

ECSEL Research and Innovation actions (RIA)



AMASS

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

**Prototype for cross/intra-domain reuse (b)
D6.5**

Work Package:	WP6: Cross-Domain and Intra-Domain Reuse
Dissemination level:	PU = Public
Status:	Final
Date:	29 December 2017
Responsible partner:	Borja López (TRC)
Contact information:	borja.lopez@reusecompany.com
Document reference:	AMASS_D6.5_WP6_TRC_V1.0

PROPRIETARY RIGHTS STATEMENT

This document contains information that is proprietary to the AMASS consortium. Permission to reproduce any content for non-commercial purposes is granted, provided that this document and the AMASS project are credited as source.

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	Maelardalens Hoegskola (MDH)
Stefano Puri	Intecs (INT)
Jose María Álvarez, Roy Mendieta, Miguel Rozalen	Universidad Carlos III de Madrid (UC3)

Reviewers

Names	Organisation
Helmut Martin (Peer-reviewer)	Virtual Vehicle (VIF)
Morayo Adedjouma (Peer-reviewer)	Commissariat a L'énergie Atomique et aux Energies Alternatives (CEA)
Cristina Martinez (Quality Manager)	Tecnalia Research & Innovation (TEC)
Jose Luis de la Vara (TC review)	Universidad Carlos III de Madrid (UC3)



TABLE OF CONTENTS

Executive Summary.....	6
1. Introduction.....	7
2. Implemented Functionality	9
2.1 Scope.....	9
2.2 Implemented Requirements	10
2.2.1 Modelling of standards	12
2.2.2 Tailoring of Standards models to specific projects	15
2.2.3 Process Compliance (informal) management	17
2.2.4 Compliance Monitoring.....	23
2.2.5 Reuse Assistance.....	23
2.2.6 Product/Process/Assurance Case Line Specification	29
2.2.7 Reuse Discovery	30
2.3 Installation and User Manuals.....	32
3. Implementation Description.....	33
3.1 Implemented Modules.....	33
3.1.1 Compliance Management	33
3.1.2 Management of families/lines.....	33
3.1.3 Cross/Intra Domain Reuse.....	34
3.2 Source Code Description	35
4. Conclusions.....	40
Abbreviations and Definitions.....	41
References	43



List of Figures

Figure 1.	AMASS Building blocks	7
Figure 2.	Functional decomposition for the AMASS platform	9
Figure 3.	Standard and Process Modelling.....	12
Figure 4.	Reference Framework Editor	13
Figure 5.	ISO-26262: Recommendation Table associated to a Requirement	13
Figure 6.	Authoring perspective of the EPF composer	14
Figure 7.	Standard modelled in the EPF composer	15
Figure 8.	Recommendation table modelled in the EPF composer	15
Figure 9.	Tailoring of Baseline Models from Standard Models	16
Figure 10.	Baseline tailoring.....	17
Figure 11.	Baseline graphical editor	17
Figure 12.	Assurance Project and System Component Specification structure	18
Figure 13.	How to create Compliance Maps	19
Figure 14.	Compliance Set view	19
Figure 15.	Mapping Table view	21
Figure 16.	Import View in OpenCert for EPF: Result of the import operation	21
Figure 17.	Modelling of compliance mappings in the EPF composer	22
Figure 18.	Mapped Requirements in the EPF composer	22
Figure 19.	Compliance report in web client	23
Figure 20.	Cross-project Reuse View	24
Figure 21.	Using the Reuse View	25
Figure 22.	How to create Equivalence Map.	26
Figure 23.	Equivalence Map form.....	27
Figure 24.	Cross Domain button	27
Figure 25.	Cross domain window	28
Figure 26.	SysML content transformed into RSHP metamodel.....	31
Figure 27.	SysML model content as a query, to search for similar artefacts	31
Figure 28.	Reuse Discovery operations integrated in AMASS.....	32
Figure 29.	Tool components for Compliance Management	33
Figure 30.	Tool components for Automatic generation of process-based arguments.....	34
Figure 31.	Tool components for Automatic generation of product-based arguments	34
Figure 32.	Tool components for Cross/Intra Domain Reuse	35
Figure 33.	Cross-Domain and Intra-Domain Reuse plugins	38



List of Tables

Table 1. Requirements implemented in the first prototype of the AMASS platform (Core Prototype)10

Table 2. Requirements implemented in the second prototype of the AMASS platform (Prototype P1) ...10



Executive Summary

The deliverable D6.5 Prototype for cross/intra-domain reuse (b), is the second 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. D6.5 is the evolution of D6.4, which described the first prototype.

This deliverable reports the status of the aforementioned tooling framework for the AMASS platform second prototype release (P1) 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. Finally, the benchmarking of the prototype implementation shall guide further refinement of the conceptual approach.

The WP6-related part of the second prototype iteration (P1) 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 P1 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.

D6.5 relates to other AMASS outcomes referred in this deliverable:

- Installable AMASS platform tools for P1.
- 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 OPENCOS [1] and SafeCer [2] towards an architecture-driven, multi-concern assurance, reuse-oriented, and seamlessly interoperable tool platform.

The expected tangible AMASS results are:

- The **AMASS Reference Tool Architecture**, which will extend the OPENCOS 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 specifications [12]).
- 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.
- The **Open AMASS Community**, which will manage the project outcomes, for maintenance, evolution and industrialization. The Open Community will be supported by a governance board, and by rules, policies, and quality models. This includes support for AMASS base tools (tool infrastructure for database and access management, among others) and extension tools enriching AMASS platform functionalities. As Eclipse Foundation is part of the AMASS consortium, the Polarsys/Eclipse community (www.polarsys.org) is going to host 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.

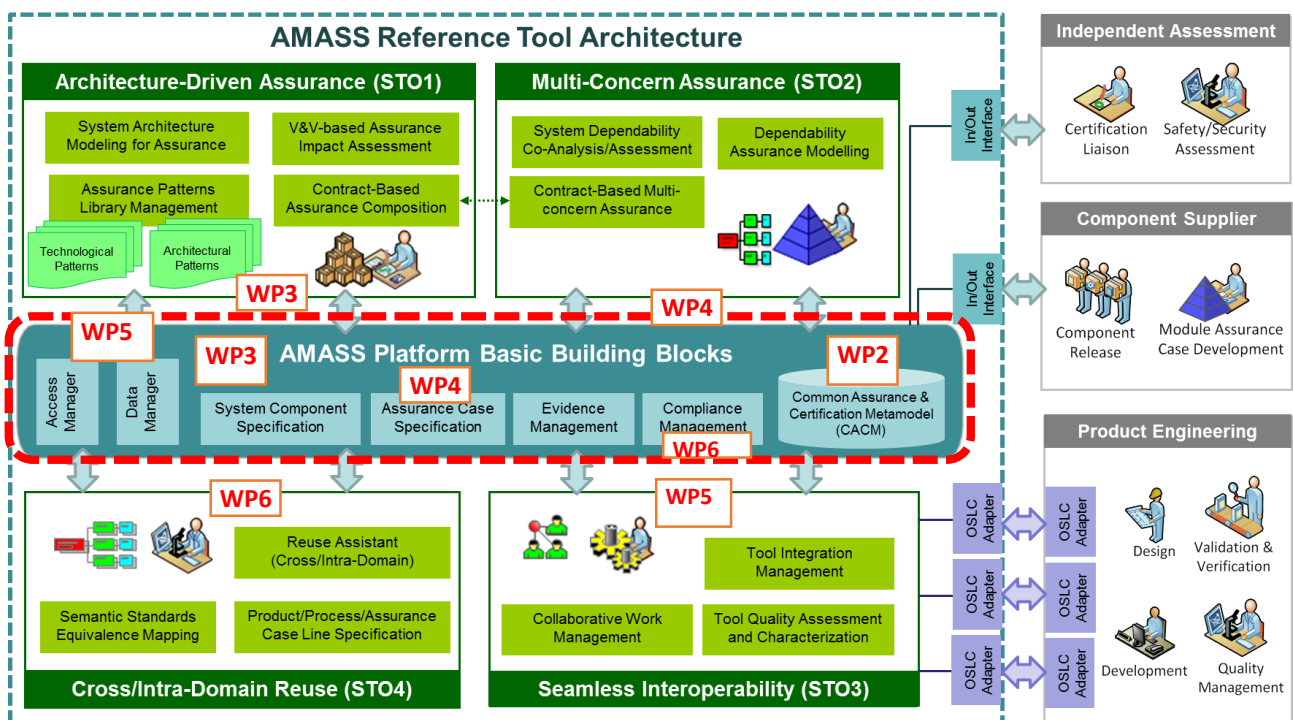


Figure 1. AMASS Building blocks



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), will be aligned, merged and consolidated at Technology Readiness Levels TRL4¹. Concerning this first prototype, the basic building block assigned to WP6 is Compliance Management.
2. During the **second prototyping** iteration (Prototype P1), the AMASS-specific Building Blocks will be developed and benchmarked at TRL4; this comprises the blue basic building blocks as well as the green building blocks in Figure 1. Regarding WP6, in this second prototype the specific building blocks include 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 shall be integrated in a comprehensive toolset operating at TRL5. WP6 functionalities developed during the second prototype iteration shall be 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 application of the building blocks in case studies for this first prototype are described in D1.1 [20].

As part of the Prototype P1, 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**”, “**Semantics Standards Equivalence Mapping**” and “**Product/Process/Assurance Case Line Specification**” (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 D6.5 are:

- Installable AMASS Platform tools for the Prototype P1.
- User manuals and Installation instructions. [18]
- Source code. [17]

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 D3.5 [14]). WP6 is also in charge of developing the Assurance Project Lifecycle Management functionality to provide a common frame and repository for any assurance asset created, managed or removed in the lifecycle of a specific assurance project.

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

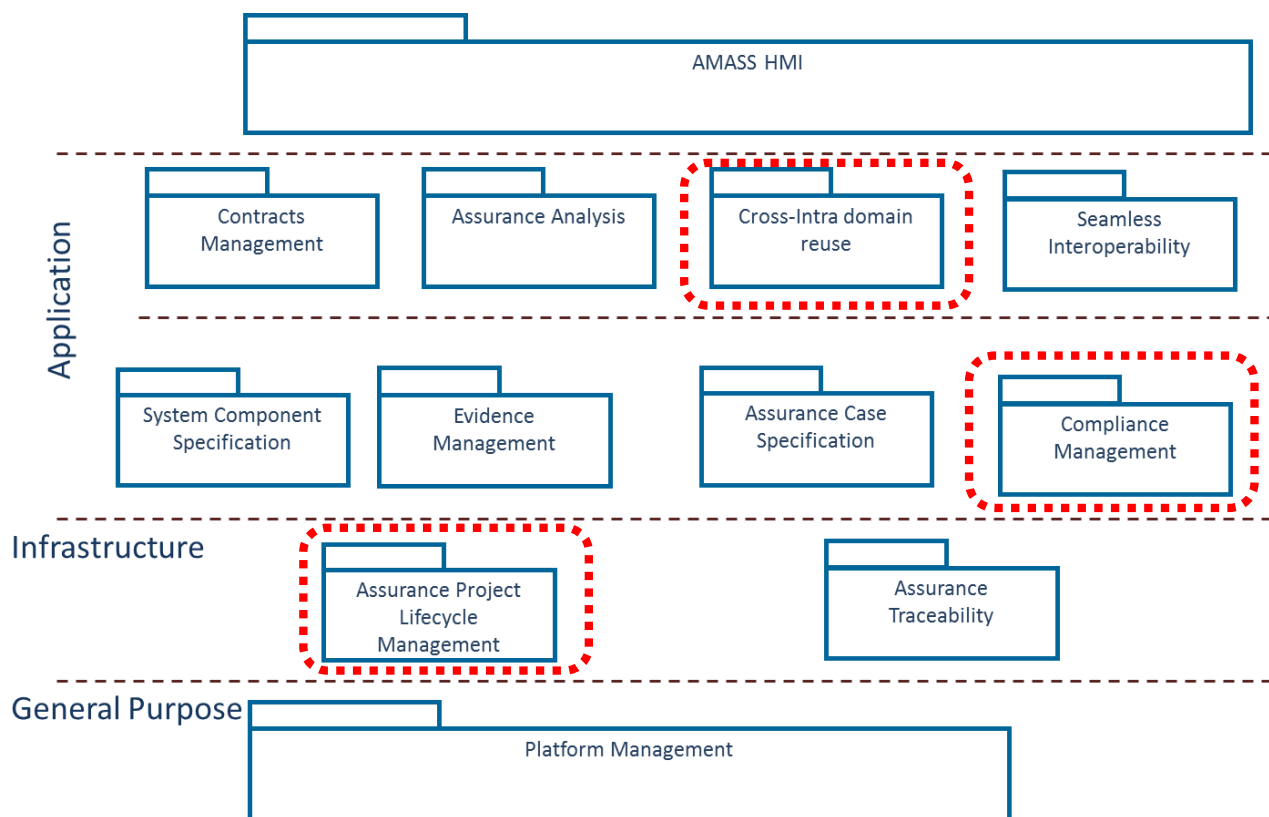


Figure 2. Functional decomposition for the AMASS platform

The Compliance Management building block has been improved in this second prototype to allow monitoring the compliance progress using filtering by criticality levels (e.g., SIL, ASIL, DAL). 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 project².
- Tools from the EPF (Eclipse Process Framework) project³.

² Further information about the OPENCROSS toolset can be found at www.opencross-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 P1.

2.2 Implemented Requirements

From the requirements point of view, D6.5 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 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	<p>The AMASS tools shall enable users to visualize process compliance. This means showing the links between the requirements and the applicant's evidence (during the planning as well as execution phase).</p> <p>This visualization could be done via compliance maps (matrix) or via arguments aimed at justifying the satisfaction of the requirements coming from the standards.</p>

In addition to that, this phase focuses on a second set of AMASS requirements as defined in deliverable D2.1 [13]. The requirements listed in Table 2 have been implemented in the second iteration of the WP6-related part of the AMASS platform. The column "Related requirement" refers to the list of items gathered in deliverable D2.1.

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 characterizes the different projects should be recognized 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)).



Function name	Related Requirement	Description
		The commonality that characterizes the different projects should be recognized and proposed as a reusable process structure.
Intra-Domain, Cross versions, Reuse Assistance	WP6_RA_003	<p>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)).</p> <p>The commonality that characterizes the different projects should be recognized and proposed as reusable process structure.</p>
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 characterizes the different projects should be recognized 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 system shall enable users to specify what varies (and what remains unchanged) from one process and its family members.
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 family members (e.g., its evolved versions 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 assurance case and its family members.
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.

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 for requirement **WP6_CM_001**, described in the table above.

This feature is supported by both the OpenCert [26] and EPF toolsets [27]. Figure 3 shows examples of models for these two toolsets. The dashed circle indicates that it has not been fully integrated yet and the diamond form denotes a model that maps information from other models.

- OpenCert uses Reference Framework Editor (as part of 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). Please note that EPF can be also used to model Standards, but this has not been fully integrated with the OpenCert tools.

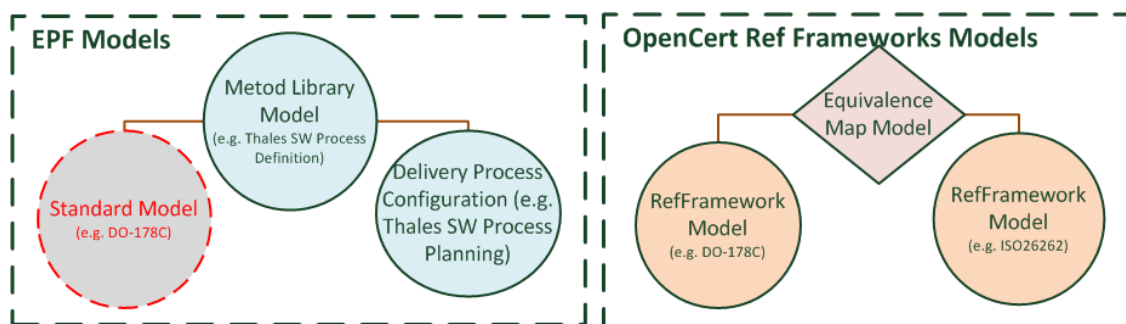


Figure 3. Standard and Process Modelling

2.2.1.1 Reference Framework Editor

The Reference Framework Editor is composed of five views (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.

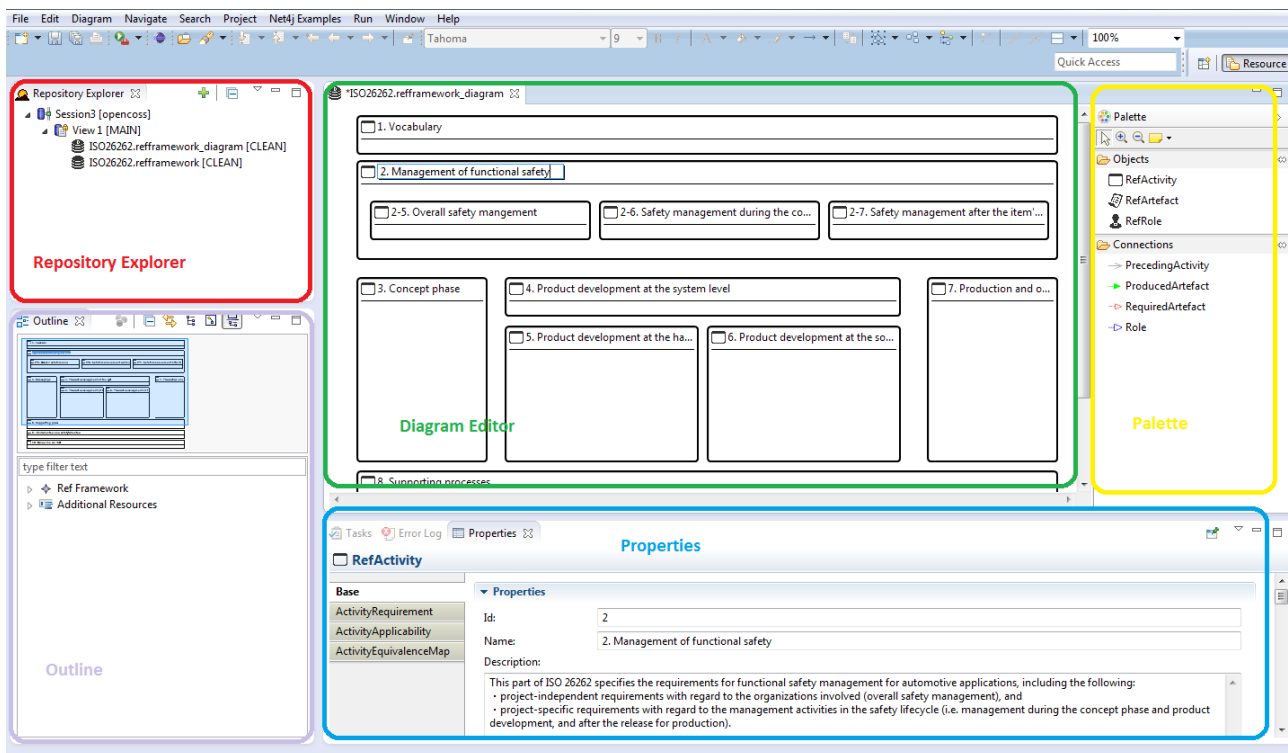


Figure 4. Reference Framework Editor

In addition, the recommendation or applicability tables 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.

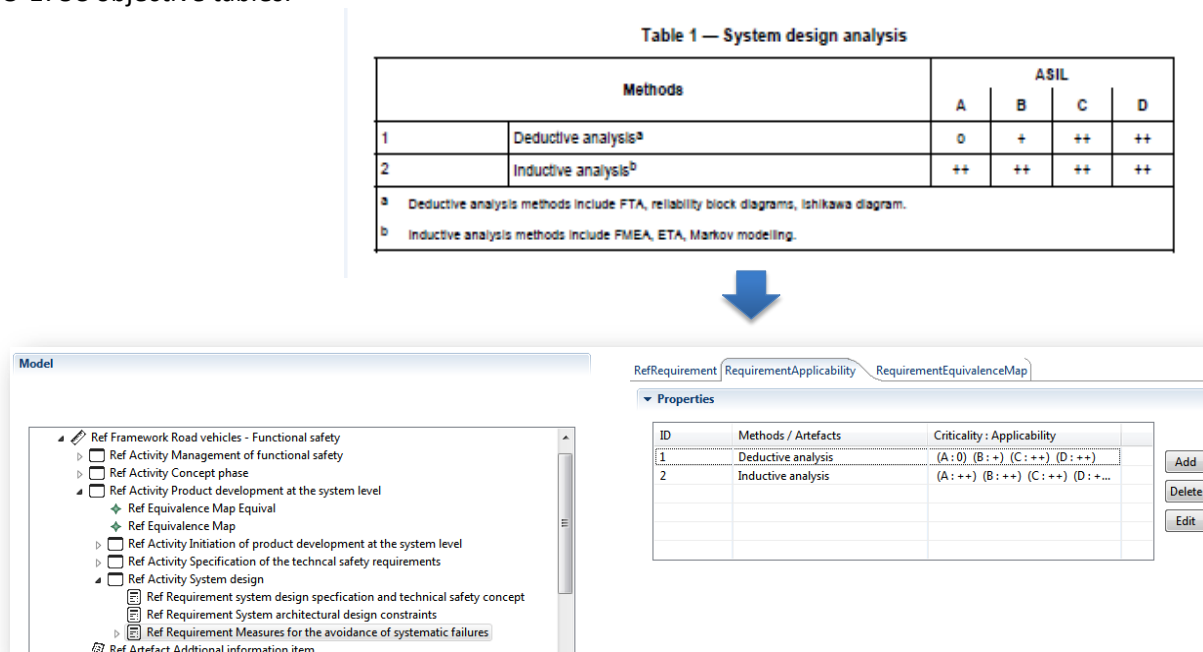


Figure 5. ISO-26262: Recommendation Table associated to a Requirement

2.2.1.2 Standards modelling with EPF

The EPF composer is a tool for the modelling of engineering process based on the SPEM 2.0 OMG Standard [25]. The functionality of the EPF composer is organized 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.

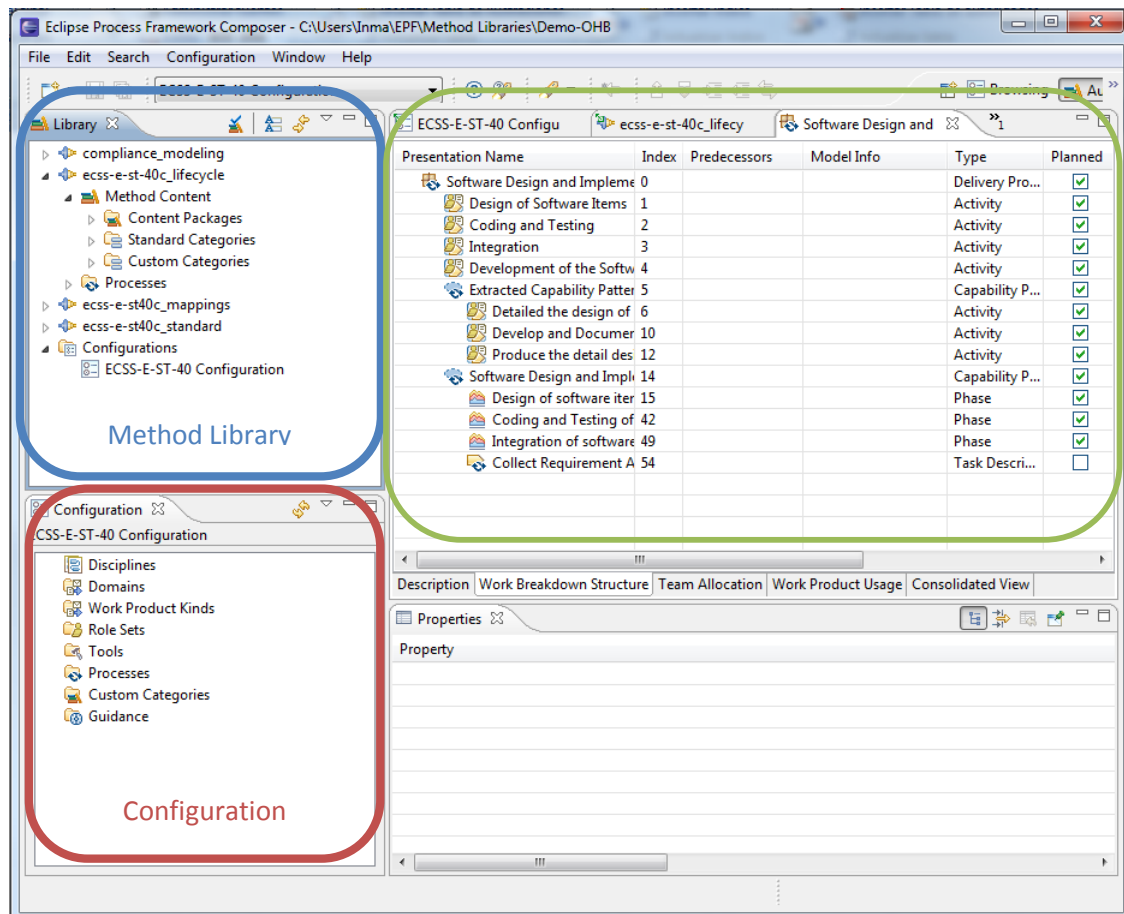


Figure 6. Authoring perspective of the EPF composer

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. customized 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 modelling of a recommendation table from RTCA DO-178C.

Table A-1 Software Planning Process

	Objective	Ref	Activity	Applicability by Software Level				Output	Ref	Control Category by Software Level			
				A	B	C	D			A	B	C	D
1	The activities of the software life cycle processes are defined.	4.1.a	4.2.a 4.2.c 4.2.d 4.2.e 4.2.g 4.2.i 4.2.j 4.3.c	○	○	○	○	PSAC SDP SVP SCM Plan SQA Plan	11.1 11.2 11.3 11.4 11.5	① ① ① ① ①	① ① ② ② ②	② ② ② ② ②	
2	The software life cycle(s), including the inter-relationships between the processes, their sequencing, feedback mechanisms, and transition criteria, is defined.	4.1.b	4.2i 4.3.b	○	○	○		PSAC SDP SVP SCM Plan SQA Plan	11.1 11.2 11.3 11.4 11.5	① ① ① ① ①	① ① ② ② ②		
3	Software life cycle environment is selected and defined.	4.1.c	4.4.1 4.4.2.a 4.4.2.b 4.4.2.c 4.4.3	○	○	○		PSAC SDP SVP SCM Plan SQA Plan	11.1 11.2 11.3 11.4 11.5	① ① ① ① ①	① ① ② ② ②		
4	Additional considerations are addressed.	4.1.d	4.2.f 4.2.h 4.2.i 4.2.j 4.2.k	○	○	○	○	PSAC SDP SVP SCM Plan SQA Plan	11.1 11.2 11.3 11.4 11.5	① ① ① ① ①	① ① ② ② ②	① ② ② ② ②	
5	Software development standards are defined.	4.1.e	4.2.b 4.2.g 4.5	○	○	○	○	SW Requirements Standards SW Design Standards SW Code Standards	11.6 11.7 11.8	① ① ①	① ① ②		
6	Software plans comply with this document.	4.1.f	4.3.a 4.6	○	○	○	○	Software Verification Results	11.14	②	②	②	
7	Development and revision of software plans are coordinated.	4.1.g	4.2.g 4.6	○	○	○	○	Software Verification Results	11.14	②	②	②	

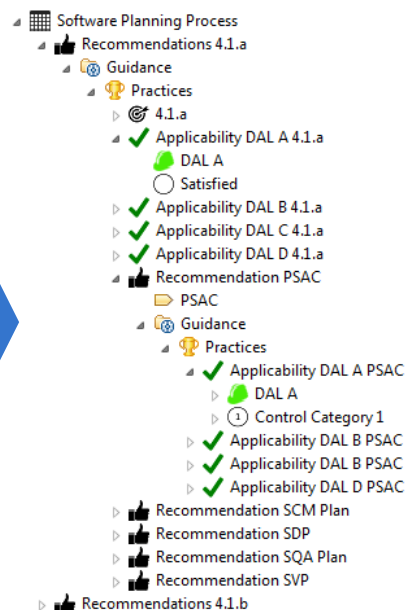


Figure 8. Recommendation table modelled in the EPF composer

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 the table above.

The information managed by 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 Reference Framework models. It is possible to import Baseline models based on the Standard processes modelled in EPF.

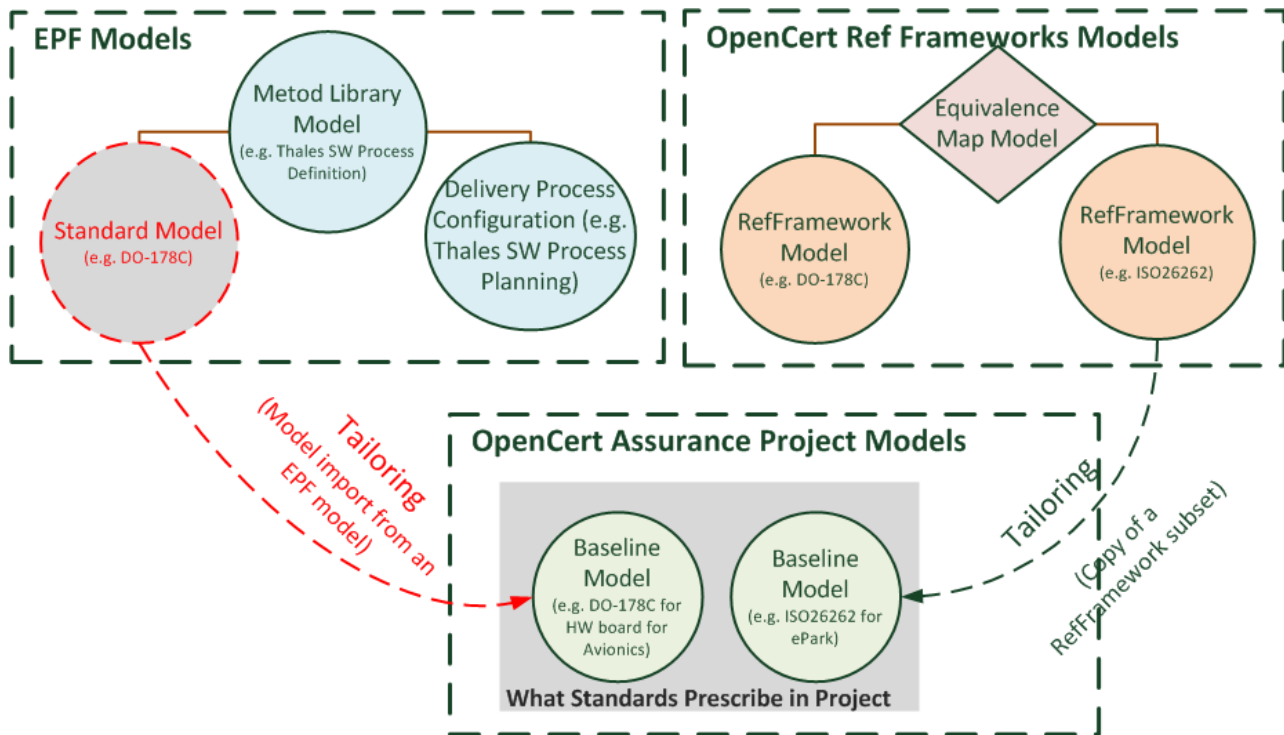


Figure 9. Tailoring of Baseline Models from Standard Models

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).

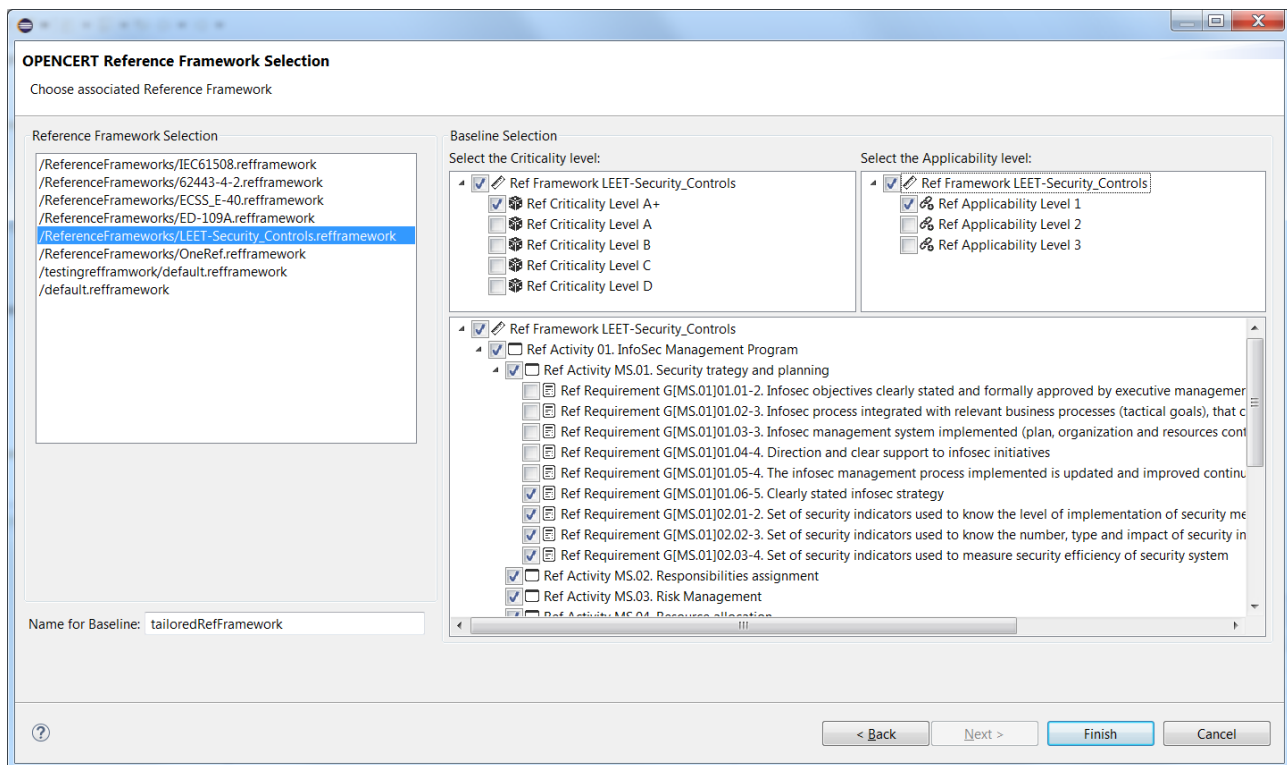


Figure 10. Baseline tailoring

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.

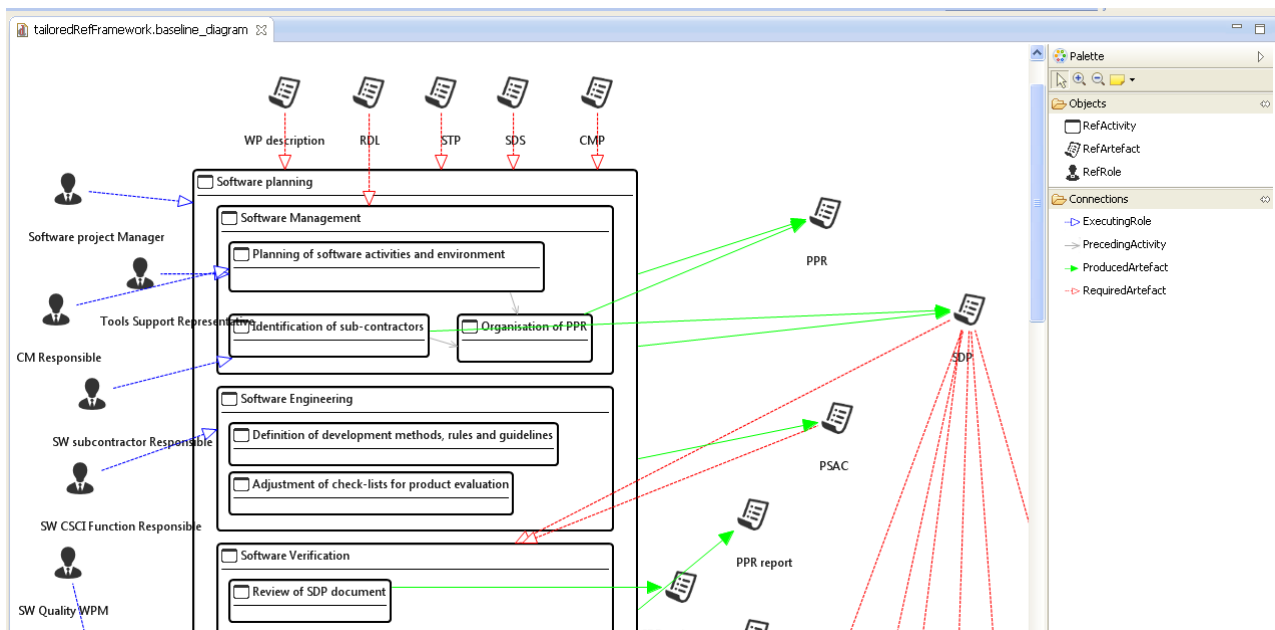


Figure 11. Baseline graphical editor

2.2.3 Process Compliance (informal) management

This chapter describes the implementation for requirement **WP6_CM_008**, described in the table above.

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 has not been implemented yet (implementation plan for this point will be revised for P2). It will support 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 CHES toolset (see deliverable D3.4 [15]). The link of Assurance Assets Package with System models has not been implemented yet (implementation plan for this point will be revised for P2 due to CDO technical problems). 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.

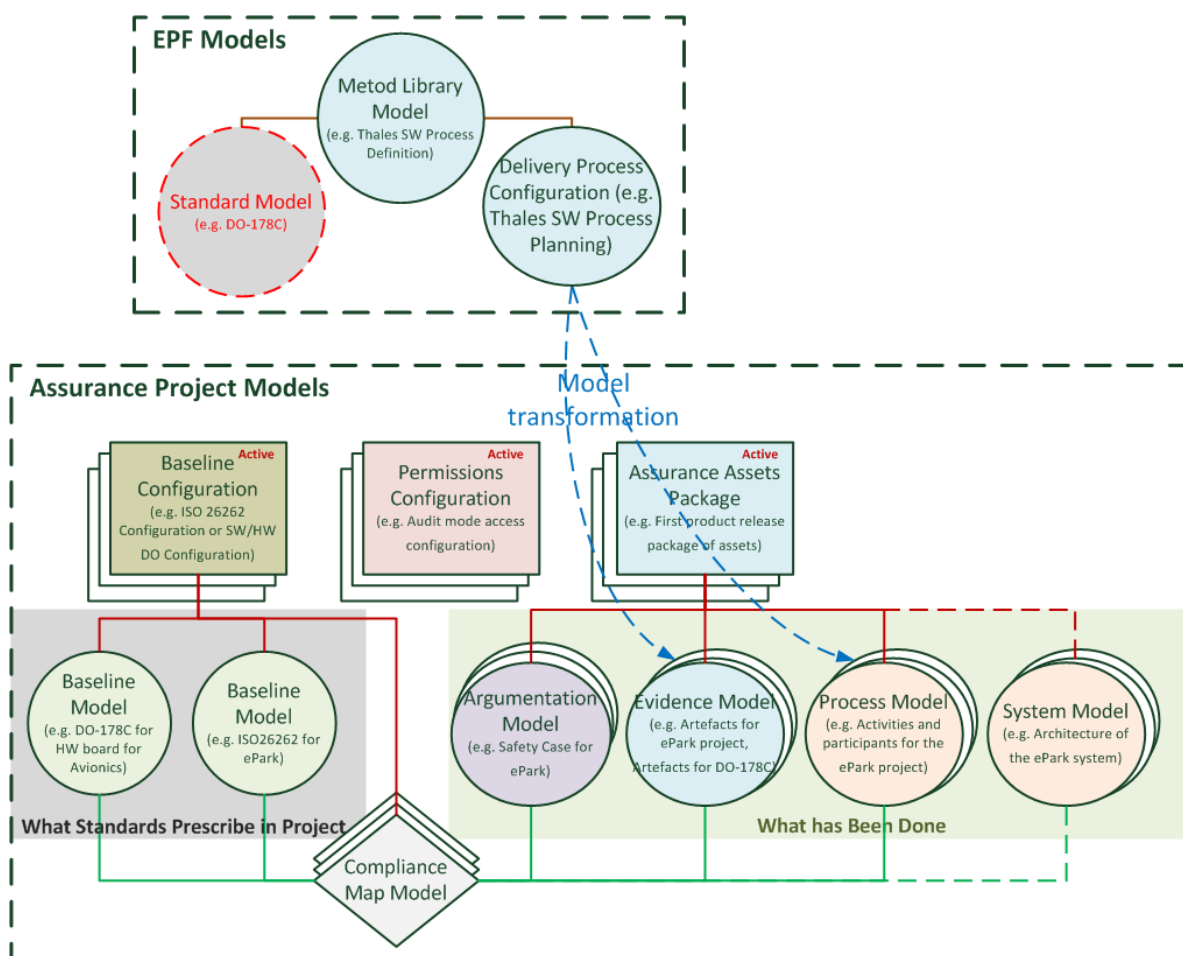


Figure 12. Assurance Project and System Component Specification structure



2.2.3.1 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.

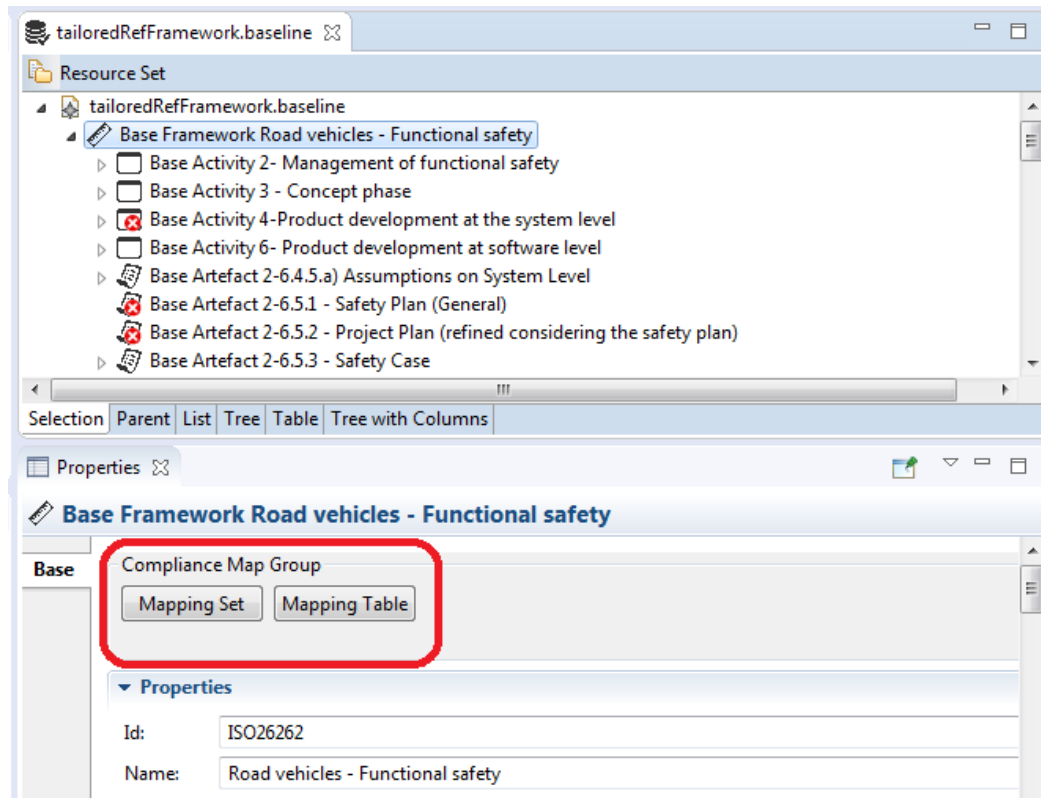


Figure 13. How to create Compliance Maps

Figure 14 shows the Compliance Set view.

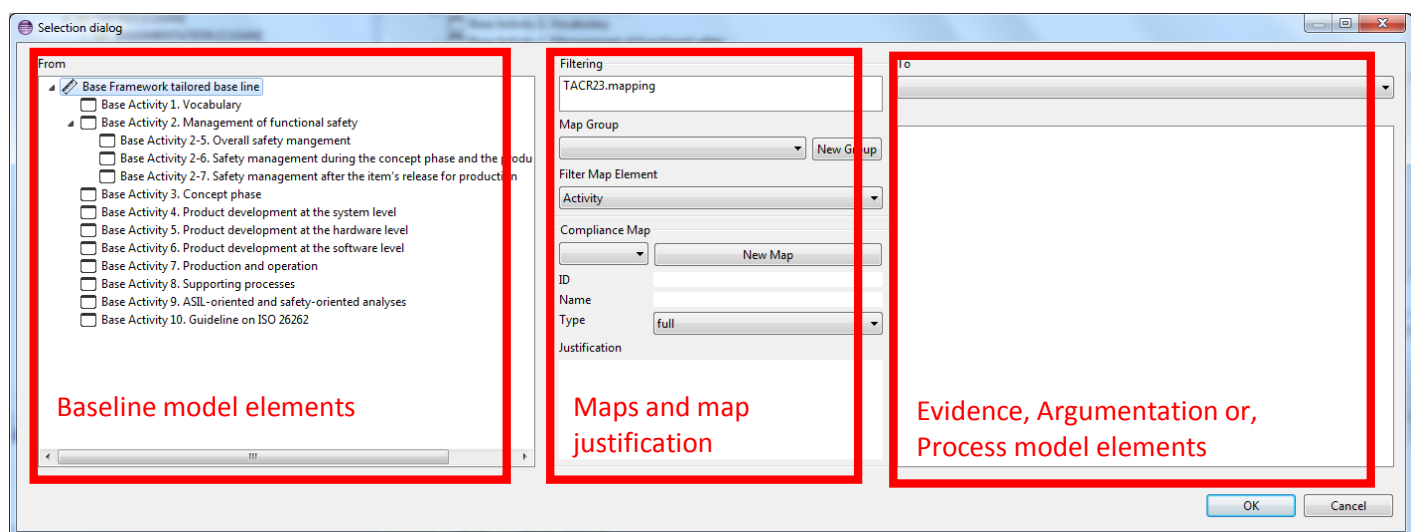


Figure 14. Compliance Set view

The Compliance Set view form is organized 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 organized 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.

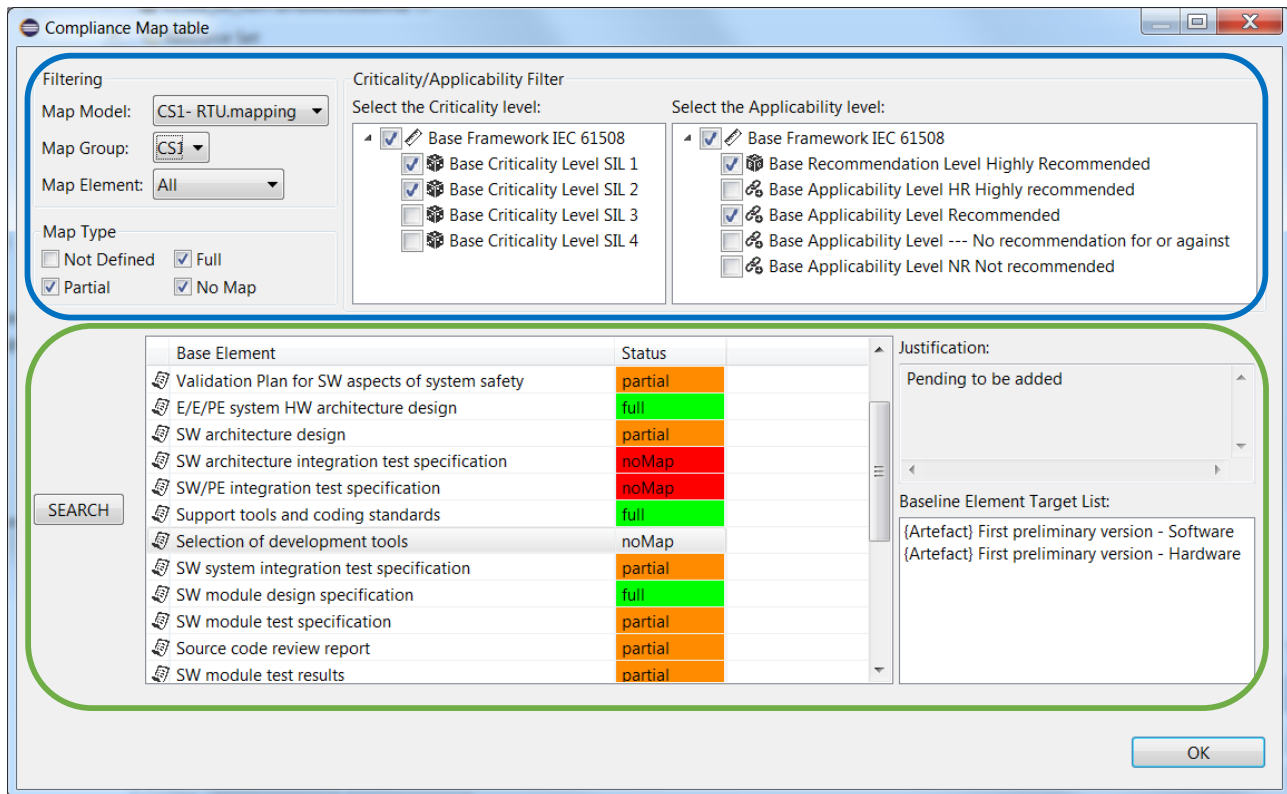


Figure 15. Mapping Table view

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.

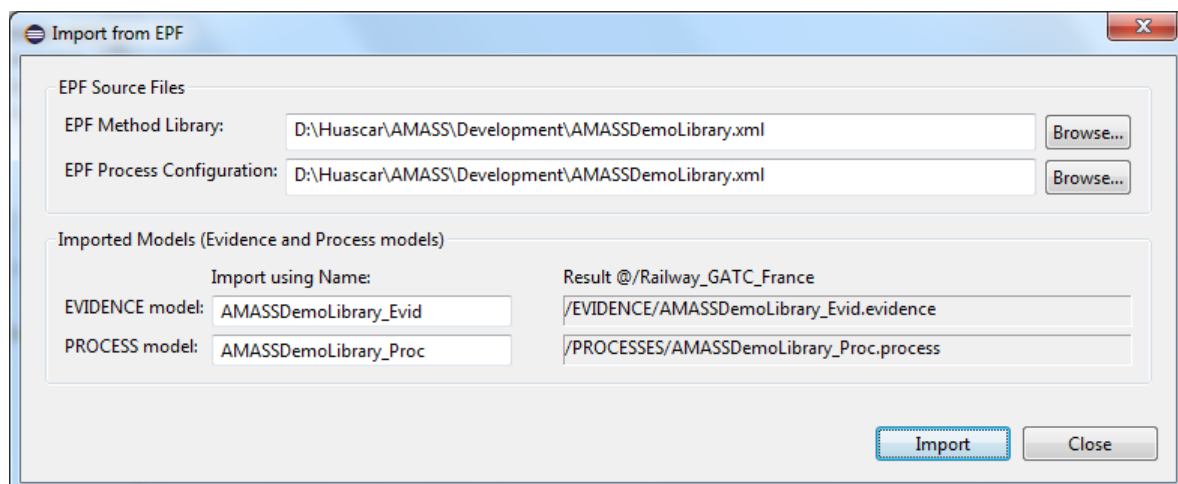


Figure 16. Import View in OpenCert for EPF: Result of the import operation

2.2.3.2 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. Currently, this compliance information cannot be imported in OpenCert but this functionality is planned as a future work.

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.

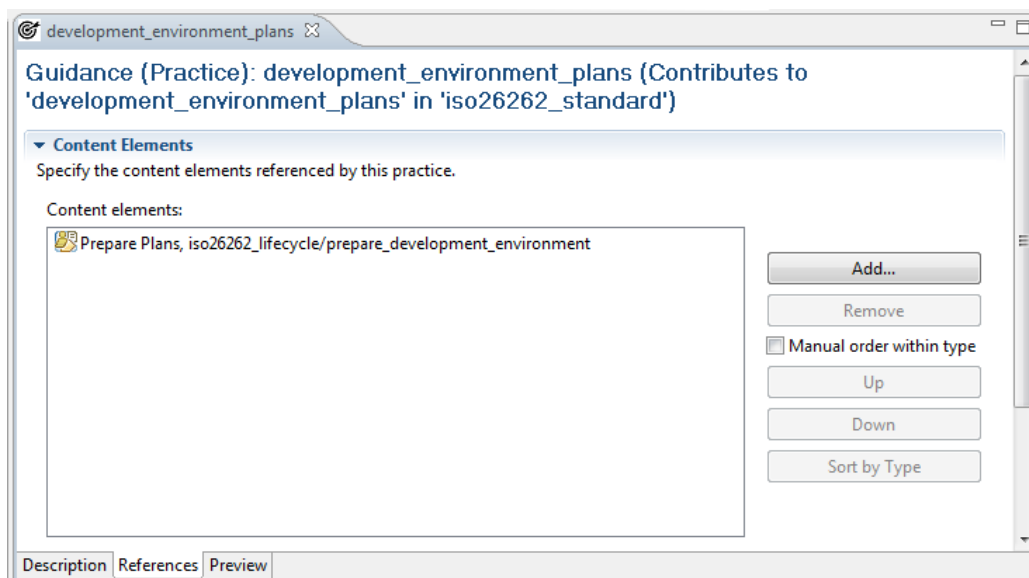


Figure 17. Modelling of compliance mappings in the EPF composer

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.

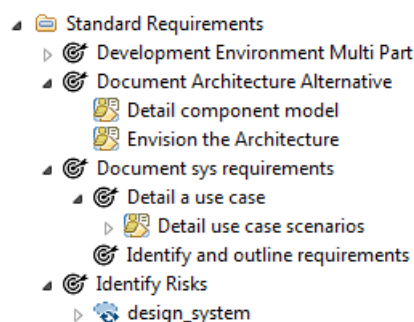


Figure 18. Mapped Requirements in the EPF composer

2.2.4 Compliance Monitoring

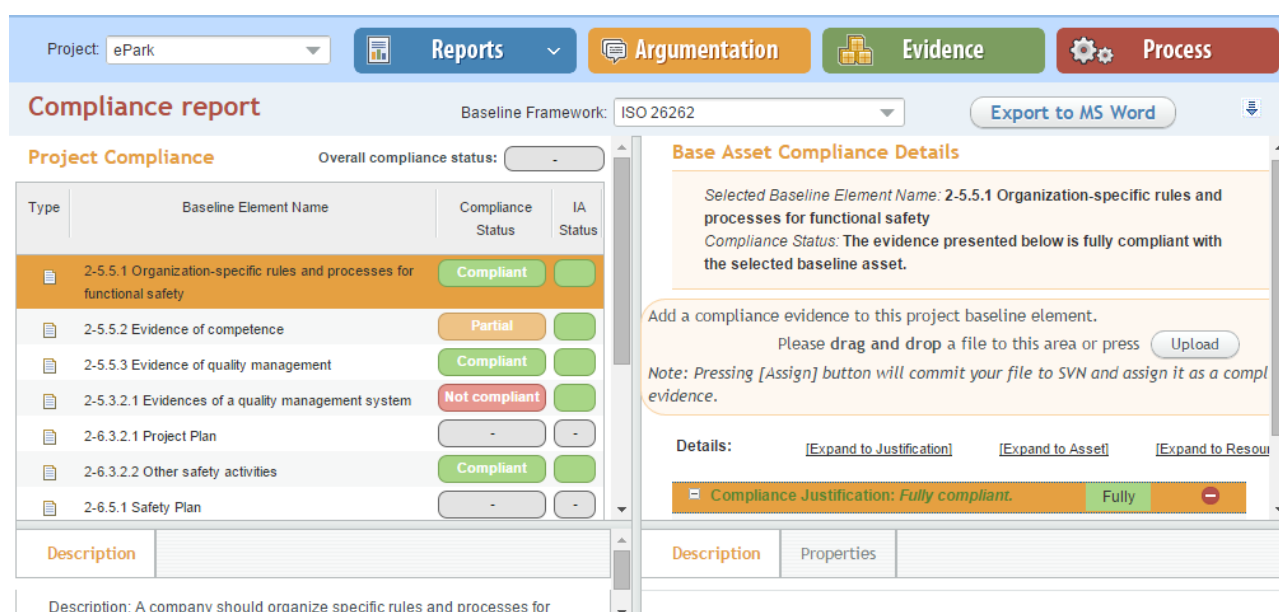
This chapter describes the implementation for requirement **WP6_CM_005**, described in the table above.

Compliance report provides extensive functionality that helps 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 19 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.



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	-	-

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 [Upload](#)

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

Details: [\[Expand to Justification\]](#) [\[Expand to Asset\]](#) [\[Expand to Resource\]](#)

Compliance Justification: Fully compliant. Fully [-](#)

Description: A company should organize specific rules and processes for

Figure 19. Compliance report in web client

2.2.5 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 above.

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.

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

This functionality implies reuse of assurance assets when a product or system evolve 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. Future versions of the Reuse View will allow users to filter a specific subset of model elements by different criteria such as those featured by the Reuse Discovery functionalities (see Section 2.2.7). We will also implement impact analysis in order to understand the consequences of the reuse operation and to identify the set of assurance assets that may be affected when executing the reuse operation. Impact analysis will use in turn a traceability module (based on the CAPRA tool and documented in AMASS deliverable D5.2 [16]).

Figure 20 and Figure 21 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.

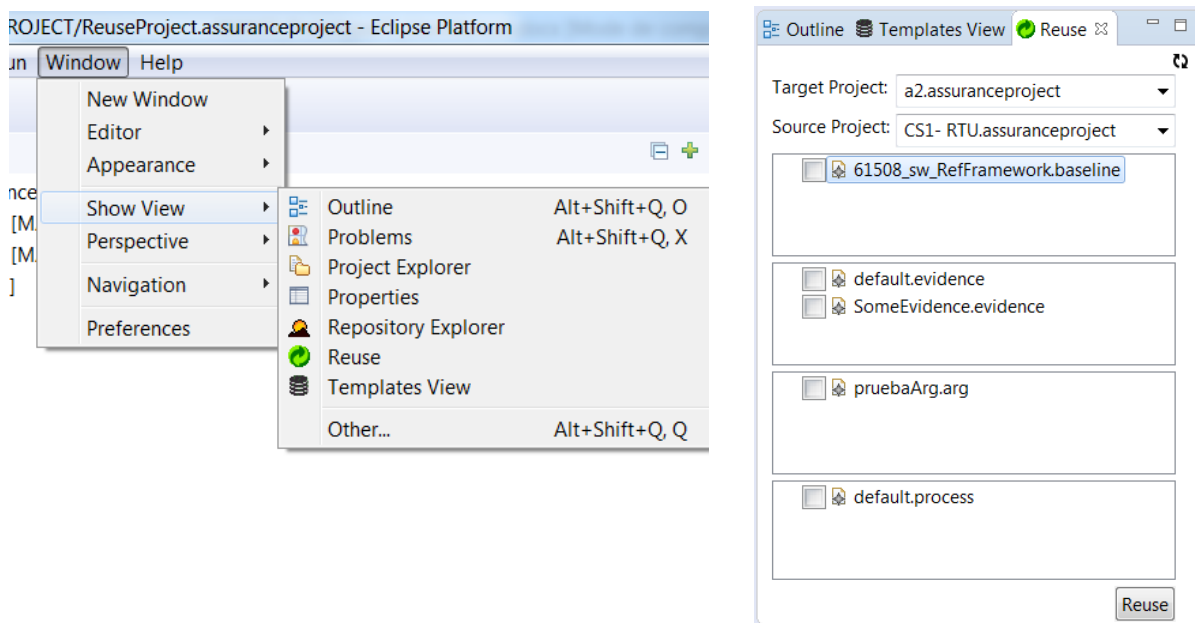


Figure 20. Cross-project Reuse View

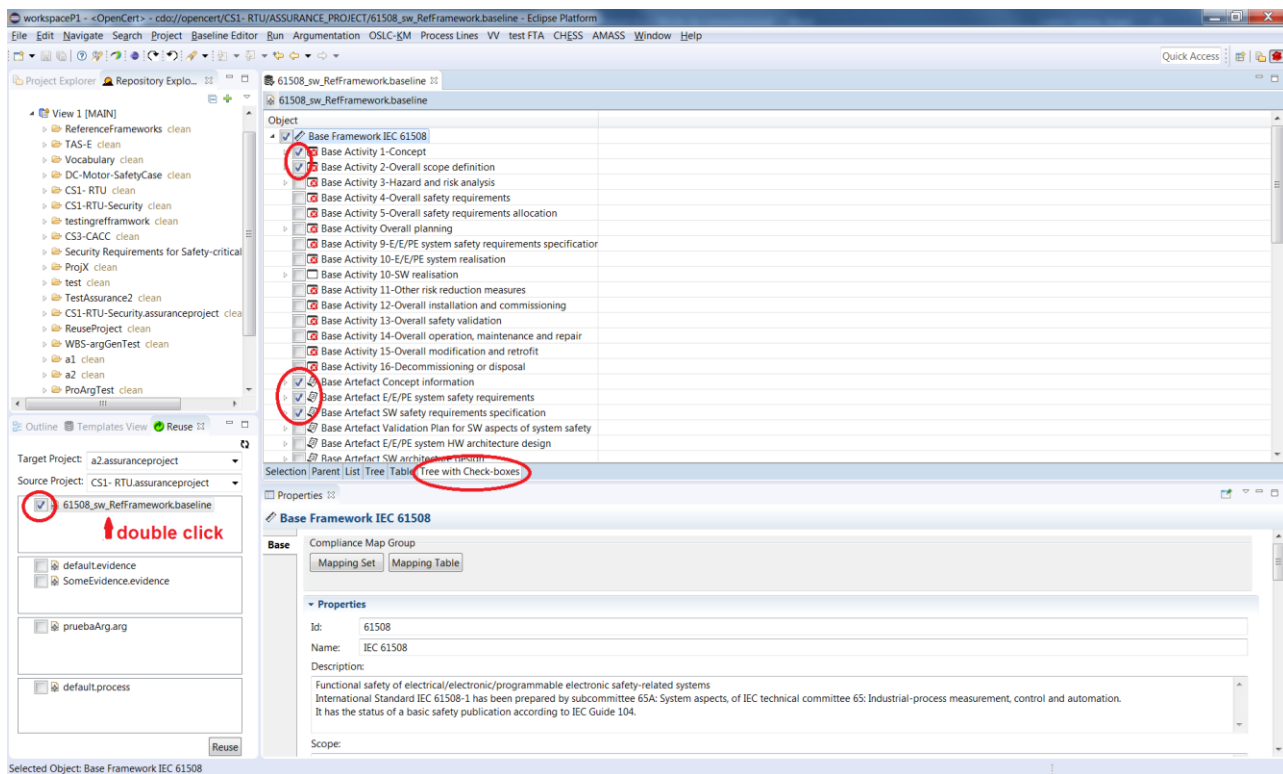


Figure 21. Using the Reuse View

2.2.5.2 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 button called “Mapping Set” on the properties form of the reference framework using the tree view editor (see Figure 22).

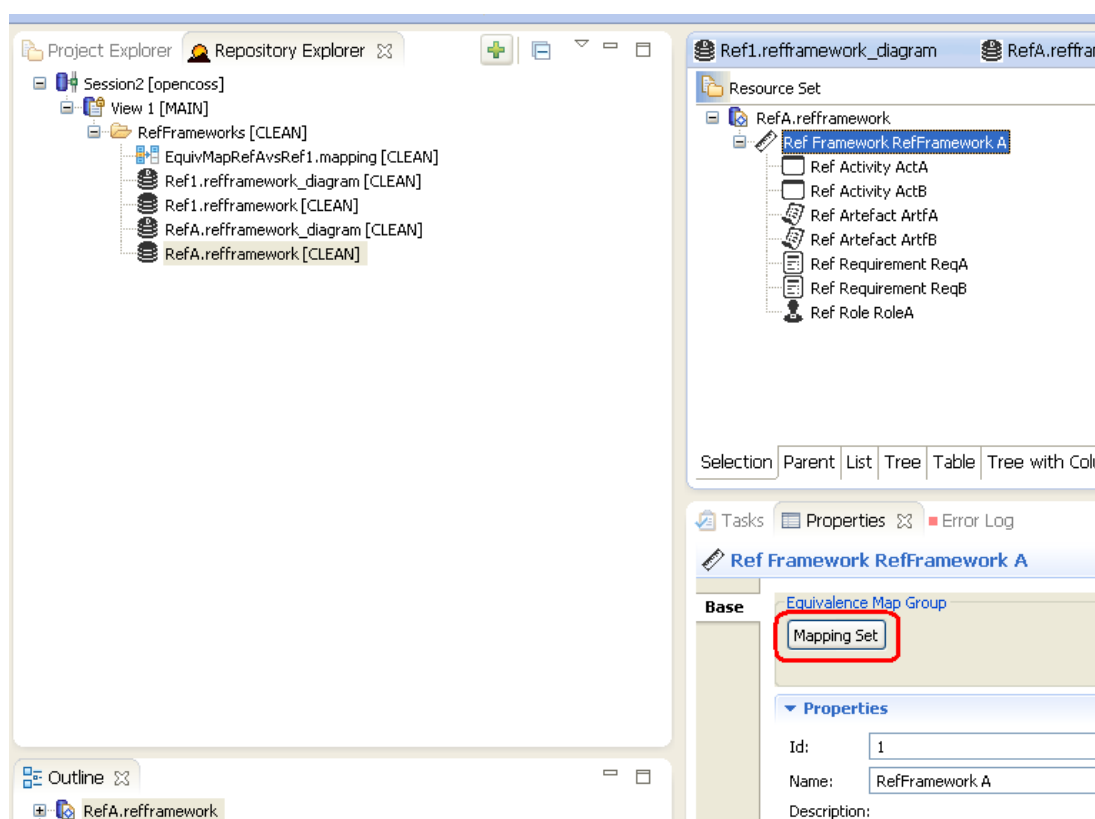


Figure 22. How to create Equivalence Map.

Figure 23 shows the form for Equivalence Map. The Equivalence Map form is organized 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 in the 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.

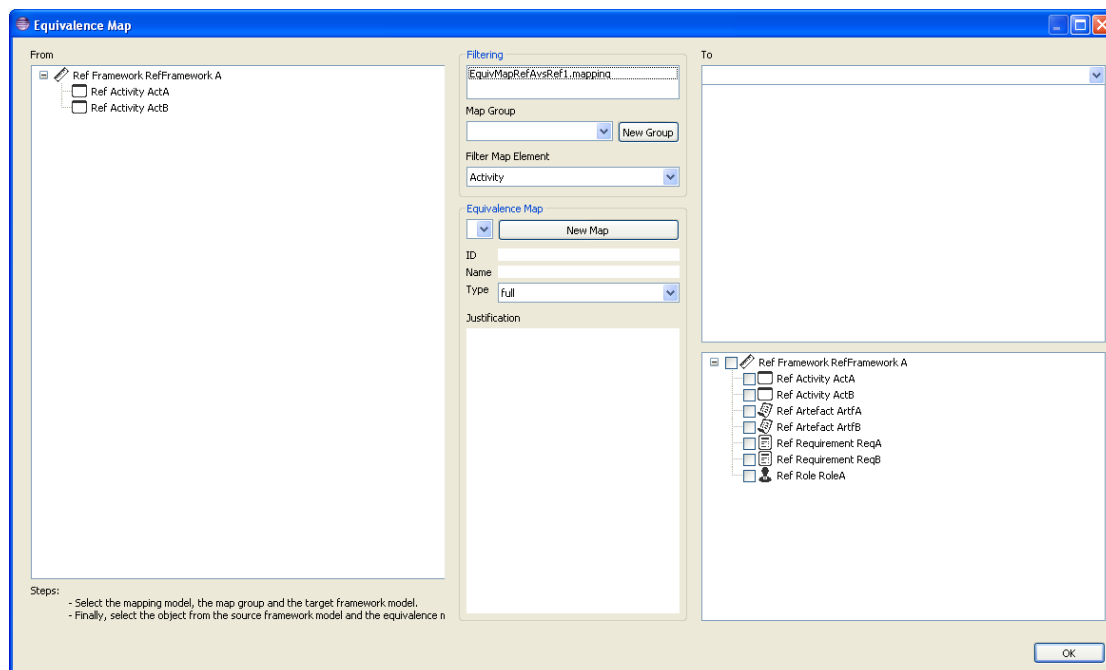


Figure 23. Equivalence Map form.

For the reuse assistant, it is mandatory that the target assurance project is based in a refframework with equivalence maps associated to the refframework in which the source assurance project is based. Figure 24 shows how to open the reuse assistant for cross-domain reuse. We focus on evidence reuse for this version of AMASS prototype.

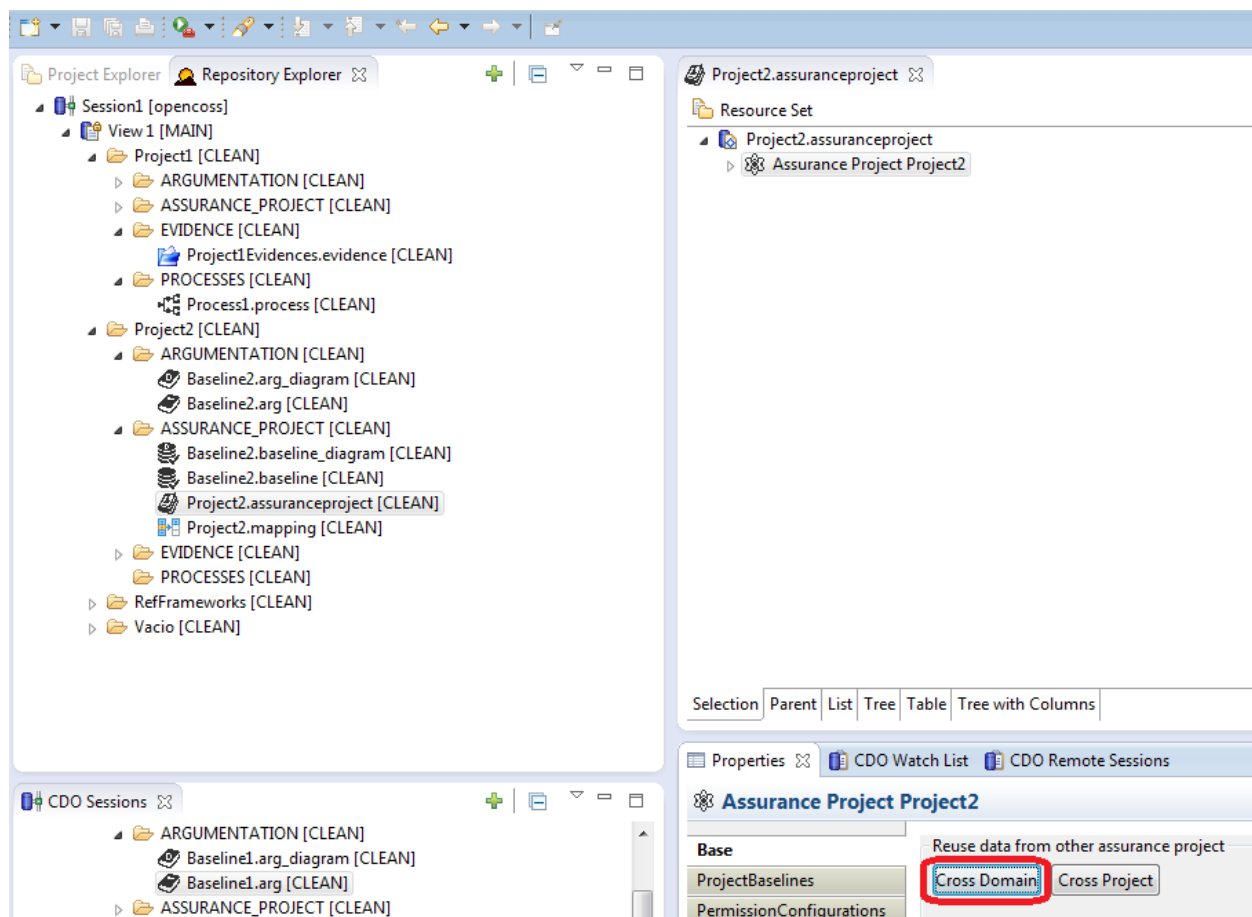


Figure 24. Cross Domain button



Figure 25 shows the Cross-Domain assistant form. It is organized 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 post-conditions in a list (to see the ID, Name and description one post-condition 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.

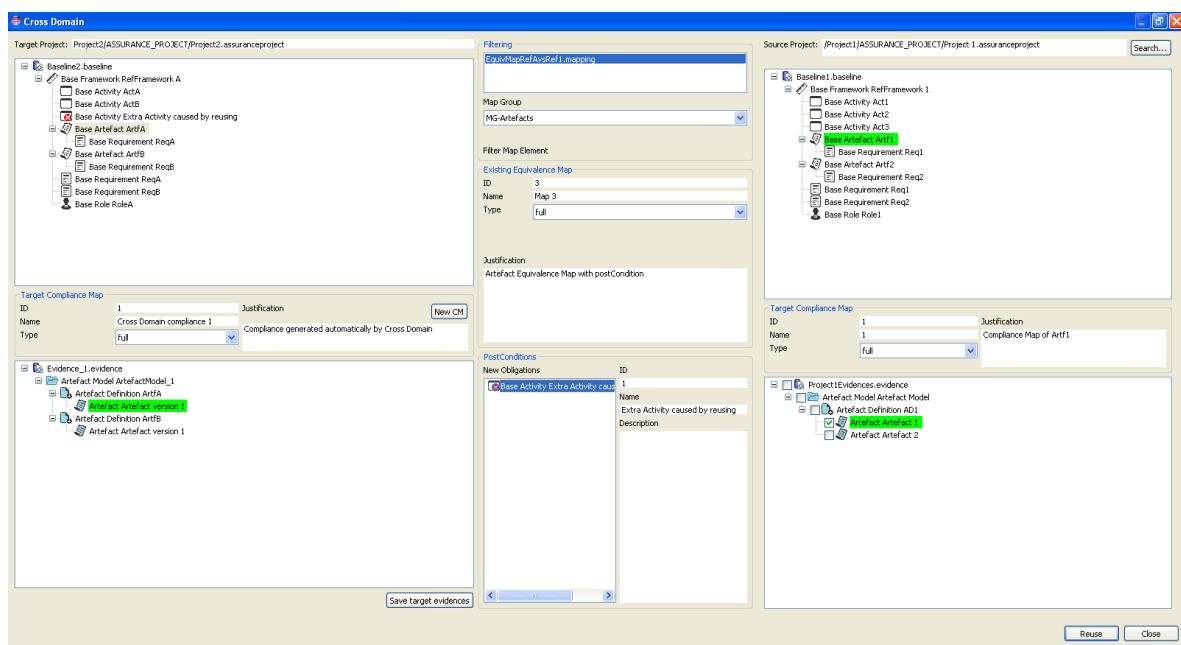


Figure 25. Cross domain window

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 post-conditions will be selected in the target baseline model.



2.2.6 Product/Process/Assurance Case Line Specification

To manage families/lines, it is necessary to have at disposal modelling means for systematizing 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

2.2.6.1 Variability management support at process level

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

This section discusses the seamless integration between EPF Composer and Base Variability Resolution (BVR) tool. EPF Composer provides support for authoring, tailoring and deploying (software) systems development processes. BVR tool supports feature modelling, resolution, realization and derivation of specific family members, as well as their testing and analysis.

EPF Composer is based on the Unified Method Architecture (UMA) metamodel, which defines library as a root container. The library contents are persisted in their own folders as XMI files, in particular, method configurations, method plugins, method content description and processes. However, the XMI files produced by EPF Composer cannot be directly opened. The problems in XMI files have to be resolved for mapping the elements of a target configuration and variability abstractions in the BVR tool. The analysis reveals that the hypertext references (hrefs) in XMI files are based on the globally unique identifiers (GUIDs) for example `uma://_ErexoKA4EeaPp8nsuu2eew`; therefore, the `MalformedURLException` is produced. This is the case with method packages, tasks, roles, work products, tool mentors, selected steps, includes patterns, process views, method plugin selections, method package selections, variability based on element, bases, categorized elements, activity references, sub managers and presentations. The platform specific paths or otherwise Uniform Resource Identifiers (URIs) should be used instead. The problems are resolved for all XMI files. 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 is just required for configured processes. The method library, plugins, configurations, content descriptions and processes might be considered for variability management with the BVR tool.

The feature diagram associated to the development process with variability is modelled in the VSpec editor, the process configurations are performed in the resolution editor, and the placement and replacement fragments are defined in the realization editor. In order to specify the fragments, the model elements are dragged and dropped to the realization editor, and then "create placement" or "create replacement option" is selected from the context menu. This produces the `IllegalOperationException` for UMA compliant models, with a message that the "selected object is not contained anywhere or simply a top most element, not allowed yet". The integration between realization editor and UMA compliant models brings an additional challenge. The analysis reveals that the break down elements, tasks, roles, artefacts, contents, work products, activities, descriptors and deliverables have associated description implementations. Their naming structure "parent name», «parent GUID" is not allowed. Accordingly, we have performed temporary naming adaptations for supporting placements and replacements in the realization editor. 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.

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?" 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 in exported models.

2.2.6.2 Specification of variability at component level

This chapter describes the implementation for requirement **WP6_PPA_004**, described in the table above.



The integration between CHES 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. CHES model is stored in .di, .notation and .uml files. The realization fragments have been specified for the .uml model. However, the commands for back propagation of injected elements into diagrams and removal of orphan (deleted) entries are implemented.

2.2.6.3 Variability management support at assurance case level

This chapter describes the implementation for requirement **WP6_PPA_005**, described in the table above.

To support the variability management at assurance case level, the integration between OpenCert and BVR tools has been achieved. Similar to Product and Product Lines, the generation of target model is performed with three editors: VSpec, Resolution, and Realization. The argumentation is stored in .arg and .arg_diagram files. The command for propagation of injected model elements into diagram is however needed, so that this integration will be implemented as part of P2.

2.2.6.4 Semi-automatic generation of product arguments

This chapter describes the implementation for requirement **WP6_PPA_002**, described in the table above.

The generation of product-based arguments functionality uses the information specified in a CHES 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 contract 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.

2.2.6.5 Semi-automatic generation of process arguments

This chapter describes the implementation for requirement **WP6_PPA_003**, described in the table above.

The generation of process-based arguments functionality takes the exported XML file of EPF delivery process as an input model, which is automatically transformed into safety argument fragments (i.e., model and diagram) using Epsilon Transformation Language (ETL) [8]. The input process model is exported using export of one or more method plug-ins option in EPF. The mapping of elements is focused on the Work Breakdown Structure of processes in EPF, such as Task Descriptor, Role Descriptor and Work Product Descriptor. These descriptors have references to their base elements (such as task, role and work product) in Method Content as well as additional information that is relevant to the local process. The id, name and description of process elements are mapped to the id, name and description of argumentation model elements. 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 functionally.

2.2.7 Reuse Discovery

This chapter describes the implementation for requirement **WP6_RA_002**, described in the table above.

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. The implemented functionality for Prototype P1 is 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.

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 26, so that this SKB can receive input queries in order to get similar artefacts by a given input artefact (Figure 27).

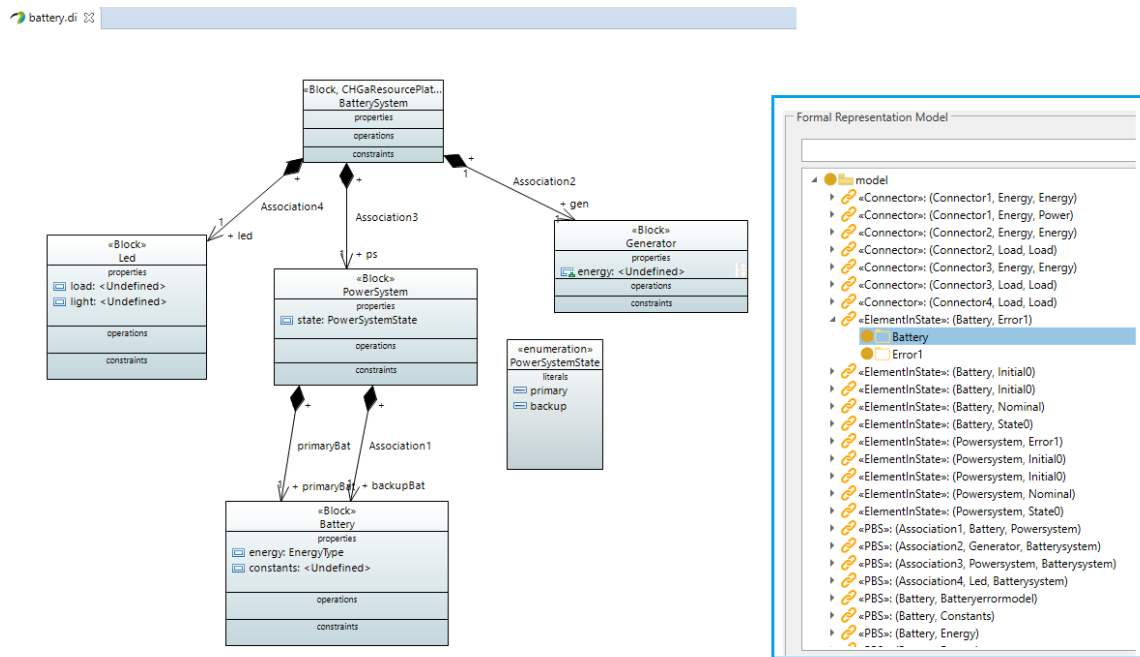


Figure 26. SysML content transformed into RSHF metamodel

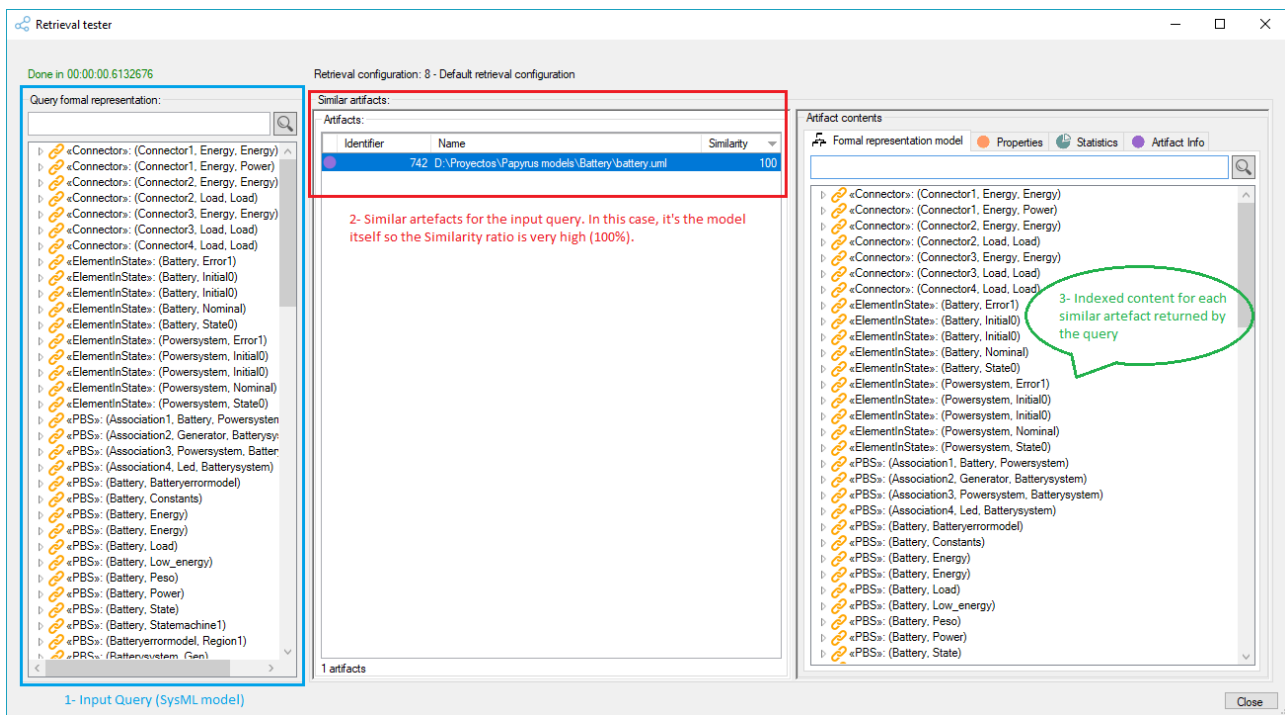


Figure 27. SysML model content as a query, to search for similar artefacts

It has also been integrated within the AMASS platform as a technological demonstrator of the OSLC-KM capabilities in order to integrate this core functionality in P1 (Figure 28).

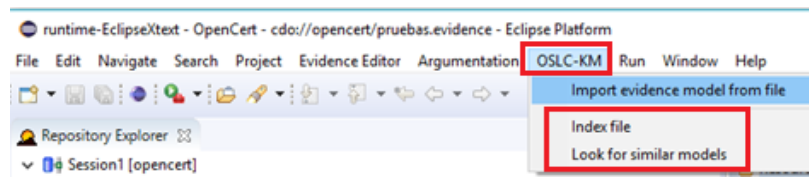


Figure 28. Reuse Discovery operations integrated in AMASS

The integration for P1 includes 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 will allow the user to select other kind of SysML models, and it will be also integrated with CDO.

2.3 Installation and User Manuals

The steps necessary to install the AMASS Prototype P1 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* (Figure 29). 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.

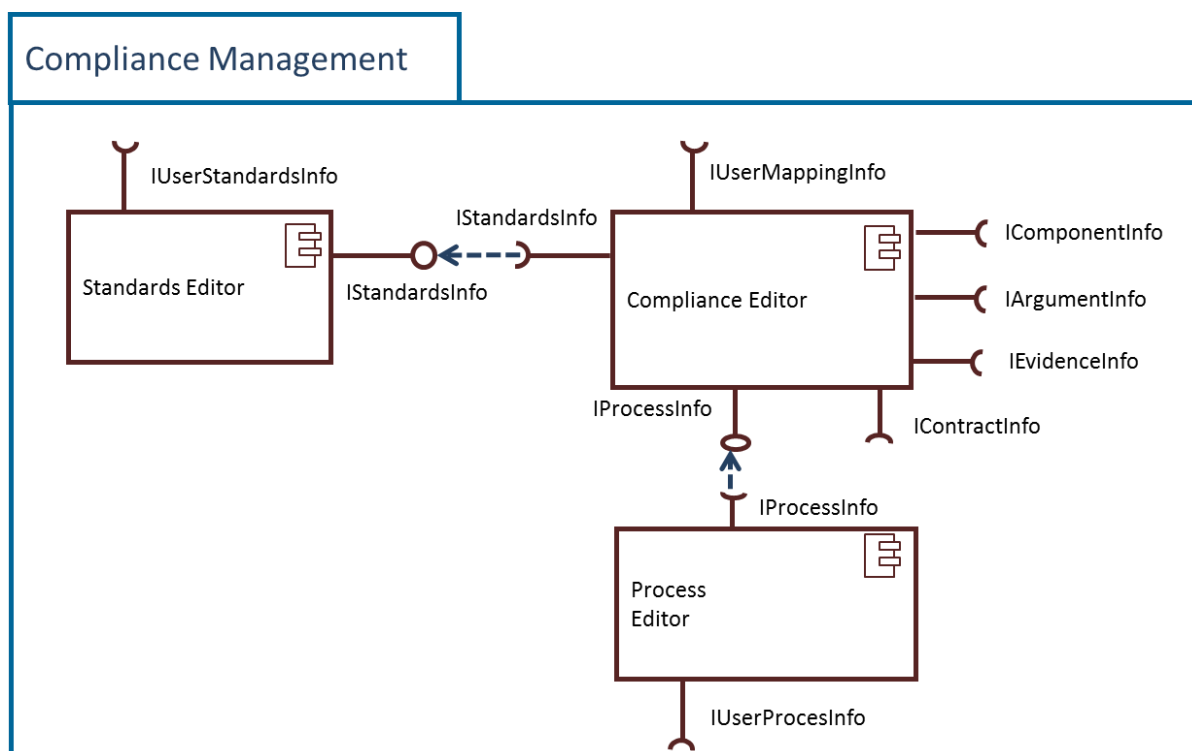


Figure 29. Tool components for Compliance Management

3.1.2 Management of families/lines

To manage families/lines, it is necessary to have at disposal modelling means for systematizing 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 30). 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.

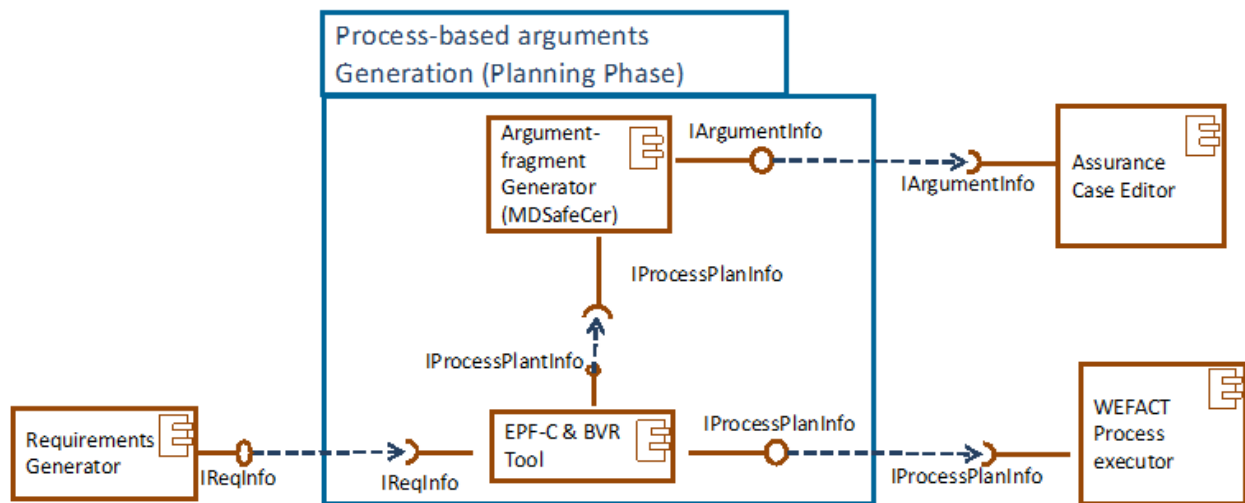


Figure 30. Tool components for Automatic generation of process-based arguments

3.1.2.2 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 31). 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.

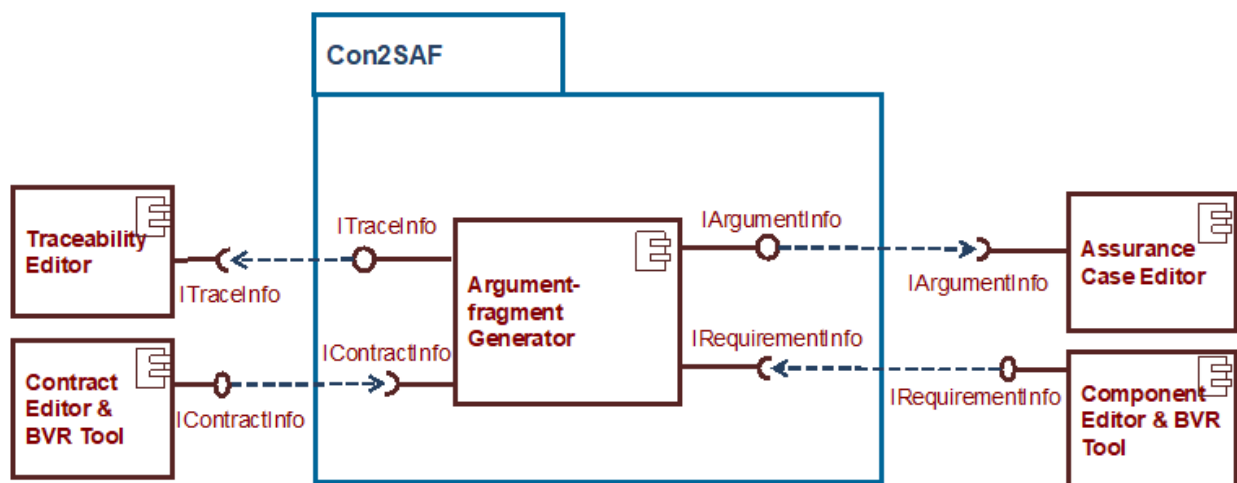


Figure 31. Tool components for Automatic generation of product-based arguments

3.1.3 Cross/Intra Domain Reuse

The Cross/Intra domain reuse module is composed by two main components: *Reuse Assistance* and *Line Specification* (Figure 32). 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.

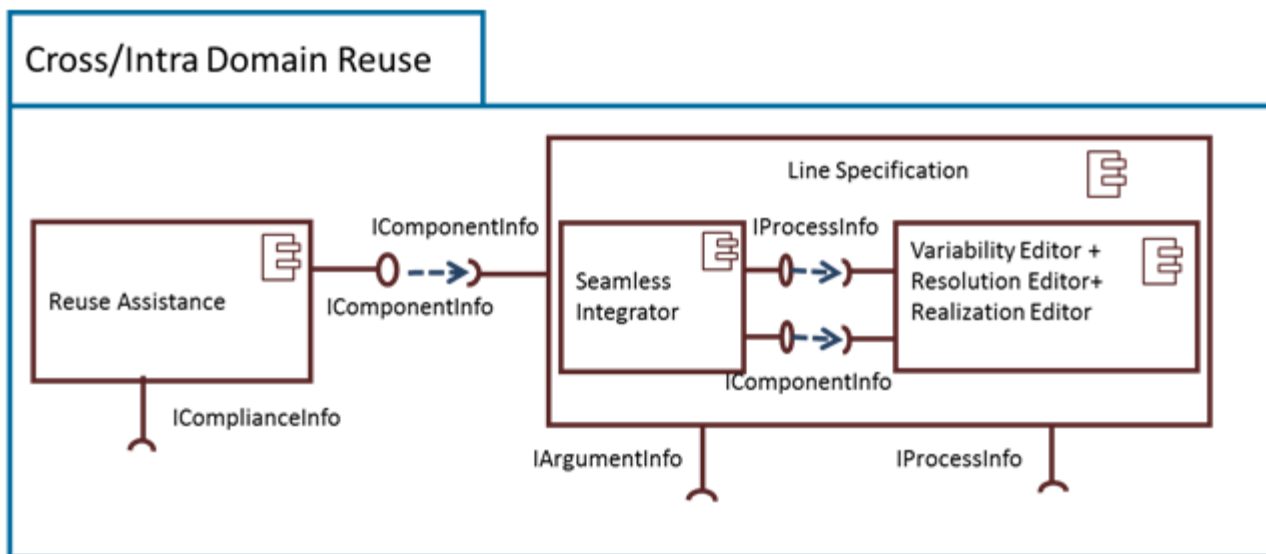


Figure 32. Tool components for Cross/Intra Domain Reuse

3.2 Source Code Description

The source code of the WP6-related components in AMASS prototype P1 can be found in the source code SVN 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 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.opencoss.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.amass.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.amass.lines**



This plugin provides a user interface for copying a method library and resolving the validation errors while opening the XMI files. The errors in XMI files need to be resolved for variability management with the BVR tool. The resolved files are copied under the project folder “error_free_models”.

- **org.eclipse.opencert.process2assurproj**

This plugin provides the transformation of EPF process (compliant with UMA metamodel) to argumentation model and diagram (compliant with CACM metamodel). The generated model and diagram are stored in the corresponding assurance project in the CDO repository.

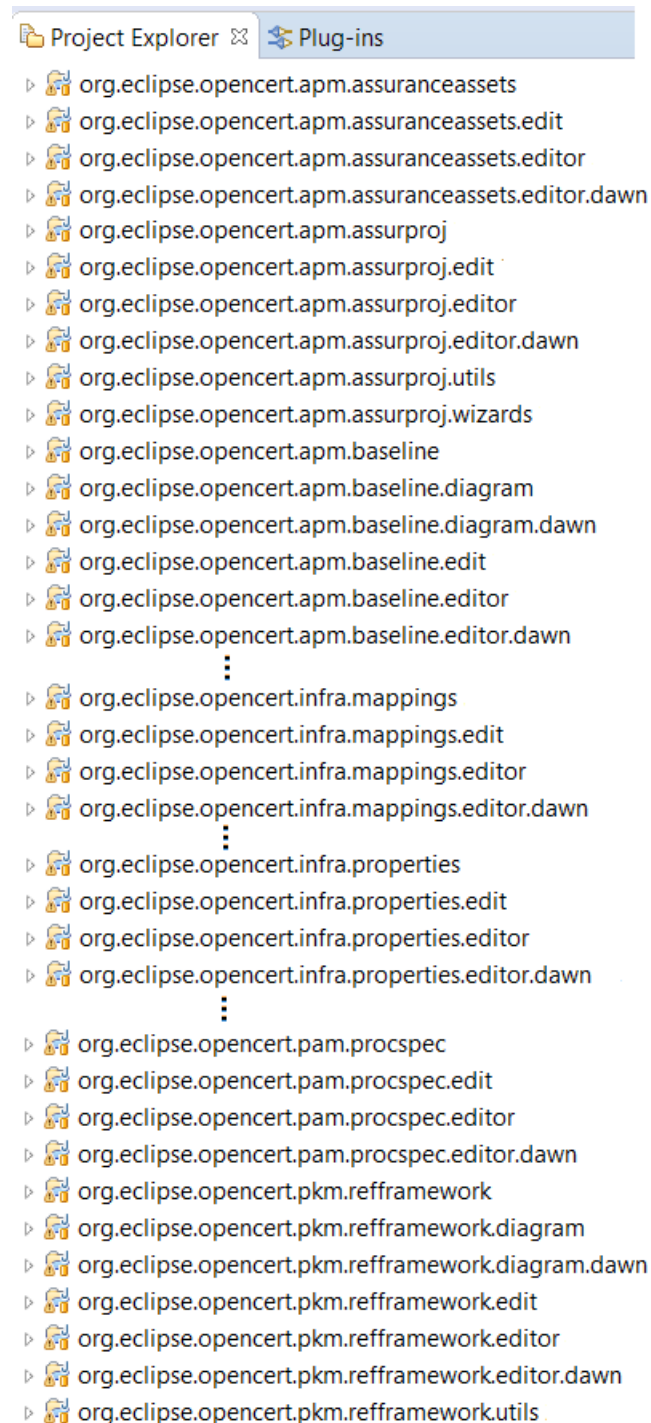


Figure 33. Cross-Domain and Intra-Domain Reuse plugins

In addition, the following plugin from the CHES 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 through the BVR tool (see section 2.2.6.2).

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`.



4. Conclusions

This deliverable has presented the implementation work performed for Cross-Domain and Intra-Domain Reuse in AMASS Prototype P1, which is the second version of the AMASS Tool Platform. The current support in the Platform allows a user to manage variability at the process and component level, semi-automatic generation of 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 P1 has **TRL 4** (technology has been validated in lab). 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), Prototype P2 will include the final decision upon the implementation of new features targeted at: reuse discovery, reuse assistant, variability management at the process/component level, and generation of process arguments.



Abbreviations and Definitions

<i>Abbreviation</i>	<i>Explanation</i>
API	Application Programming Interface
ASIL	Automotive Safety Integrity Level
BVR	Base Variability Resolution
CACM	Common Assurance and Certification Metamodel
CDO	Connected Data Objects
CHESSML	CHESS Modelling Language
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
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
SIL	Safety Integrity Level
SKB	System Knowledge Base
SL	Security Level



<i>Abbreviation</i>	<i>Explanation</i>
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 OPENCROSS project <http://www.opencross-project.eu/>
- [2] The SafeCer project <http://www.safecer.eu/>
- [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 [D2.1 Business cases and high-level requirements](#) (28 February 2017)
- [14] AMASS [D3.5 Prototype for architecture-driven assurance \(b\)](#) (24 October 2017)
- [15] AMASS [D3.4 Prototype for architecture-driven assurance \(a\)](#) (23 December 2016).
- [16] AMASS D5.2 Design of the AMASS tools and methods for seamless interoperability (a) (11 October 2017)
- [17] AMASS project: source code https://services.medini.eu/svn/AMASS_source/ ⁴
- [18] AMASS Platform – Prototype P1 User Manual https://services.medini.eu/svn/AMASS_collab/WP-transversal/ImplementationTeam/PrototypeP1/AMASS_PrototypeP1_UserManual.docx Version 0.1 (2017)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 proposal <https://www.polarsys.org/proposals/opencert>
- [27] Eclipse Process Framework Project (EPF) <https://eclipse.org/epf/>
- [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.

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 current User Manual is a draft document; the final version of the manual will be integrated in D2.5 AMASS User guidance and methodological framework (m31).