool, OpenCert Tool and BVR Tool is achieved for the establishment of lines. For the multiple/integrated lines, the cross-cutting constraints between variability models, joining of resolutions and simultaneous execution of realization fragments belonging to multiple base models are supported.
- **org.eclipse.opencert.epf.generateArgumentation (*)**
  This plugin provides the transformation of an EPF process (compliant with UMA metamodel) to argumentation model and diagram (compliant with CACM metamodel). The generated model and diagram are stored in the ARGUMENTATION folder of the selected assurance case project in the CDO repository. The user is prompted to select both source process model (Capability Pattern/Delivery Process) and target assurance case project. The target model (argument) is also saved locally in a new project into the current Workspace under the name Argumentation.
- **org.eclipse.opencert.epf.detectFallacies (*)**
  This plugin facilitates the detection of fallacies by identifying whether the safety process modelled in EFP Composer contains the sufficient information corresponding to the key evidence for supporting the specific requirement. The user is prompted to select the folder where he or she wants to store the validation reports .txt files. Moreover, the results are presented on the console.
- **org.eclipse.opencert.epf.transfromRequirements (*)**
  This plugin provides the transformation of standard's requirements modelled in EPF Composer (compliant with UMA metamodel) to baseline models (compliant with CACM metamodel). The generated models will be stored in selected assurance case project in the CDO repository.

- **org. eclipse.opencert.evm.evidspec.editor (*)**
  This plugin provides the user interface to view instances of the evidence model in a tree-based way using several common viewers, and to add, remove, cut, copy and paste model objects, or modify the objects in a standard property sheet.
  This plugin also allows the introduction of the searching parameters to find reusable artefacts by means of the OSLC-KM Reuse Discovery API or the Elastic Search Reuse Discovery API and shows the results to the user.



```
org.eclipse.opencert.apm.assuranceassets
org.eclipse.opencert.apm.assuranceassets.edit
org.eclipse.opencert.apm.assuranceassets.editor
org.eclipse.opencert.apm.assuranceassets.editor.dawn
org.eclipse.opencert.apm.assurproj
org.eclipse.opencert.apm.assurproj.edit
org.eclipse.opencert.apm.assurproj.editor
org.eclipse.opencert.apm.assurproj.editor.dawn
org.eclipse.opencert.apm.assurproj.utils
org.eclipse.opencert.apm.assurproj.wizards
org.eclipse.opencert.apm.baseline
org.eclipse.opencert.apm.baseline.diagram
org.eclipse.opencert.apm.baseline.diagram.dawn
org.eclipse.opencert.apm.baseline.edit
org.eclipse.opencert.apm.baseline.editor
org.eclipse.opencert.apm.baseline.editor.dawn
⋮
org.eclipse.opencert.infra.mappings
org.eclipse.opencert.infra.mappings.edit
org.eclipse.opencert.infra.mappings.editor
org.eclipse.opencert.infra.mappings.editor.dawn
⋮
org.eclipse.opencert.infra.properties
org.eclipse.opencert.infra.properties.edit
org.eclipse.opencert.infra.properties.editor
org.eclipse.opencert.infra.properties.editor.dawn
⋮
org.eclipse.opencert.pam.procspec
org.eclipse.opencert.pam.procspec.edit
org.eclipse.opencert.pam.procspec.editor
org.eclipse.opencert.pam.procspec.editor.dawn
org.eclipse.opencert.pkm.refframework
org.eclipse.opencert.pkm.refframework.diagram
org.eclipse.opencert.pkm.refframework.diagram.dawn
org.eclipse.opencert.pkm.refframework.edit
org.eclipse.opencert.pkm.refframework.editor
org.eclipse.opencert.pkm.refframework.editor.dawn
org.eclipse.opencert.pkm.refframework.utils
```

Figure 68. Cross-Domain and Intra-Domain Reuse plugins

In addition, the following plugin from the CHESS Polarsys project has been updated:

- **org.polarsys.chess.service**
  This plugin provides some utility functions for the CHESS editor; the plugin has been extended in the context of WP6 with a new feature regarding the removal of orphan views from the Papyrus UML diagrams. Orphan views are graphical elements of a Papyrus diagram that do not have any corresponding semantic element in the UML model. For instance, orphan views can originate in a Papyrus diagram if the associated entities in the UML model are removed without using the Papyrus editor facilities; this is the case when the CHESS model is modified to implement variability at product level trough the BVR tool (see section 3.1.2.8).

  The plugin is available from https://git.polarsys.org/c/chess/chess.git repository, together with the other CHESS related plugins.

Furthermore, the functionality for Reuse Discovery has been integrated within the following plugin:

- **org.eclipse.opencert.evm.oslc.km.importevid**
  This plugin provides both, the functionality for indexing Papyrus models into the KM database, through OSLC-KM, and the capability to look for similar Papyrus models given a Papyrus model as a query.

The functionality for the product-based argument generation is located in the following plugin:

- **org.eclipse.opencert.chess.argumentGenerator**
  This plugin facilitates generation of argument-fragments for each component of a CHESS model. It stores the argument files in the ARGUMENTATION folder of the selected CDO repository assurance case. The user is prompted to select both source CHESS model and target assurance case. The plugin is available on the AMASS_source.

3.3 Source Code Description for External Tools

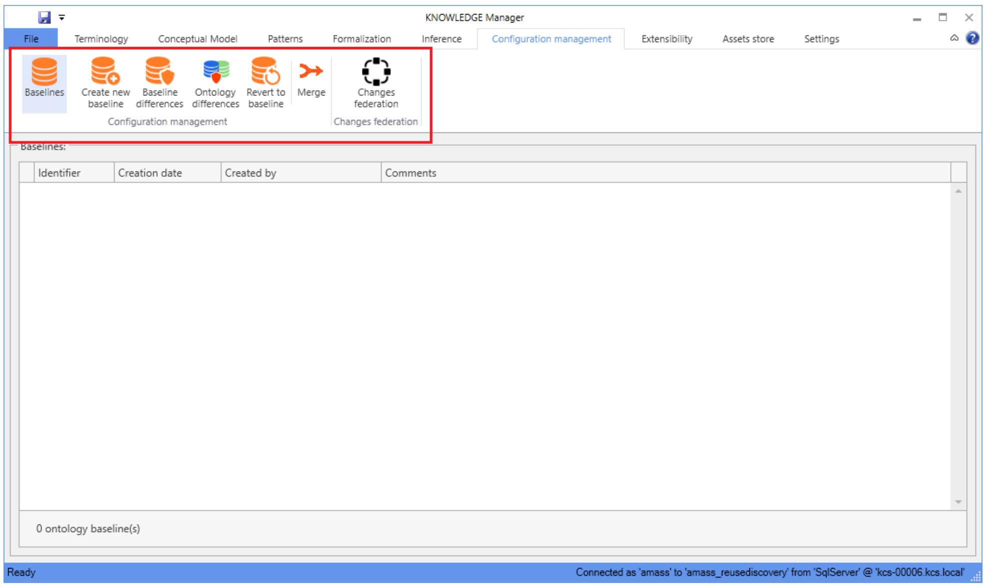
3.3.1 OSLC-KM integration

The core functionality for the Reuse Discovery section has been developed within the TRC toolset (mainly Knowledge Manager for the WP6). The interoperability with the AMASS platform has been performed via the OSLC-KM, using some of the operations defined in the context of WP5. Specially, the implementation of the functionality of the WP6 takes advantage of the following components:

- **Rqa.Face.OslcKM**: it implements the connection of the OSLC-KM model instance with the rest of the functionalities of the TRC toolset.
- **System Representation Language (SRL)**: metamodel of the OSLC-KM layer.

3.3.2 Ontology Reuse Operations

In the context of AMASS WP6, some operations for ontology reuse have been developed in the TRC toolset (KM):



0 ontology baseline(s)

Ready

Connected as 'amass' to 'amass_reusediscovery' from 'SqlServer' @ 'kcs-00006.kcs.local'

Figure 69. Ontology Reuse Operations

4. Conclusions

This deliverable presented the implementation work performed for Cross-Domain and Intra-Domain Reuse in AMASS Prototype P2, which is the latest version of the AMASS Tool Platform. The current support in the Platform allows a user to manage variability at the product, process and component level, semi-automatic generation of product/process arguments, and some external support provided for reuse discovery.

As its current state and prior to validation in WP2 and application in WP1, Cross-Domain and Intra-Domain Reuse support for Prototype P2 should have TRL 5 (technology validated in real use cases). Basic technological components are integrated to establish that they work together.

In addition to the implementation of further requirements in the AMASS tool Platform (and also to the general revision of some implementation for enhancement), the Prototype P2 includes the final decision upon the implementation of new features targeted at: reuse discovery, reuse assistant, variability management at the product/process/component level, and generation of product/process arguments.

Abbreviations and Definitions

Abbreviation	*Explanation*
API	Application Programming Interface
ASIL	Automotive Safety Integrity Level
BVR	Base Variability Resolution
CACM	Common Assurance and Certification Metamodel
CDO	Connected Data Objects
CHESS	Composition with Guarantees for High-Integrity Embedded Software Components Assembly
CHESSML	CHESS Modelling Language
CPF	Canonical Predictive Form
CPS	Cyber-Physical Systems
DAL	Development Assurance Levels
DIN	Deutsches Institut für Normung
EBIOS	Expression des Besoins et Identification des Objectifs de Sécurité (Expression of Needs and Identification of Security Objectives)
ECSEL	Electronic Components and Systems for European Leadership
EMF	Eclipse Modelling Framework
EN	European Norm
EPF	Eclipse Process Framework
ETL	Epsilon Transformation Language
EU	European Union
FT	Fault Tolerance
GSN	Goal Structured Notation
GUID	Globally Unique Identifier
HW	Hardware
IEC	International Electrotechnical Commission
IED	Intelligent Electronic Devices
IEEE	Institute of Electrical and Electronics Engineers
IL	Impact Level
ISO	International Organization for Standardization
ITS	Intelligent Transport Systems
JU	Joint Undertaking
KM	Knowledge Manager
OCRA	Othello Contracts Refinement Analysis
OMG	Object Management Group
OSLC	Open Services for Lifecycle Collaboration
OSLC-KM	OSLC for Knowledge Management
PhD	Philosophiae Doctor (neolatin; = doctor of philosophy)
RSHP	Relationship Model
RTCA	Radio Technical Commission for Aeronautics
RT	Real-Time
SACM	Structured Assurance Case Metamodel
SDLC	(Microsoft) Secure Development Life Cycle

Abbreviation	*Explanation*
SIL	Safety Integrity Level
SKB	System Knowledge Base
SL	Security Level
SoPLs	Safety-oriented Process Lines
SPEM	Software & Systems Process Engineering Metamodel
STO	Scientific and Technical Objective
SVN	Subversion
SW	Software
SysML	System Modelling Language
TOM	Trade-Off Method
TRC	The REUSE Company
TRL	Technology Readiness Level
UMA	Unified Method Architecture
UML	Unified Modelling Language
V&V	Verification and Validation
XMI	XML Metadata Interchange
XML	Extensible Markup Language
WP	Work Package

References

[1] The OPENCOSS project http://www.opencoss-project.eu/

[2] The SafeCer project http://cordis.europa.eu/project/rcn/103721_en.html and http://cordis.europa.eu/project/rcn/105610_en.html

[3] OMG - Semantics of Business Vocabulary and Rules™ (SBVR™) version 1.3, 2015 http://www.omg.org/spec/SBVR/1.3

[4] OMG - SACM - Object Management Group version 1.1, 2015 http://www.omg.org/spec/SACM/1.1

[5] Origin Consulting GSN Community Standard Version 1 (2011)

[6] Graphical Modelling Project (GMP) http://www.eclipse.org/modeling/gmp/

[7] Eclipse Modelling Framework (EMF) https://www.eclipse.org/modeling/emf/

[8] Epsilon Transformation Language http://www.eclipse.org/epsilon/doc/etl/

[9] Xtext http://www.eclipse.org/Xtext/

[10] Eugenia http://www.eclipse.org/epsilon/doc/eugenia/

[11] CDO http://www.eclipse.org/cdo/

[12] OSLC http://open-services.net/specifications/

[13] AMASS project: D2.1 Business cases and high-level requirements (28 February 2017)

[14] AMASS project: D3.6 Prototype for architecture-driven assurance (c) (31 August 2018)

[15] AMASS project: D3.4 Prototype for architecture-driven assurance (a) (23 December 2016).

[16] AMASS project: D5.3 Design of the AMASS tools and methods for seamless interoperability (b) (29 June 2018)

[17] AMASS project: source code https://git.polarsys.org/r/opencert/opencert [4]

[18] AMASS Platform – Prototype P2 User Manual https://services.medini.eu/svn/AMASS_collab/WP-transversal/ImplementationTeam/PrototypeP2/AMASS_PrototypeP2_UserManual.docx (2018)[5]

[19] WEFACT http://www.ait.ac.at/en/research-fields/verification-validation/methods-and-tools/wefact/

[20] AMASS D1.1 Case studies description and business impact (30 November 2016)

[21] Richard Hawkins, Software Contribution Safety Argument Pattern (2009) http://www.goalstructuringnotation.info/archives/234

[22] McIsaac, B.: Ibm rational method composer: Standards mapping. Tech. rep., IBM Developer Works (2015)

[23] Eclipse Process Framework Project. https://eclipse.org/epf/

[24] Rational Method Composer. http://www-03.ibm.com/software/products/es/rmc

[25] Object Management Group: Software & systems process engineering meta-model specification. Tech. rep. (2008), http://www.omg.org/spec/SPEM/2.0/

[26] OpenCert project https://www.polarsys.org/opencert/

[27] EPF Composer Architecture Overview. https://www.eclipse.org/epf/composer_architecture/

[28] OSLC-KM for Knowledge Management http://trc-research.github.io/spec/km/

[29] Llorens, J., Morato, J., Genova, G., Fuentes, M., Quintana, V., & Díaz, I. (2004). RHSP: An information representation model based on relationship. *Studies in fuzziness and soft computing*, *159*, 221-253.

[30] AMASS D6.3 Design of the AMASS tools and methods for intra/cross domain reuse (b), 28 July 2018.

[4] The AMASS SVN code repository is open to AMASS partners with the same credentials as the SVN document repository. In case that people outside the project need access, please contact the AMASS Project Manager (alejandra.ruiz@tecnalia.com)

[5] The final version of the manual will be integrated in D2.5 AMASS User guidance and methodological framework (m31).

[31] AMASS D1.5 AMASS Demonstrators (b), 21 April 2018

[32] B. Gallina. A Model-driven Safety Certification Method for Process Compliance. 2nd IEEE International Workshop on Assurance Cases for Software-intensive Systems (ASSURE), joint event of ISSRE, Naples, Italy, doi: 10.1109/ISSREW.2014.30, pp. 204-209, November 3-6, 2014.

[33] Regorous Process Designer is available under an evaluation license http://www.regorous.com

[34] B. McIsaac, "IBM Rational Method Composer: Standards Mapping." 2015.

[35] M. A. Javed and B. Gallina, Safety-oriented process line engineering via seamless integration between EPF Composer and BVR Tool, 22nd International Conference on Systems and Software Product Line (SPLC '18), Gothenburg, Sweden, September 10-14, 2018.

[36] M. A. Javed and B. Gallina, Get EPF Composer back to the future: A trip from Galileo to Photon after 11 years, EclipseCon, Toulouse, France, June 13 - 14, 2018.

[37] AMASS D2.5 User Guidance and Methodological Framework, October 2018.

Appendix A: Document changes with respect to D6.5

New Sections:

Section	*Title*
2.2.4	Solution for Process Compliance Checking
3.1.2.10	Variability Management and Change Impact Analysis in Multiple Lines
2.2.7.1.1	Detecting fallacies in process models
2.2.7.1.2	Semi-automatic generation of process arguments representing plans
3.1.2.13	Reuse Discovery based on OSLC-KM
3.1.2.14	Reuse Discovery based on ElasticSearch
3.1.2.16	Semi-automatic generation of process-based arguments representing plans
3.3	Source Code Description for External Tools
Appendix A	Appendix A: Document changes with respect to D6.5

Modified Sections:

Section	*Title*	*Change*
2.2.2	Tailoring of Standards Models to Specific Projects	Completed with implementation in P2
3.1.2.4	Compliance Mappings in EPF	Added transformation of standard's requirements modelled in EPF composer to baseline models
3.1.2.5	Cross-systems reuse (intra standard or intra domain product upgrade)	Completed with the integration of the Reuse Discovery functionalities
3.1.2.7	Variability management support at process level	
3.1.2.8	Specification of variability at component level	
3.1.2.9	Variability management support at assurance case level	
2.2.8	Reuse Discovery	Implemented functionalities for P2 (ElasticSearch and OSLC-KM)
3.2	Source Code Description for the AMASS Tool Platform	Revision of plugins
4	Conclusions	Included requirements completeness table
	Abbreviations and Definitions	Completed list
	References	Completed list