

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 (a)
D6.4**

Work Package:	WP6 Cross-Domain and Intra-Domain Reuse
Dissemination level:	PU = Public
Status:	Final
Date:	31 March 2017
Responsible partner:	Borja López (TRC)
Contact information:	borja.lopez@reusecompany.com
Document reference:	AMASS_D6.4_WP6_TRC_V1.0

PROPRIETARY RIGHTS STATEMENT

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

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
Ángel López, Huáscar Espinoza	Tecnia Research & Innovation
Barbara Gallina, Inmaculada Ayala	Maelardalens Hoegskola

Reviewers

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

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	10
2.2.2 Tailoring of Standards models to specific projects	15
2.2.3 Process Compliance (informal) management	16
2.2.4 Compliance Monitoring.....	21
2.3 Installation and User Manuals.....	22
3. Implementation Description.....	23
3.1 Implemented Modules.....	23
3.2 Source Code Description	23
Abbreviations and Definitions	28
References	29

List of Figures

Figure 1. AMASS Building blocks	7
Figure 2. Functional decomposition for the AMASS platform	9
Figure 3. Standard and Process Modelling.....	11
Figure 4. Reference Framework Editor	11
Figure 5. ISO-26262: Recommendation Table associated to a Requirement	12
Figure 6. Authoring perspective of the EPF composer	13
Figure 7. Standard modelled in the EPF composer	14
Figure 8. Recommendation table modelled in the EPF composer	14
Figure 9. Tailoring of Baseline Models from Standard Models	15
Figure 10. Baseline tailoring.....	16
Figure 11. Baseline graphical editor	16
Figure 12. Assurance Project and System Component Specification structure	17
Figure 13. How to create Compliance Maps	18
Figure 14. Compliance Set view	18
Figure 15. Mapping Table view	19
Figure 16. Import View in OpenCert for EPF: Result of the import operation	20
Figure 17. Modelling of compliance mappings in the EPF composer	21
Figure 18. Mapped Requirements in the EPF composer	21
Figure 19. Compliance report in web client.....	22
Figure 20. Tool components for Compliance Management	23
Figure 21. Cross-Domain and Intra-Domain Reuse plugins	27

List of Tables

Table 1.	Requirements implemented in the first prototype of the AMASS platform	10
----------	---	----

Executive Summary

The deliverable D6.4 – Implementation for Cross Domain and Intra Domain Reuse (a), is the 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. 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, as the latter 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 first prototype iteration (Prototype Core) releases the following basic building block from the AMASS tool architecture:

- *Compliance Management*, for (a) creation and edition of process and standards' information as well as of any other information derived from them, and (b) demonstrating conformance with the standards in the context of specific assurance projects.

More concretely, the developed tools in the WP6-related part of the first prototype 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.

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

D6.4 relates to other AMASS outcomes referred in this deliverable:

- Installable AMASS platform tools for the first prototype
- User manual and installation instructions
- Source code description

1. Introduction

The AMASS approach focuses on the development and consolidation of an open and holistic assurance and certification framework for CPS, which constitutes the evolution of the approaches proposed by the EU projects OPENCROSS [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 OPENCROSS and SafeCer conceptual, modelling and methodological frameworks for architecture-driven and multi-concern assurance, as well as for further cross-domain and intra-domain reuse capabilities and seamless interoperability mechanisms (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.

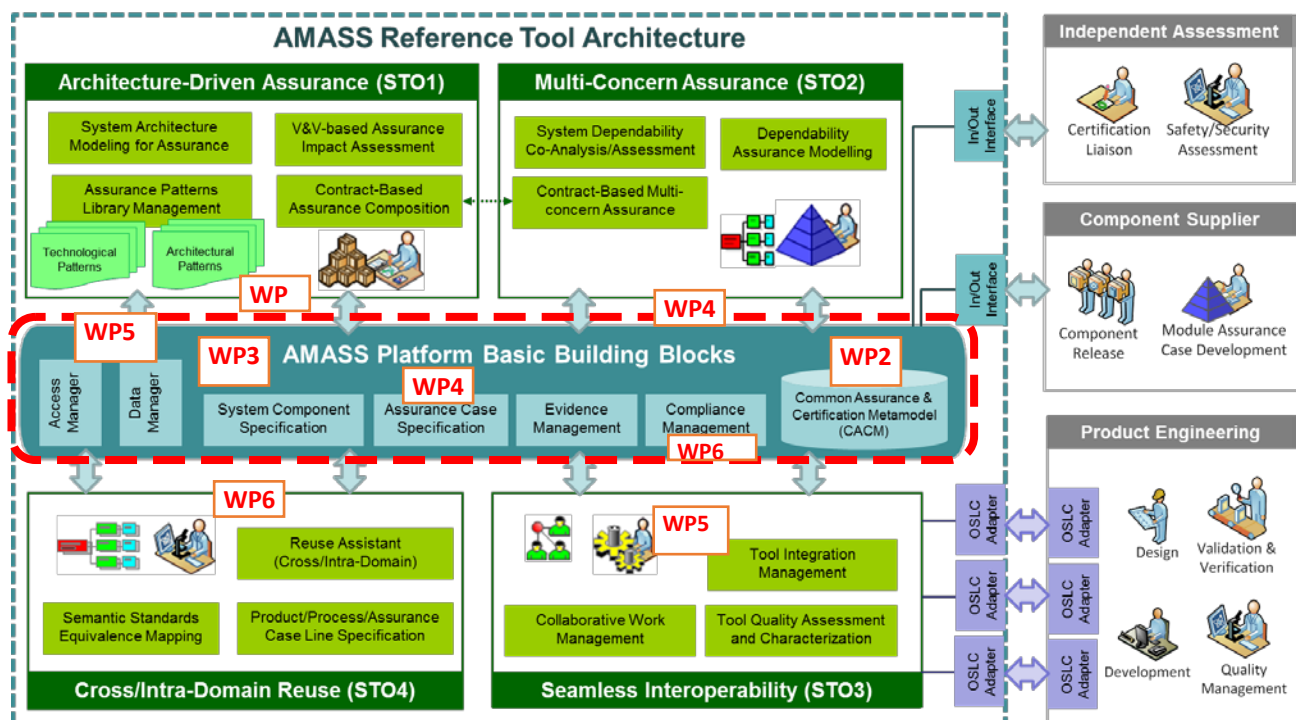


Figure 1. AMASS Building blocks

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 [19].

As part of the Prototype Core, WP6 is responsible for consolidating the previous works on process modelling, standards modelling and conformance demonstration in order to design and implement the basic building block called “**Compliance Management**” (Figure 1). Compliance management of cyber-physical systems builds upon the industrial standards; that is: (a) what the standards define and (b) how compliance/conformance with the standards can be demonstrated.

This deliverable reports the **tool development** results of the “Compliance Management” basic building block. 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.

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 basic building block assigned to WP6 in the first prototype is Compliance Management. Compliance Management supports edition, search, transfer, etc. of process and standards' information as well as of any other information derived from them, such as interpretations about intents and mapping between processes and standards. This functional group maintains a knowledge database about "Standards & Processes", which can be used by other AMASS functionalities (see Figure 2).

The Compliance Management is highlighted with a red circle in Figure 2 showing the general functional overview of the AMASS platform (from deliverable D2.2 [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.

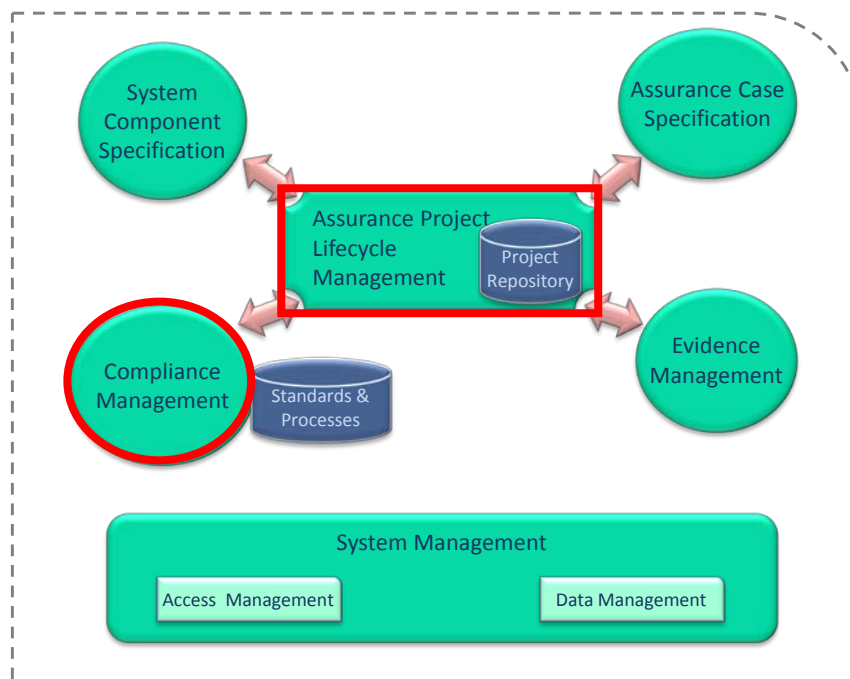


Figure 2. Functional decomposition for the AMASS platform

Additionally, the Compliance Management building block makes possible the monitoring of the compliance progress. For this prototype, the monitoring will be 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.

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

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/>

2.2 Implemented Requirements

From the requirements point of view, this phase focuses on a set of AMASS requirements as defined in deliverable D2.1 [13]. The requirements listed in Table 1 have been implemented in the first 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 1. Requirements implemented in the first prototype of the AMASS platform

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, 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 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>

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 feature is supported by both the OpenCert [25] and EPF toolsets [26]. 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.

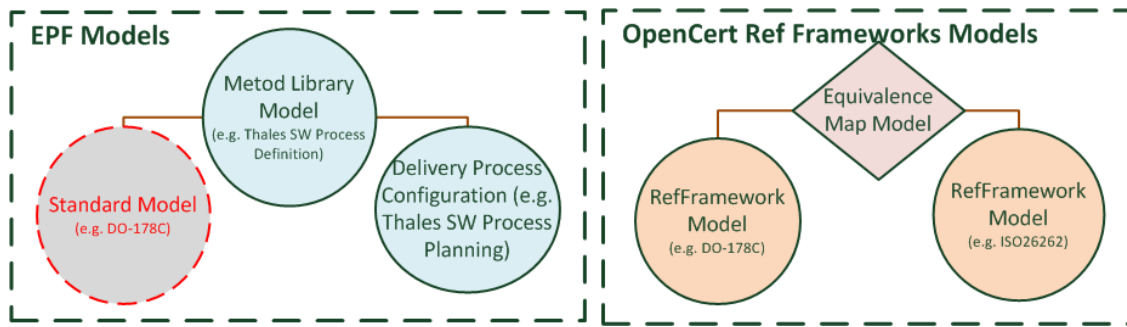


Figure 3. Standard and Process Modelling

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.

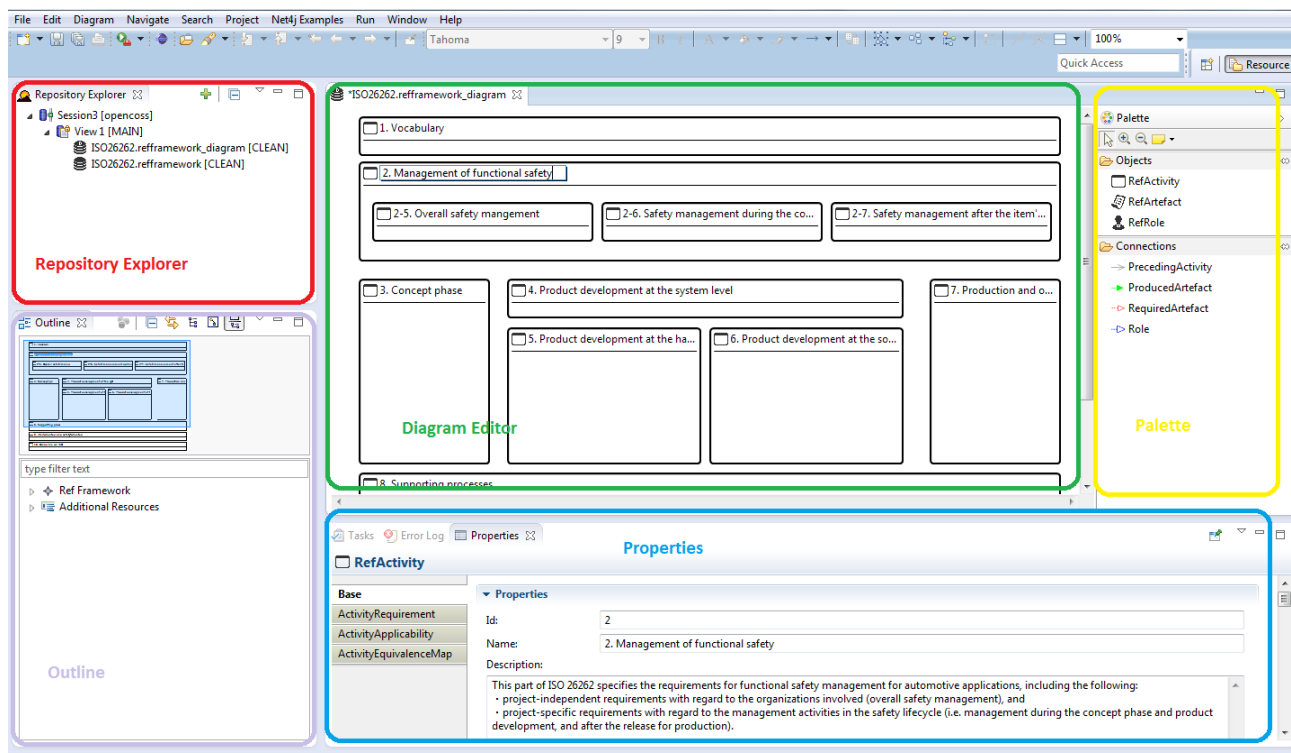


Figure 4. Reference Framework Editor

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.

Table 1 — System design analysis

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.

↓

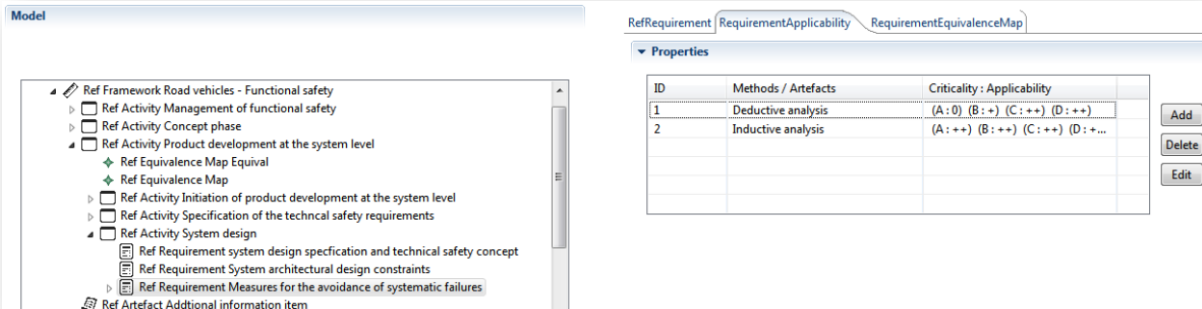


Figure 5. ISO-26262: Recommendation Table associated to a Requirement

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 [24]. The functionality of the EPF composer is organized in two views, the Authoring (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.

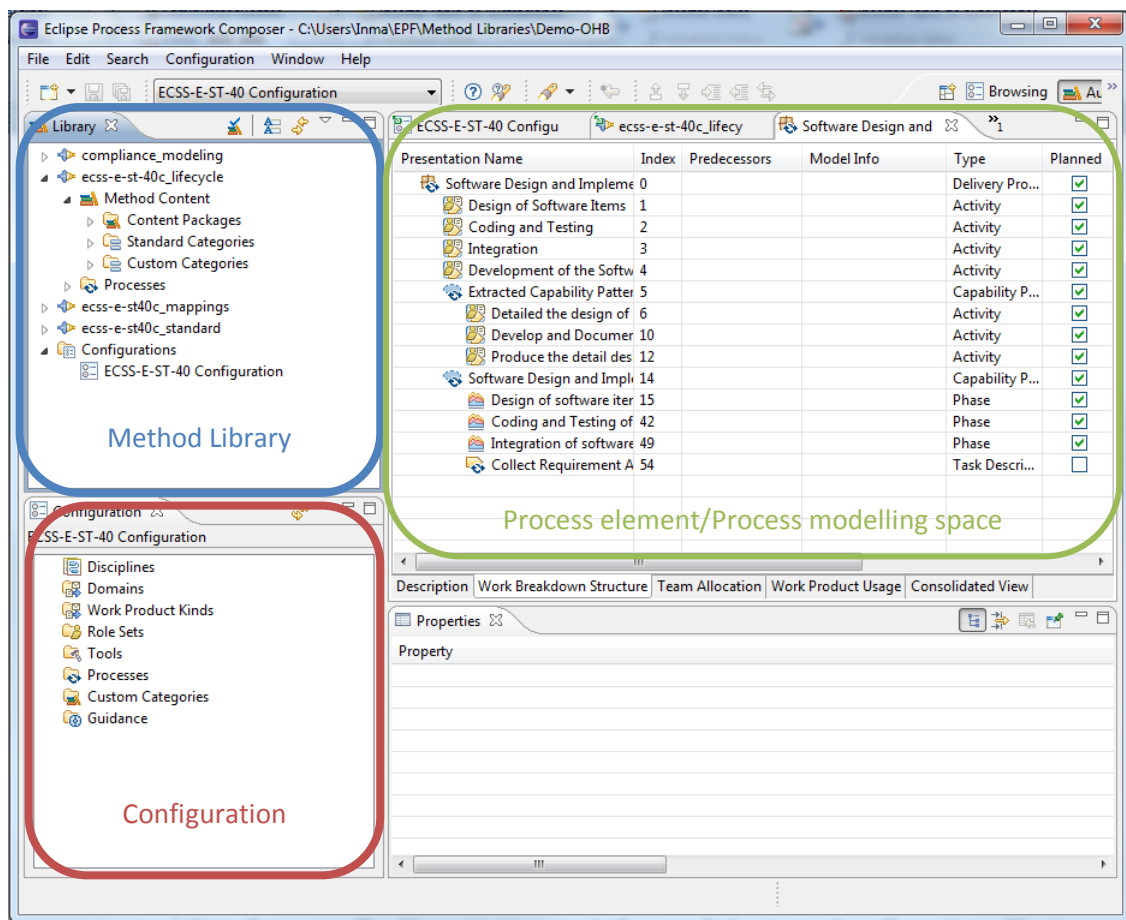


Figure 6. Authoring perspective of the EPF composer

Standards can be modelled in the EPF composer tool [22] following an approach similar to those described in [21] for the IBM Rational Method Composer. In the work presented in [21], 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. So, 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		Activity	Applicability by Software Level				Output		Control Category by Software Level			
	Description	Ref		A	B	C	D	Data Item	Ref	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 ② ② ② ②	① ② ② ② ②	① 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 ② ② ② ②	① ② ② ② ②	① ② 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 ② ② ② ②	① ② ② ② ②	① ② 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 ② ② ② ②	① ② 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 ② ②	① ② ②	① ② ②
6	Software plans comply with this document.	4.1.f	4.3.a 4.6	○	○	○	○	Software Verification Results	11.14	② ② ②	② ② ②	② ② ②	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	② ② ②	② ② ②	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

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. We plan to create Baseline models from EPF Standards models as well. However, this functionality has not been implemented yet.

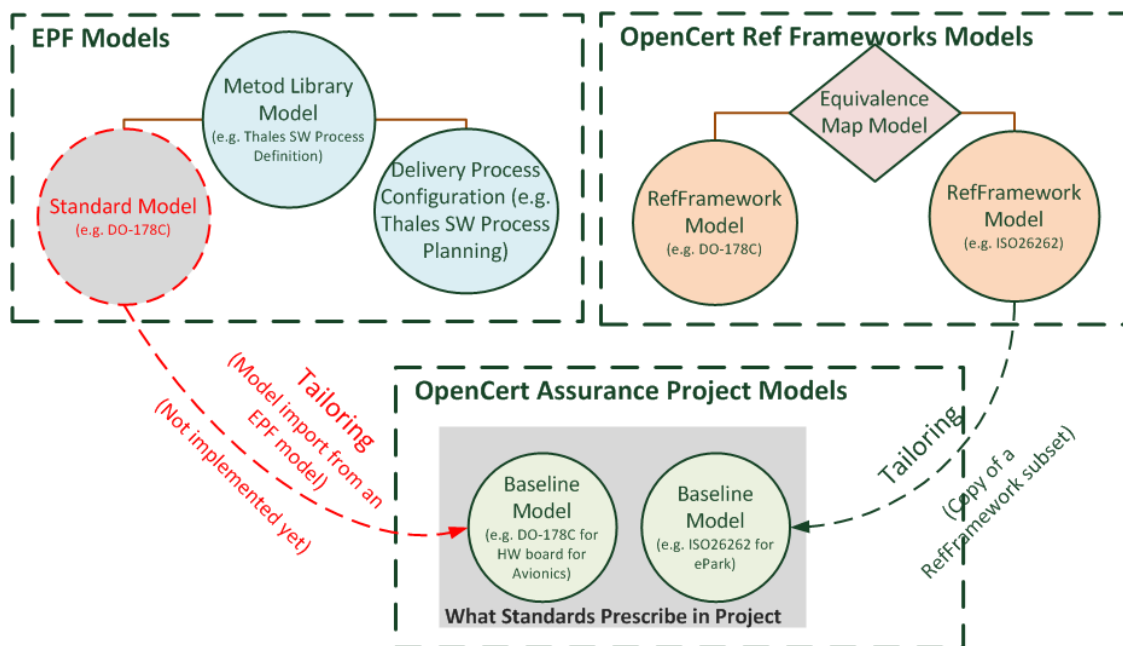


Figure 9. Tailoring of Baseline Models from Standard Models

Each baseline model results from importing (copying) a Reference Framework project 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).

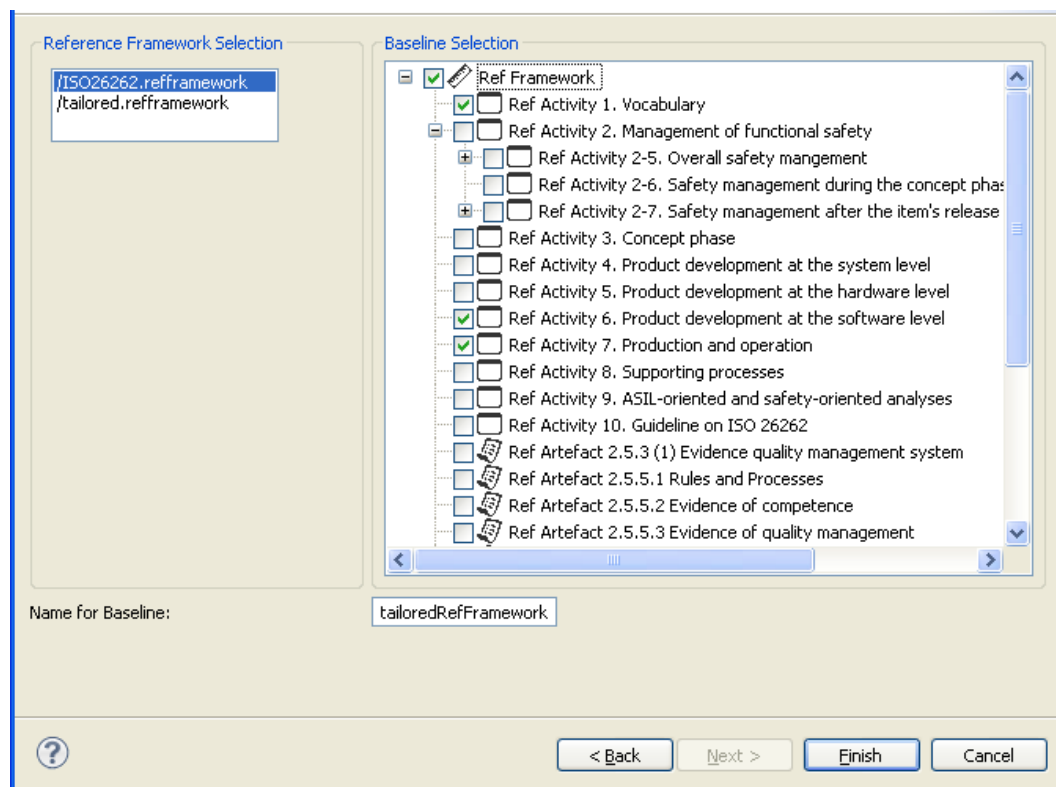


Figure 10. Baseline tailoring

Figure 11 shows an example of baseline model automatically generated from the reference framework model.

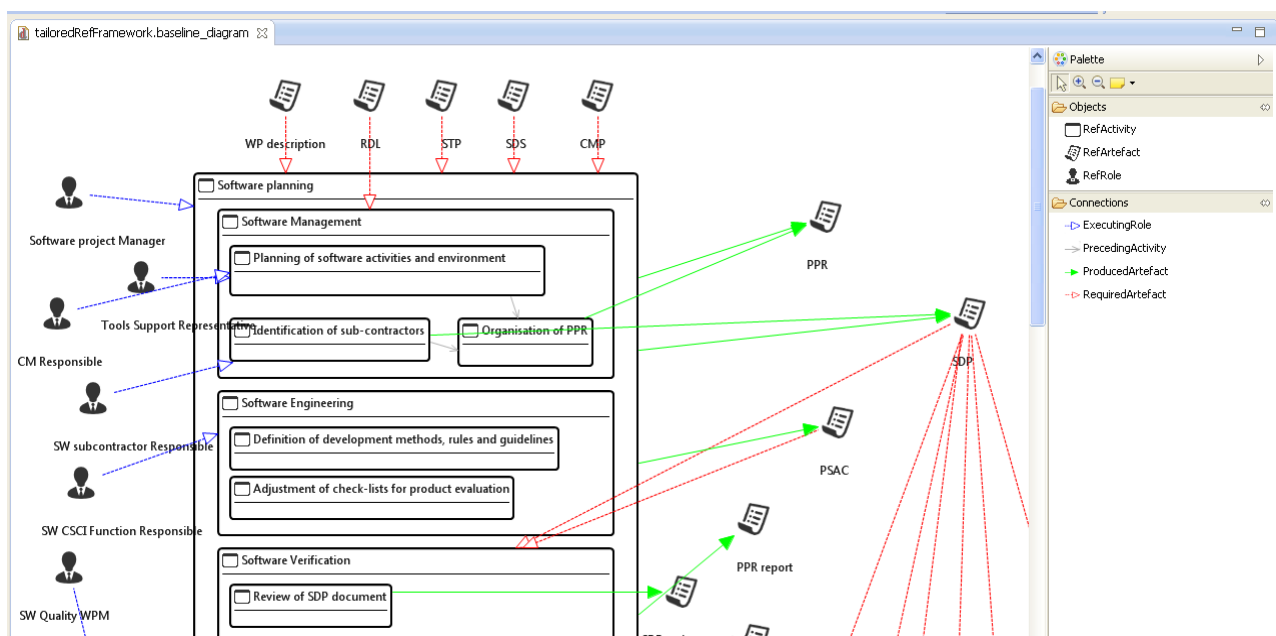


Figure 11. Baseline graphical editor

2.2.3 Process Compliance (informal) management

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. 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.

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 to get a first version of their evidence and executed process models to demonstrate compliance with standards.

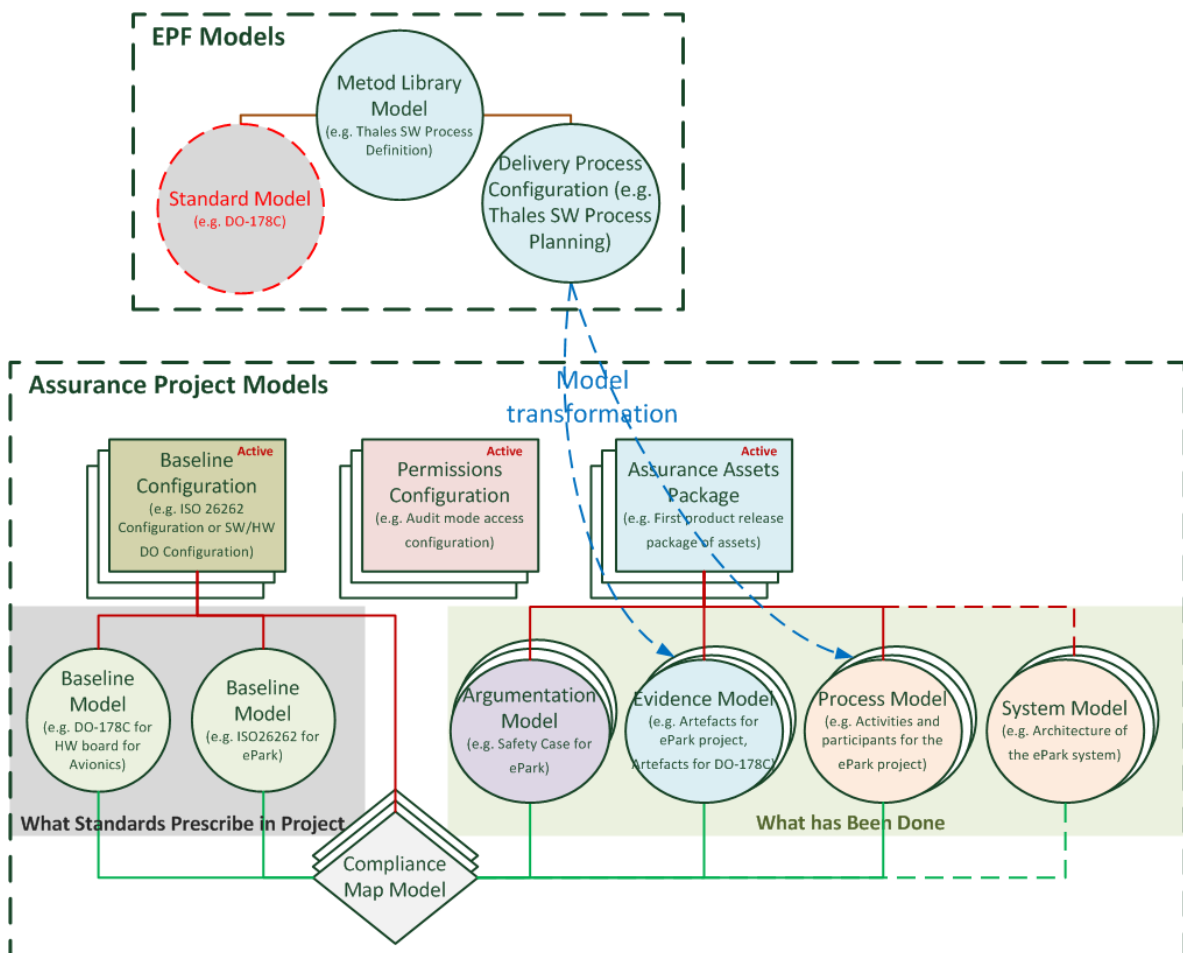


Figure 12. Assurance Project and System Component Specification structure

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.

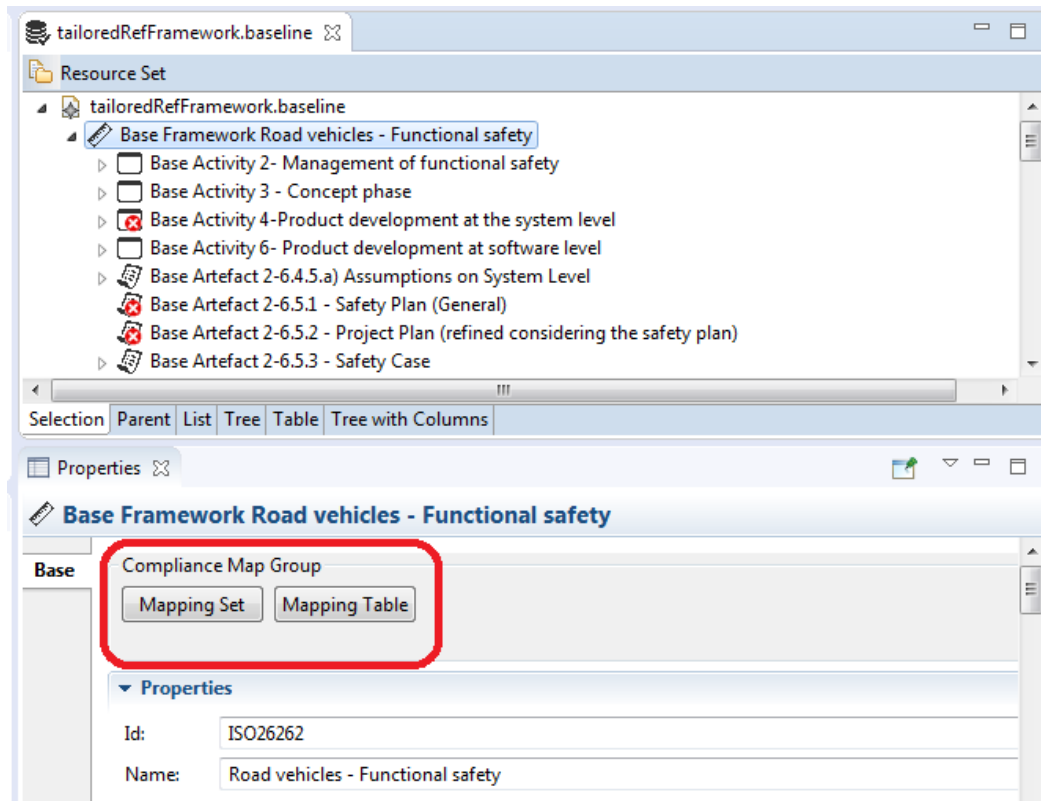


Figure 13. How to create Compliance Maps

Figure 14 shows the Compliance Set view.

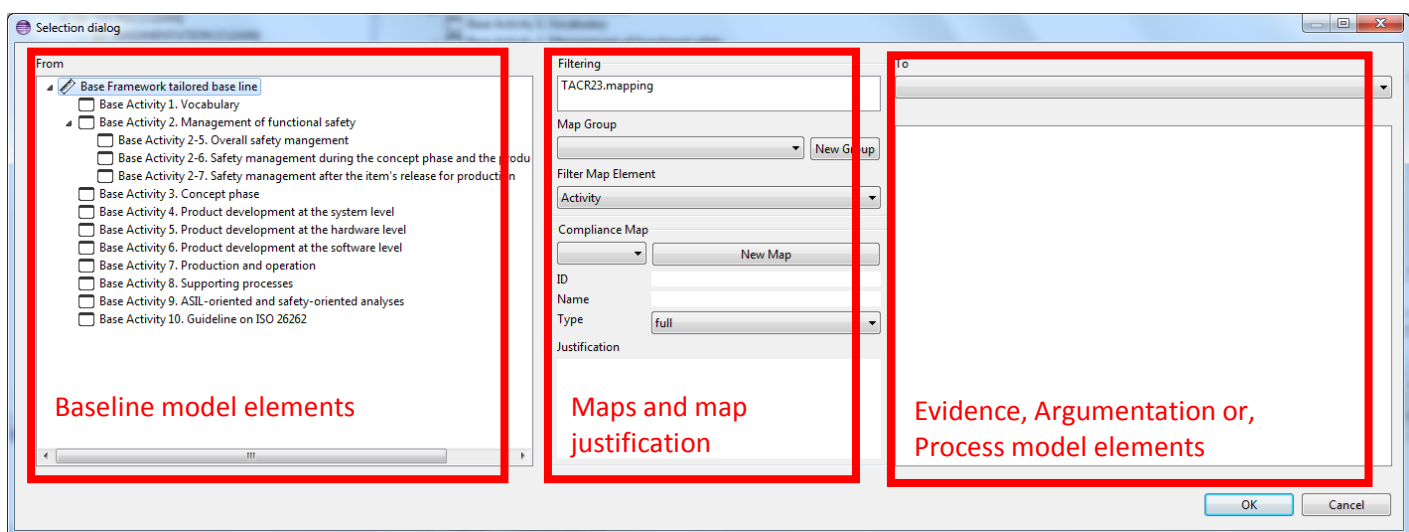


Figure 14. Compliance Set view

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 zone* has controls to allow filtering.
- The *bottom zone* showing two areas:
 - The compliance mapping table that shows all the baseline elements that accomplish with the searching criteria selected by the user.
 - A list that shows all the target elements of the base element selected in the table with a simple left click.

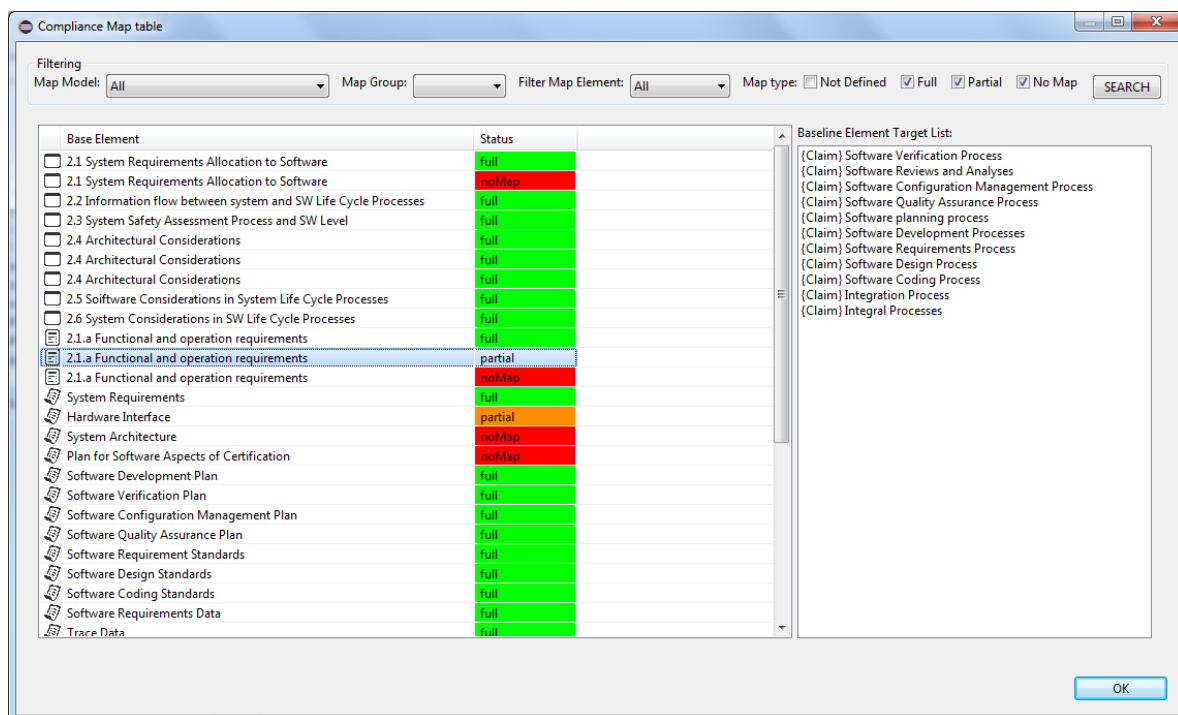


Figure 15. Mapping Table view

2.2.3.1.1 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 modeled in EPF and generates an evidence model and a process model in OpenCert.

The selection of EPF Composer is based on consideration regarding AMASS-needs in terms of reuse-support. A more in depth justification for choosing EPF Composer was documented within deliverable D6.1 [27].

Figure 16 shows the Importing View implemented in OpenCert.

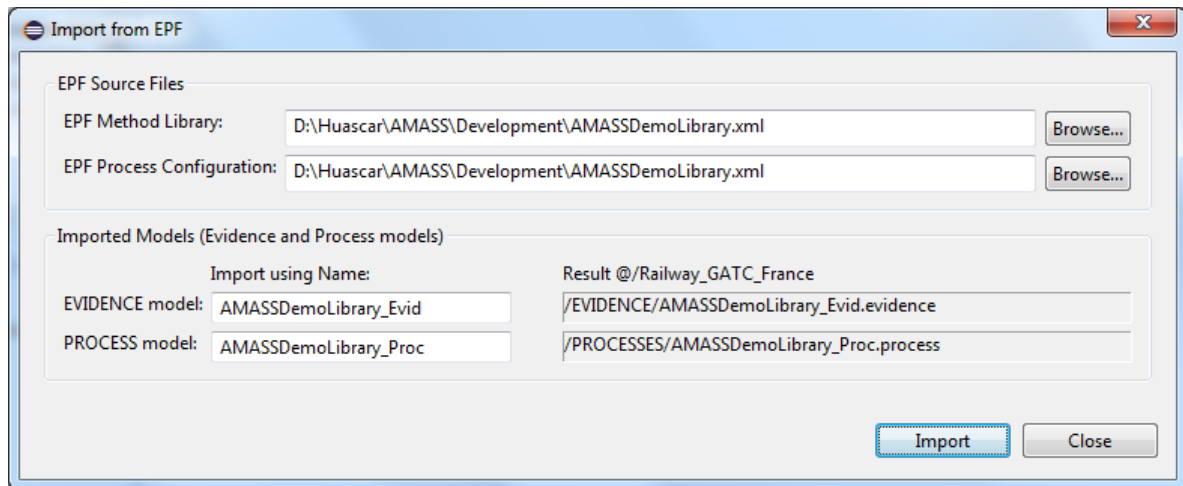


Figure 16. Import View in OpenCert for EPF: Result of the import operation

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 [21]. The use of EPF for compliance modelling allows planning how our process will address standard requirements. Currently, this compliance information cannot be imported to OpenCert but this functionality is planned as a future work.

As a part of the approach presented in [21], 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.

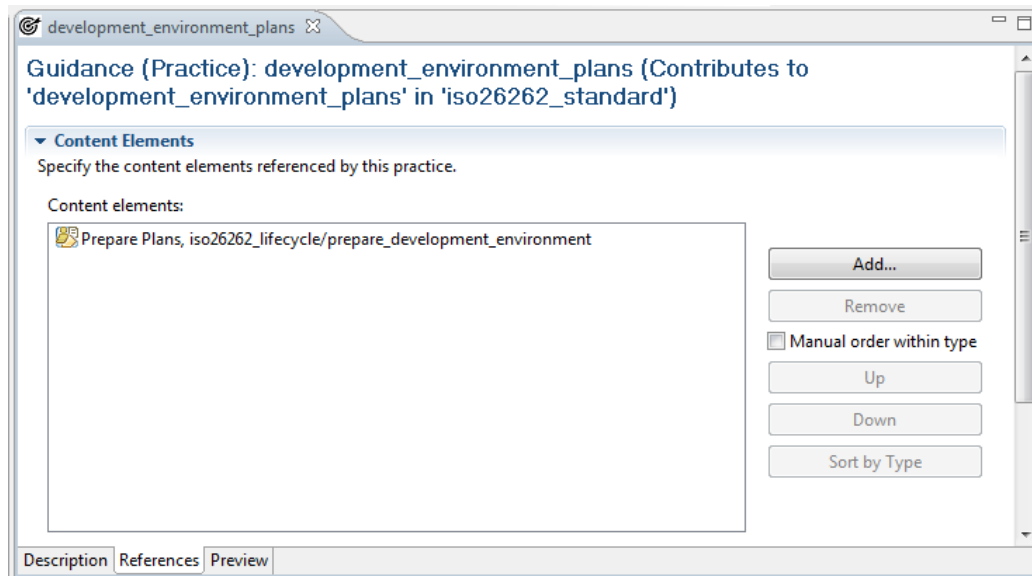


Figure 17. Modelling of compliance mappings in the EPF composer

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.

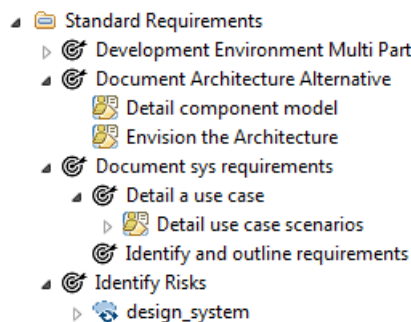


Figure 18. Mapped Requirements in the EPF composer

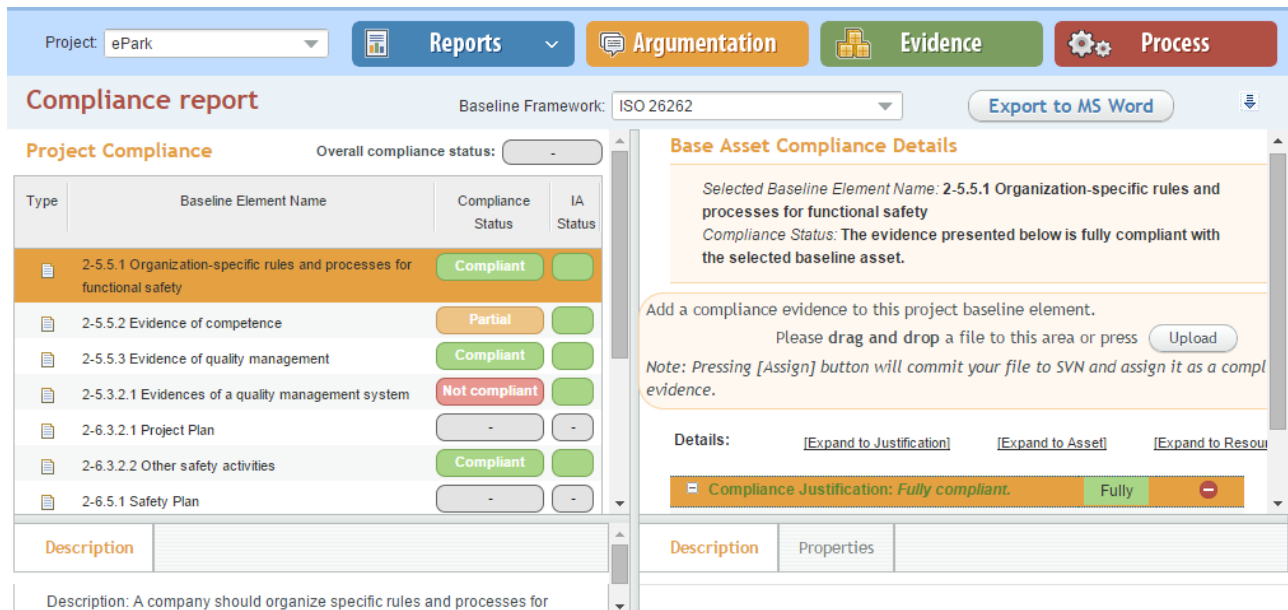
2.2.4 Compliance Monitoring

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 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 buttons 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

2.3 Installation and User Manuals

The steps necessary to install the first prototype are exhaustively described in the AMASS User Manual [17] and will not be repeated in this deliverable. That document contains all required steps and document references to set up the tools, allowing the users to find the installation instructions, the tool environment description, and the functionalities for the creation of Standards and Process models (models representing Standards, Regulations, or Company-specific Processes), Assurance Projects and the associated Evidence models (Artefacts), Compliance Maps (so far, compliance maps from Reference Artefacts to Artefacts), and Argumentation models, in addition to Architecture models.

3. Implementation Description

3.1 Implemented Modules

We have decomposed the Compliance Management module into three components: *Standards Editor*, *Process Editor* and *Compliance Editor* (Figure 20). The first tool component includes services to capture the knowledge from the standards, while the second captures information of the life-cycles that are described in the process. The focus of the third component, the Compliance Editor, 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; EPF (Eclipse Process Framework) is used for modelling processes and for modelling standards as well.

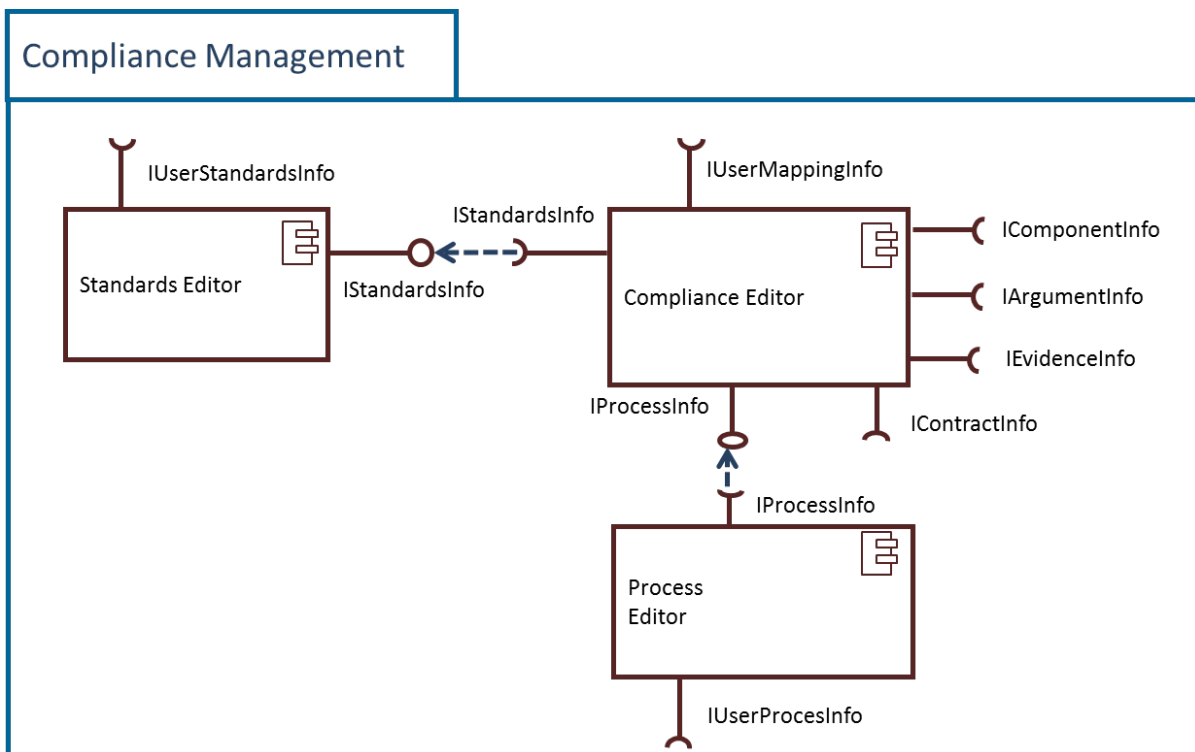


Figure 20. Tool components for Compliance Management

3.2 Source Code Description

The source code of the first AMASS prototype can be found in the source code SVN repository [16]. The code for the Compliance Management module in its first prototype is stored together with the code of the other basic building blocks in the SVN repository under “tag” to distinguish the state of the code at the time of the integrated release.

Once all the plugins are installed, these are the necessary ones for the Standards, Projects, Compliance and Processes Management Specification:

- **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. Additionally, it includes the transformations from EPF to OpenCert that generates initial models for artifacts and process.
- **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.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.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.

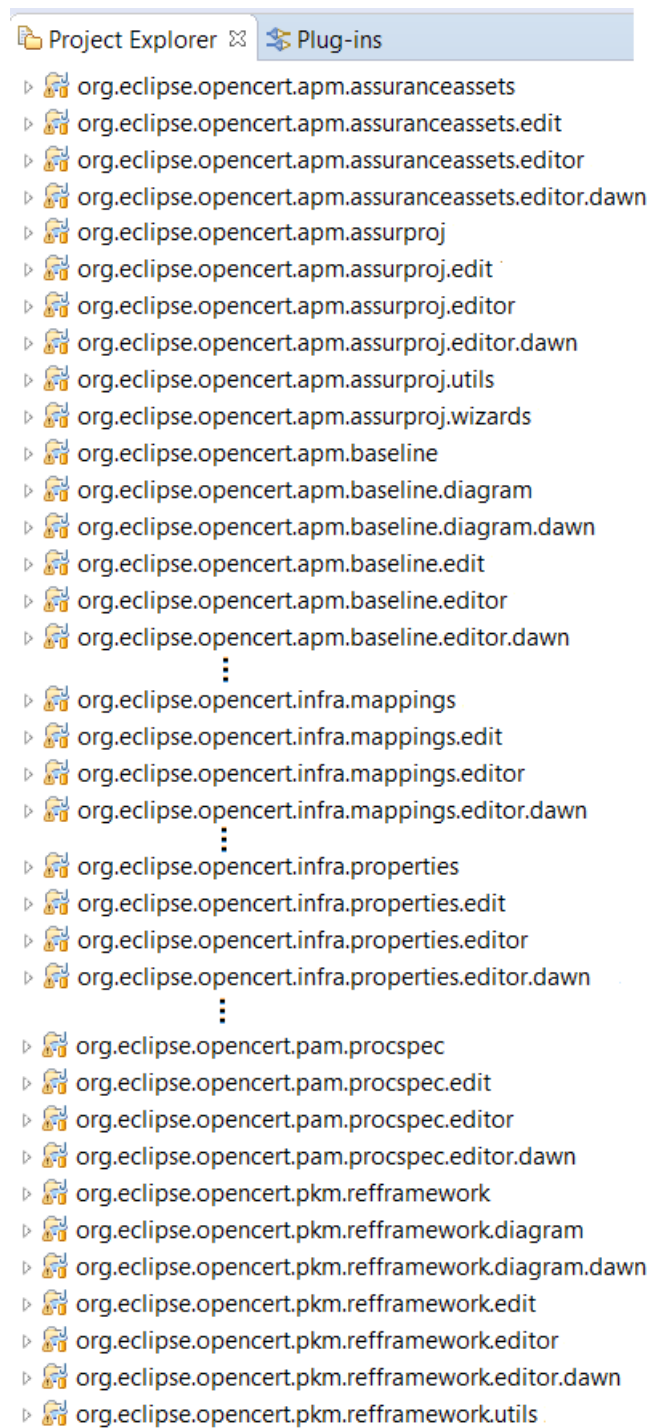


Figure 21. Cross-Domain and Intra-Domain Reuse plugins

Abbreviations and Definitions

<i>Abbreviation</i>	<i>Explanation</i>
API	Application Programming Interface
CPS	Cyber-Physical Systems
ECSEL	Electronic Components and Systems for European Leadership
EN	European Norm
GSN	Goal Structured Notation
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Organization for Standardization
JU	Joint Undertaking
OMG	Object Management Group
OSLC	Open Services for Lifecycle Collaboration
PhD	Philosophiae Doctor (neolatin; = doctor of philosophy)
RAMS	Reliability, Availability, Maintainability, Safety (and Security)
RTCA	Radio Technical Commission for Aeronautics
SACM	Structured Assurance Case Metamodel
SPEM	Software & Systems Process Engineering Metamodel
SysML	System Modelling Language
S&S	Safety and Security
V&V	Verification and Validation
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 Modeling Project (GMP) <http://www.eclipse.org/modeling/gmp/>
- [7] Eclipse Modeling 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 D2.2 AMASS Reference Architecture (a) Version 1.0 (30 November 2016)
- [15] AMASS [D3.4 Prototype for architecture-driven assurance \(a\)](#) (23 December 2016).
- [16] AMASS source code https://services.medini.eu/svn/AMASS_source/ ⁴
- [17] AMASS Platform – Prototype Core User Manual https://services.medini.eu/svn/AMASS_collab/WP-transversal/ImplementationTeam/PrototypeCore/AMASS_Prototype1_UserManual.docx Version 0.1 (2017)⁵
- [18] WEFAC <http://www.ait.ac.at/en/research-fields/verification-validation/methods-and-tools/wefact/>
- [19] AMASS [D1.1 Case studies description and business impact](#) (30 November 2016)
- [20] Richard Hawkins, Software Contribution Safety Argument Pattern (2009) <http://www.goalstructuringnotation.info/archives/234>
- [21] McIsaac, B.: Ibm rational method composer: Standards mapping. Tech. rep., IBM Developer Works (2015)
- [22] Eclipse Process Framework Project. <https://eclipse.org/epf/>
- [23] Rational Method Composer. <http://www-03.ibm.com/software/products/es/rmc>
- [24] Object Management Group: Software & systems process engineering meta-model specification. Tech. rep. (2008), <http://www.omg.org/spec/SPEM/2.0/>
- [25] OpenCert proposal <https://www.polarsys.org/proposals/opencert>
- [26] Eclipse Process Framework Project (EPF) <https://eclipse.org/epf/>
- [27] AMASS [D6.1 Baseline and requirements for cross/intra-domain reuse](#) (29 September 2016)

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 (huascar.espinoza@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).