

ECSEL Research and Innovation actions (RIA)



AMASS

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

**Prototype for seamless interoperability (c)
D5.6**

Work Package:	WP5 Seamless Interoperability
Dissemination level:	PU = Public
Status:	Final
Date:	28 September 2018
Responsible partner:	Luis M. Alonso (TRC)
Contact information:	luis.alonso@reusecompany.com
Document reference:	AMASS_D5.6_WP5_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.

Contributors¹

Names	Organisation
Luis M. Alonso, Borja López	The REUSE Company (TRC)
Jose Luis de la Vara, Jose María Álvarez, Eugenio Parra, Roy Mendieta, Francisco Rodríguez	Universidad Carlos III de Madrid (UC3)
Ángel López, Alejandra Ruiz, Estibaliz Amparan	Tecnalia Research & Innovation (TEC)
Pietro Braghieri, Stefano Tonetta, Alberto Debiasi	Fondazione Bruno Kessler (FBK)
Stefano Puri	Intecs (INT)
Marc Sango	ALL4TEC (A4T)
Tomáš Kratochvíla, Vit Koksa	Honeywell (HON)
Ivana Černá	Masaryk University (UOM)
Jan Mauersberger	Ansys medini Technologies (KMT)
Markus Grabowski	Assystem Germany (B&M)
Morayo Adedjouma, Botella Bernard, Huascar Espinoza, Thibaud Antignac	CEA LIST (CEA)
Staffan Skogby, Detlef Scholle	Alten Sweden (ALT)

Reviewers

Names	Organisation
Daniel Wright (Peer review), Ran Bi (Peer review)	Rapita Systems (RPT)
George Bravos (Peer review)	Infineon (IFX)
Cristina Martinez (Quality Manager)	Tecnalia Research & Innovation (TEC)

¹ The list includes the contributors to D5.5, which is evolved in D5.6

TABLE OF CONTENTS

Abbreviations and Definitions.....	9
Executive Summary.....	11
1. Introduction (*).....	13
2. Implemented Functionality (*).....	16
2.1 Scope (*).....	16
2.2 Implemented Requirements (*).....	16
2.2.1 Evidence Management Functionality (**).....	22
2.2.1.1 'Characterise Artefact' with OpenCert	22
2.2.1.2 'Link Artefact with External Tool' with OpenCert.....	22
2.2.1.3 'Specify Artefact Lifecycle' with OpenCert.....	24
2.2.1.4 'Evaluate Artefact' with OpenCert.....	25
2.2.1.5 'Evaluate Artefact' through Management of V&V evidence (**).....	26
2.2.1.6 'Specify Process Information for Artefacts' with OpenCert.....	29
2.2.2 Assurance Traceability Functionality (**).....	30
2.2.2.1 'Specify Traceability between Assurance Assets' with OpenCert	30
2.2.2.2 'Specify Traceability between Assurance Assets' with Capra (*)	30
2.2.2.3 'Specify Traceability between Assurance Assets' for Knowledge-Centric Automated Traceability (**).....	32
2.2.2.4 'Conduct Impact Analysis of Assurance Asset Change' with OpenCert	33
2.2.2.5 'Conduct Impact Analysis of Assurance Asset Change' with Knowledge-Centric Automated Traceability (**)	34
2.2.3 Tool integration Functionality (**).....	34
2.2.3.1 'Characterise Toolchain' with OpenCert (**)	34
2.2.3.2 'Characterise Toolchain' with Papyrus (**).....	34
2.2.3.3 'Characterise Toolchain' with Systems Engineering Suite (**)	35
2.2.3.4 'Specify Tool Connection Information' with OpenCert.....	37
2.2.3.5 'Specify Tool Connection Information' for OSLC-KM-based Integration (*)	37
2.2.3.6 'Specify Tool Connection Information' for Integration with V&V Manager (*)	46
2.2.3.7 'Specify Tool Connection Information' for Integration of CHESS and V&V Tools (*)	49
2.2.3.8 'Specify Tool Connection Information' for Papyrus Safety and Security Engineering (**).....	52
2.2.3.9 'Specify Tool Connection Information' for Systems Engineering Suite through Ad-hoc Tool Integration (*)	53
2.2.3.10 'Specify Tool Connection Information' for Safety/Cyber Architect (**)	58
2.2.3.11 'Specify Tool Connection Information' for Farkle (**).....	65
2.2.3.12 'Specify Tool Connection Information' for Sabotage (**).....	65
2.2.3.13 'Specify Tool Connection Information' for SAVONA (**).....	66
2.2.3.14 'Specify Tool Connection Information' though Automatic Generation of OSLC KM-based Connectors (**)	66
2.2.4 Platform Management Functionality (**)	67
2.2.4.1 'Configure Access to Assurance Assets' in OpenCert (**)	67
2.2.4.2 'Log in the platform' in OpenCert (**).....	72
2.2.4.3 'Concurrent Assurance Information Edition' with Web-based Technologies	73
2.2.4.4 'Concurrent Assurance Information Edition' with Data Mining Technologies.....	74
2.2.4.5 'Concurrent Assurance Information Edition' through Automatic Translations (**)	76
2.2.4.6 'Concurrent Assurance Information Edition' in OpenCert (**)	80

2.2.4.7 Concurrent System Architecture Edition in OpenCert (**)	85
2.3 Installation and User Manuals (*)	86
3. Implementation Description (*)	87
3.1 Implemented Modules (*)	87
3.2 Implemented Metamodel	89
3.3 Source Code Description for the AMASS Tool Platform (*)	90
3.4 Source Code Description for External Tools (*)	93
3.4.1 Seamless Interoperability Features in Systems Engineering Suite by TRC (**)	93
3.4.1.1 OSLC-KM standard and OSLC-KM implementation (**)	93
3.4.1.2 ReqIF Connector (**)	96
3.4.1.3 PTC Integrity Connector (**)	97
3.4.1.4 RAT for Rhapsody Plugin (**)	98
3.4.1.5 DOORS Next Generation Connector (**)	99
3.4.1.6 Automatic translations (**)	100
3.4.2 Seamless Interoperability Features for Safety/Cyber Architect tools (**)	101
3.4.3 Integration of CHES and V&V Tools (**)	101
4. Conclusion (*)	105
References	106

List of Figures

Figure 1.	AMASS Building blocks	13
Figure 2.	Functional decomposition for the AMASS platform	16
Figure 3.	Artefact definition creation	22
Figure 4.	Artefact data specification	23
Figure 5.	Use of SVN repository as artefact repository	23
Figure 6.	Resource specification for an artefact	24
Figure 7.	Resource properties	24
Figure 8.	Artefact event properties	25
Figure 9.	Artefact evaluation properties	25
Figure 10.	VERIFICATION Studio displaying some requirements in a specification	26
Figure 11.	AMASS Repository connection parameters	26
Figure 12.	Importing evidence into the AMASS repository	27
Figure 13.	Detailed requirement information	28
Figure 14.	Details of the export of the assessment of a metric for a requirement	28
Figure 15.	Details of the export of the metadata of the assessment of a metric for a requirement	29
Figure 16.	Process model	29
Figure 17.	Activity data	30
Figure 18.	Advanced CAPRA trace creation view (drop sensitive)	31
Figure 19.	Tracing a claim to a contract	32
Figure 20.	Knowledge-Centric Automated Traceability	33
Figure 21.	Modification event of an artefact	33
Figure 22.	Impact analysis information	34
Figure 23.	Impact analysis with Knowledge-Centric Automated Traceability	34
Figure 24.	SE Suite new connectors	36
Figure 25.	RAT plugin for Rhapsody, create new requirement with RAT	36
Figure 26.	RAT plugin for Rhapsody, edit requirement description with RAT	37
Figure 27.	OSLC-KM Importing an Evidence Model from a model file	37
Figure 28.	Fragment of a Papyrus model to be imported	38
Figure 29.	Step #1 of the OSLC-KM Evidence Manager Importer	38
Figure 30.	Step #2 of the OSLC-KM Evidence Manager Importer	39
Figure 31.	New evidence model from a Papyrus model	40
Figure 32.	OSLC-KM Preferences. Web Service URL	41
Figure 33.	VERIFICATION Studio Connection Window	42
Figure 34.	OSLC-KM Connection (SysML Papyrus sub-type)	43
Figure 35.	OSLC-KM input type: Web Service	43
Figure 36.	OSLC-KM input type: File	44
Figure 37.	OSLC-KM input type: Database	44
Figure 38.	OSLC-KM input type: Database connection parameters window	45
Figure 39.	OSLC-KM mappings selector	45
Figure 40.	OSLC-KM mappings edition window	45
Figure 41.	OSLC-KM connection window. Optional configuration	46
Figure 42.	OSLC-KM connection window. Custom-code filtering	46
Figure 43.	V&V Manager integration	47
Figure 44.	Status of the V&V tasks and the available results	48
Figure 45.	FBK Tool Integration via files	49
Figure 46.	FBK Tool Integration via OSLC Automation	50
Figure 47.	FBK Tool Adapters Configuration	50
Figure 48.	Contract and Behaviour Verification context menu	51

Figure 49.	Contract and Behaviour Verification main menu	51
Figure 50.	FBK Tool Automation Plan example.....	52
Figure 51.	Interoperability flow between Papyrus SSE and XFTA tool	53
Figure 52.	User interface in Papyrus SSE for seamless model-checking analysis with NuSMV tool.....	53
Figure 53.	ReqIF metamodel.....	54
Figure 54.	ReqIF connection window focusing on the ReqIF specific parameters	54
Figure 55.	Mapping window for ReqIF Specifications	55
Figure 56.	Mapping window for a single ReqIF specification	55
Figure 57.	Integrity connector architecture	55
Figure 58.	PTC integrity connection window	56
Figure 59.	Rhapsody connector architecture	57
Figure 60.	DNG Connection window	58
Figure 61.	Interoperability between AMASS platform (CHESS, OpenCert) and Safety/Security analysis tools (Safety Architect and Cyber Architect)	59
Figure 62.	Flow of UML/SysML Model to Farkle	65
Figure 63.	Extended Farkle interface for AMASS Tools	65
Figure 64.	CHESS Export function of SAVONA	66
Figure 65.	GUI to allow creation of XSLT files to customise the mapping of XML file nodes to elements in the OSLC-KM metamodel	67
Figure 66.	Manage Security menu	68
Figure 67.	Manage Users Window	69
Figure 68.	Manage Roles Window	69
Figure 69.	Manage Groups Window	70
Figure 70.	Repository Explorer showing models with different font styles according the access rights	70
Figure 71.	Generation of Assurance Project in Repository folders with write access	71
Figure 72.	Import data options disabled for an Assurance Project with read only access.....	72
Figure 73.	Authentication for Platform Access	73
Figure 74.	Change password window.....	73
Figure 75.	Ultimate picture of collaborative work using rich and web clients	74
Figure 76.	Screenshot of the current version of the web-based tool for collaborative model editing	74
Figure 77.	Screenshot of the Indexing configuration preferences.....	75
Figure 78.	Screenshot of context menu to index data	75
Figure 79.	Screenshot of the Data Mining platform for collaborative work	76
Figure 80.	Screenshot of the Kibana Discovery tool	76
Figure 81.	Automatic translations via Ontology patterns	77
Figure 82.	Linking of patterns across different languages.....	77
Figure 83.	INTEROPERABILITY Studio > Transformation manager	78
Figure 84.	INTEROPERABILITY Studio > Transformation parameterization.....	78
Figure 85.	INTEROPERABILITY Studio > Specification point of view	79
Figure 86.	Transformation output log	79
Figure 87.	Target specification showing new work products generated automatically	80
Figure 88.	INTEROPERABILITY Studio > SandBox window	80
Figure 89.	Conflict message when saving a model.	81
Figure 90.	Two users edit the same model elements. When the user 1 (left editor) saves the changes, the conflicted elements are marked in red to the user 2 (right editor).....	81
Figure 91.	Conflicted elements bordered in red in a graphical editor.	82
Figure 92.	Interactive Conflict Resolution context menu.....	83
Figure 93.	Rollback resolution	83
Figure 94.	Options to Lock/Unlock model elements.....	84
Figure 95.	Lock state visualization in editors (locker user editor in the left).....	85
Figure 96.	Creating a new CHESS Model in CDO.....	85
Figure 97.	Platform management modules.....	88

Figure 98.	Evidence management module	88
Figure 99.	Assurance traceability modules.....	89
Figure 100.	Tool integration modules	89
Figure 101.	Excerpt of artefact information in the CCL.....	90
Figure 102.	Evidence management and System management plug-ins	92
Figure 103.	Rqa.Face.OslcKm library.....	94
Figure 104.	Implementation of the OSLC-KM for SE Suite by TRC.....	94
Figure 105.	OSLC-KM parsers and XSLT transformation files for Papyrus and Rhapsody	95
Figure 106.	Part of the Papyrus XSLT transformation to map requirements in the model to the OSLC-KM model instance	95
Figure 107.	ReqIF connector source code libraries.....	96
Figure 108.	PTC integrity connector source code libraries.....	97
Figure 109.	RAT plugin for Rhapsody source code.....	98
Figure 110.	Java plugin for Rhapsody.....	98
Figure 111.	DNG connector source code libraries	99
Figure 112.	Automatic translations connector source code libraries	100
Figure 113.	CHESS to SA plugins	101
Figure 114.	Source project structure of Tool Adapter plugins	102
Figure 115.	Tool Function Hierarchy	103
Figure 116.	Tool Runner Hierarchy	103
Figure 117.	FBK Tool Eclipse Command	104

List of Tables

Table 1.	Summary of the status of WP5 requirements.....	17
Table 2.	A Mapping table between CHES and Safety Architect (name, ID and description of CHES elements are preserved by default).....	59

Abbreviations and Definitions

API	Application Programming Interface
ARTA	AMASS Reference Tool Architecture
ASCE	Assurance and Safety Case Environment
ASIL	Automotive Safety Integrity Level
BPMN	Business Process Model and Notation
CACM	Common Assurance and Certification Metamodel
CHES	Composition with Guarantees for High-integrity Embedded Software Components Assembly
CDO	Connected Data Objects
CCL	Common Certification Language
CPS	Cyber-Physical Systems
CSV	Comma-Separated Values
DNG	DOORS Next Generation
DOORS	Dynamic Object-Oriented Requirements System
ECSEL	Electronic Components and Systems for European Leadership
EEF	Extended Editing Framework
EMF	Eclipse Model Framework
FMEA	Failure Mode and Effects Analysis
FMI	Functional Mock-up Interface
FMU	Functional Mock-up Unit
FMVEA	Failure Modes, Vulnerabilities and Effect Analysis
FTA	Fault Tree Analysis
GSN	Goal Structuring Notation
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
OCRA	Othello Contracts Refinement Analysis
OLEDDB	Object Linking and Embedding for Databases
OPENCOS	Open Platform for Evolutionary Certification of Safety-critical Systems
OSLC	Open Services for Lifecycle Collaboration
OSLC-KM	OSLC for Knowledge Management
PSA	Open Initiative for Next Generation of Probabilistic Safety Assessment
RAT	Requirements Authoring Tool
RM	Requirements Management
RMS	Requirements Management System
RQA	Requirements Quality Analyzer
RQS	Requirements Quality Suite
RSA	Rational Software Architect
RTF	Rich Text Format
SA	Safety Architect
SACM	Structured Assurance Case Metamodel
SafeCer	Safety Certification of Software-Intensive Systems with Reusable Components
SE Suite	Systems Engineering Suite by TRC (formerly known as RQS or Requirements Quality Suite)

SMV	Symbolic Model Verifier
SSE	Safety and Security Engineering
SQL	Structured Query Language
SRL	System Representation Language
SVN	Apache Subversion
SysML	Systems Modelling Language
TRC	The REUSE Company
TRL	Technology Readiness Level
UML	Unified Modelling Language
URL	Uniform Resource Locator
V&V	Verification and Validation
WP	Work Package
XMI	XML Metadata Interchange
XML	eXtensible Markup Language
xSAP	eXtended Safety Assessment Platform
XSLT	eXtensible Stylesheet Language Transformations

Executive Summary

The document is AMASS deliverable D5.6 - Prototype for seamless interoperability (c). It is the third and final output of task T5.3 - Implementation for Seamless Interoperability, and is based on the results of tasks T5.1 - Consolidation of Current Approaches for Seamless Interoperability and T5.2 - Conceptual Approach for Seamless Interoperability, as well as on the first and second outputs of T5.3 (D5.4 - Prototype for seamless interoperability (a) and D5.5 - Prototype for seamless interoperability (b)).

Task T5.3 develops a tooling framework to implement prototype support for seamless interoperability in CPS assurance and certification. T5.3 is being carried out iteratively, in close connection with the conceptual tasks (T5.2 and Tx.2 in the other technical WPs), and with validation results from the implementation being used to guide further refinement of the conceptual approach. The implementation is closely guided by the requirements of the case studies, which are used to evaluate the prototype.

The third prototype iteration extends the implementation of basic building blocks for the AMASS Core Prototype, which was a consolidation and integration of results from previous projects, and the new features implemented for the AMASS Prototype P1. More concretely, the Seamless Interoperability features of the AMASS Prototype P2 are:

- Access Management (already in Core Prototype)
- Data Management (already in Core Prototype)
- Evidence Management (already in Core Prototype)
- Tool Integration (already in Prototype P1)
- Collaborative Work (already in Prototype P1)
- Traceability Management (already in Prototype P1)

The developed tools for the Prototype P2 support the following use cases:

- Configure Access to Assurance Assets
- Log in the platform
- Characterise Artefact
- Link Artefact with External Tool
- Specify Artefact Lifecycle
- Evaluate Artefact
- Specify Process Information for Artefacts
- Specify Traceability between Assurance Assets
- Conduct Impact Analysis of Assurance Asset Change
- Characterise Toolchain
- Specify Tool Connection Information
- Concurrent Assurance Information Edition

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

D5.6 relates to other implementation-related AMASS deliverables:

- Installable AMASS Tool Platform for Prototype P2
- User manuals and installation instructions
- Source code description

In addition, D5.6 is related to the following AMASS deliverables:

- D2.1 (Business cases and high-level requirements) includes the requirements that have been implemented in D5.6.
- D2.4 (AMASS reference architecture (c)) presents the abstract architecture based on which D5.6 has been created.

- D2.8 (Integrated AMASS platform (c)) and D2.9 (AMASS platform validation) report the results from validating the implementation described in D5.6 and integrating it with the results from other implementation tasks.
- D5.1 (Baseline requirements for seamless interoperability) reviews the main background on seamless interoperability for AMASS and proposes a way forward. D5.6 corresponds to the realisation of this way forward as of September 2018.
- D5.4 (Prototype for seamless interoperability (a)) describes the first version of the Seamless Interoperability support in the AMASS Tool Platform, and D5.5 (Prototype for seamless interoperability (b)) the second version.
- D5.8 (Methodological guide for seamless interoperability (b)) presents the guidelines to effectively use the tool support reported in D5.6 for Seamless Interoperability in CPS assurance and certification.

Note: the sections modified with respect to D5.5 are marked with an asterisk in the headline, i.e. (*), and the new sections with two asterisks, i.e. (**). Some new sections contain modified (or non-modified) sections because the former has been added and the latter was already included in D5.5. In other words, the structure of the sections and subsections has been revised for D5.6.

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 OPENCOS [19] and SafeCer [23] approaches 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 [21]).
- 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 functionality). As Eclipse Foundation is part of the AMASS consortium, the PolarSys/Eclipse community (www.polarsys.org) is a strong candidate 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.

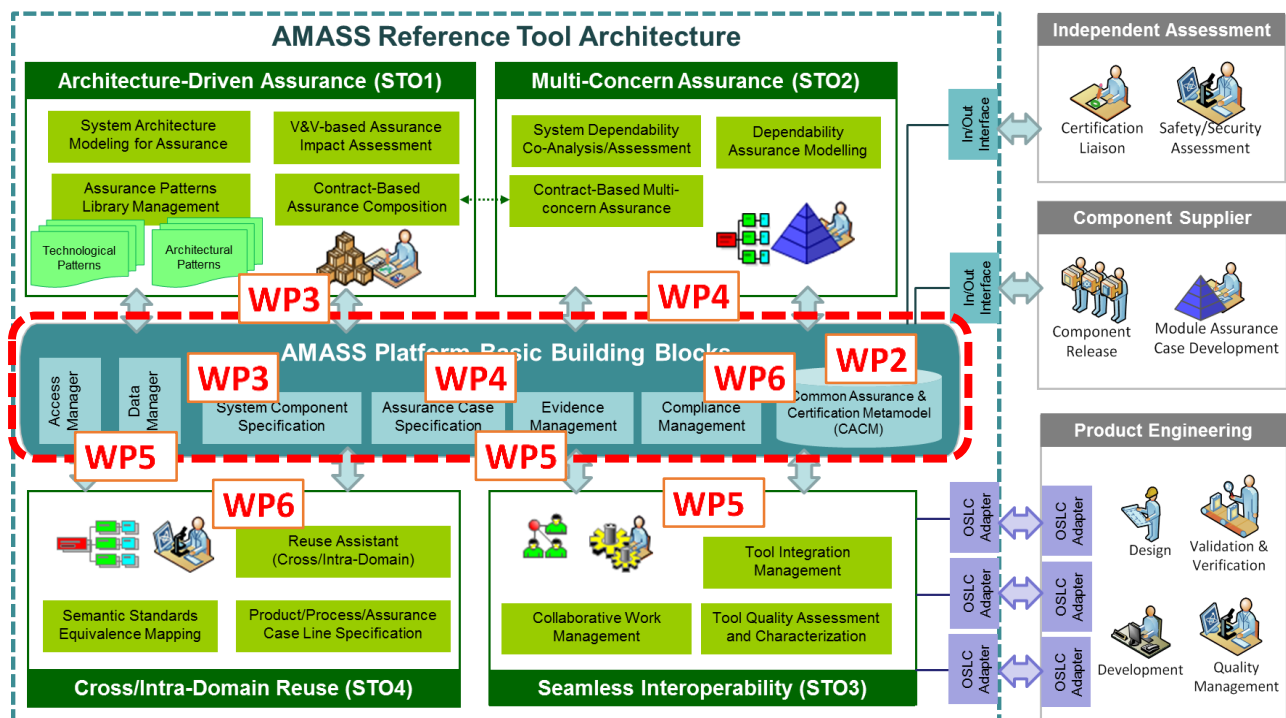


Figure 1. AMASS Building blocks

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 [2]), will be aligned, merged and consolidated at TRL4².
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 (Figure 1). Regarding seamless interoperability, in this second prototype, the specific building blocks will provide advanced functionalities regarding tool integration, collaborative work, and tool quality characterisation and assessment.
3. Finally, at the **third prototyping** iteration (Prototype P2), all AMASS building blocks will be integrated in a comprehensive toolset operating at TRL5. Functionalities specific for seamless interoperability developed for the second prototype will be enhanced and integrated with functionalities from 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 (see D1.1 [1]) using the tool-supported solutions.

As part of the Prototype Core, WP5 was responsible for consolidating the previous works on specification of evidence characteristics, handling of evidence evolution, and specification of evidence-related information (e.g. process information) in order to design and implement the basic building block called “Evidence Management” (Figure 1). In addition, WP5 was responsible for the implementation of the “Access Manager” and “Data Manager” basic building blocks. Nonetheless, the functionality of these latter blocks is used not only in WP5, but in all the WPs, e.g. for data storage and access (of system components, of assurance cases, of standards’ representations, etc.). For P1, WP5 has refined and extended the existing implementation with support for specific seamless interoperability based on the development of new functionality, and not only the integration of available tools. P2 completes the implementation for seamless interoperability of the AMASS tool platform by enhancing the available features and by supporting several further WP5 requirements for which no functionality had been provided yet.

This deliverable reports the **tool development results of the “Evidence Management”, “Access Manager”, “Data Manager”, “Tool Integration Management”, and “Collaborative Work Management” building blocks**. It presents in detail the technological design of the functionality implemented in the AMASS Tool Platform, the building blocks’ software architecture, the technology used, and some source code references. The design is based on the investigated state of the art and state of practice approaches presented in D5.1 [9], on the ARTA specification in D2.2³ [2], D2.3 [3], and D2.4 [4], and on the conceptual design presented in D5.2 [10] and D5.3 [11]. Gaps were identified and analysed to determine a way forward for seamless interoperability, enabling the formulation of requirements to achieve the interoperability

² 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

³ D2.2 and D2.3 are non-public descriptions of the ARTA. The deliverable that presents the final version (D2.4) is public.

vision of AMASS. This vision covers tool integration, collaborative work, and tool quality assessment and characterisation.

The rest of the deliverable presents the requirements implemented (Section 2) and describes the implementation performed (Section 3).

2. Implemented Functionality (*)

This section presents the scope of the implementation work reported in this deliverable and the implemented requirements.

2.1 Scope (*)

The scope for the current prototype for seamless interoperability is the provision of tools for: (1) access and data management; (2) specification and management of evidence-related assurance information, mostly artefact information; (3) traceability management; (4) tool integration, and; (5) collaborative work. The overall scope is highlighted in Figure 2, which shows the general functional overview of the AMASS Tool Platform as presented in D2.3 [3] and later evolved for D2.4 [4].

The *Platform Management* block includes generic functionality for security, permissions and profiles, data storage, visualization, and reporting, and including collaborative work. The *Evidence Management* block handles the full lifecycle of evidence artefacts and evidence chains. The *Seamless Interoperability* block manages the interoperability between the AMASS modules, as well as the connections with external tools. The *Assurance Traceability* block provides generic support for traceability management and impact analysis.

The next section presents the use cases that the above building blocks support in the scope of WP5.

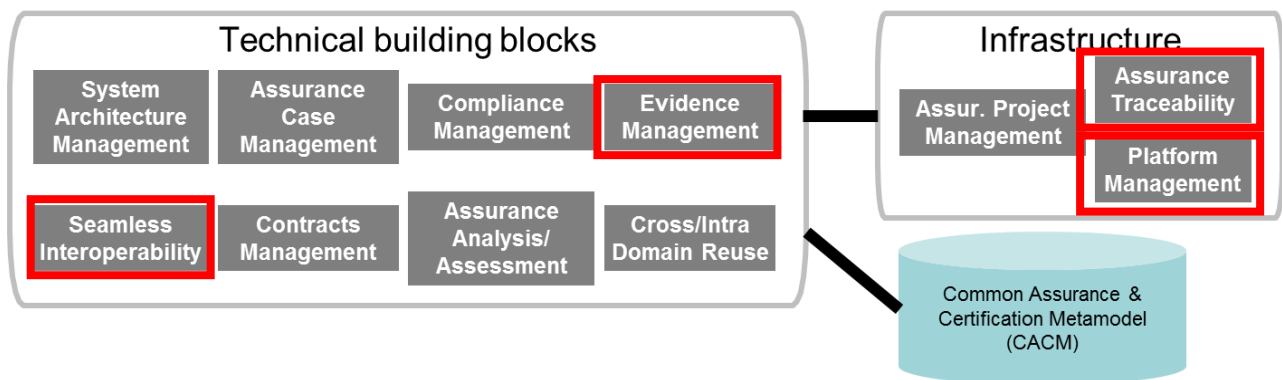


Figure 2. Functional decomposition for the AMASS platform

2.2 Implemented Requirements (*)

The implemented requirements correspond to 12 use cases specified in D2.4 [4]. The following subsections include a short description of how the implementation performed supports each use case, and the main tools and technologies supporting the use cases. Some use cases are supported by several tools and technologies. For example, there exist several means for tool integration in the AMASS Tool Platform. The use case functionality has been grouped as in D2.4. For example, Evidence Management functionality includes the Characterise Artefact, Link Artefact with External Tool, Specify Artefact Lifecycle, Evaluate Artefact, and Specify Process Information for Artefacts use cases.

Table 1 shows the requirements and the status of the requirements for the WP5.

Table 1. Summary of the status of WP5 requirements

Req. No.	Name	Description	Status	Tool	Partners
WP5_EM_001	Evidence characteristics specification	The AMASS Tool Platform shall allow an assurance engineer to specify the characteristics of assurance evidence.	Solved	AMASS Platform	TEC
WP5_EM_002	Evidence traceability	The AMASS Tool Platform shall allow an assurance engineer to specify relationships between evidence artefacts.	Solved	AMASS Platform	TEC, AMT, INT
WP5_EM_003	Evidence change impact analysis	When an evidence artefact is changed, the AMASS Tool Platform shall indicate how the change impacts other evidence artefacts.	Solved	AMASS Platform	TEC, AMT
WP5_EM_004	Evidence evaluation	The AMASS Tool Platform shall allow an assurance manager engineer to specify information about the results from evaluating an evidence artefact.	Solved	AMASS Platform	TEC, TRC
WP5_EM_005	Evidence information import	The AMASS Tool Platform shall be able to import information about evidence artefacts.	Solved	AMASS Platform	TEC, UC3, TRC
WP5_EM_006	Evidence information export	The AMASS Tool Platform shall be able to export information about evidence artefacts.	Solved	AMASS Platform	TEC, UC3, TRC
WP5_EM_007	Derivation of evidence characterization model	The AMASS Tool Platform shall derive an evidence characterisation model from the baseline of an assurance project.	Solved	AMASS Platform	MDH, TEC
WP5_EM_010	Evidence lifecycle information storage	The AMASS Tool Platform shall allow an assurance engineer to specify the events that have occurred during the lifecycle of an evidence artefact.	Solved	AMASS Platform	TEC
WP5_EM_011	Interactive evidence change impact analysis	The AMASS Tool Platform shall allow an assurance manager to indicate what evidence artefacts are impacted by the changes to a given evidence artefact.	Solved	AMASS Platform	TEC, AMT
WP5_EM_013	Link of evidence to other assets	The AMASS Tool Platform shall allow an assurance manager to link evidence artefacts with other assurance assets.	Solved	AMASS Platform	TEC, INT, AMT
WP5_EM_014	Evidence resource specification	The AMASS Tool Platform shall allow an assurance engineer to indicate the location of the resource that an evidence artefact represents in the system.	Solved	AMASS Platform	TEC
WP5_EM_016	Evidence report generation	The AMASS Tool Platform shall be able to automatically generate reports, checklists, and evidence for certification purposes.	Solved	AMASS Platform	TEC

Req. No.	Name	Description	Status	Tool	Partners
WP5_AM_003	User action log	The AMASS Tool Platform shall maintain a log with all the actions performed by the users.	Solved	AMASS Platform	TEC
WP5_DM_001	Multi-platform availability	The AMASS Tool Platform shall be accessible from desktop, Web, and cloud environments.	Solved	AMASS Platform	TEC
WP5_DM_002	Simultaneous data access	The AMASS Tool Platform shall allow users to access data simultaneously.	Solved	AMASS Platform	TEC, AMT, INT
WP5_DM_005	System artefact information storage	The AMASS Tool Platform shall be able to store information about any type of system artefact.	Solved	AMASS Platform	TEC
WP5_DM_006	Standard formats storage	The AMASS Tool Platform shall be able to store system artefacts represented in standard formats (OSLC RM, ReqIF, UML, Sims, FMI, FMU...).	Solved	AMASS Platform	TEC
WP5_DM_007	Data versioning	The AMASS Tool Platform shall support data versioning.	Solved	AMASS Platform	TEC
WP5_TI_001	Automatic data collection	The AMASS Tool Platform shall automatically collect data from external tools.	Solved	AMASS Platform	TEC, UC3, TRC, HON
WP5_TI_002	Automatic data export	The AMASS Tool Platform shall be able to automatically export data to external tools.	Solved	AMASS Platform	UC3, TRC
WP5_TI_007	Version management tools interoperability	The AMASS Tool Platform shall be able to interoperate with version management tools.	Solved	AMASS Platform	TEC
WP5_CW_003	Collaborative management of compliance with standards and of process assurance	The AMASS Tool Platform shall support the collaboration among systems engineers, assurance managers for management of compliance with standards and of process assurance.	Solved	AMASS Platform	TEC
WP5_CW_004	Collaborative re-certification needs & consequences analysis	The AMASS Tool Platform shall support the collaboration among assurance managers and assurance engineers for re-certification needs & consequences analysis.	Solved	AMASS Platform	TEC
WP5_CW_007	Collaborative assurance evidence management	The AMASS Tool Platform shall support the collaboration among assurance managers and systems engineers for assurance evidence management.	Solved	AMASS Platform	TEC
WP5_CW_008	Collaborative product reuse needs & consequences analysis	The AMASS Tool Platform shall support the collaboration among systems engineers and assurance managers for product reuse needs & consequences analysis.	Solved	AMASS Platform	TEC

Req. No.	Name	Description	Status	Tool	Partners
WP5_CW_009	Collaborative assurance case specification	The AMASS Tool Platform shall support the collaboration among assurance managers and assurance engineers for assurance case specification.	Solved	AMASS Platform	TEC
WP5_CW_010	Collaborative compliance needs specification	The AMASS Tool Platform shall support the collaboration among assurance managers for compliance needs specification.	Solved	AMASS Platform	TEC
WP5_CW_011	Collaborative assurance assessment	The AMASS Tool Platform shall support the collaboration among assurance managers, assurance engineers, and assurance assessors for assurance assessment.	Solved	AMASS Platform	TEC
WP5_CW_012	Collaborative compliance assessment	The AMASS Tool Platform shall support the collaboration among assurance managers, assurance engineers, and assurance assessors for compliance assessment.	Solved	AMASS Platform	TEC
WP5_TQ_001	Tool qualification information needs	The AMASS Tool Platform shall allow an assurance manager to specify the needs regarding qualification for the engineering tools used in a CPS' lifecycle.	Solved	AMASS Platform	MDH, TEC
WP5_TQ_002	Tool quality evidence management	The AMASS Tool Platform shall manage evidence of tool quality.	Solved	AMASS Platform	MDH, TEC
WP5_TQ_003	Tool quality information import	The AMASS Tool Platform shall be to import tool quality information such as tool qualification dossiers.	Solved	AMASS Platform	MDH, TEC
WP5_TQ_004	Tool quality needs indication	The AMASS Tool Platform should indicate the tool quality needs that need to be fulfilled in a given assurance project.	Solved	AMASS Platform	MDH, TEC
WP5_TQ_005	Tool quality requirements fulfilment	The AMASS Tool Platform should indicate the degree to which tool quality requirements for the engineering tools used in a CPS' lifecycle have been fulfilled.	Solved	AMASS Platform	MDH, TEC
WP5_EM_008	Visualization of chains of evidence	The AMASS Tool Platform shall display the chains of evidence to which an evidence artefact belongs.	Solved	AMASS Platform	AMT, INT
WP5_EM_015	Resource part selection	When indicating the location of the resource that an evidence artefact represents in the system, the AMASS Tool Platform shall allow an assurance engineer to select a part of the resource (e.g. a section inside a document or a component model file within a large system model).	Solved	AMASS Platform	AMT

Req. No.	Name	Description	Status	Tool	Partners
WP5_TI_003	Tool chain deployment support	The AMASS Tool Platform shall support the specification, configuration, and deployment of tool chains for CPS assurance and certification on a single environment.	Solved	AMASS Platform	UC3, TRC, FBK, HON
WP5_TI_005	System specification tools interoperability	The AMASS Tool Platform shall be able to interoperate with system specification tools.	Solved	AMASS Platform	UC3, TRC, FBK
WP5_TI_006	V&V tools interoperability	The AMASS Tool Platform shall be able to interoperate with V&V tools.	Solved	AMASS Platform	UC3, TRC, FBK, HON, UOM
WP5_TI_014	Client-server support	The AMASS Tool Platform shall support data and tool integration in client-server architectures.	Solved	AMASS Platform	HON
WP5_TI_017	Standards-based interoperability	The AMASS Tool Platform shall support standard mechanisms for tool interoperability.	Solved	AMASS Platform	UC3, TRC, FBK, HON
WP5_TI_018	Extended standard-based interoperability	The AMASS Tool Platform shall provide extended means to standard mechanisms for tool interoperability.	Solved	AMASS Platform	UC3, TRC, FBK, HON
WP5_EM_009	Suggestion of evidence traces	When specifying relationships for an evidence artefact, the AMASS Tool Platform shall suggest evidence artefacts to which the first evidence artefact might relate.	Solved	AMASS Platform	UC3, TRC
WP5_EM_012	Evidence trace verification	The AMASS Tool Platform shall analyse the quality of the relationships between evidence artefacts.	Solved	AMASS Platform	UC3, TRC
WP5_AM_001	User authentication	The AMASS Tool Platform shall require users to be authenticated for Platform access.	Solved	AMASS Platform	TEC
WP5_AM_002	User access	The AMASS Tool Platform shall provide users with different options for data access and for action permission.	Solved	AMASS Platform	TEC
WP5_AM_004	User profiles	The AMASS Tool Platform shall allow users to have different profiles for Platform access.	Solved	AMASS Platform	TEC
WP5_AM_005	Access rights groups	The AMASS Tool Platform shall allow users to belong to different access rights groups.	Solved	AMASS Platform	TEC
WP5_DM_003	Consistent data access	When users are accessing data simultaneously, the AMASS Tool Platform shall manage the possible conflicts.	Solved	AMASS Platform	AMT, TEC

Req. No.	Name	Description	Status	Tool	Partners
WP5_DM_004	Real-time data access feedback	The AMASS Tool Platform shall provide users with feedback about how data is being accessed by other users on real time.	Solved	AMASS Platform	AMT, TEC
WP5_DM_008	Secure data access	The AMASS Tool Platform shall provide a secure standard API for data access.	Cancelled	AMASS Platform	
WP5_TI_004	System analysis tools interoperability	The AMASS Tool Platform shall be able to interoperate with system analysis tools.	Solved	AMASS Platform	A4T
WP5_TI_008	Quality management tools interoperability	The AMASS Tool Platform shall be able to interoperate with quality management tools.	Solved	AMASS Platform	UC3, TRC
WP5_TI_009	MS Office applications interoperability	The AMASS Tool Platform shall be able to interoperate with MS Office applications (Word, Excel, Visio, etc.).	Solved	AMASS Platform	TEC, UC3, TRC
WP5_TI_010	Interoperability throughout CPS lifecycle	The AMASS Tool Platform shall be able to interoperate with some tool in all CPS lifecycle phases.	Solved	AMASS Platform	UC3, TRC
WP5_TI_011	Non-proprietary data exchange	The AMASS Tool Platform shall provide exchange data in non-proprietary formats.	Solved	AMASS Platform	UC3, TRC, HON
WP5_TI_012	Data entry effort	The AMASS Tool Platform shall allow users to create and enter data only once.	Solved	AMASS Platform	UC3, TRC, HON
WP5_TI_013	Continuous data management	The AMASS Tool Platform shall support continuous data analysis, verification, and integration.	Solved	AMASS Platform	AMT
WP5_TI_015	Service offer and discovery	The AMASS Tool Platform shall allow clients to ask for a server's services and to discover servers.	Solved	AMASS Platform	HON
WP5_TI_016	Performance monitoring	The AMASS Tool Platform shall allow continuous performance monitoring of the servers.	Solved	AMASS Platform	HON
WP5_CW_001	Collaborative system analysis	The AMASS Tool Platform shall support the collaboration among systems engineers, safety engineers, and security engineers for system analysis.	Solved	AMASS Platform	A4T
WP5_CW_002	Collaborative system specification	The AMASS Tool Platform shall support the collaboration among systems engineers, safety engineers, and security engineers for system modelling.	Solved	AMASS Platform	A4T
WP5_CW_005	Collaborative system V&V	The AMASS Tool Platform shall support the collaboration among systems engineers for system V&V.	Solved	AMASS Platform	HON, FBK

Req. No.	Name	Description	Status	Tool	Partners
WP5_CW_006	Collaborative model-based systems engineering	The AMASS Tool Platform shall support the collaboration among systems engineers, safety engineers, and security engineers for model-based systems engineering.	Solved	AMASS Platform	A4T
WP5_CW_013	Metrics & measurements reports	The AMASS Tool Platform shall manage metrics and measurements about collaborative work.	Solved	AMASS Platform	AMT

2.2.1 Evidence Management Functionality (**)

2.2.1.1 ‘Characterise Artefact’ with OpenCert

For artefact characterization (i.e. evidence artefact characterisation), the AMASS Tool Platform lets a user create artefact models and add artefact definitions to the model via a tree-view based editor (Figure 3). A user can later specify artefacts for the artefact definitions (Figure 4). For each artefact, a user can specify basic data such as name, description, version information, and preceding version. Examples of evidence artefact types include system plans, system analysis results, system specifications, and V&V results.

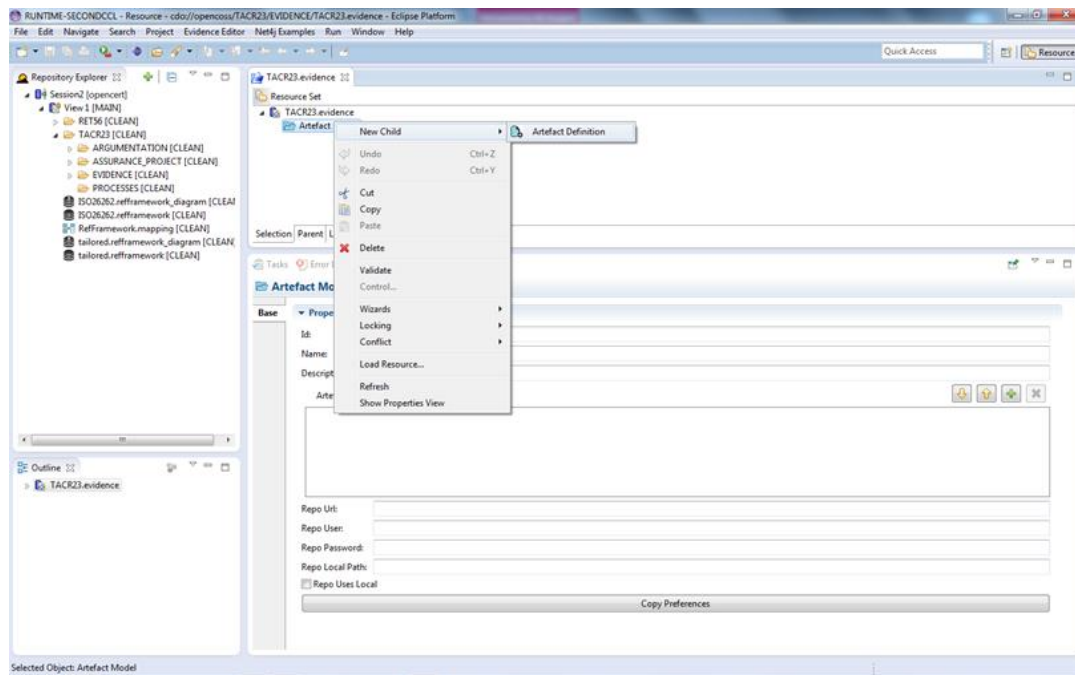
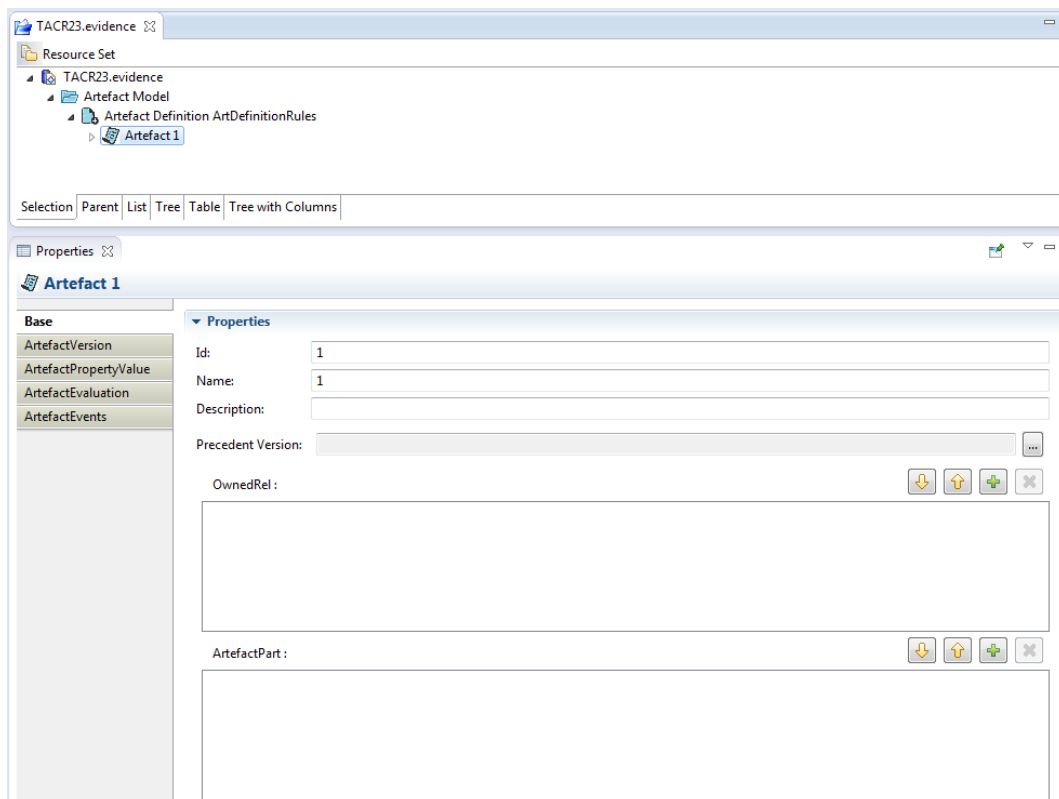


Figure 3. Artefact definition creation

2.2.1.2 ‘Link Artefact with External Tool’ with OpenCert

Artefacts can be linked to external tools in two main ways. First, a user can specify that the artefact repository for an assurance project corresponds to an SVN repository (Figure 5). Second, a user can add a resource to an artefact (Figure 6) and, in its properties (Figure 7), indicate the external location and format of the file that corresponds to the artefact.



The screenshot shows the 'TACR23.evidence' application window. The left sidebar displays a tree view with the following structure:

- Resource Set
 - TACR23.evidence
 - Artefact Model
 - Artefact Definition ArtDefinitionRules
 - Artefact 1

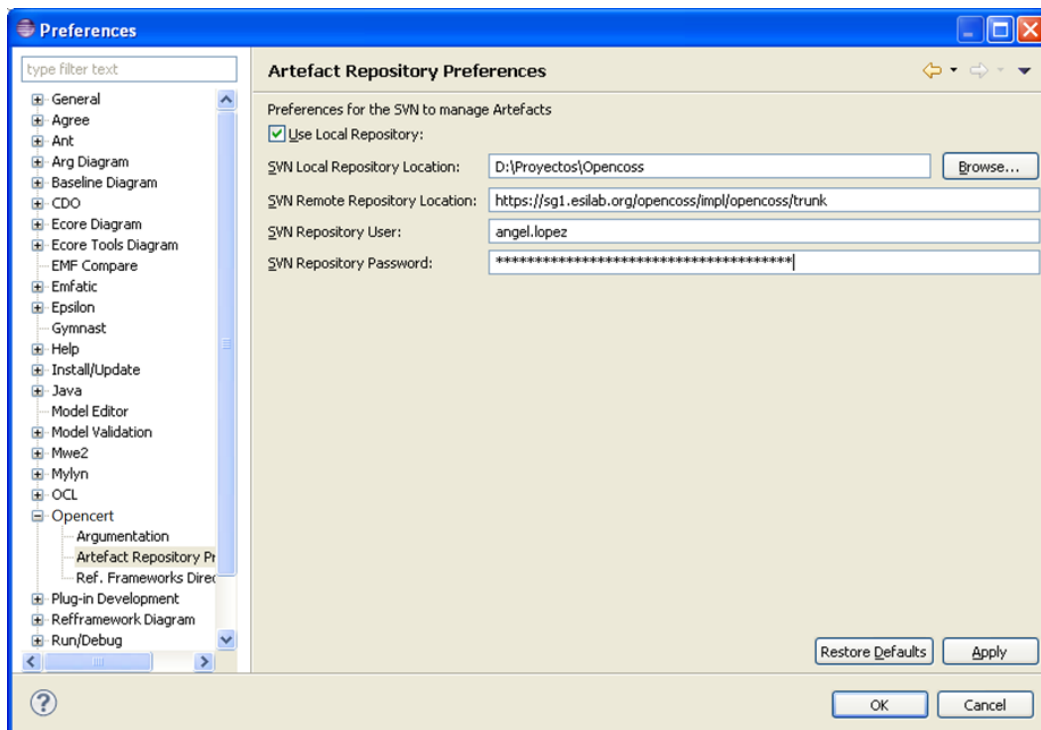
Below the tree view, there are tabs for 'Selection', 'Parent', 'List', 'Tree', 'Table', and 'Tree with Columns'. The 'Properties' tab is active, showing the 'Artefact 1' properties. The 'Base' section on the left lists the following properties:

- ArtefactVersion
- ArtefactPropertyValue
- ArtefactEvaluation
- ArtefactEvents

The 'Properties' section on the right contains the following fields:

- Id:** 1
- Name:** 1
- Description:** (empty text box)
- Precedent Version:** (empty text box with a dropdown arrow)
- OwnedRel:** (empty text box with icons: down arrow, up arrow, plus, minus)
- ArtefactPart:** (empty text box with icons: down arrow, up arrow, plus, minus)

Figure 4. Artefact data specification



The screenshot shows the 'Preferences' dialog box with the 'Artefact Repository Preferences' section selected. The 'type filter text' field is empty. The left sidebar lists the following categories:

- General
- Agree
- Ant
- Arg Diagram
- Baseline Diagram
- CDO
- Ecore Diagram
- Ecore Tools Diagram
- EMF Compare
- Emfatic
- Epsilon
- Gymnast
- Help
- Install/Update
- Java
- Model Editor
- Model Validation
- Mwe2
- Mylyn
- OCL
- Opencert
 - Argumentation
 - Artefact Repository Preferences
 - Ref. Frameworks Direct
- Plug-in Development
- Refreframework Diagram
- Run/Debug

The 'Artefact Repository Preferences' section contains the following fields:

- Use Local Repository:** ☒
- SVN Local Repository Location:** D:\Proyectos\Opencoss (with a 'Browse...' button)
- SVN Remote Repository Location:** https://sg1.esilab.org/opencoss/impl/opencoss/trunk
- SVN Repository User:** angel.lopez
- SVN Repository Password:** (password field with asterisks)

At the bottom right, there are buttons for 'Restore Defaults', 'Apply', 'OK', and 'Cancel'.

Figure 5. Use of SVN repository as artefact repository

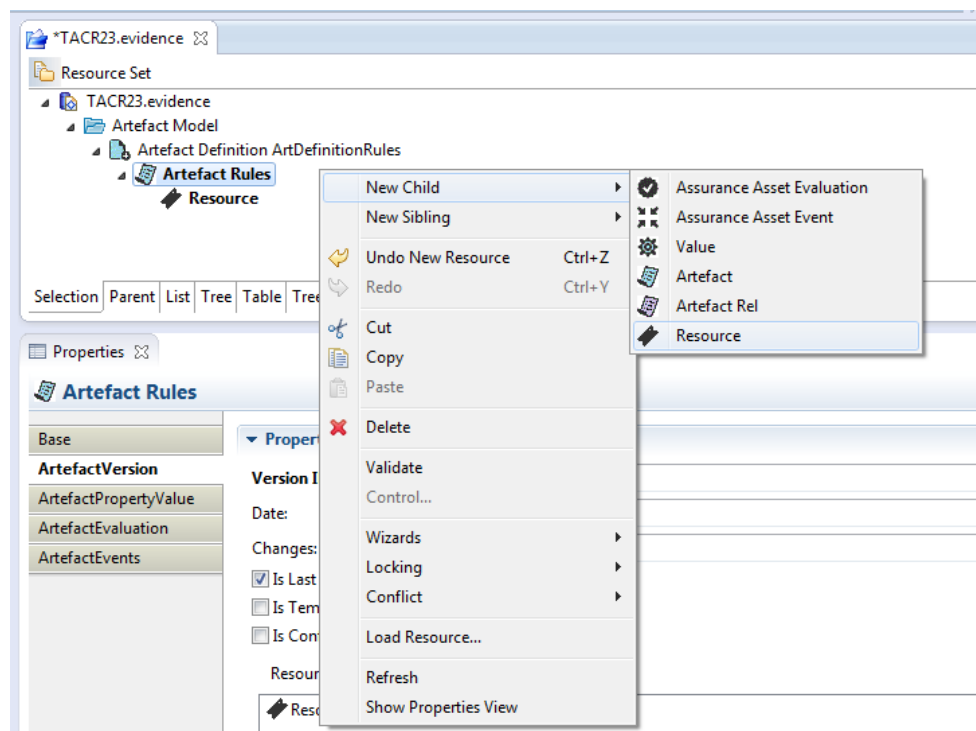


Figure 6. Resource specification for an artefact

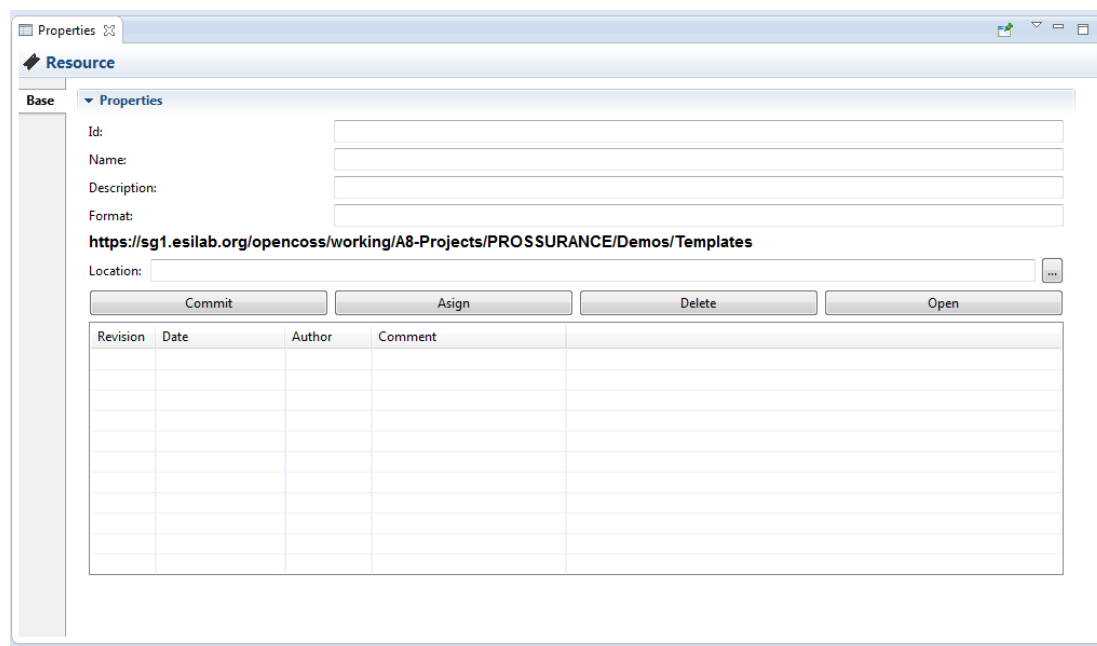
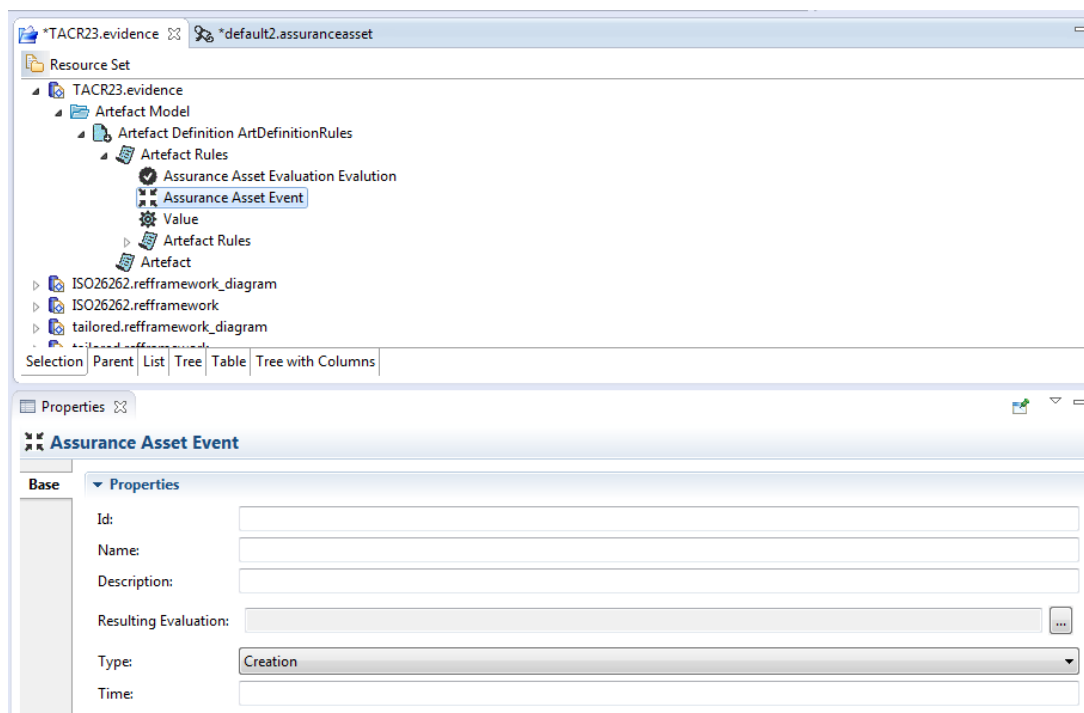


Figure 7. Resource properties

2.2.1.3 'Specify Artefact Lifecycle' with OpenCert

Once an artefact has been created, its lifecycle can be specified by adding events and specifying event data (Figure 8), such as the event type (creation, modification, evaluation, and revocation) and when the event happened.

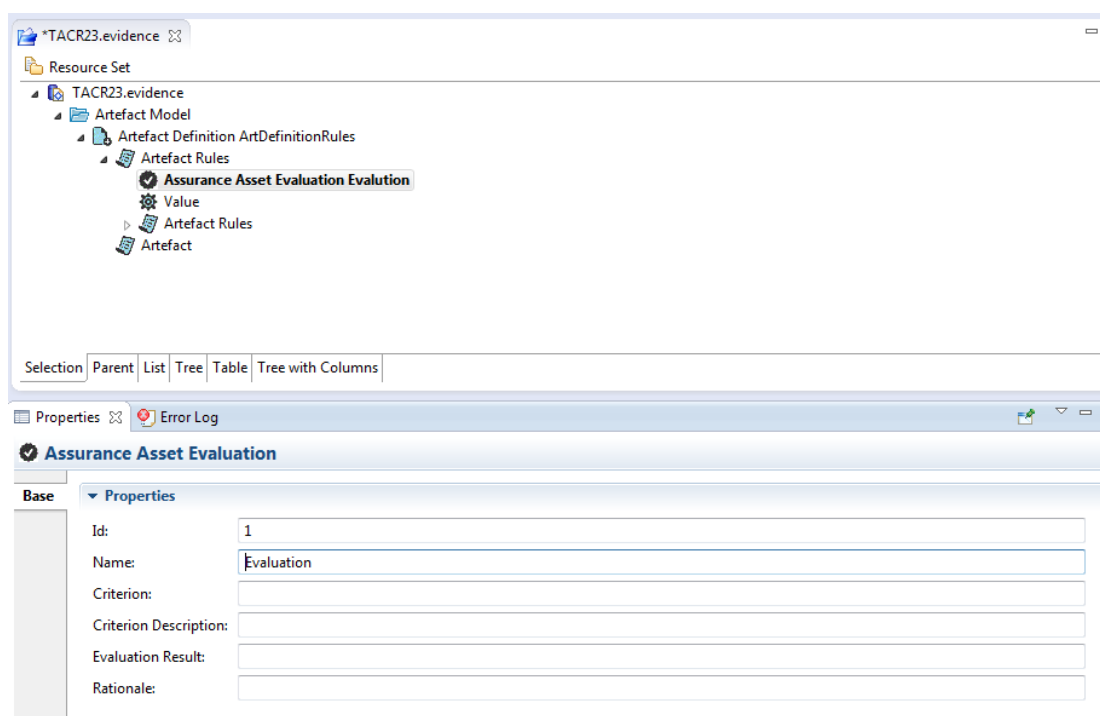


The screenshot shows the 'Assurance Asset Event' properties form. The left pane displays a tree view of the 'TACR23.evidence' resource set, with 'Assurance Asset Event' selected. The right pane shows the 'Properties' tab for this event. The form includes fields for 'Id', 'Name', 'Description', 'Resulting Evaluation' (with a dropdown arrow), 'Type' (set to 'Creation'), and 'Time'.

Figure 8. Artefact event properties

2.2.1.4 'Evaluate Artefact' with OpenCert

A user can add evaluations to artefacts, and can specify the evaluation criterion, the criterion description, the evaluation result, and its rationale, among other properties (Figure 9).

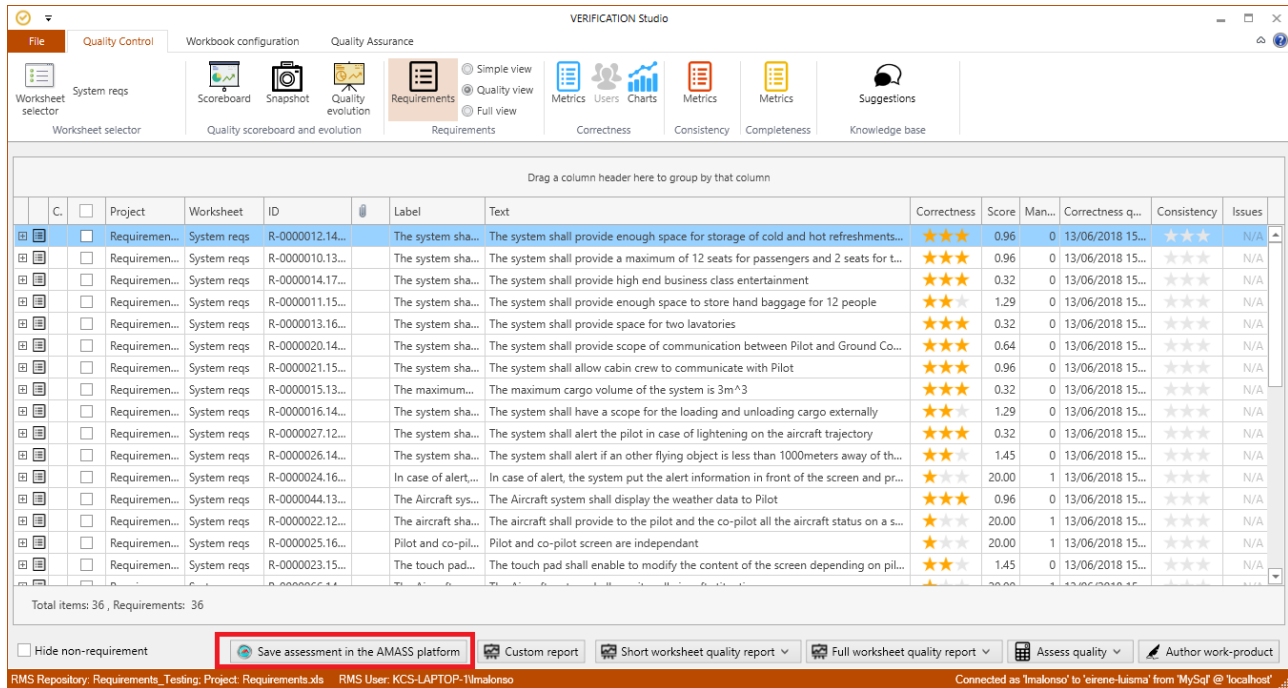


The screenshot shows the 'Assurance Asset Evaluation' properties form. The left pane displays a tree view of the 'TACR23.evidence' resource set, with 'Assurance Asset Evaluation Evaluation' selected. The right pane shows the 'Properties' tab for this evaluation. The form includes fields for 'Id' (set to '1'), 'Name' (set to 'Evaluation'), 'Criterion', 'Criterion Description', 'Evaluation Result', and 'Rationale'.

Figure 9. Artefact evaluation properties

2.2.1.5 'Evaluate Artefact' through Management of V&V evidence (**)

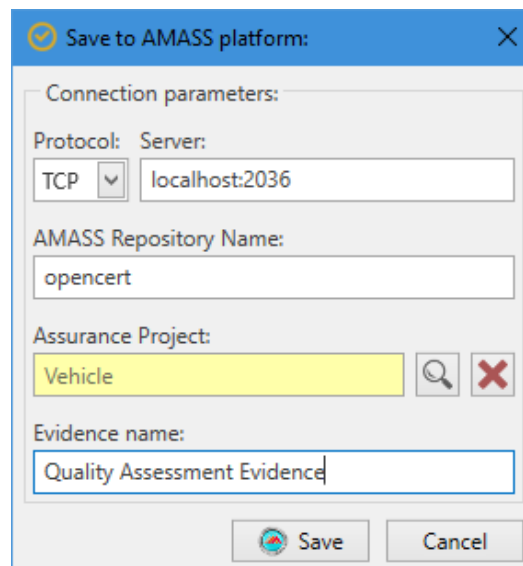
A user can store evidence created by external tools that can connect to the CDO repository, and more concretely from the VERIFICATION Studio (Figure 10; formerly known as Requirements Quality Analyzer or System Quality Analyzer).



C.	Project	Worksheet	ID	Label	Text	Correctness	Score	Man...	Correctness q...	Consistency	Issues
<input checked="" type="checkbox"/>	Requiremen...	System reqs	R-0000012.14...	The system sha...	The system shall provide enough space for storage of cold and hot refreshments...	★★★★	0.96	0	13/06/2018 15...	★★★★	N/A
<input checked="" type="checkbox"/>	Requiremen...	System reqs	R-0000010.13...	The system sha...	The system shall provide a maximum of 12 seats for passengers and 2 seats for t...	★★★★	0.96	0	13/06/2018 15...	★★★★	N/A
<input checked="" type="checkbox"/>	Requiremen...	System reqs	R-0000014.17...	The system sha...	The system shall provide high end business class entertainment	★★★★	0.32	0	13/06/2018 15...	★★★★	N/A
<input checked="" type="checkbox"/>	Requiremen...	System reqs	R-0000011.15...	The system sha...	The system shall provide enough space to store hand baggage for 12 people	★★★★	1.29	0	13/06/2018 15...	★★★★	N/A
<input checked="" type="checkbox"/>	Requiremen...	System reqs	R-0000013.16...	The system sha...	The system shall provide space for two lavatories	★★★★	0.32	0	13/06/2018 15...	★★★★	N/A
<input checked="" type="checkbox"/>	Requiremen...	System reqs	R-0000020.14...	The system sha...	The system shall provide scope of communication between Pilot and Ground Co...	★★★★	0.64	0	13/06/2018 15...	★★★★	N/A
<input checked="" type="checkbox"/>	Requiremen...	System reqs	R-0000021.15...	The system sha...	The system shall allow cabin crew to communicate with Pilot	★★★★	0.96	0	13/06/2018 15...	★★★★	N/A
<input checked="" type="checkbox"/>	Requiremen...	System reqs	R-0000015.13...	The maximum...	The maximum cargo volume of the system is 3m^3	★★★★	0.32	0	13/06/2018 15...	★★★★	N/A
<input checked="" type="checkbox"/>	Requiremen...	System reqs	R-0000016.14...	The system sha...	The system shall have a scope for the loading and unloading cargo externally	★★★★	1.29	0	13/06/2018 15...	★★★★	N/A
<input checked="" type="checkbox"/>	Requiremen...	System reqs	R-0000027.12...	The system sha...	The system shall alert the pilot in case of lightening on the aircraft trajectory	★★★★	0.32	0	13/06/2018 15...	★★★★	N/A
<input checked="" type="checkbox"/>	Requiremen...	System reqs	R-0000026.14...	The system sha...	The system shall alert if an other flying object is less than 1000meters away of th...	★★★★	1.45	0	13/06/2018 15...	★★★★	N/A
<input checked="" type="checkbox"/>	Requiremen...	System reqs	R-0000024.16...	In case of alert...	In case of alert, the system put the alert information in front of the screen and pr...	★★★★	20.00	1	13/06/2018 15...	★★★★	N/A
<input checked="" type="checkbox"/>	Requiremen...	System reqs	R-0000044.13...	The Aircraft sys...	The Aircraft system shall display the weather data to Pilot	★★★★	0.96	0	13/06/2018 15...	★★★★	N/A
<input checked="" type="checkbox"/>	Requiremen...	System reqs	R-0000022.12...	The aircraft sha...	The aircraft shall provide to the pilot and the co-pilot all the aircraft status on a s...	★★★★	20.00	1	13/06/2018 15...	★★★★	N/A
<input checked="" type="checkbox"/>	Requiremen...	System reqs	R-0000025.16...	Pilot and co-pil...	Pilot and co-pilot screen are independant	★★★★	20.00	1	13/06/2018 15...	★★★★	N/A
<input checked="" type="checkbox"/>	Requiremen...	System reqs	R-0000023.15...	The touch pad...	The touch pad shall enable to modify the content of the screen depending on pil...	★★★★	1.45	0	13/06/2018 15...	★★★★	N/A

Figure 10. VERIFICATION Studio displaying some requirements in a specification

After providing connection details and a name for the evidence (Figure 11), it is imported to the AMASS Tool Platform from VERIFICATION Studio (Figure 12). Each artefact contains all the assessments of a requirement and detailed information about them (Figure 13).



Save to AMASS platform:

Connection parameters:

Protocol:

AMASS Repository Name:

Assurance Project:

Evidence name:

Figure 11. AMASS Repository connection parameters

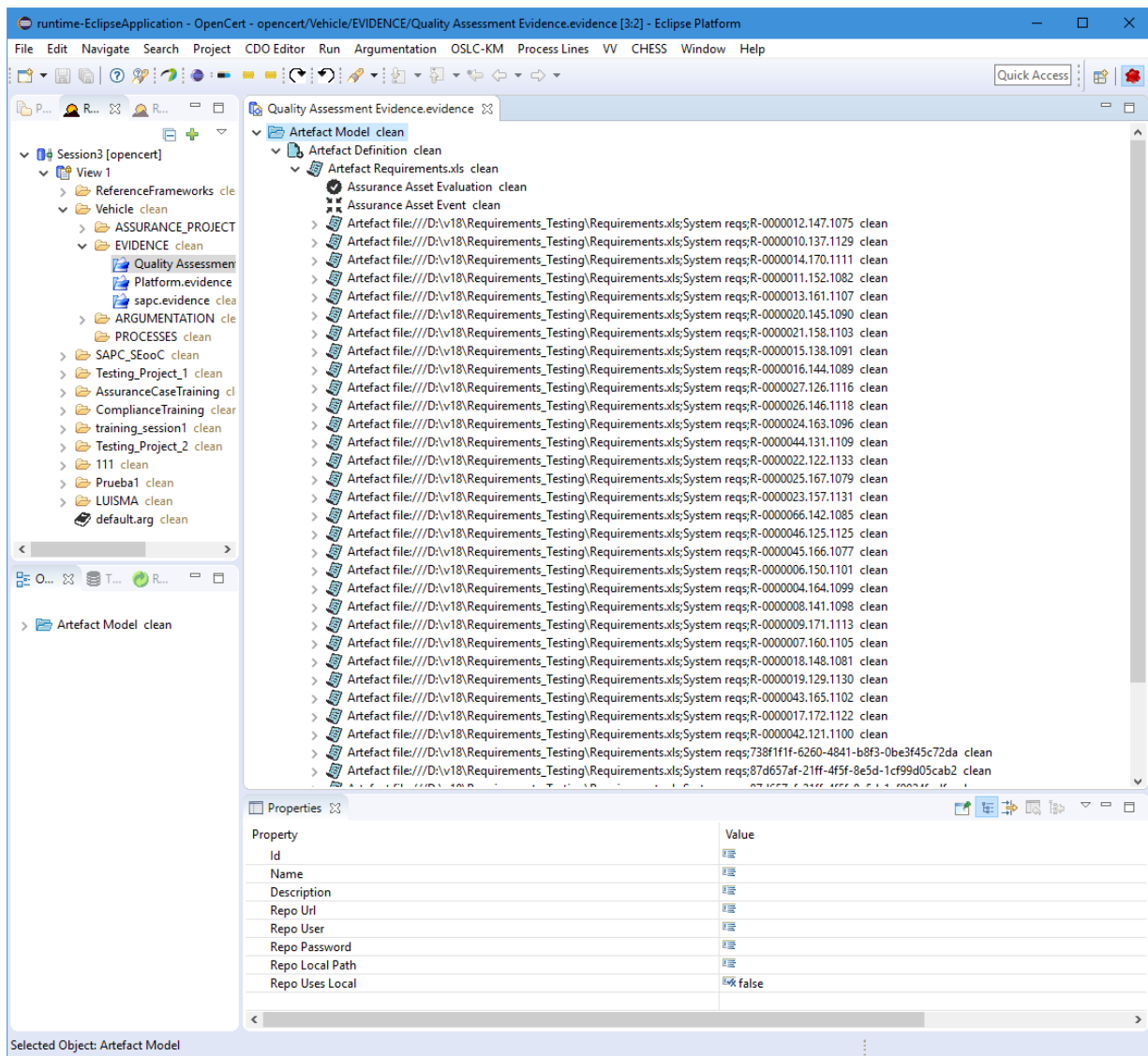


Figure 12. Importing evidence into the AMASS repository

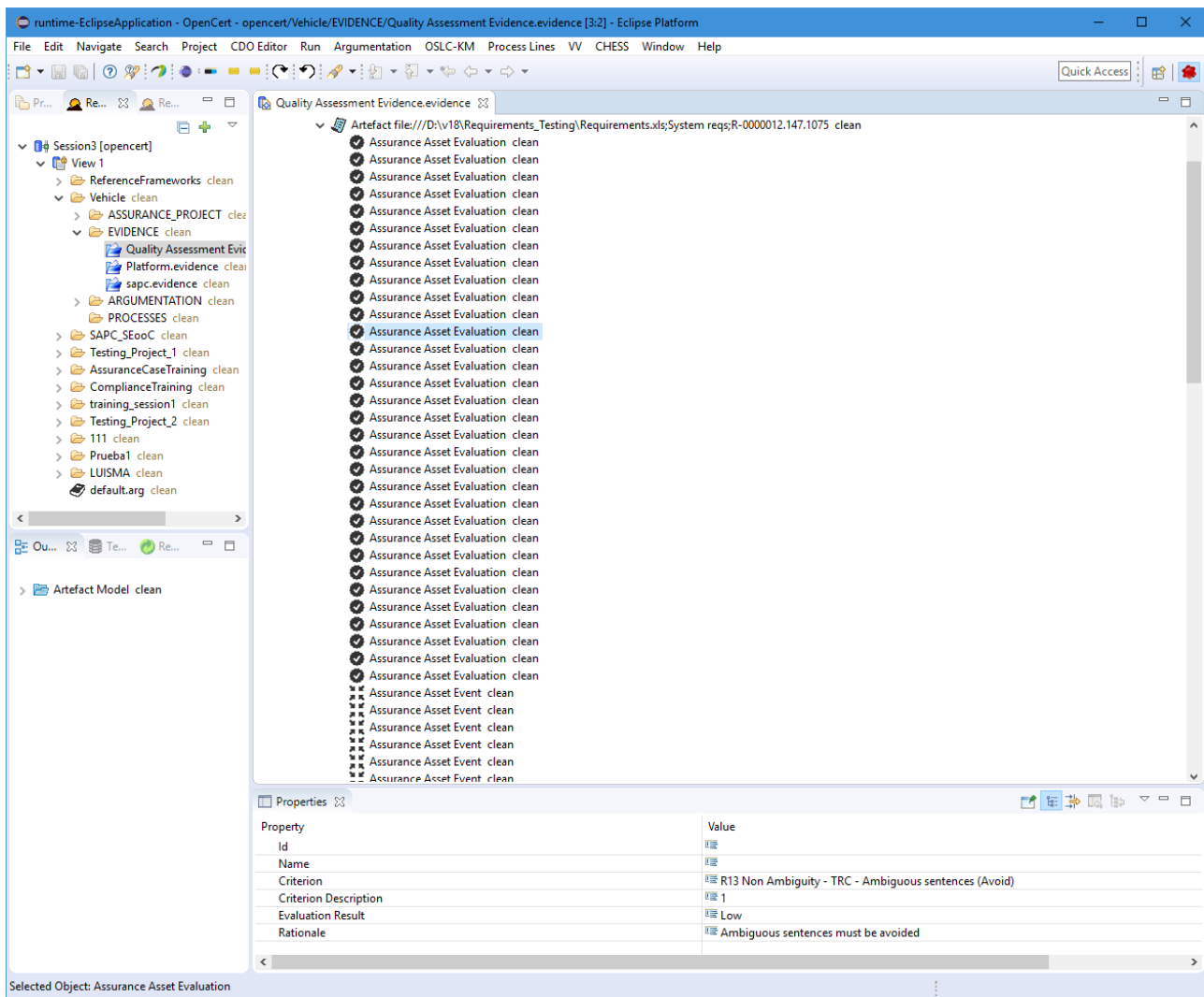


Figure 13. Detailed requirement information

A user can assess and manage the quality of all the work products of a Cyber-Physical System (CPS) by using TRC tools. This lets them demonstrate that quality procedures and processes have been executed to develop the CPS.

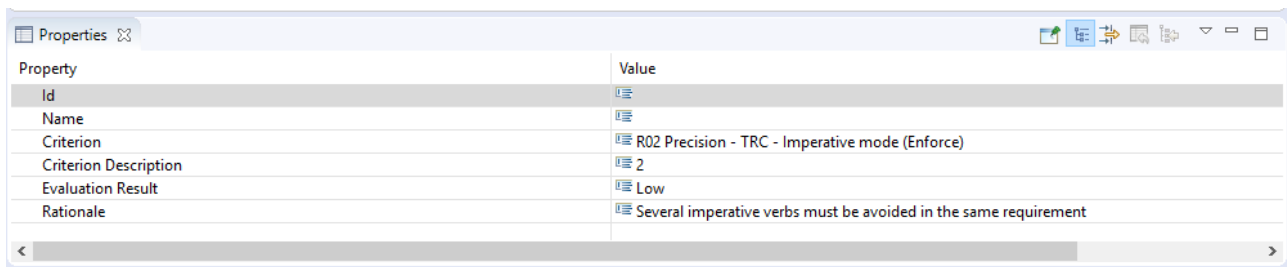
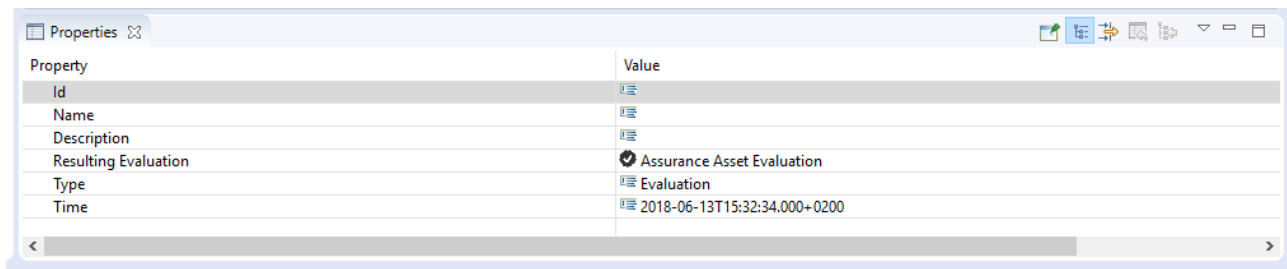


Figure 14. Details of the export of the assessment of a metric for a requirement



Property	Value
Id	
Name	
Description	
Resulting Evaluation	Assurance Asset Evaluation
Type	Evaluation
Time	2018-06-13T15:32:34.000+0200

Figure 15. Details of the export of the metadata of the assessment of a metric for a requirement

2.2.1.6 ‘Specify Process Information for Artefacts’ with OpenCert

Process-related artefact information is specified by means of process models (Figure 16). These models can contain information about activities, participants, people, tools, organizations, and techniques involved in the processes within an assurance project. Artefacts can later be associated to these elements. For example, ‘activity artefacts’ is a set of activity data (Figure 17) with which the input and output artefact of an activity can be specified.

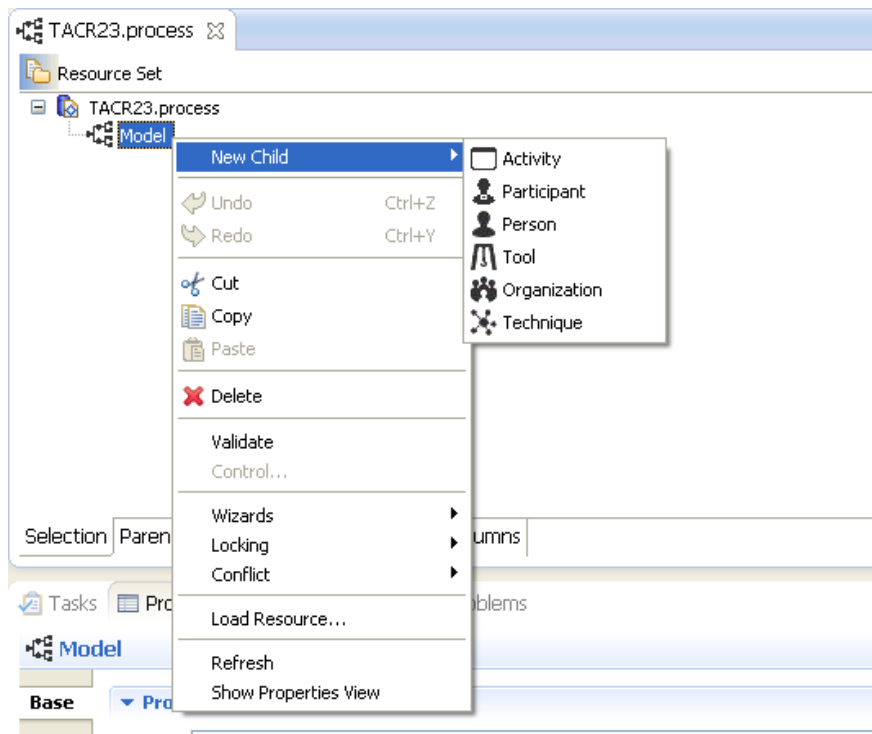


Figure 16. Process model

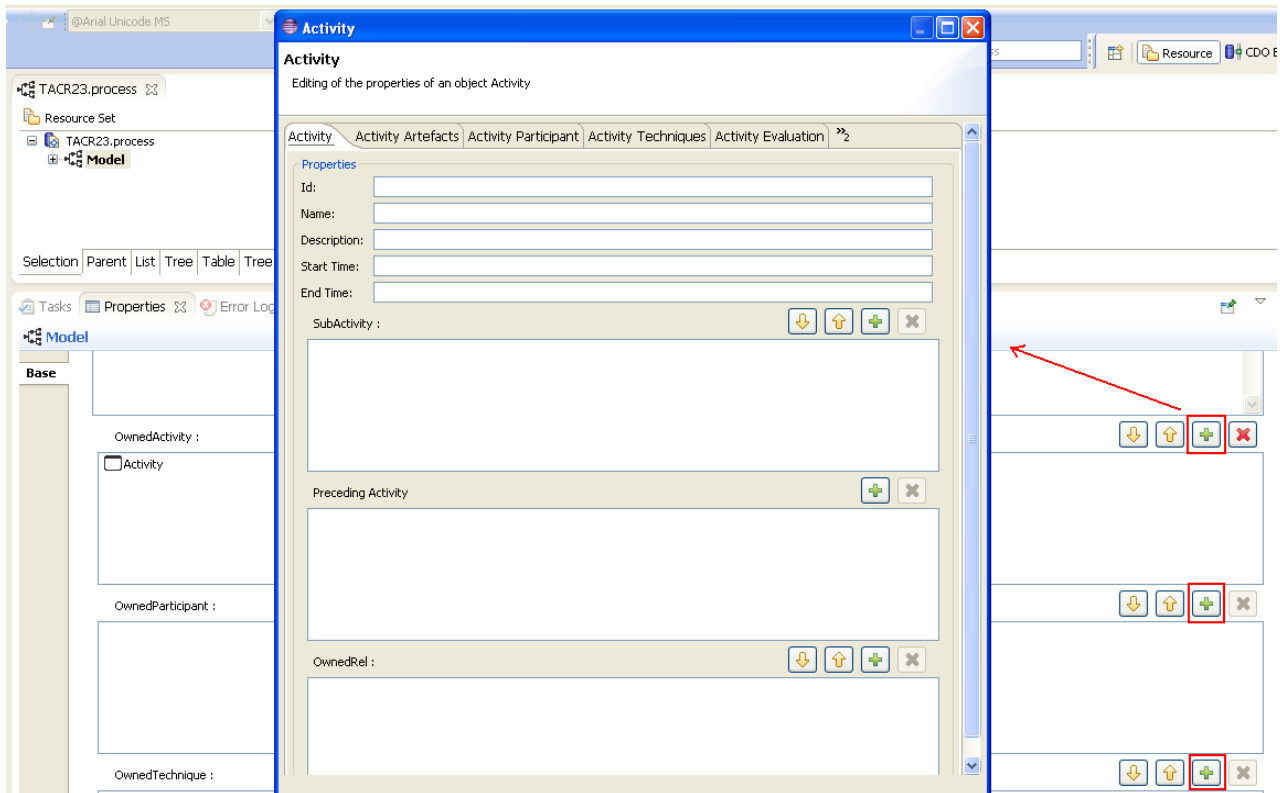


Figure 17. Activity data

2.2.2 Assurance Traceability Functionality (**)

2.2.2.1 'Specify Traceability between Assurance Assets' with OpenCert

OpenCert lets a user specify traceability between evidence artefacts (part of its functionality to characterise artefacts, see Section 2.2.1). More concretely, relationships can be created to specify artefact components with 'ArtefactPart' and any other type of relationship with 'OwnedRel'.

2.2.2.2 'Specify Traceability between Assurance Assets' with Capra (*)

Capra Eclipse project⁴ offers basic support for the creation, management and visualisation of trace links between resources within Eclipse. In WP5, Capra basic support has been extended to support references to resources external to the Eclipse environment (e.g. external files, requirements modelled with DOORS, etc.) and to references to Eclipse resources stored in CDO. The trace model was extended to include trace directionality, allowing users to express upstream-downstream relationships. Based on trace directionality, the software can calculate the impact of changes in upstream artefacts on downstream artefacts. Furthermore, the user interface was extended so that the creation of traces between internal artefacts and external artefacts is much easier. A new view is available which lets the user add items (URLs, files, objects) by dragging & dropping (Figure 18).

At the time of writing, the aforementioned extensions are under discussion with the Capra project team, in particular to evaluate their possible integration in the Capra project, as AMASS contribution.

⁴ <https://projects.eclipse.org/projects/modeling.capra>

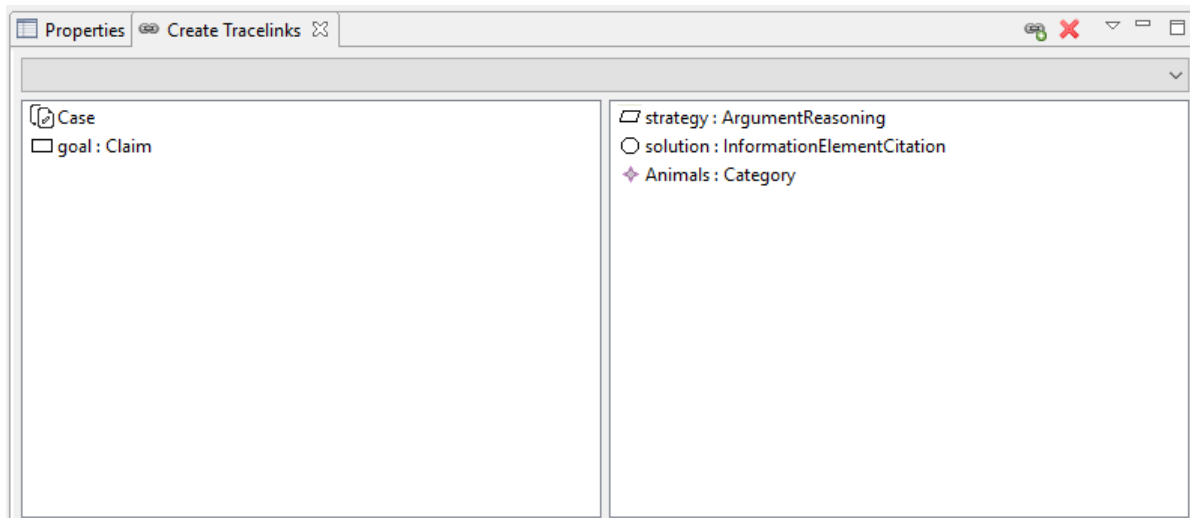


Figure 18. Advanced CAPRA trace creation view (drop sensitive)

Per WP3, the use of Capra was selected for the storage of the traceability links between system architectural entities such as components and contracts, and assurance related entities, like claims and evidences. In this work package, a specific support/user interface has been developed to assist the user creating traceability links, by letting them specify the kind of traceability links allowed by the system architecture (abstract) metamodel (see D3.3 [5] , section 3.2.2.5).

Figure 19 shows an example of the aforementioned support: by selecting a contract in the system architecture model ((through the Papyrus/CHESS editor), a dedicated tab (named OpenCert) is enabled in the properties view. The OpenCert tab allows to check the current assurance case entities already traced to the contract itself, and also allows to create new traceability links. For instance, a trace link between the selected contract and the *Goal1* claim available in the assurance case model, showed in the left part of the figure, can be created by using the *Claim table* in the OpenCert tab. The link to be created will be automatically stored in the Capra model, by using the Capra API's facilities. The possibility to retrieve existing traceability link associated to the selected architectural entity and the possibility to create new links, requires that the location of the Capra model must be known by the tool (it can be set in the CHESS preference page or by using some setting at the assurance project level, the definition of this part is ongoing).

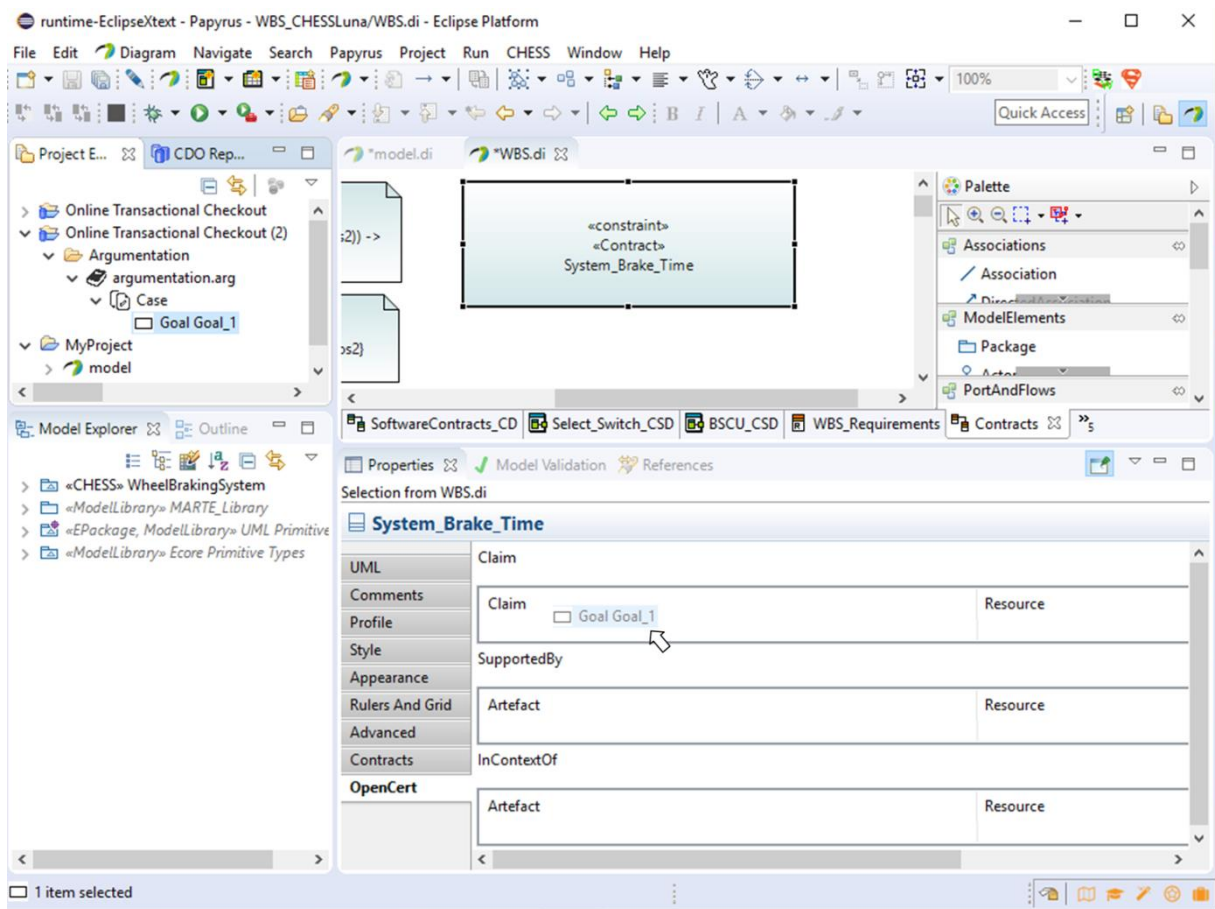


Figure 19. Tracing a claim to a contract

2.2.2.3 ‘Specify Traceability between Assurance Assets’ for Knowledge-Centric Automated Traceability (**)

Knowledge-Centric Automated Traceability has been implemented in the Traceability Studio tool [26] by TRC (Figure 20). This enables the definition and implementation of trace links between two sources of information, which can be of different types: V&V results, requirements, architecture data, design blocks, risk management information, etc. This functionality supports traceability management between heterogeneous and isolated work products, exploiting OSLC-KM as base technology for solving the “connectivity” challenge with different tools. Trace link suggestion is also supported.

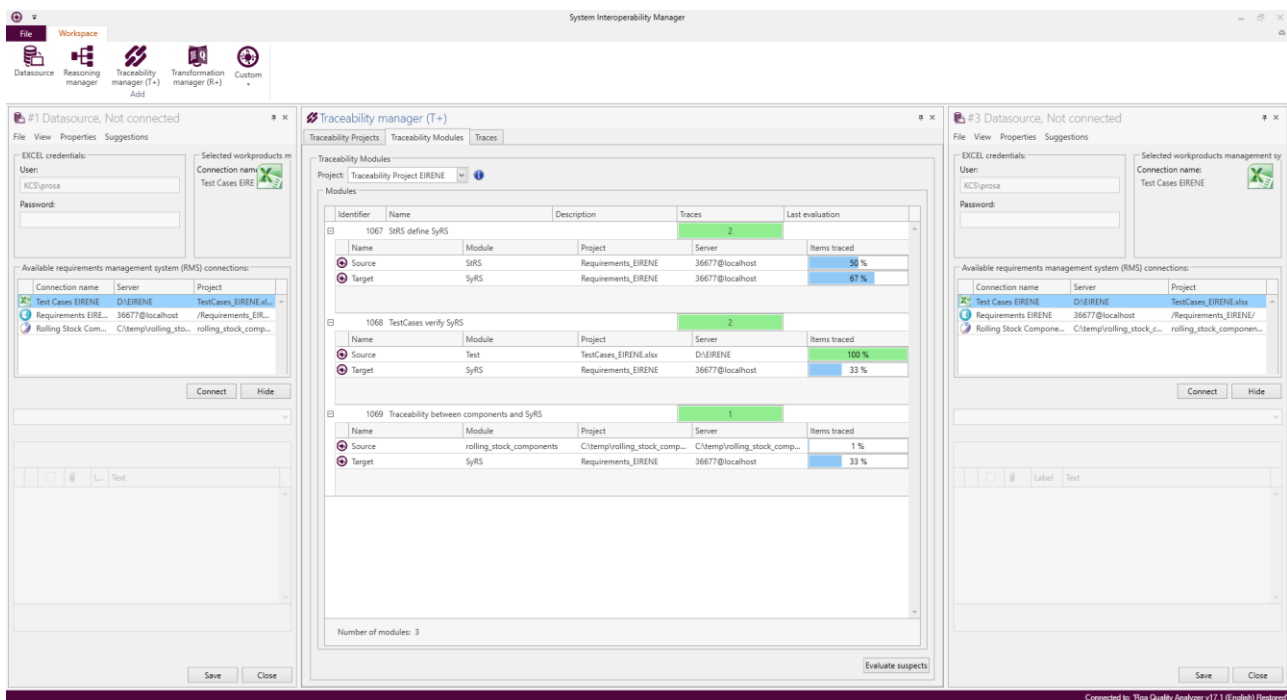


Figure 20. Knowledge-Centric Automated Traceability

2.2.2.4 'Conduct Impact Analysis of Assurance Asset Change' with OpenCert

When changes are made to artefacts and these changes result in modification events (Figure 21), users can determine the impact of such changes in other artefacts and accept or reject the changes (Figure 22).

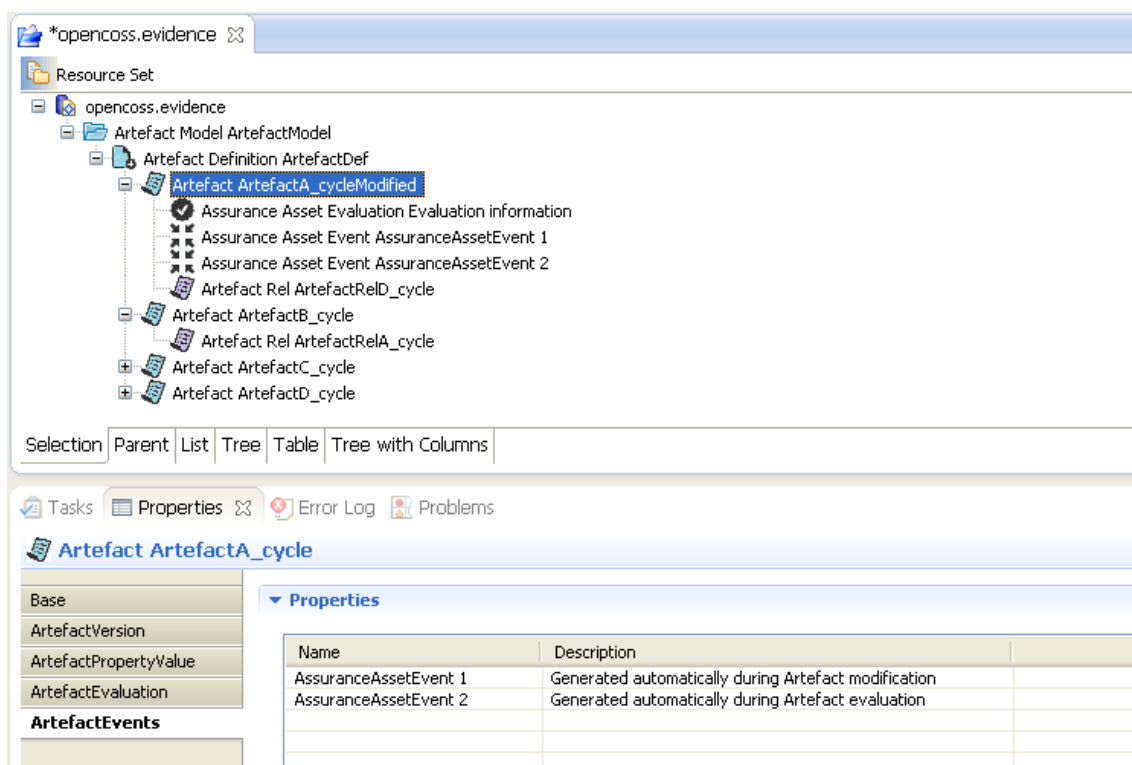


Figure 21. Modification event of an artefact

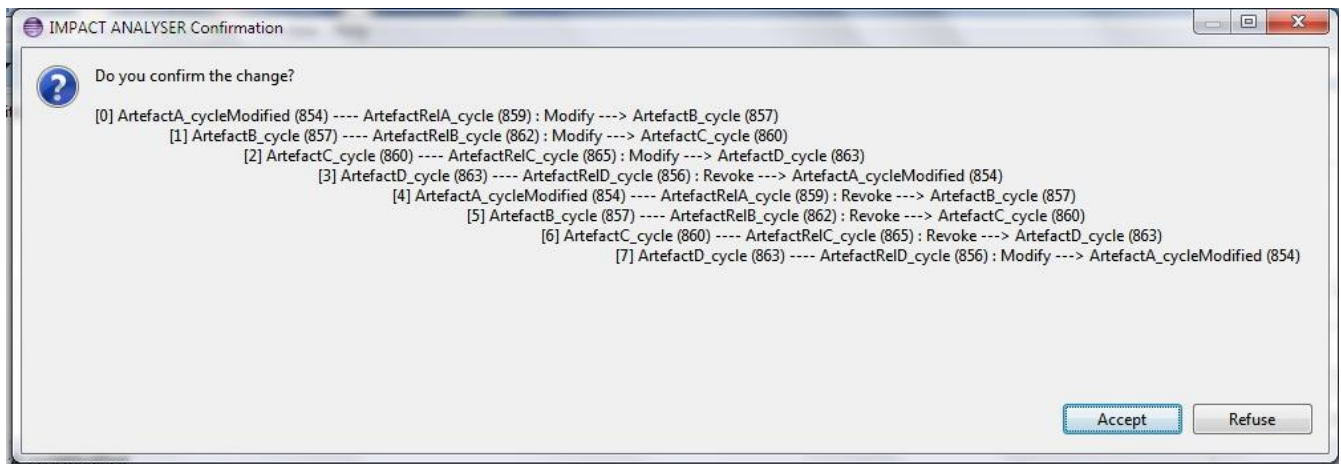


Figure 22. Impact analysis information

2.2.2.5 ‘Conduct Impact Analysis of Assurance Asset Change’ with Knowledge-Centric Automated Traceability (**)

As part of the functionality in Traceability Studio for Knowledge-Centric Automated Traceability, the tool supports impact analysis when changes occur in the traced elements (Figure 23).

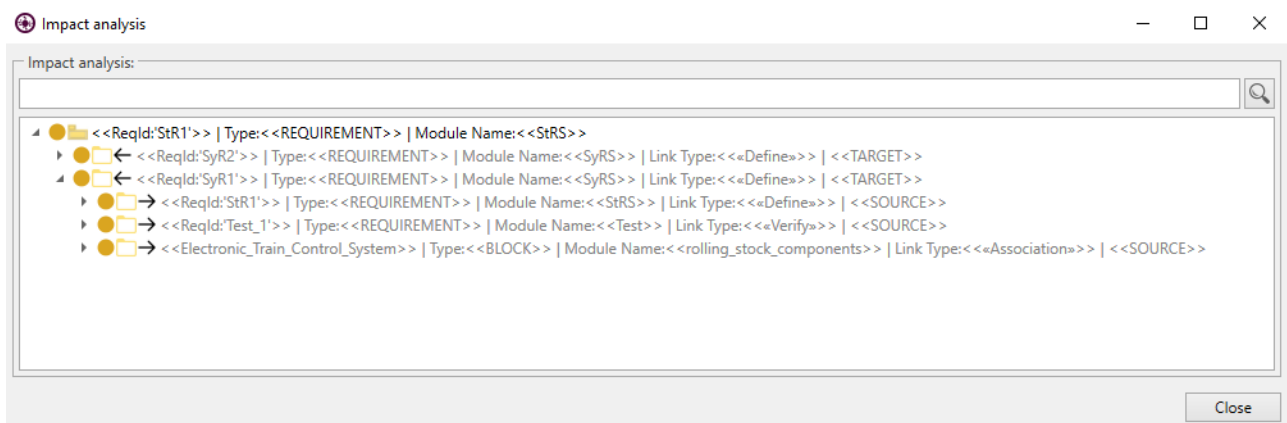


Figure 23. Impact analysis with Knowledge-Centric Automated Traceability

2.2.3 Tool integration Functionality (**)

2.2.3.1 ‘Characterise Toolchain’ with OpenCert (**)

Toolchains can be configured in OpenCert through the different connectors implemented in the tool, which are presented below. The configuration is not performed from a single, unified user interface, but the different technologies have their own user interface. This facilitates maintenance and extensibility of the connectors with different external tools in OpenCert.

2.2.3.2 ‘Characterise Toolchain’ with Papyrus (**)

Papyrus fosters seamless interoperability with different external tools and technologies via different means. Example of such interoperability means have been presented in D5.3 [11] and are listed below:

- Integration using Eclipse API with CDO repository to support object-level locking, changes to Papyrus’s editing behavior, etc., together with workbench-based server and user administration facilities.
- Integration using Eclipse API with file-based remote repositories, e.g. git, SVN, etc.

- Integration with the RSA tool using Papyrus additional component (specifically the RSA model importer plugin) to be able to import RSA models into Papyrus.
- Integration with the Rhapsody tool using Papyrus additional component (specifically the Rhapsody model importer plugin) to be able to import Rhapsody models into Papyrus.
- Capability to import and export Simulink and Autosar models via integrated model-to-model (M2M) transformations gateways.
- Connector to retrieve and import information from ReqIF specifications, Excel and CSV files, and by extension to be interoperable with DOORS tool using its ReqIF export component.
- Integration with the Eclipse debug framework to provide (co-)execution of UML models and alternative execution semantics (BPMN, Simulink, etc.), and enable control, observation and animation facilities over the executions.

Each connector mean provides its proper interface that can be activated or deactivated depending of the need. This allows to configure within a platform a set of different tool-chains for different engineering purpose: requirement management, system design, architecture optimization, process management, system analysis, safety analyses, security analyses, simulation, product lines, etc.

Papyrus tool, as a core component of the AMASS platform, takes also benefits of many tool connections developed within the platform and presented in the next sections.

2.2.3.3 ‘Characterise Toolchain’ with Systems Engineering Suite ()**

Systems Engineering Suite (SE Suite) is the new name of the already existing TRC toolsuite: Requirements Quality Suite (RQS).

Within SE Suite, the following set of toolchain connectors (Figure 24) have been added to or updated in WP5:

- OSLC-KM to create a universal language to communicate among different engineering tools in a semantic way.
- ReqIF: a new connector able to integrate requirement specifications from tools that can export to a ReqIF standard format. This is useful when no other connector is available in the SE Suite.
- PTC Integrity: a new connector able to retrieve requirements from PTC Integrity documents.
- IBM Rhapsody: a new connector able to understand models from this modelling tool. This has been integrated in a different way to the other connectors, in the form of a plugin that provides capabilities to assess requirements quality on-the-fly as the modeller describes them within the Rhapsody Model (Figure 25 and Figure 26).
- DOORS Next Generation: a new connector to be able to integrate with this IBM platform, which is the successor to the IBM Rational DOORS software, which was already integrated in the toolchain.

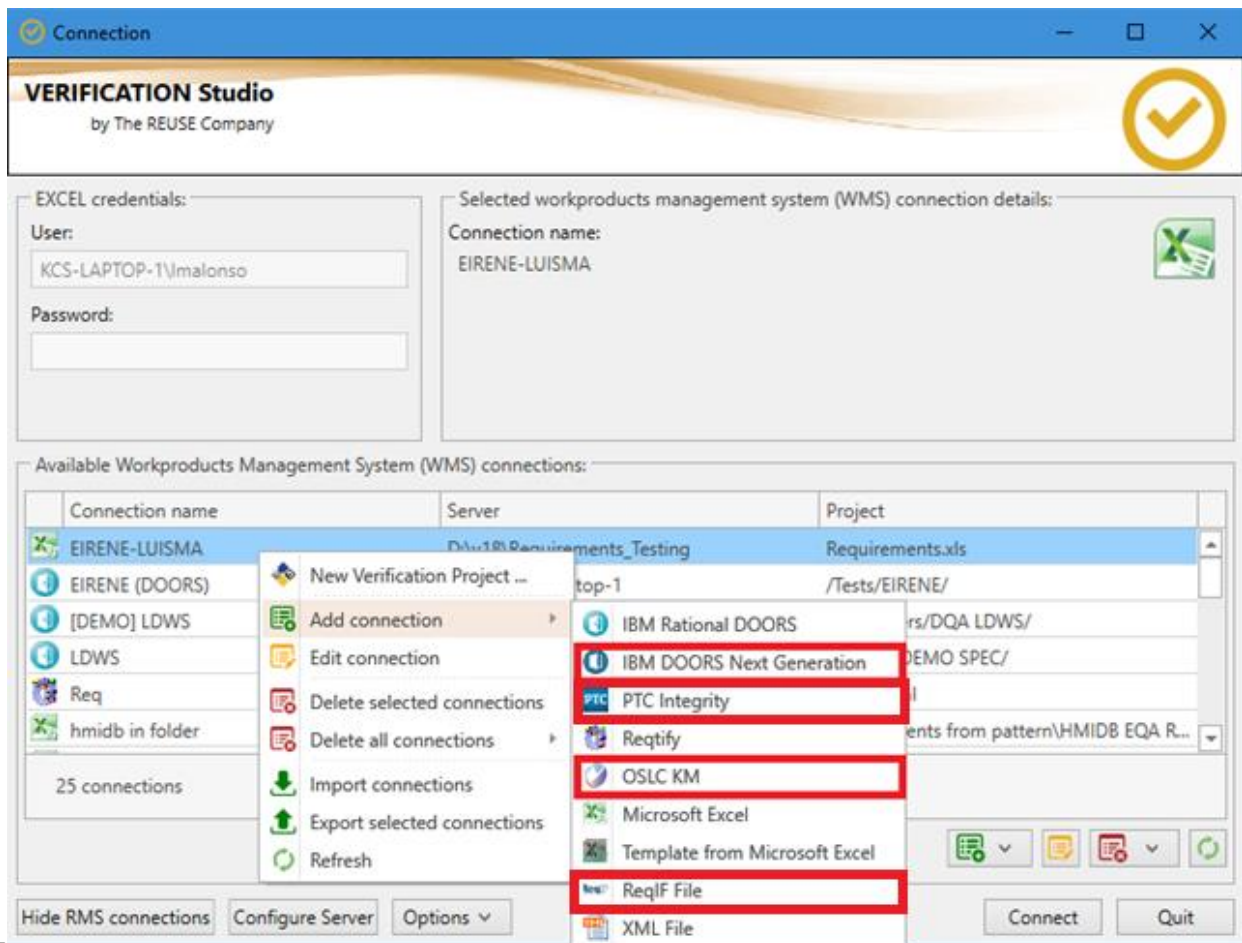


Figure 24. SE Suite new connectors

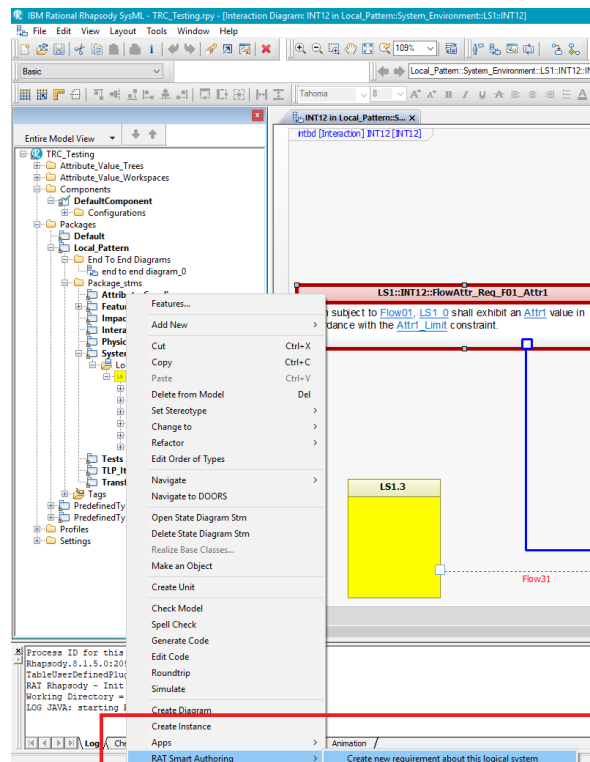


Figure 25. RAT plugin for Rhapsody, create new requirement with RAT

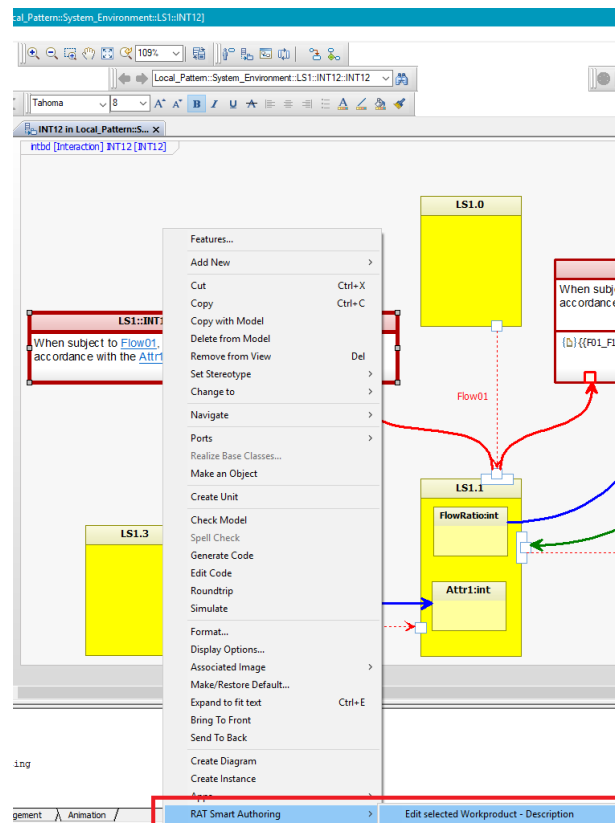


Figure 26. RAT plugin for Rhapsody, edit requirement description with RAT

The OSLC-KM connector will be described in depth in ‘Specify Tool Connection Information’ for OSLC-KM-based Integration.

2.2.3.4 ‘Specify Tool Connection Information’ with OpenCert

The default OpenCert support to specify tool connection information is presented in Section 2.2.1.2. The extended support developed for Prototypes P1 and P2 is presented in the following sections.

2.2.3.5 ‘Specify Tool Connection Information’ for OSLC-KM-based Integration (*)

From AMASS platform:

Artefact evidence can be gathered from external tools aiming at different specification or V&V targets. All of them can populate the artefact evidence database for an assurance project by implementing a producer of the OSLC-KM standard.

An example use case implemented for the AMASS platform is as follows.

In the AMASS Tool Platform menu bar, select the “OSLC-KM” menu, and the option “Import evidence model from file” (Figure 27), then select a Papyrus file (Figure 28).

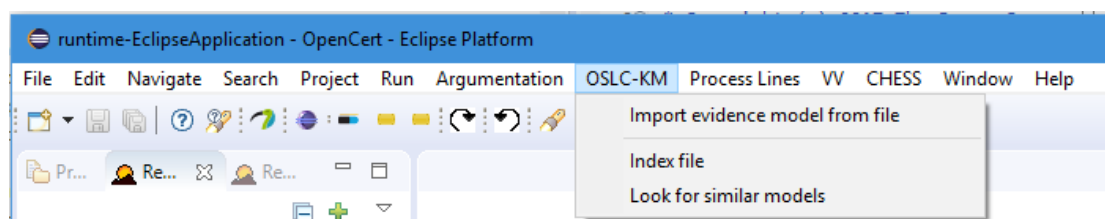


Figure 27. OSLC-KM Importing an Evidence Model from a model file

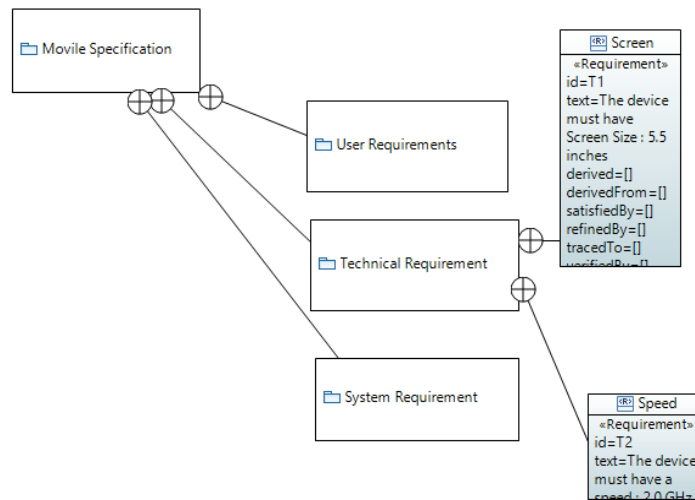


Figure 28. Fragment of a Papyrus model to be imported

A wizard will be shown to include all the parameters needed for this importation. This wizard comprises two steps.

The first one is designed to gather all the inputs related to the OSLC-KM file to be imported (Figure 29):

- The type of file to be parsed
- The file itself
- An additional transformation file to fine-tune the default OSLC-KM generation from the file type.

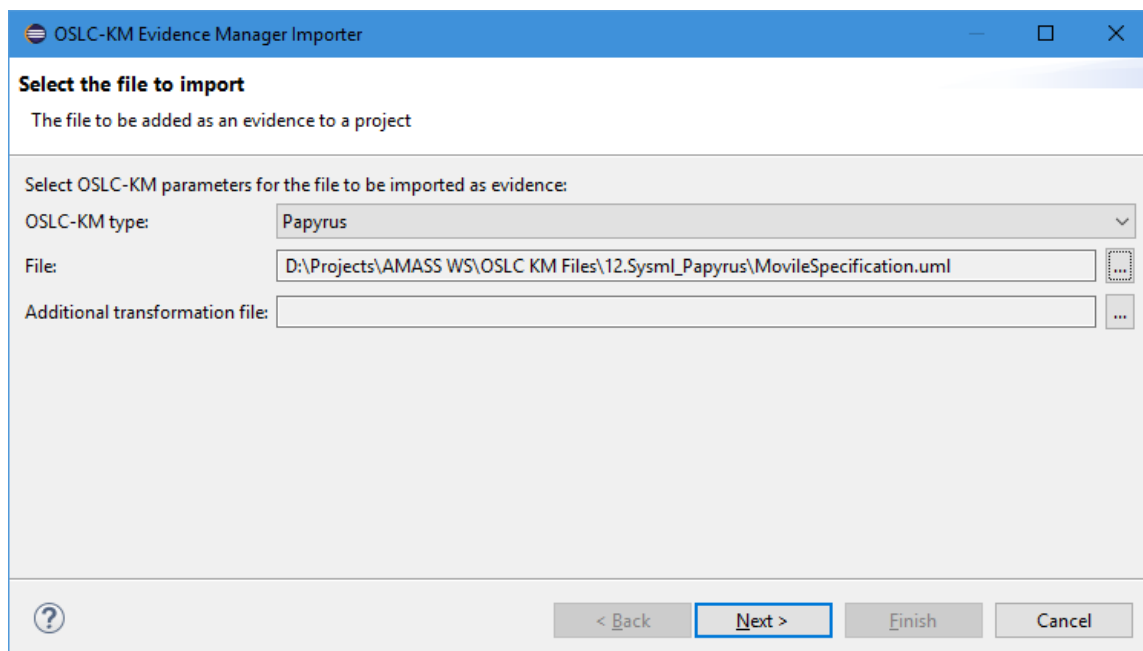
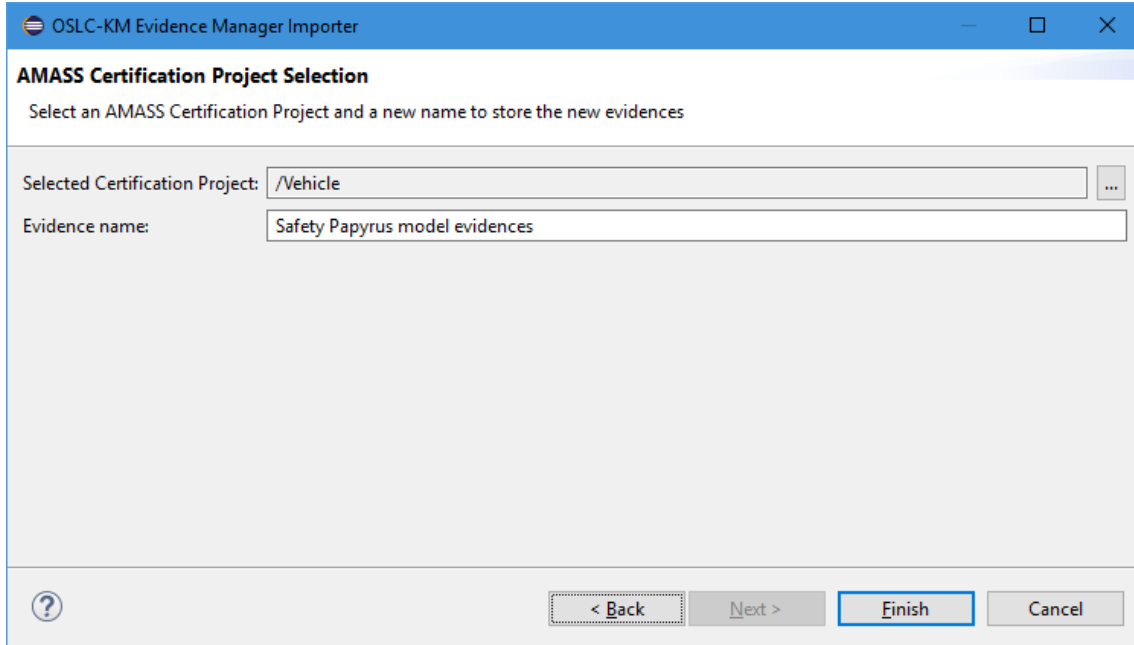


Figure 29. Step #1 of the OSLC-KM Evidence Manager Importer

The second one is designed to gather all the inputs about the storage of the new Evidence data (Figure 30): its Assurance Project and the name it will have.

As a result, its content is sent to a VERIFICATION Studio web service that works as an OSLC producer. The web method returns the OSLC-KM instance, then the AMASS platform loads the model by the Java

implementation of the OSLC-KM standards and maps its content to an Artefact Model inside the current assurance project (Figure 31).



OSLC-KM Evidence Manager Importer

AMASS Certification Project Selection

Select an AMASS Certification Project and a new name to store the new evidences

Selected Certification Project: /Vehicle

Evidence name: Safety Papyrus model evidences

< Back Next > Finish Cancel

Figure 30. Step #2 of the OSLC-KM Evidence Manager Importer

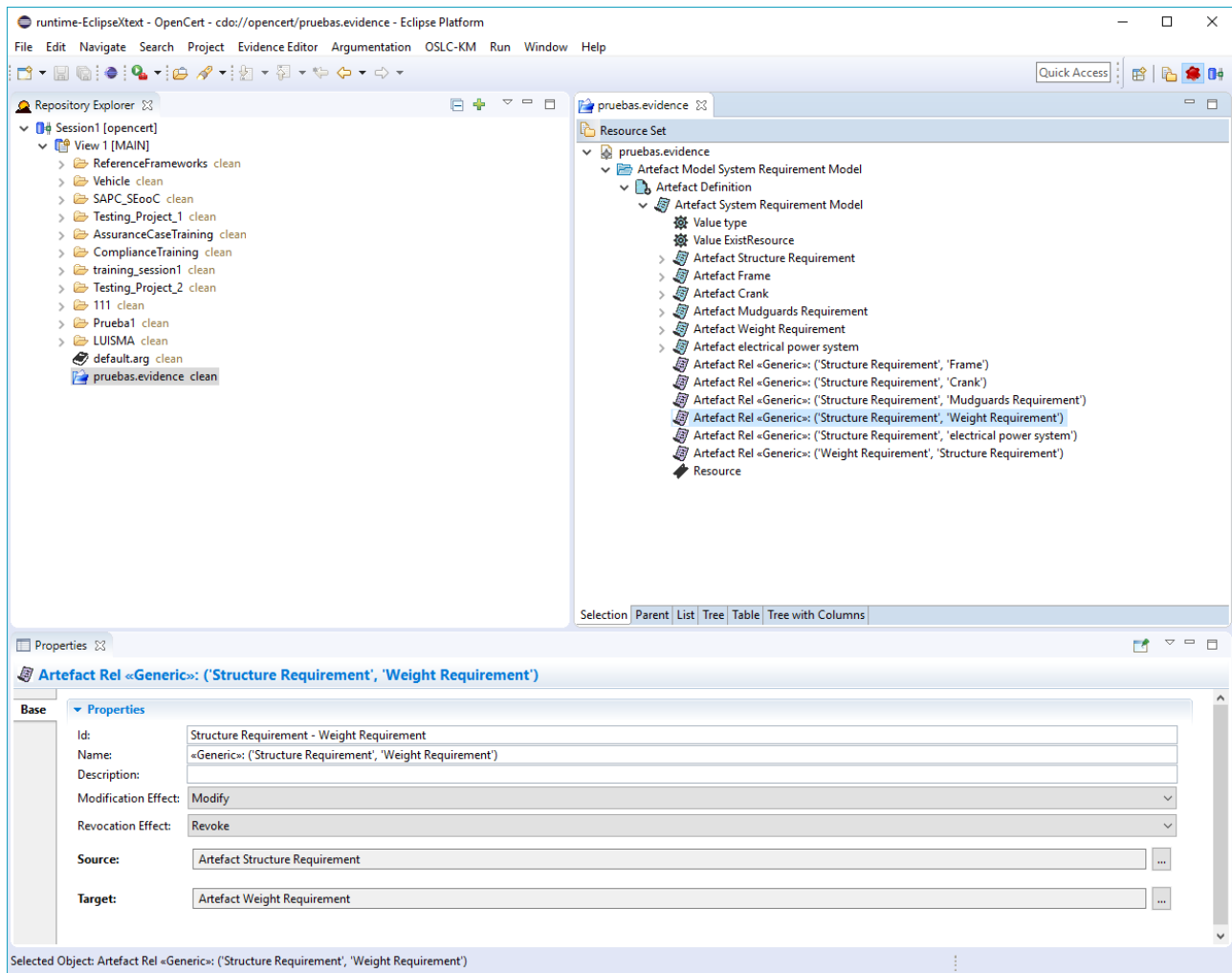


Figure 31. New evidence model from a Papyrus model

The location of the VERIFICATION Studio web service can be changed in the Window > Preferences Menu.

In the Preferences window there is an option with the name “OSLC-KM Preferences” where the URL of this web service is configured (Figure 32).

This web service is publicly available at the following URL:

<http://authoring.reusecompany.com:9999/OSlcKmService.svc/GetSrlFromContent>

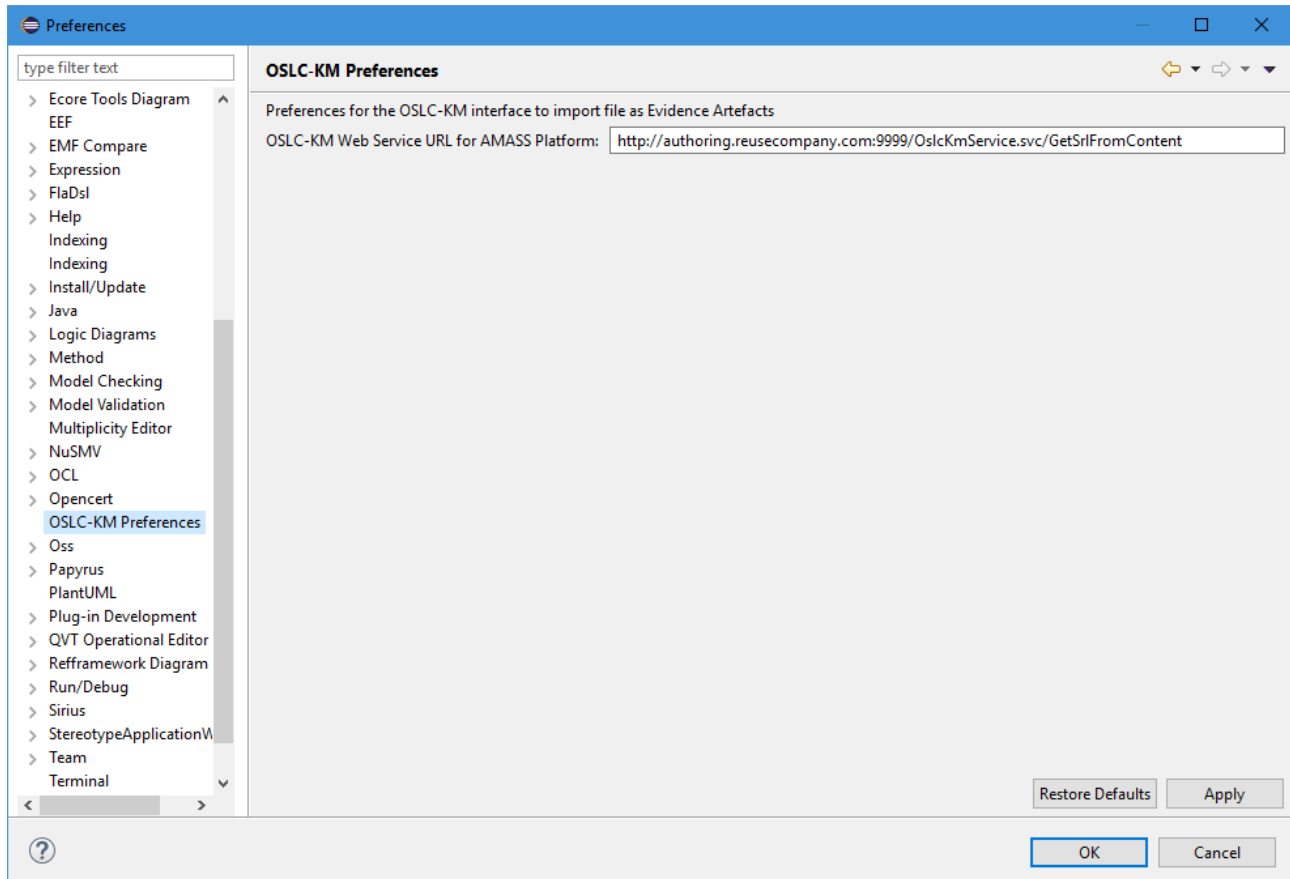


Figure 32. OSLC-KM Preferences. Web Service URL

This web service can transform files created from many different tools into OSLC-KM:

- Microsoft Excel
- Standard XMI (output from many UML tools)
- SysML from Rhapsody
- SysML from Papyrus
- SysML from Magic Draw
- SysML from Other tool providers
- Simulink
- ASCE
- FMI/FMU
- Pure Variance
- Metadata
- SQL
- XML

- SRL encoded in JSON format

From SE Suite tools

In the VERIFICATION Studio, the OSLC-KM standard can be the source to retrieve work-products from many different tools to assess its quality.

1. The tools expect these work-products either directly (the source tool is also provider of the OSLC-KM instance) via web service or via application API.
2. Other possibility, is to perform the parsing of the structured information from the source file to produce the instance of OSLC-KM, provided that the metamodel of the information is known, such as an XMI file coming out of a modelling tool.
3. The last possibility is like the previous one, but the input is not a file, but the output of an SQL query to a database.

This can be done in the VERIFICATION Studio connection window by selecting a new OSLC-KM connection (Figure 33) and then, in the new window, selecting a suitable source (Figure 34).

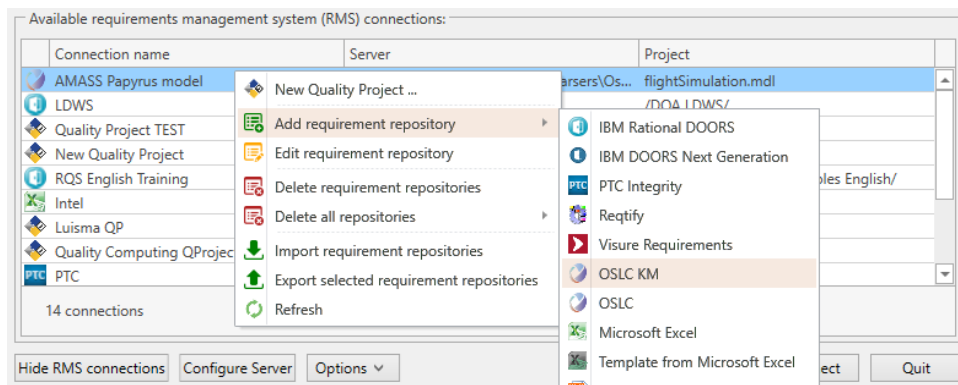


Figure 33. VERIFICATION Studio Connection Window

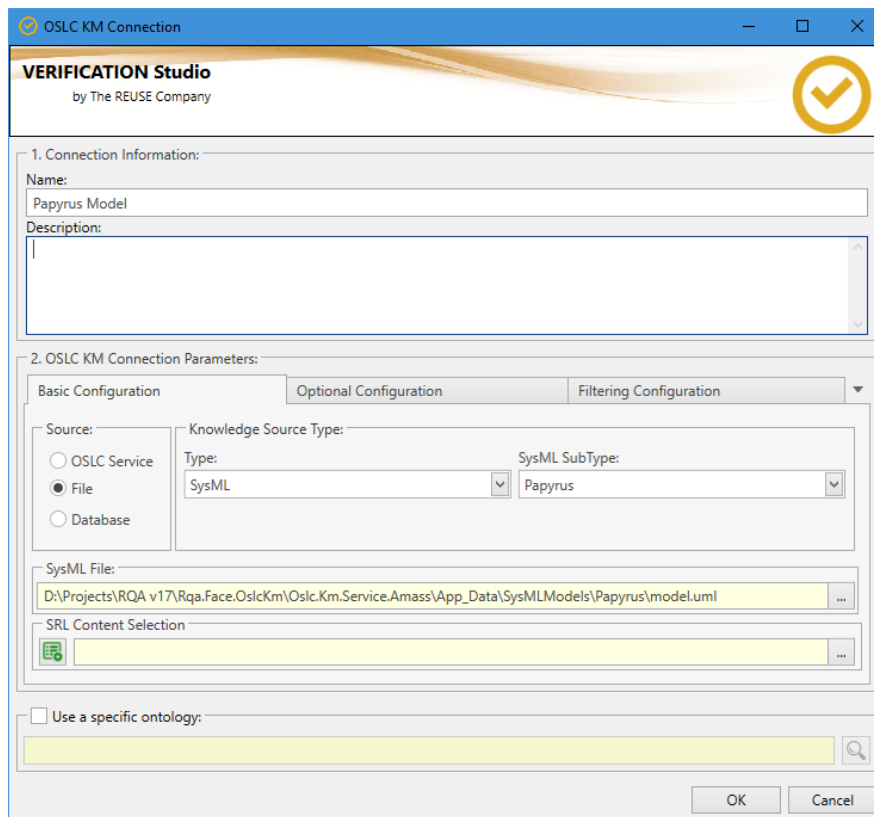


Figure 34. OSLC-KM Connection (SysML Papyrus sub-type)

In the OSLC-KM connection edition window, there are three different options to select the source of the information to match the input possibilities described at the top of this chapter. Once the input type source is selected, the window reconfigures automatically to ask for further parameters, depending on the source selected:

- **OSLC Service:** this is a provider of OSLC-KM in the form of a web service. It can be the web service described in the integration from the AMASS platform, where the input content of a file is transformed in OSLC-KM, or any other web service that offers it in its API.

Extra parameters needed to be provided for this type:

- The type of input.
- The URL of the web service.

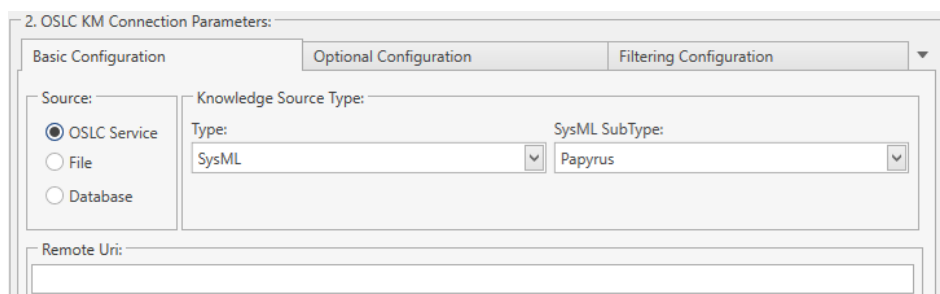


Figure 35. OSLC-KM input type: Web Service

- **File:** when selecting a file input format, the transformation will be done using the default parameters in the OSLC-KM connector. These can be fine-tuned if necessary to add additional parameters or an extra XSLT transformation schema.

Extra parameters needed to be provided for this type:

- The type of input.
- The location of the file.

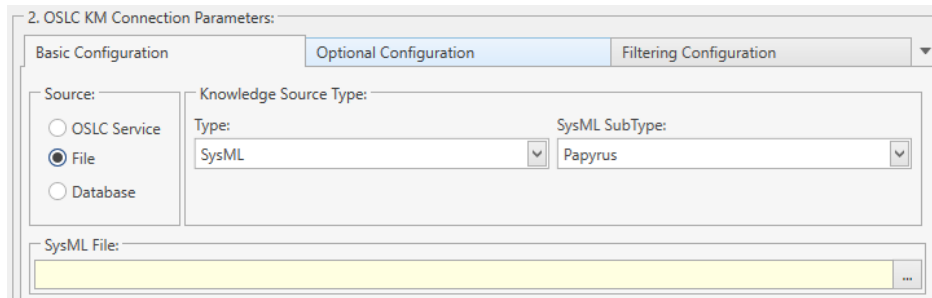


Figure 36. OSLC-KM input type: File

- Database: with this source a user can describe all the parameters to connect to a database that can use OLEDB drivers (Access, SQL Server, Oracle) and provide a SQL query that will retrieve a set of rows. These rows will be parsed into OSLC-KM and loaded into the tool.

Extra parameters needed to be provided for this type:

- The connection parameters to the database.
- The SQL query to be executed.

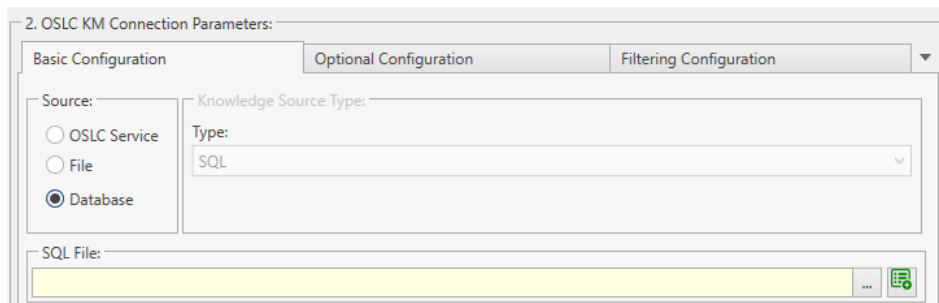
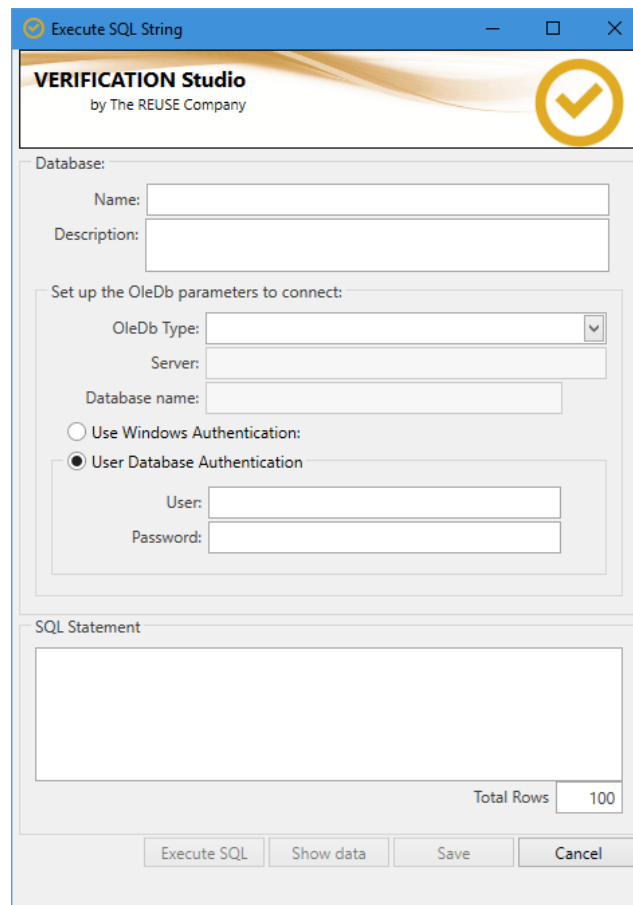


Figure 37. OSLC-KM input type: Database



Execute SQL String

VERIFICATION Studio
by The REUSE Company

Database:

Name:

Description:

Set up the OleDb parameters to connect:

OleDb Type:

Server:

Database name:

☐ Use Windows Authentication:

☒ User Database Authentication

User:

Password:

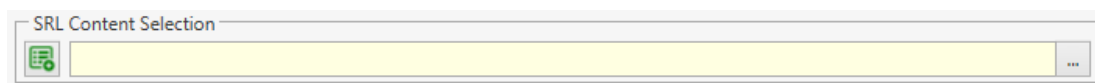
SQL Statement

Total Rows: 100

Execute SQL Show data Save Cancel

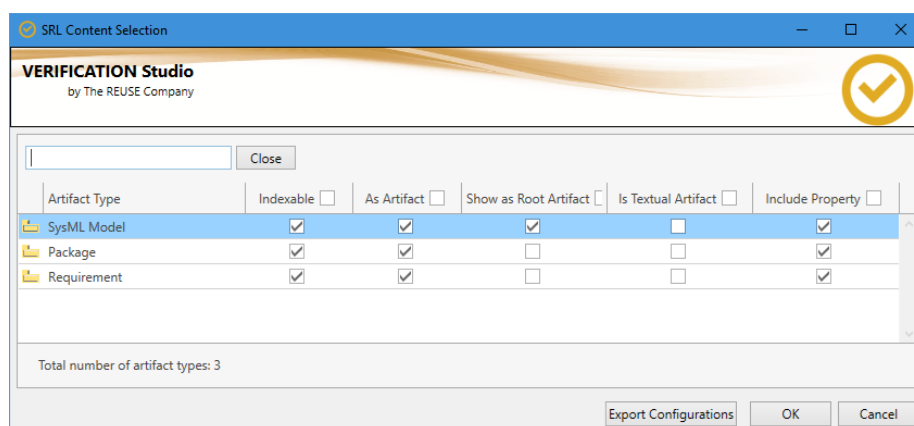
Figure 38. OSLC-KM input type: Database connection parameters window

After selecting the type of the input and basic parameters to retrieve its contents, further refinement of the parsing of the content can be done; from the OSLC-KM connection window, the SRL Content selection window lets users specify the exact mappings between the inputs in the file and the SE Suite categories. You can use an ad-hoc mapping (the green button shown in the Figure 39 will show the mappings edition window as in Figure 40) or an existing mapping file (the file selector on the right side of Figure 39).



SRL Content Selection

Figure 39. OSLC-KM mappings selector



SRL Content Selection

VERIFICATION Studio
by The REUSE Company

Close

Artifact Type	Indexable <input type="checkbox"/>	As Artifact <input type="checkbox"/>	Show as Root Artifact <input type="checkbox"/>	Is Textual Artifact <input type="checkbox"/>	Include Property <input type="checkbox"/>
SysML Model	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Package	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Requirement	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Total number of artifact types: 3

Export Configurations OK Cancel

Figure 40. OSLC-KM mappings edition window

After dealing with the input type, its basic parameters and mappings, a user can specify optional configuration to do the following:

- Include a work-product containing all the work products found in the OSLC-KM source to allow execution of completeness and consistency metrics over the whole source regardless the parts it's divided into.
- Include a “missing information” work product if the source of the OSLC-KM is not present in the given configuration.

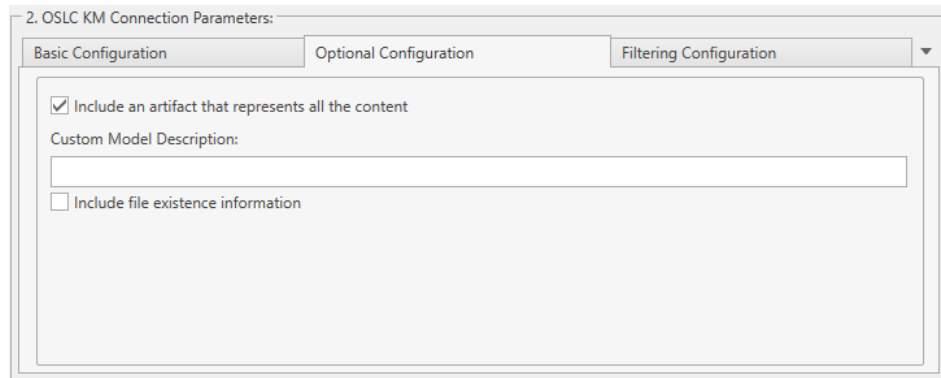


Figure 41. OSLC-KM connection window. Optional configuration

Finally, the user can set rules based on the knowledge database which the SE Suite tools can use to distinguish work products that are useful for quality assessment from those that are not.

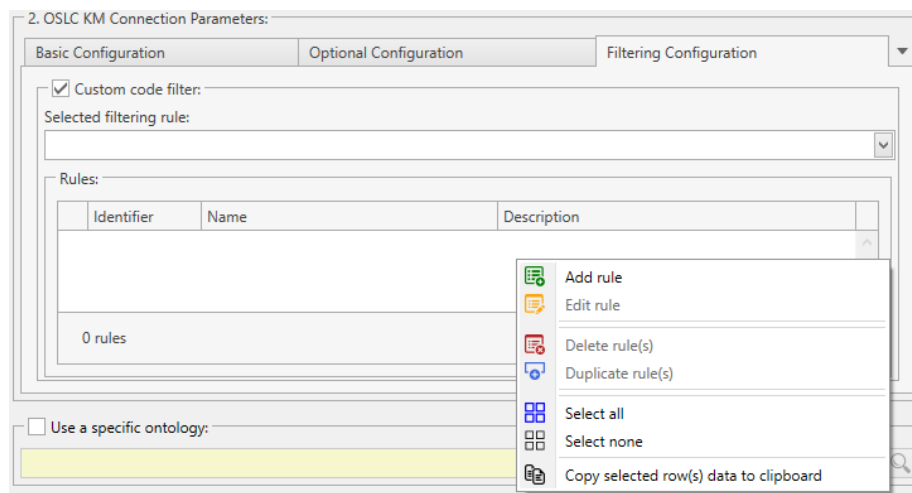


Figure 42. OSLC-KM connection window. Custom-code filtering

2.2.3.6 ‘Specify Tool Connection Information’ for Integration with V&V Manager (*)

V&V Manager allows formal verification of the contracts by external V&V tools that are installed on remote verification servers (WP5_TI_014 Client-server support).

The contracts (or individual formal properties) are selected, for example, from a Block Definition Diagram or at the level of components, using the related contracts. The verification or validation is invoked using a contextual menu Validation → V/V Manager, as depicted in Figure 43. Before any V&V activity is started, the availability of the Verification servers is checked and the communication is established with one of the available servers (WP5_TI_015 Service offer and discovery).

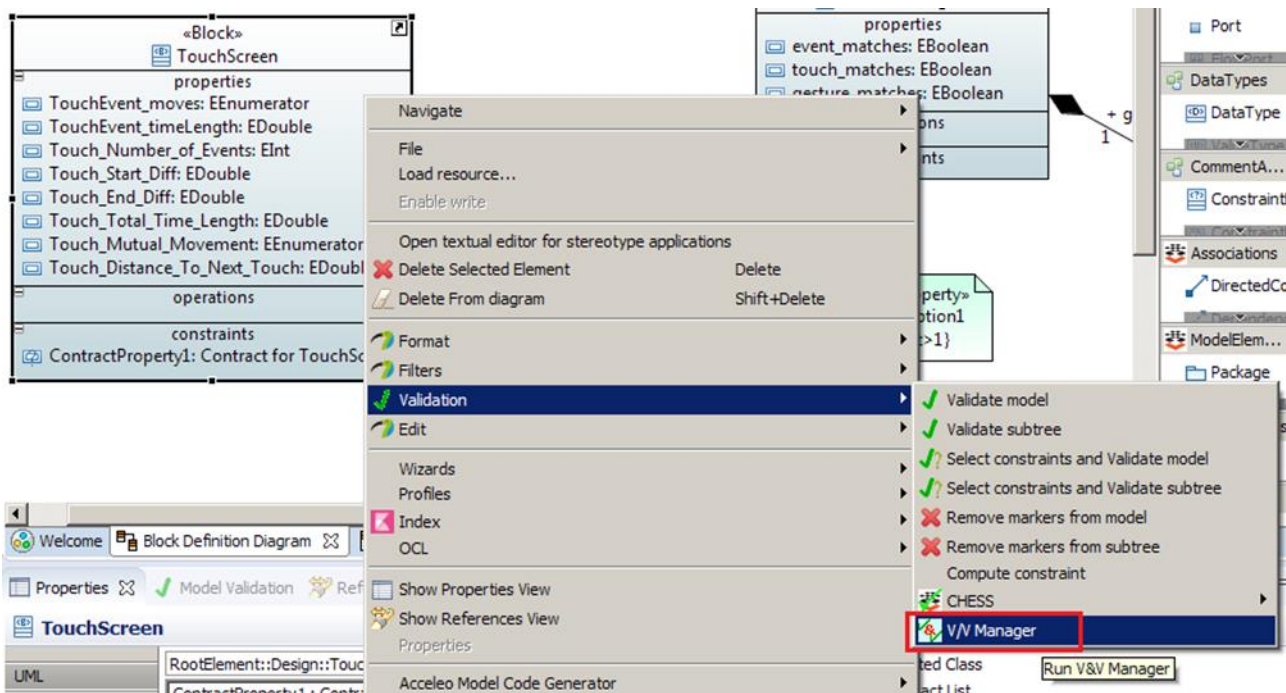
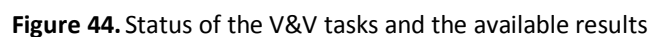


Figure 43. V&V Manager integration

Until a V&V activity is finished, the V&V Manager sends monitoring requests periodically to the Verification Server. The responses to these monitoring requests indicate the current status of the V&V tasks performed by the external V&V tools and the responses also contain the information that the tasks were completed (WP5_TI_016 Performance monitoring). The status and result of the verification or validation is depicted in the *V&V Result* tab, see Figure 44.

Requests and responses exchanged between the V&V Manager, the Verification Server and the V&V tools comply to the non-proprietary OSLC standard and are encoded in the RDF format (WP5_TI_006 V&V tools interoperability, WP5_TI_011 Non-proprietary data exchange, and WP5_TI_017 Standards-based interoperability).

Verification servers are installed as Linux servers in the current prototype, where a Proxygen or Apache-Tomcat server acts as an OSLC Automation service provider (by default on port 6080, or 8080). All verification tools installed on the verification servers get the OSLC Automation Plan and Request, and if the V&V tool can execute the verification plan, it is executed. When the execution performed by the V&V tool finishes, the server returns the OSLC Automation Response with Verification Results. All Verification results from all tools are seamlessly and continuously consolidated into a complete V&V result. Currently, all integrated V&V tools are command line based.



In summary, if the tool binary name, its parameters and the artefacts under verification can be passed to the command line tool in a standard way, the V&V tool does not have to be registered by the verification server application.

The Verification Server collects data that are extracted from the OSLC communication going through the server. The V&V Manager is capable to store data (V&V requests and results) in an appropriate PostgreSQL database (WP5_TI_001 Automatic data collection).

2.2.3.7 ‘Specify Tool Connection Information’ for Integration of CHES and V&V Tools (*)

CHES has been extended and integrated to perform V&V activities on the models by using the FBK Tools. Currently, two kinds of tool adapters are available: the first one invokes the FBK tools locally by passing the artefacts and the command via files. The second performs the same functionality via the OSLC-Automation adapter.

Adapter to FBK Tool via files

The architecture of the integration with FBK tools via files is depicted in Figure 45. The tool adapter takes the request from CHES, converts the model to the Verification tool format, sets up the artefacts and the command files, sends them to the Verification Tools and finally returns the result to CHES, ready to be shown graphically.

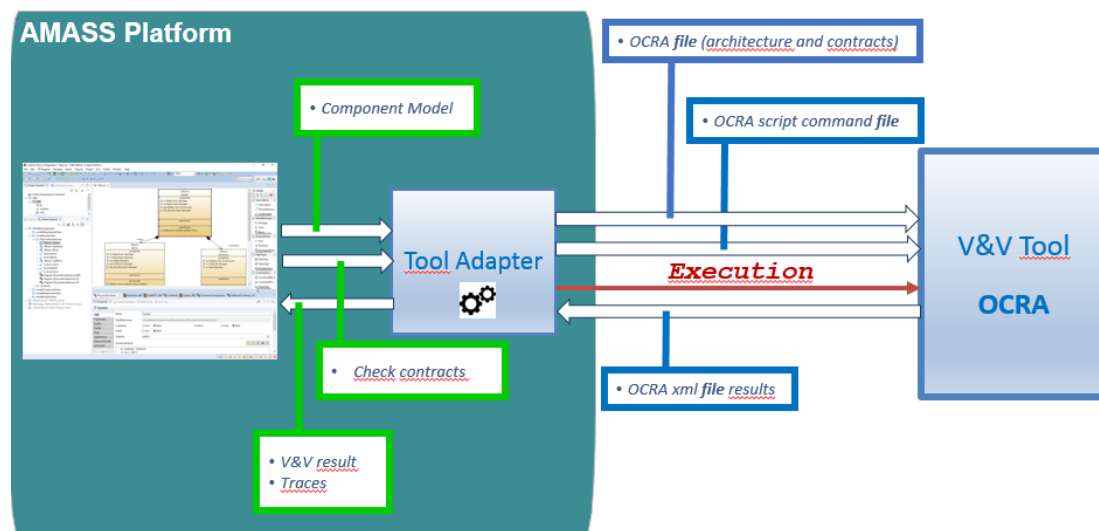


Figure 45. FBK Tool Integration via files

Adapter to FBK Tool via OSLC

Figure 46 represents the same functionality using the OSLC approach. As mentioned above, here we choose to use the OSLC Automation Domain for the integration toward the Verification Tools. From the user side, the choice of the adapter is transparent in terms of functionalities, so the user can decide to ask for a specific validation regardless of where this validation is going to be performed (locally or remotely).

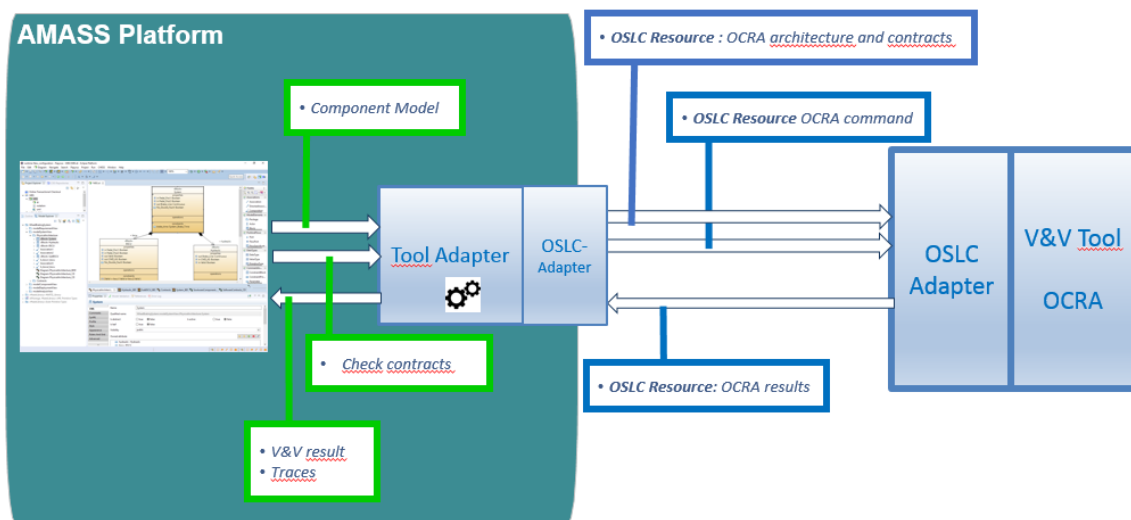


Figure 46. FBK Tool Integration via OSLC Automation

Adapter Configuration

The user can configure adapters from the Preferences menu (Figure 47). The Tools Preference Page allows the user to configure both the local adapter (via files) and the OSLC tool adapters by specifying parameters such as the executable path, the execution timeout, and the OSLC Service Provider catalogue end in the Service Provider instance.

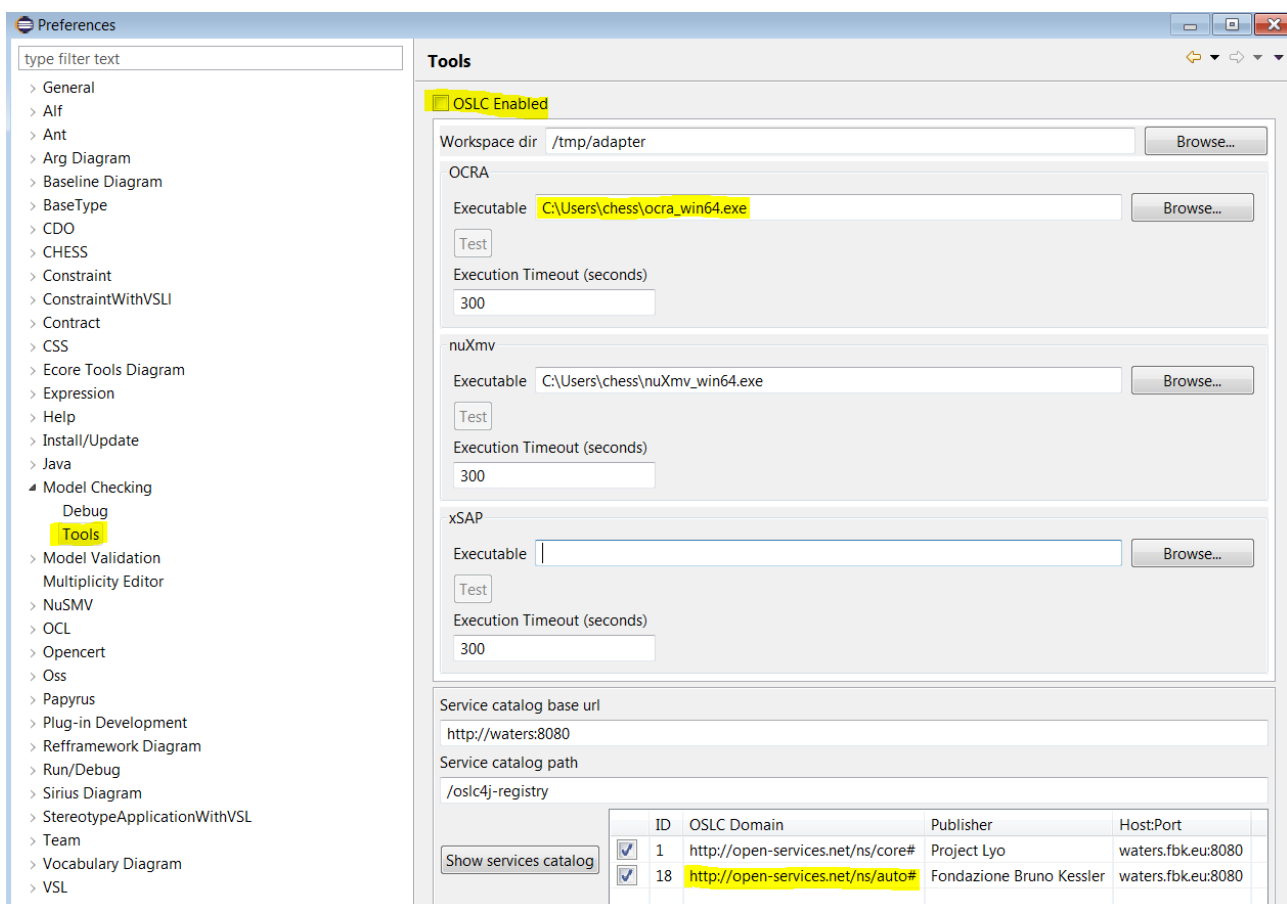


Figure 47. FBK Tool Adapters Configuration

Verification actions can be executed on CHESS models and invoked from both the main menu and context menu (Figure 48 and Figure 49). The same functions can be invoked by selecting the component in the diagram.

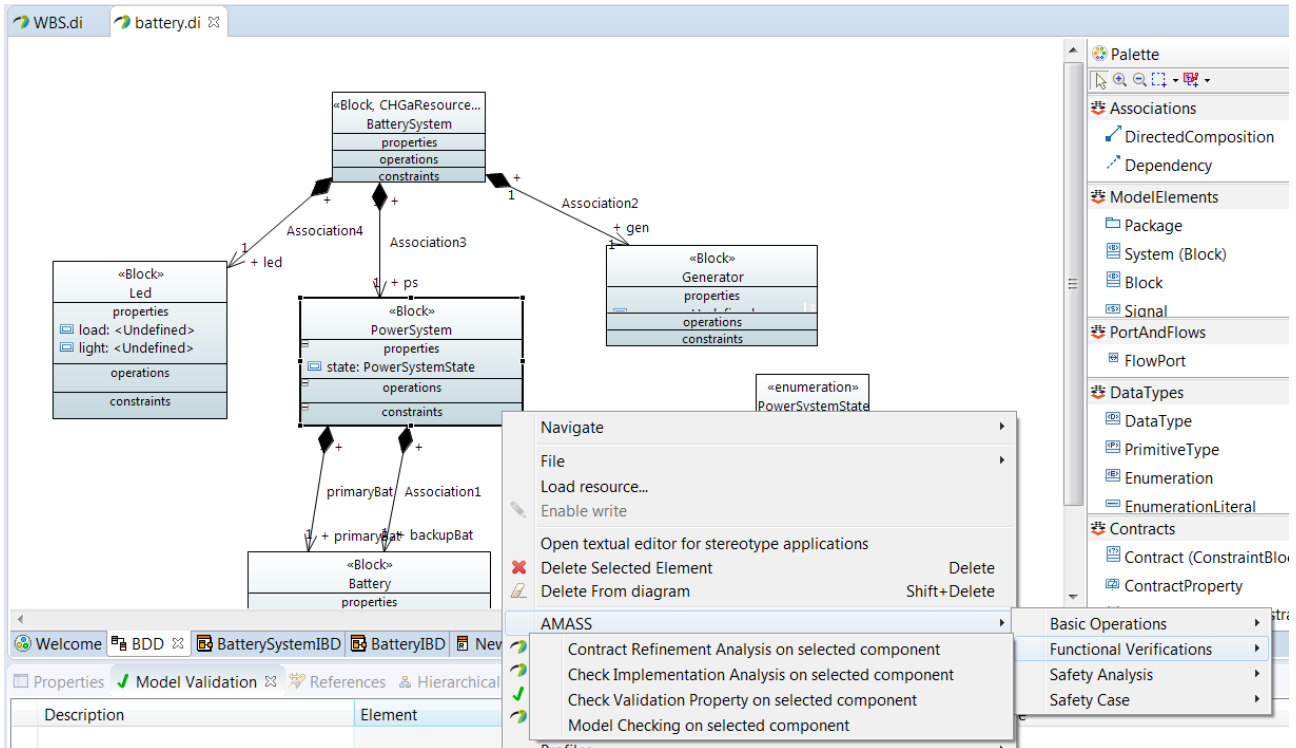


Figure 48. Contract and Behaviour Verification context menu

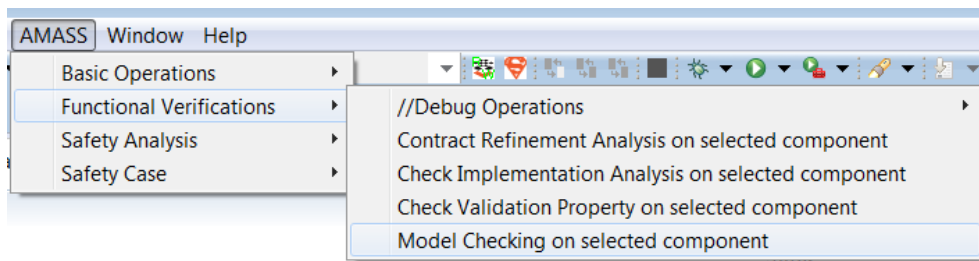


Figure 49. Contract and Behaviour Verification main menu

In the OSLC approach, all the verification functions have been mapped on *AutomationPlan* instances. The adapter on the client side maps the required function to the corresponding Automation Plan, then instantiates the Automation Request, setting up the parameter values in accordance with the plan. As an example, the Contract Refinement check is defined in the Service Provider catalogue (Figure 50).

```

▼<oslc_auto:AutomationPlan rdf:about="http://waters.fbk.eu:9000/eu.fbk.tools.oslc.provider/services/autoPlans/1">
  <oslc:serviceProvider rdf:resource="http://waters.fbk.eu:9000/oslc4j-registry/serviceProviders/13"/>
  <dcterms:identifier>1</dcterms:identifier>
  <dcterms:created rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2017-10-30T10:10:38.508Z</dcterms:created>
  ▼<oslc_auto:parameterDefinition>
    ▼<oslc:Property>
      <oslc:allowedValue>hybrid</oslc:allowedValue>
      <oslc:allowedValue>discrete</oslc:allowedValue>
      <oslc:defaultValue>discrete</oslc:defaultValue>
      <oslc:name>timeModel</oslc:name>
      <dcterms:description rdf:parseType="Literal">The type of time model (discrete or hybrid)</dcterms:description>
      <oslc:occurs rdf:resource="http://open-services.net/ns/core#Zero-or-one"/>
      <dcterms:title rdf:parseType="Literal">Time Model</dcterms:title>
      <oslc:valueType rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    </oslc:Property>
  </oslc_auto:parameterDefinition>
  ▼<dcterms:description rdf:parseType="Literal">
    Checks the contract refinement of the contract base model
  </dcterms:description>
  <dcterms:modified rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2017-10-30T10:10:38.508Z</dcterms:modified>
  ▼<oslc_auto:parameterDefinition>
    ▼<oslc:Property>
      <dcterms:description rdf:parseType="Literal">Type of algorithm</dcterms:description>
      <oslc:name>algorithmType</oslc:name>
      <oslc:allowedValue>auto</oslc:allowedValue>
      <oslc:occurs rdf:resource="http://open-services.net/ns/core#Zero-or-one"/>
      <oslc:defaultValue>auto</oslc:defaultValue>
      <oslc:allowedValue>bdd</oslc:allowedValue>
      <oslc:allowedValue>bmc</oslc:allowedValue>
      <oslc:valueType rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
      <oslc:allowedValue>ic3</oslc:allowedValue>
      <dcterms:title rdf:parseType="Literal">Algorithm Type</dcterms:title>
    </oslc:Property>
  </oslc_auto:parameterDefinition>
  ▼<oslc_auto:parameterDefinition>
    ▼<oslc:Property>
      <oslc:name>contractName</oslc:name>
      <dcterms:description rdf:parseType="Literal">The contract name</dcterms:description>
      <oslc:occurs rdf:resource="http://open-services.net/ns/core#Zero-or-one"/>
      <dcterms:title rdf:parseType="Literal">Contract Name</dcterms:title>
      <oslc:valueType rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    </oslc:Property>
  </oslc_auto:parameterDefinition>
  ▼<oslc_auto:parameterDefinition>
    ▼<oslc:Property>
      <oslc:name>contractModel</oslc:name>
      <dcterms:description rdf:parseType="Literal">The contract based model (OSS)</dcterms:description>
      <oslc:occurs rdf:resource="http://open-services.net/ns/core#Exactly-one"/>
      <dcterms:title rdf:parseType="Literal">Contract Model</dcterms:title>
      <oslc:valueType rdf:resource="http://open-services.net/ns/core#Resource"/>
    </oslc:Property>
  </oslc_auto:parameterDefinition>
  ▼<oslc_auto:parameterDefinition>
    ▼<oslc:Property>
      <oslc:defaultValue>10</oslc:defaultValue>
      <oslc:name>boundLength</oslc:name>
      <dcterms:description rdf:parseType="Literal">The bound length</dcterms:description>
      <oslc:occurs rdf:resource="http://open-services.net/ns/core#Zero-or-one"/>
      <dcterms:title rdf:parseType="Literal">Bound Length</dcterms:title>
      <oslc:valueType rdf:resource="http://www.w3.org/2001/XMLSchema#integer"/>
    </oslc:Property>
  </oslc_auto:parameterDefinition>
  <dcterms:title rdf:parseType="Literal">ocra_check_refinement</dcterms:title>
</oslc_auto:AutomationPlan>

```

Figure 50. FBK Tool Automation Plan example

2.2.3.8 'Specify Tool Connection Information' for Papyrus Safety and Security Engineering (**)

Papyrus Safety and Security Engineering (SSE) is a tool that offers model-based safety and security analysis capabilities of UML/SysML-based system models. As Papyrus SSE can be connected to Papyrus as a plugin, it offers seamless interoperability to the AMASS Platform.

Papyrus SSE also provides file-based interoperability features with other safety analyses tools like NuSMV and XFTA tools. Papyrus SSE enables annotation of the system models with fault and failure propagation information. This information is used to perform FMEA and fault tree analyses. The safety information can also be translated into a file format supported by NuSMV or XFTA (OpenPSA, SMV) to perform further FTA and model-checking analyses with these tools. The files can be manually imported in the XFTA and NuSMV tools. The files can be also directly used from the Papyrus SSE environment, as it provides seamless integration and a user interface to run these external tools and deliver their results back to the analysed model. Figure 51 presents the interoperability flow between Papyrus SSE and XFTA tool to get quantitative

FTA analyses on a model. Figure 52, meanwhile, shows excerpts of the Papyrus SSE user interface to transform a UML model into an SMV model and run the NuSMV tool for model-checking analysis.

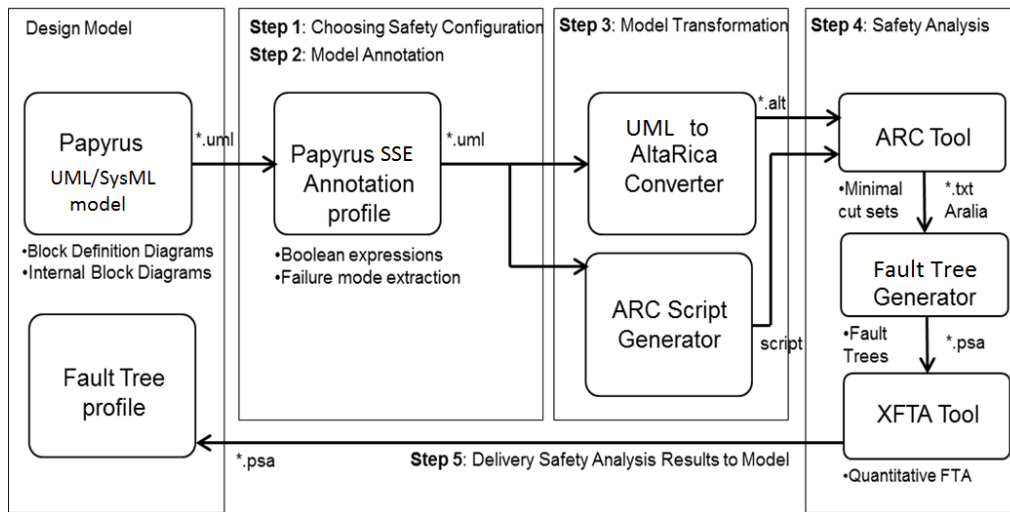


Figure 51. Interoperability flow between Papyrus SSE and XFTA tool

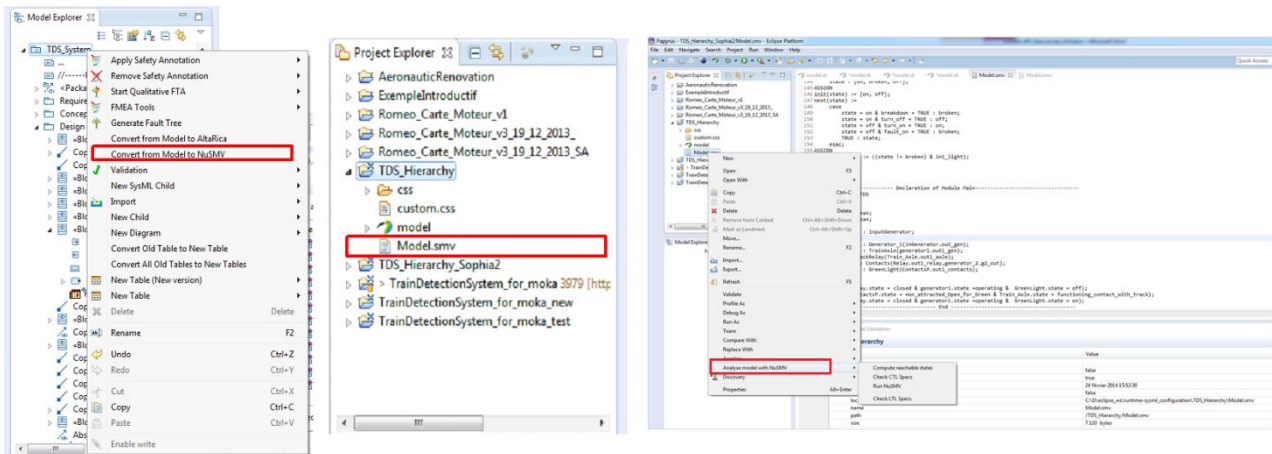


Figure 52. User interface in Papyrus SSE for seamless model-checking analysis with NuSMV tool

2.2.3.9 'Specify Tool Connection Information' for Systems Engineering Suite through Ad-hoc Tool Integration (*)

In this section, unlike in chapter 'Specify Tool Connection Information' for OSLC-KM-based Integration, the integration of the connectors for DOORS Next Generation (DNG) and Rhapsody is only available inside the SE Suite (by TRC). In the following sections, an ad-hoc integration is described.

Once the connection is defined for any of these connectors in the SE Suite tool, all the functions that are performed are the same, so the focus will be only to describe the ad-hoc integration and details of the connection windows and specific parameters needed and handled in those windows.

2.2.3.9.1 ReqIF Connector

This is a new ad-hoc connector to retrieve and author requirements from ReqIF specifications (Figure 53).

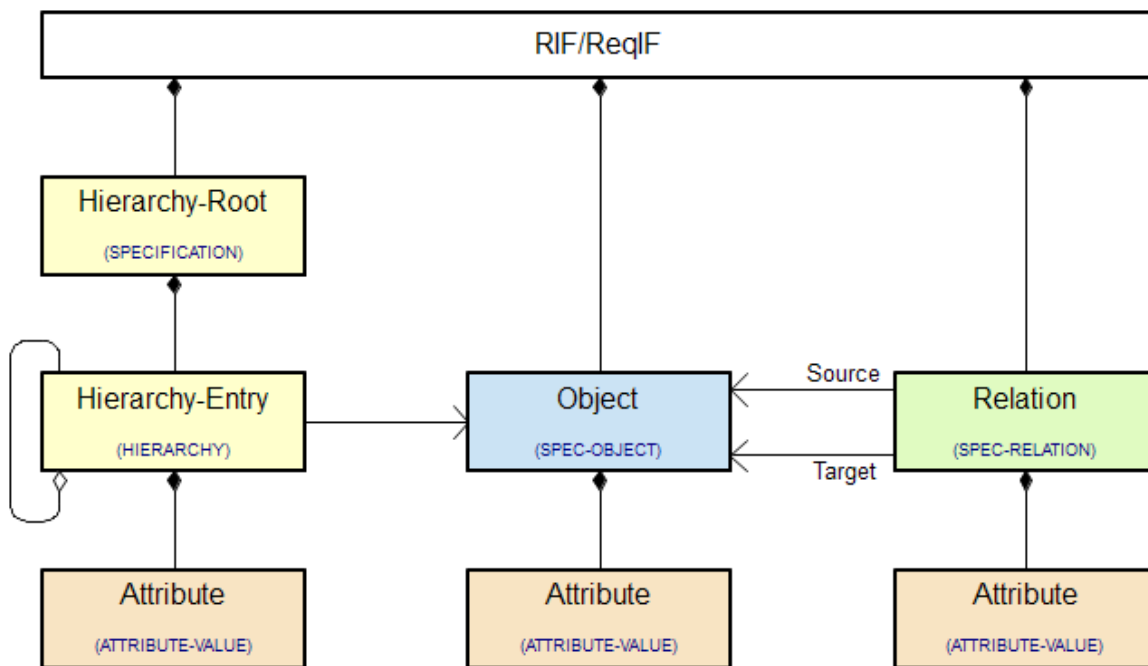


Figure 53. ReqIF metamodel

ReqIF is a well-known standard to represent requirements in XML format. Its structure supports to the addition of several specifications into a single project. Every ReqIF XML file is considered as a project, and may contain anywhere from 0 to N blocks or Specifications. Each Specification may contain both hierarchically-related Specifications and Objects (requirements). ReqIF allows traceability by letting users create Relations between Objects within the same ReqIF file.

All of the information contained within ReqIF files is meta-defined. This means that they do not contain fixed attributes for the Objects, but instead contain a meta-definition of attributes that are part of the Objects. Every attribute is defined in advance within the HEADER of the ReqIF file, and then mapped in the Objects definition.

For that reason, the ReqIF ad-hoc connector needs to pre-define a mapping of the attributes of every single ReqIF Specification to fulfil the SE Suite metamodel. This is compulsory to let the tools know where to extract the Statement, Heading, Author, etc, from each ReqIF Object (requirement).

The ReqIF connection window gathers all these parameters, from the physical location of the ReqIF file to the creation and management of the mapping between the ReqIF file and the SE Suite metamodel attributes. (Figure 54, Figure 55 and Figure 56).

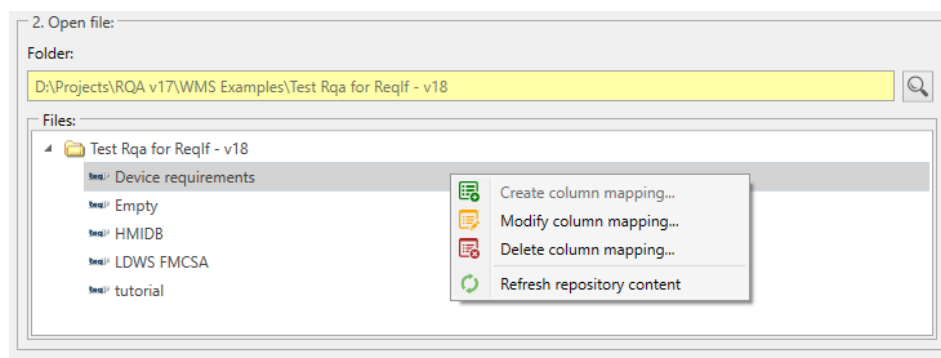


Figure 54. ReqIF connection window focusing on the ReqIF specific parameters

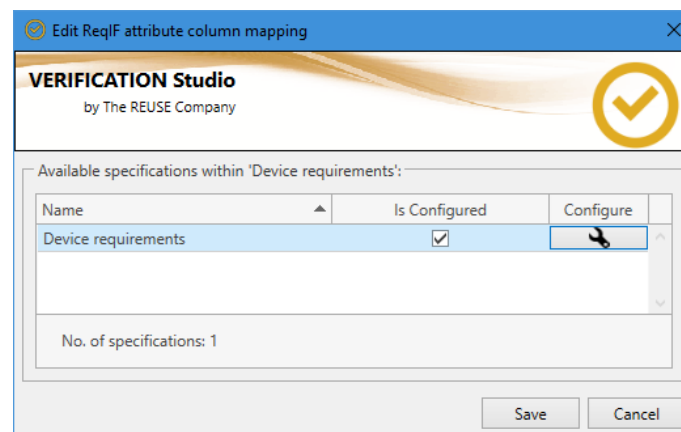


Figure 55. Mapping window for ReqIF Specifications

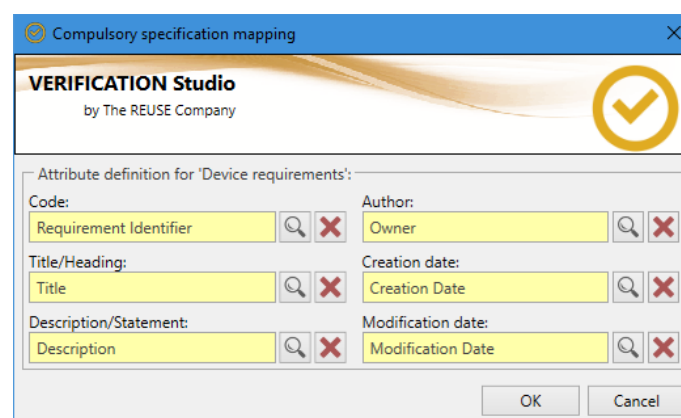


Figure 56. Mapping window for a single ReqIF specification

2.2.3.9.2 PTC Integrity

This is a new ad-hoc connector created to retrieve and author requirements from PTC documents.

PTC Integrity follows a typical client/server architecture with the specific characteristic that the server is a web server composed of many different interfaces. The client also has some possible interactions via its API and coding it in C language.

Integration with PTC Integrity has been accomplished by consuming some of these web service interfaces for SE Suite tools, and for authoring capabilities on top of the Integrity client (RAT Integrity Plugin), some interactivity has been achieved by using the Integrity client API (Figure 57).

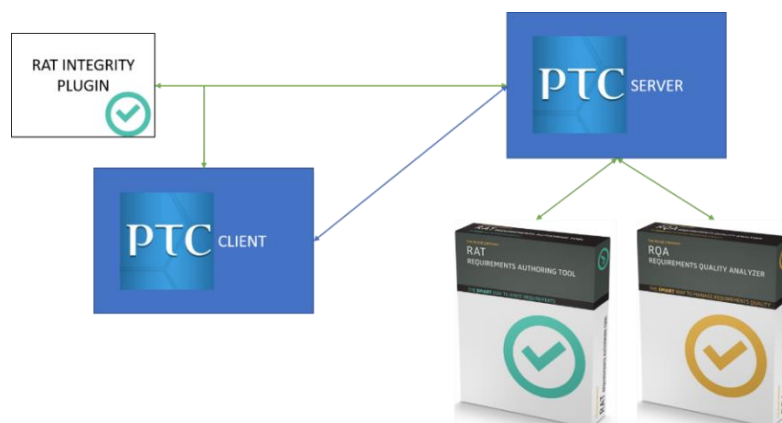


Figure 57. Integrity connector architecture

The integration with the RAT plugin has not been as seamless as that achieved with other RMS tools. Every other RAT plugin has a feature (whose name is RAT Inline), which allows the user to see directly in the requirements grid the quality assessment without opening any other user interface.

The problem arose when understanding the Integrity architecture that the changes are committed to the server and the triggers reacting to these changes were to be executed on the server, that would create an incredible amount of network traffic from RAT Integrity Plugins to the Integrity server and, in addition, the server would be overloaded executing all the trigger actions for all the changes of all the users. However, in other tools, the triggers can be handled by the client which is the source of the change, that allows to distribute the computing load and to reduce the network traffic to the minimum.

The PTC Integrity connection window lets users specify the details needed to integrate with PTC Integrity (Figure 58).

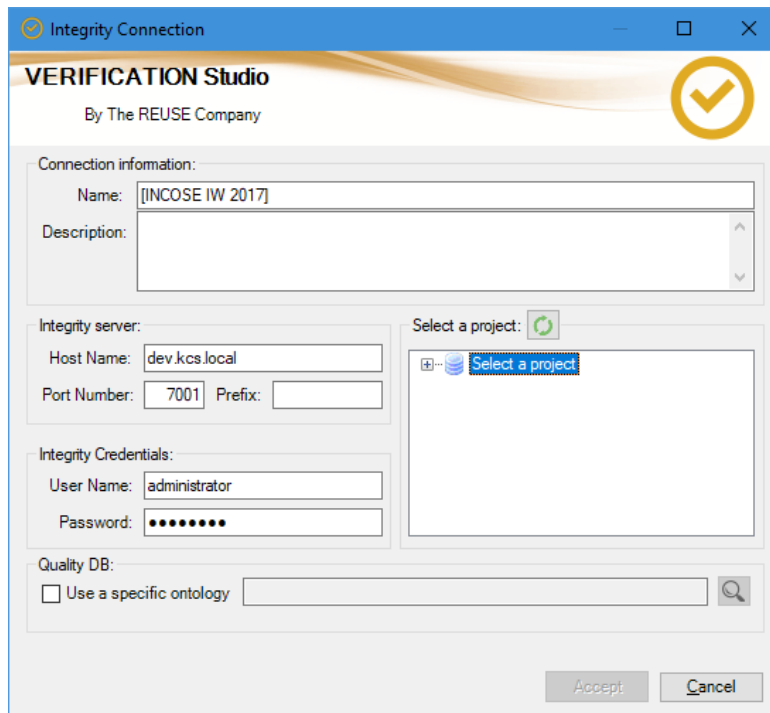


Figure 58. PTC integrity connection window

2.2.3.9.3 RAT for Rhapsody plugin

This is a new ad-hoc connector created to retrieve requirements from Rhapsody projects. Even if a Rhapsody project is composed of many different models, and these models can have requirements related to or inside them, the integration will focus on retrieving and authoring the requirements, not on the models themselves.

The Rhapsody architecture is composed of an editing environment that works with files stored either locally on the computer or on a network resource. These files may be under management control using any of the well-known version control management tools, such as Git, Subversion, etc.

Rhapsody allows interoperation with project contents using a Java interface as well as other .NET interfaces, however the latter method is obsolete. As such, integration was achieved using a Java interface.

The Java interface lets a user subscribe handlers to triggers generated by Rhapsody. By creating a suitable Java function and subscribing to the desired trigger, any functionality can be implemented.

The integration between SE Suite tools and Rhapsody has been achieved using this Java interface. The architecture (Figure 59) is composed of three different elements:

- RAT Rhapsody plugin: written in Java, this subscribes to suitable triggers in Rhapsody, such as creating and editing requirements, and transfers control to a service (XAT Resident Process) written in .NET and available via a resident process within the same computer.
- The second component of the architecture is written using .NET and consists of two parts:
 - XAT Resident Process: this uses existing technology provided by TRC to author requirements via a COM object after receiving any trigger handler.
 - Rhapsody COM interface: this interface facilitates communication between the XAT Resident Process and Rhapsody. The XAT Resident Process commits the changes performed in the RAT COM object back into Rhapsody via this interface.
- The third element is the RAT COM object, which performs any quality assessment and enables guided authoring using patterns and makes this functionality available for other RMS tool plugins.

All three elements must be deployed on the same computer.

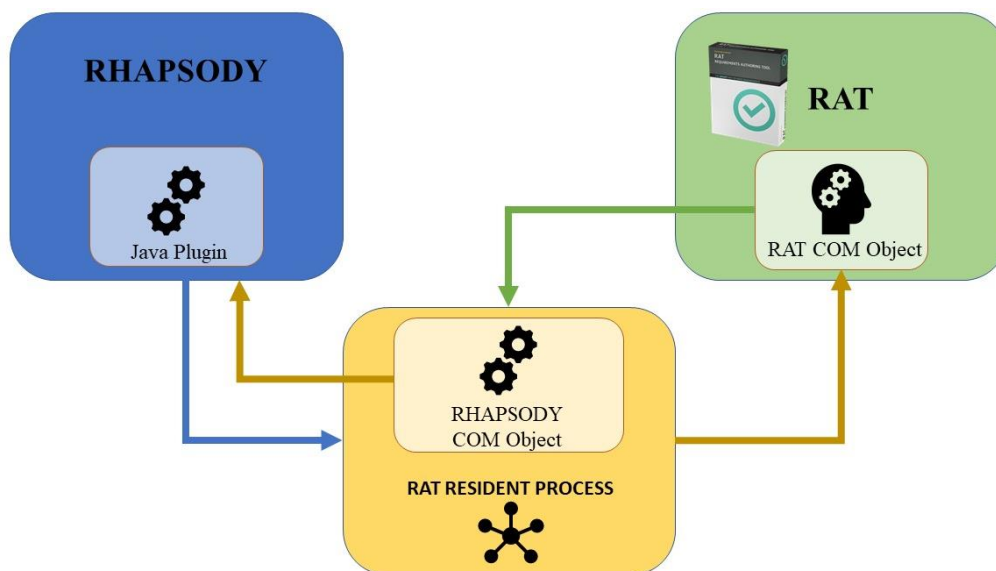


Figure 59. Rhapsody connector architecture

This integration has been achieved as a plugin on top of Rhapsody, adding several options in different Rhapsody elements, as shown in Figure 25 and Figure 26 above.

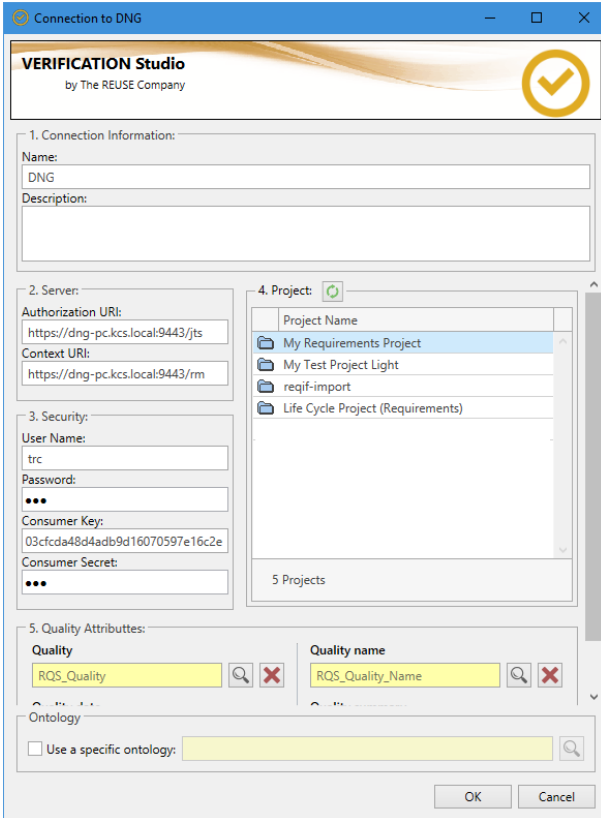
2.2.3.9.4 DOORS Next Generation (DNG)

This is a new ad-hoc connector created to retrieve and author requirements from DOORS Next Generation, which is a new requirements management tool developed by IBM and is based on the ideas and expertise gathered while creating DOORS.

DOORS NG has been released open-source under the Jazz platform, to create an ecosystem of engineering tools all running on it. It implements the OSLC-RM standard, generates a consumer of this OSLC-RM in SE Suite for DNG, and has a standard OSLC-RM connector ready to work with any other provider of OSLC-RM. When creating the first prototype of the integration, we found that the information TRC tools need to perform their assessments was not available in the DNG implementation of the OSLC-RM standard, but in custom attributes that go beyond that standard. Thus, the idea of having a standard OSLC-RM connector was rejected.

DNG Integrity follows a typical website architecture, it is installed on a web server using JAVA technology. And offers its functionality via a web browser and HTTP requests. The integration has been accomplished by requesting and consuming HTTP requests.

In WP5, we have not implemented a plugin on top of DNG. We envision that the development of several plugins on top of DNG would allow editing with RAT, and the generation of quality dashboards, meeting the requirement specifications.



The DNG connection window gathers parameters needed for an integration including connection details for the DNG Server and the project to connect to. The most significant difference with the traditional DOORS connection is that the attributes to store the quality assessment cannot be created from this integration, they must be selected *a priori* in the connection window (Figure 60).

Figure 60. DNG Connection window

2.2.3.10 'Specify Tool Connection Information' for Safety/Cyber Architect (**)

The design and method of seamless interoperability between the AMASS platform (CHESS and OpenCert tools) and safety/security analysis tools (Safety/Cyber Architect) have been presented in Deliverable D5.3 [11] (Section 3.1.6), and are depicted in Figure 61.

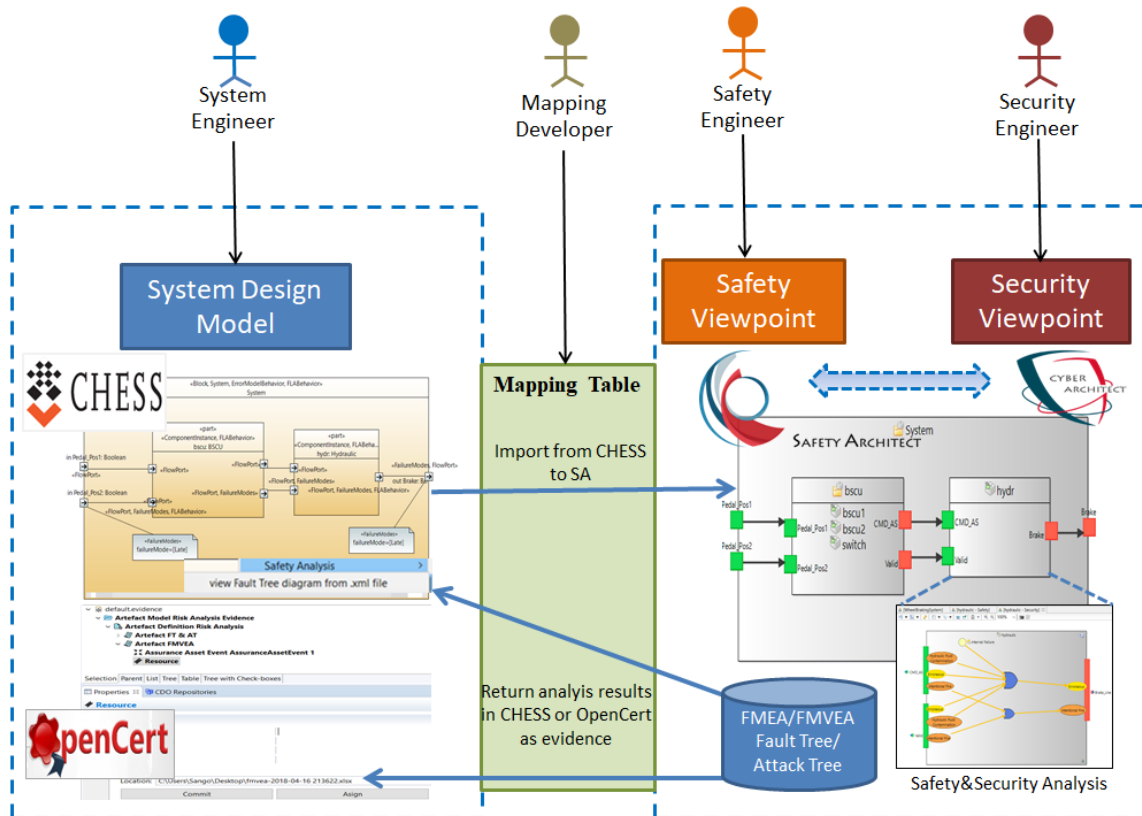


Figure 61. Interoperability between AMASS platform (CHES, OpenCert) and Safety/Security analysis tools (Safety Architect and Cyber Architect)

The design of system analysis tools interoperability (WP5_TI_004) supports a collaborative model-based systems analysis method for systems engineers, safety engineers, and security engineers (WP5_CW_001, WP5_CW_002 and WP5_CW_006). The method also supports other WP5 requirements, such as WP5_EM_014 (Evidence resource specification) because an assurance engineer can indicate in OpenCert the location of the evidence resource (FMEA/FMVEA tables) generated in Safety Architect, and WP5_EM_008 (Visualization of chains of evidence) because the fault trees generated in Safety Architect can be displayed as chains of evidence in the CHES model architecture. Table 2 provides the mapping between CHES model elements and Safety Architect model elements and the implementation status.

Table 2. A Mapping table between CHES and Safety Architect (name, ID and description of CHES elements are preserved by default)

Mapping between CHES and Safety Architect					
Separation of Concerns	CHES Model Elements	Safety Architect (SA) Model Elements	Mapping Requirements	Implementation status	Comments

Functional Elements	CHES model (".uml" file and "SystemView" Scope)	SA Model, FE Library and Data Library	<p>The mapping shall allow the import of a CHES UML file and the selection of the import scope (Requirement View, System View, Component View, Deployment View and Analysis View).</p> <p>A new SA Project composed by a Model (".sa"), a FE Library (".fearevents") and a Data Library (".data") shall be created for each import.</p>	Partially implemented	<p>In the current prototype only, the CHES Dependability View in System View is implemented. CHES Dependability View in Requirement View, software/Component View, Deployment View and Analysis View are not yet considered. Anyway, Safety Architect integrates some features such as, ReqIF Model, Functional/Logical/Physical block types, Block/Link Allocation and Global Analysis, which could be used for CHES Dependability View in Requirement View, software View, Deployment View and Analysis View, respectively. Note also that the Contract-Based View and Real Time View of CHES is not also considered in this mapping.</p> <p>Data library file has not been created yet in SA when importing.</p>
	SysML::Block UML::Component	Block	<p>A CHES Component/Block who does not have inner blocks shall be mapped to a Black box in SA. Otherwise, the SA block kind is a White box. If a domain modelling stereotype is used in CHES to distinguish the nature of a block (function, software or hardware) then this domain modelling constraint shall be mapped to a SA block</p>	Implemented	<p>A block kind can be also an Actor. An actor represents an element outside the system, i.e., human or environment, which interacts with the system under analysis. SA does not yet support the "Contract". All stereotypes <<Contract>> are currently imported as blocks in SA.</p>

			type: Function, Software, Hardware. Otherwise, the block type is Untype.		
	Port	Port	CHESS port's orientation (IN, OU, INOUT) shall be mapped to SA port type (IN, OUT, IN_OUT)	Implemented	
	Connector	Link	CHESS Connector shall be mapped to SA Link	Implemented	
	Data type / Data	Data type / Data	Some elementary data types in Chess should be mapped with SA Data Type (e.g., BooleanType, StringType, EnumerationType, NumericType, PhysicalQuantity et Unit).	Pending	In SA, the data are intended to be exchanged by the data links. It is possible to define the data concerned by a failure mode, which can impact the propagation. Complex data types in Chess (SysML – UML) are not yet supported in SA.
	Marte.Alloc.Assign	Allocation link	CHESS Marte.Alloc.Assign shall be mapped to SA AllocationLink.	Implemented	In SA, the allocation links are used to allocate blocks to other blocks or data links to other data links.

Non-functional Elements	Port FailureModes stereotype	Port Failure Modes	A CHES Port FailureModes stereotype shall be mapped to SA Specific failure modes related to port	Implemented	In SA, a failure mode specifies how Port or Barrier can be failed. By default, five failure modes are available: – Absent: an expected data flow is missing (in Safety and Security viewpoints). – Erroneous: a data flow is in error (in Safety and Security viewpoints). – Untimely: an unexpected data flow appears (in Safety viewpoint). – Malicious: an intentionally harmful data flow (in Security viewpoint). – Specific : a custom failure mode, identified by the name given by the user.
	ErrorModel State Machine		The CHES ErrorModel State Machine concept is not present in SA	Not implemented	In CHES , a dedicated ErrorModel State Machine is used to detail any possible relationship between incoming errors and outgoing errors (<i>InternalPropagation</i> , <i>ErrorState</i> and <i>failure conditions</i>). In SA , a <i>propagation link</i> is used to indicate how a failure is propagated within a black box, a <i>failure condition</i> of a black box is analysed thank to local or systems events, port failure modes, barriers and logical gates. <i>Error states</i> is not modelled in SA

	ErrorState		The CHESSE ErrorState concept is not present in SA	Not implemented	<p>In CHESSE, error state models a state of the component that is considered erroneous (i.e., not complying with the specifications).</p> <p>In SA, the error, normal or degraded states are not modelled, they are supposed to be foreseen in the specification. In addition, SA does not perform state-based analysis.</p>
	InternalPropagation	Propagation Link	CHESSE InternalPropagation shall be mapped to SA Propagation link	Pending	<p>CHESSE Internal Propagation is inside an ErrorModel State Machine, which is not handled during the import, so Internal Propagation is not yet mapped to local event in SA.</p> <p>In SA, a Propagation link is used to indicate how a failure is propagated within a black box. A propagation link is oriented and has a source and target.</p> <p>The source of a propagation link can be:</p> <ul style="list-style-type: none"> – a failure mode of an input port, an in/out port, a barrier or an outgoing allocation link – a system event or a local event – a logical gate <p>The target of a propagation link can be:</p> <ul style="list-style-type: none"> – a failure mode of an output port, of an in/out port, or of an incoming allocation link – a logical gate

	InternalFault	Local Event	CHESS Internal Fault shall be mapped to SA Local Event.	Pending	CHESS Internal Fault is inside an ErrorModel State Machine, which is not handled during the import, so Internal Fault is not yet mapped to Propagation Link in SA.
		System Event	The SA System Event concept is not present in CHESS. If the <<System Event>> Stereotype is defined in CHESS it should be mapped to SA	Pending	SA allows to configure system events impacting all the blocks of the model. Those events can be – for example – electromagnetic interference or an increase in the temperature, ... Those events are then usable in the dysfunctional equation of each block during the local analysis
		Barrier	The SA barrier concept is not present in CHESS. If the <<Barrier>> Stereotype is defined in CHESS it should be mapped to SA.	Pending	The SA barrier concept is an inner control of a block, which can be used to reduce the failure probability at the block's outputs. The barrier can have its own failure modes.
	Logical Gate	Logical gates	CHESS logical gate shall be mapped to SA logical gate	Pending	In SA, several gate types (AND, OR, NAND, NOR, K/N) are managed and can be used in the local analysis.
		Feared Events	The SA feared event concept is not present in CHESS. If the <<feared event>> Stereotype is defined in CHESS it should be mapped to SA	Pending	In SA, the feared event is not the failure itself but the consequence of this failure. A feared event can be triggered by a failure of the system under study. A feared event can be a Safety only, Security only or Safety&Security feared event. Feared events can be grouped by family.

The implementation description of the above mapping table from CHESS to SA is provided in Section 3.4.2.

2.2.3.11 'Specify Tool Connection Information' for Farkle (**)

The Farkle tool (Figure 62 and Figure 63) is developed as an external tool to the AMASS Tool Platform. The implementation of Farkle using a consumer and a producer for running an iterative learning-based tool extended a previous version of the tool towards the system under test. The abstract cases are interconnected with OSLC interfaces.

The interfaces for the AMASS Tool Platform is also handled with OSLC interfaces. The main input is a system model created with Papyrus.

Farkle is a test execution framework that enables testing systems in their target environment; through a communication mechanism between the host and the target, based on signal passing to the target system under test.

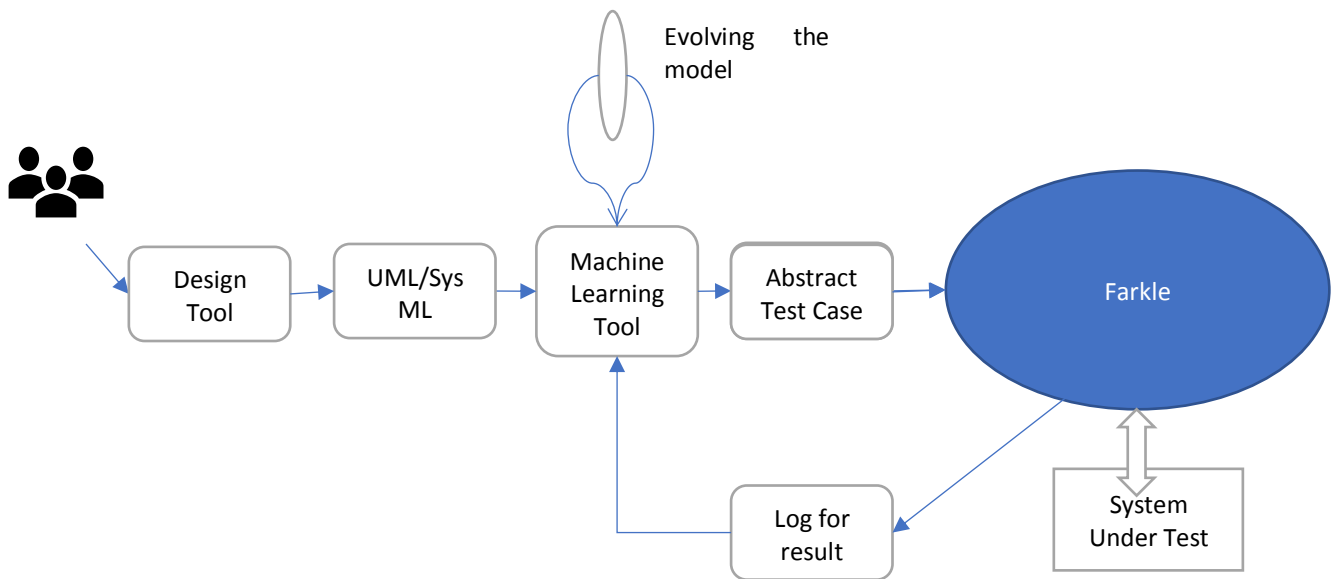


Figure 62. Flow of UML/SysML Model to Farkle

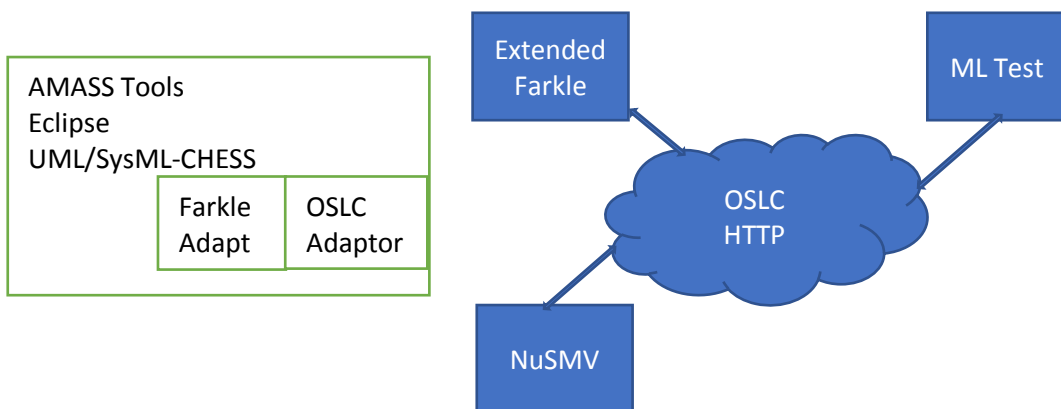


Figure 63. Extended Farkle interface for AMASS Tools

2.2.3.12 'Specify Tool Connection Information' for Sabotage (**)

The Sabotage tool is developed in the Eclipse framework using different technologies, e.g. Eclipse Modelling Framework (EMF), which is used for the development of the user interface to generate the fault list, and the Xtend technology for the code generation.

As the AMASS Platform is developed in the Eclipse framework, the Sabotage tool can be connected as an Eclipse plugin. After installing the Sabotage plugin in the AMASS platform, a new Sabotage perspective is opened which starts generation of fault list configurations and system simulations.

2.2.3.13 'Specify Tool Connection Information' for SAVONA (**)

As SAVONA is based on Papyrus, it offers seamless interoperability with the AMASS Platform/CHESS. A CHESS export function lets us convert the SAVONA model to a CHESS model (see Figure 64). That way, the initial architecture design as well as the functional specification with contracts can be performed in SAVONA and later reopened in CHESS to perform various V&V activities on the model.

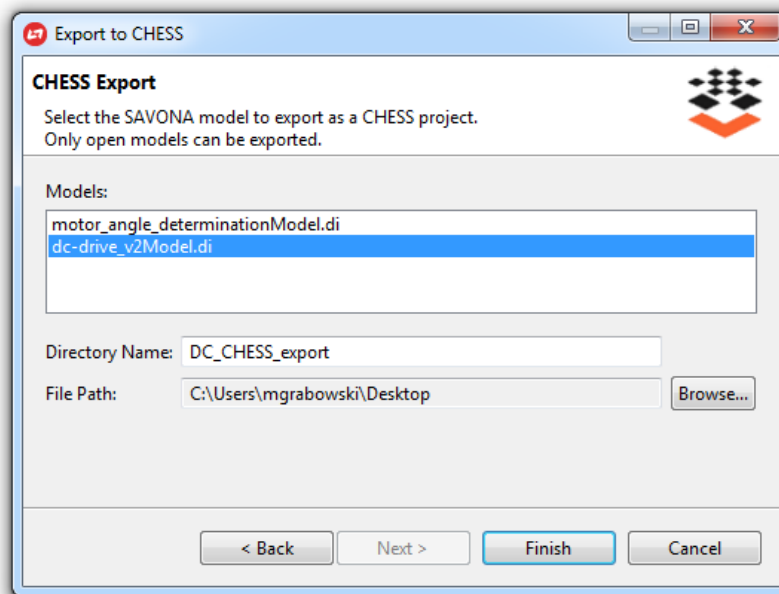


Figure 64. CHESS Export function of SAVONA

2.2.3.14 'Specify Tool Connection Information' though Automatic Generation of OSLC KM-based Connectors (**)

The approach for automatic generation of OSLC-KM connectors, whose design is given in D5.3 [11], has been implemented in the Verification Studio tool [27] by TRC. In broad terms, a user can select a structured source format (e.g. some XML file) and the mapping of the format is performed through a user interface to the SRL formalism [11]. The target elements correspond to e.g. Artefacts, Properties, relationships (RSHP), and Sub-Artifacts. Based on the mapping, an XSLT file is created with the transformation information necessary to import files of the source format.

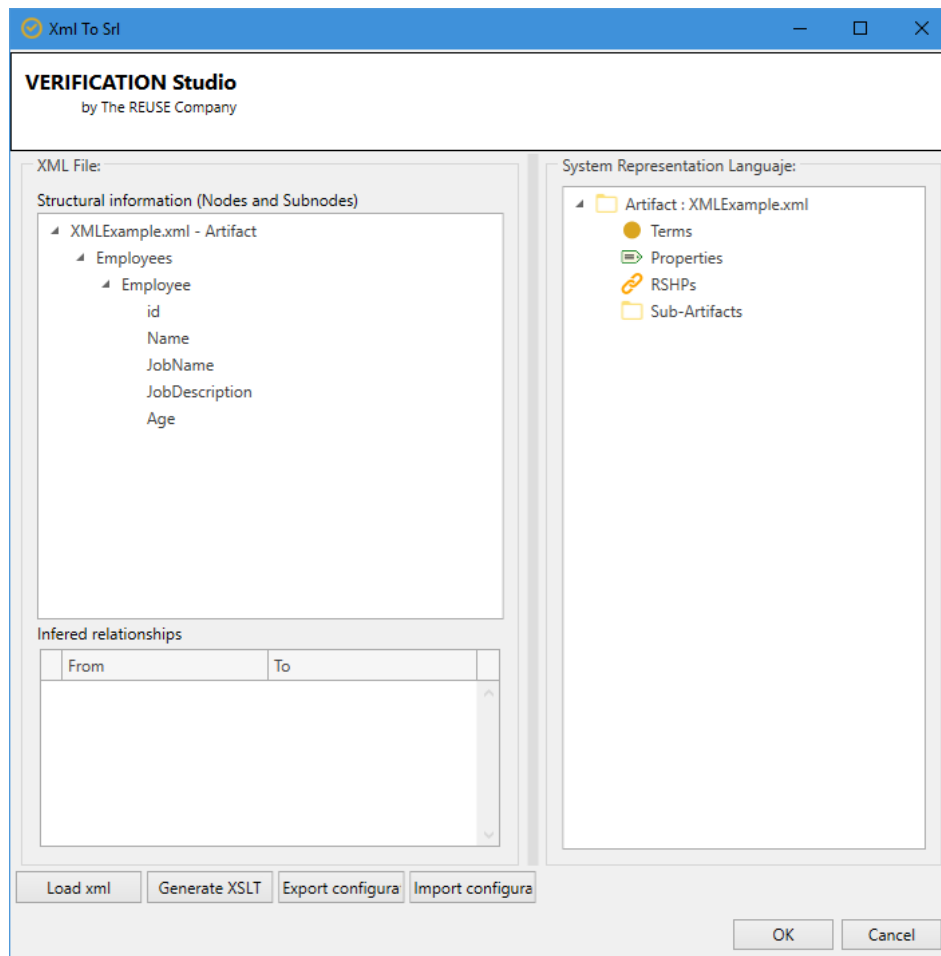


Figure 65. GUI to allow creation of XSLT files to customise the mapping of XML file nodes to elements in the OSLC-KM metamodel

2.2.4 Platform Management Functionality (**)

2.2.4.1 'Configure Access to Assurance Assets' in OpenCert (**)

CDO includes a security model to specify access rights to the models stored in CDO repositories. This model is supported by an API that has been used to develop this functionality.

To enable or disable the Security Manager module, a new parameter has been introduced to the CDO server configuration file "opencoss-properties.xml" (see [14]) generated in the windows user directory:

```
<entry key="isCDOSecurityEnabled">true/false</entry>
```

This parameter will be read in the *init()* method of the StandaloneCDOServer class (in the org.eclipse.opencert.storage.cdo plugin) , which is responsible for running the CDO server. The parameter calls the following methods, which are responsible for enabling and initializing the Security Manager on the AMASS CDO Server:

```
private ISecurityManager createSecurity(IRepository repository) {
    IManagedContainer managedContainer = IPluginContainer.INSTANCE;
    String realmPath = "/security";
    SecurityManagerUtil.prepareContainer(managedContainer);
    ISecurityManager securityManager =
SecurityManagerUtil.createSecurityManager(realmPath, managedContainer);
    CommitHandler handler = getHandler(managedContainer, "home(/home)");
```

```

        ((InternalSecurityManager) securityManager).addCommitHandler(handler);

        return securityManager;
    }

    private void initSecurity(ISecurityManager securityManager) {
        if (securityManager instanceof InternalSecurityManager) {
            ((InternalSecurityManager)
securityManager).setRepository((InternalRepository) repository);
            LifecycleUtil.activate(securityManager);
        }
    }
}

```

The CDO Server automatically adds a user account with an administration role when the Security Manager is started for the first time. The account user name is “Administrator” and the password is “0000”. Using this account, a user can read and modify access rights to the assurance assets stored on the AMASS Server. This policy is an instance of the CDO Security metamodel explained in the section 4.4 of the D5.3 [11].

The AMASS platform has a set of windows that lets you create and edit access policies for the data stored in the AMASS repository. The user can access these from the “Manage Security” context menu if you have an administration role.

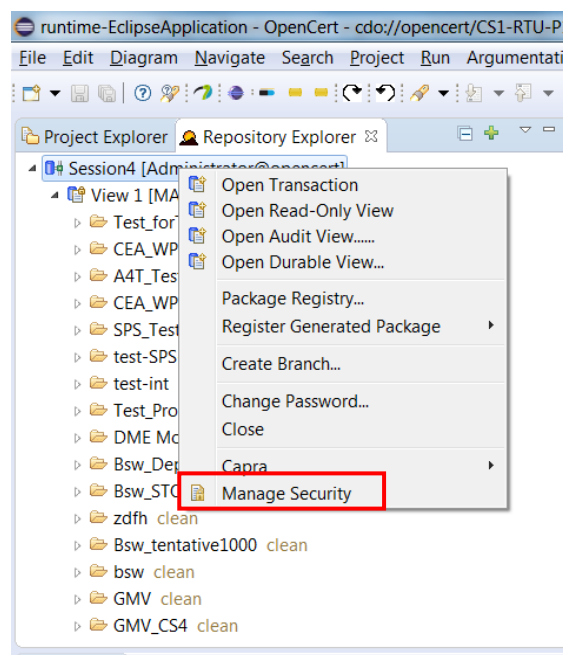


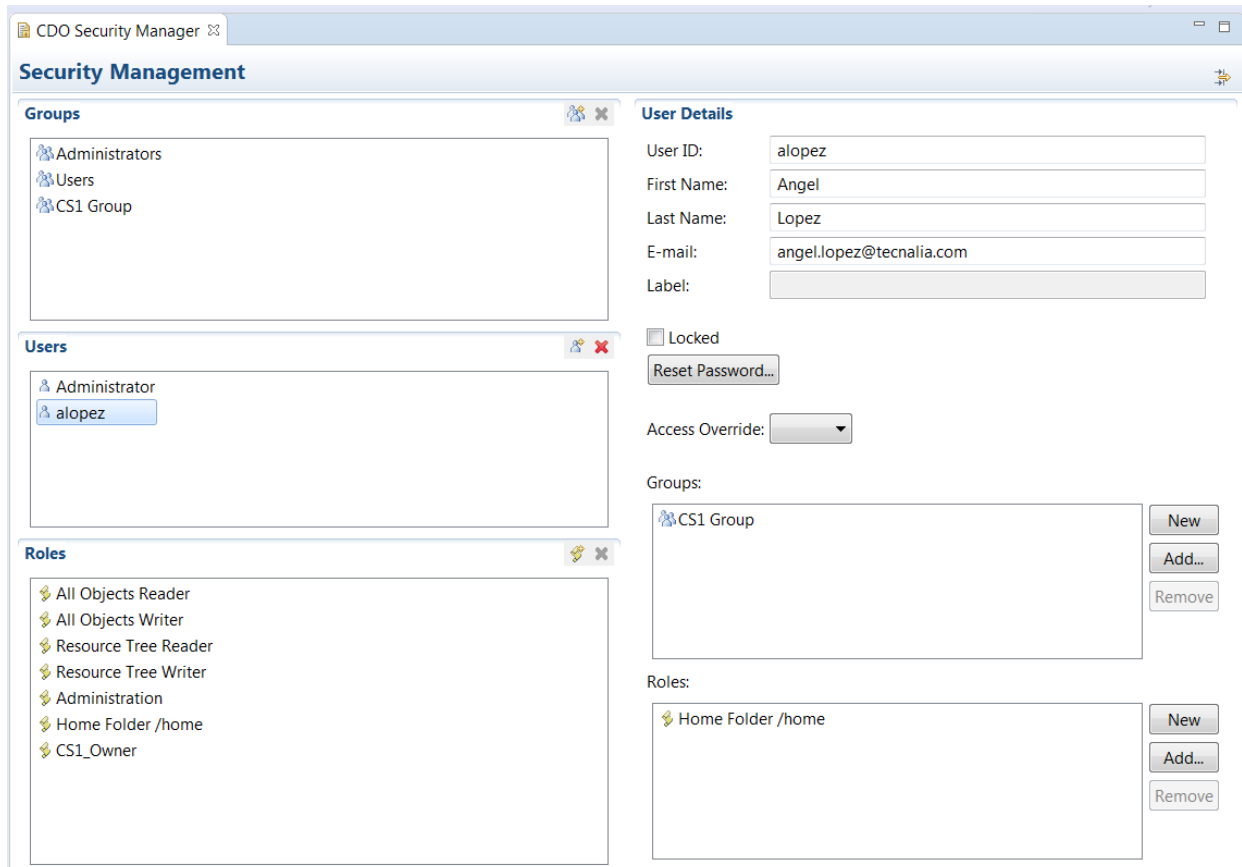
Figure 66. Manage Security menu

From the “Manage Security” menu, the user can manage users, user groups and roles (e.g. Figure 67).

Roles define the access rights (read or write) that users with the role have on the data stored in the CDO Repository. The role management window lets users browse through all stored assurance assets and select the asset whose access level the user with administration role wants to set or modify (Figure 68).

The user can create groups of users and assign roles to those groups (Figure 69).

The “Repository Explorer” shows all models stored in the AMASS CDO Server as a tree view. It displays writable resources with a black caption and (only) readable ones with a dark-green caption. If a user doesn’t have at least read permission to a model, that model is not displayed. If a user opens an (only) readable resource, the “Save changes” button is disabled (Figure 70).



CDO Security Manager

Security Management

Groups

- Administrators
- Users
- CS1 Group

Users

- Administrator
- alopez

Roles

- All Objects Reader
- All Objects Writer
- Resource Tree Reader
- Resource Tree Writer
- Administration
- Home Folder /home
- CS1_Owner

User Details

User ID: alopez

First Name: Angel

Last Name: Lopez

E-mail: angel.lopez@tecnalia.com

Label:

☐ Locked

[Reset Password...](#)

Access Override:

Groups:

- CS1 Group

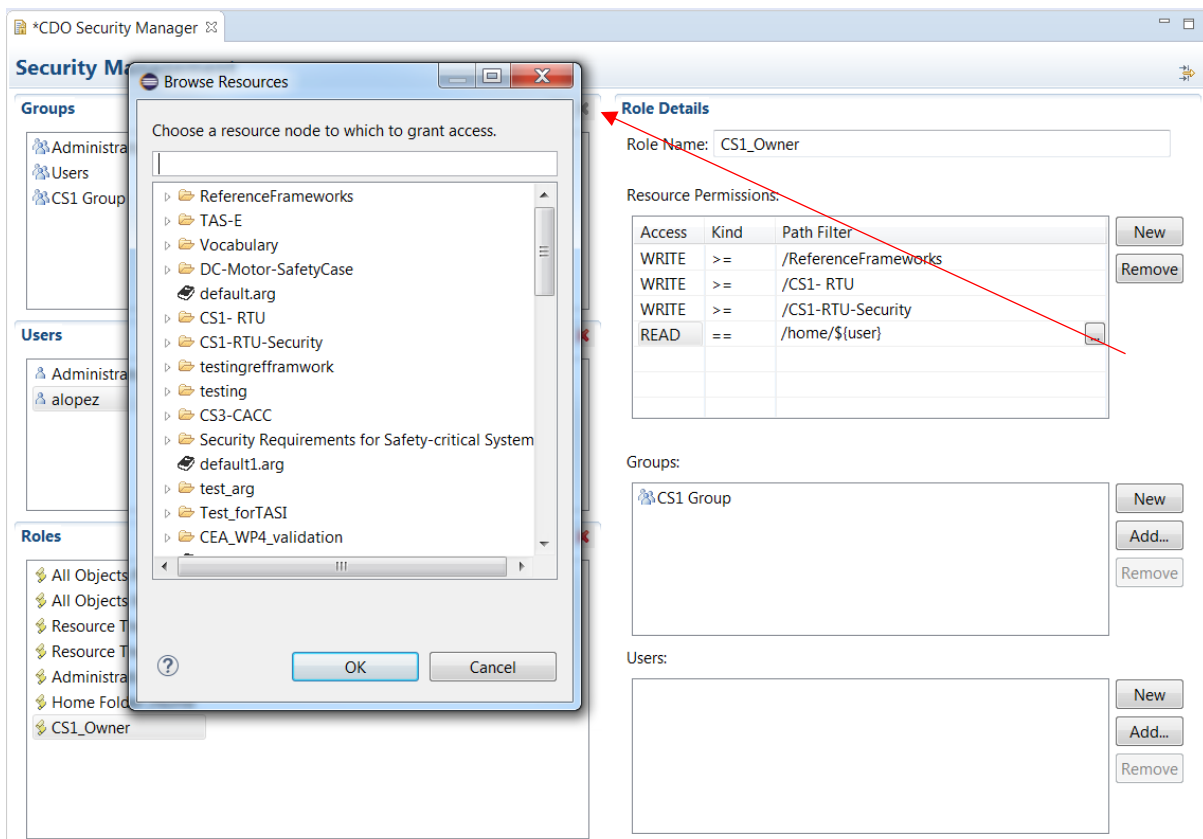
[New](#) [Add...](#) [Remove](#)

Roles:

- Home Folder /home

[New](#) [Add...](#) [Remove](#)

Figure 67. Manage Users Window



CDO Security Manager

Security Management

Groups

- Administrators
- Users
- CS1 Group

Users

- Administrator
- alopez

Roles

- All Objects Reader
- All Objects Writer
- Resource Tree Reader
- Resource Tree Writer
- Administration
- Home Folder /home
- CS1_Owner

Role Details

Role Name: CS1_Owner

Resource Permissions:

Access	Kind	Path Filter
WRITE	>=	/ReferenceFrameworks
WRITE	>=	/CS1- RTU
WRITE	>=	/CS1-RTU-Security
READ	==	/home/\${user}

[New](#) [Remove](#)

Groups:

- CS1 Group

[New](#) [Add...](#) [Remove](#)

Users:

- Administrator
- alopez

[New](#) [Add...](#) [Remove](#)

Figure 68. Manage Roles Window

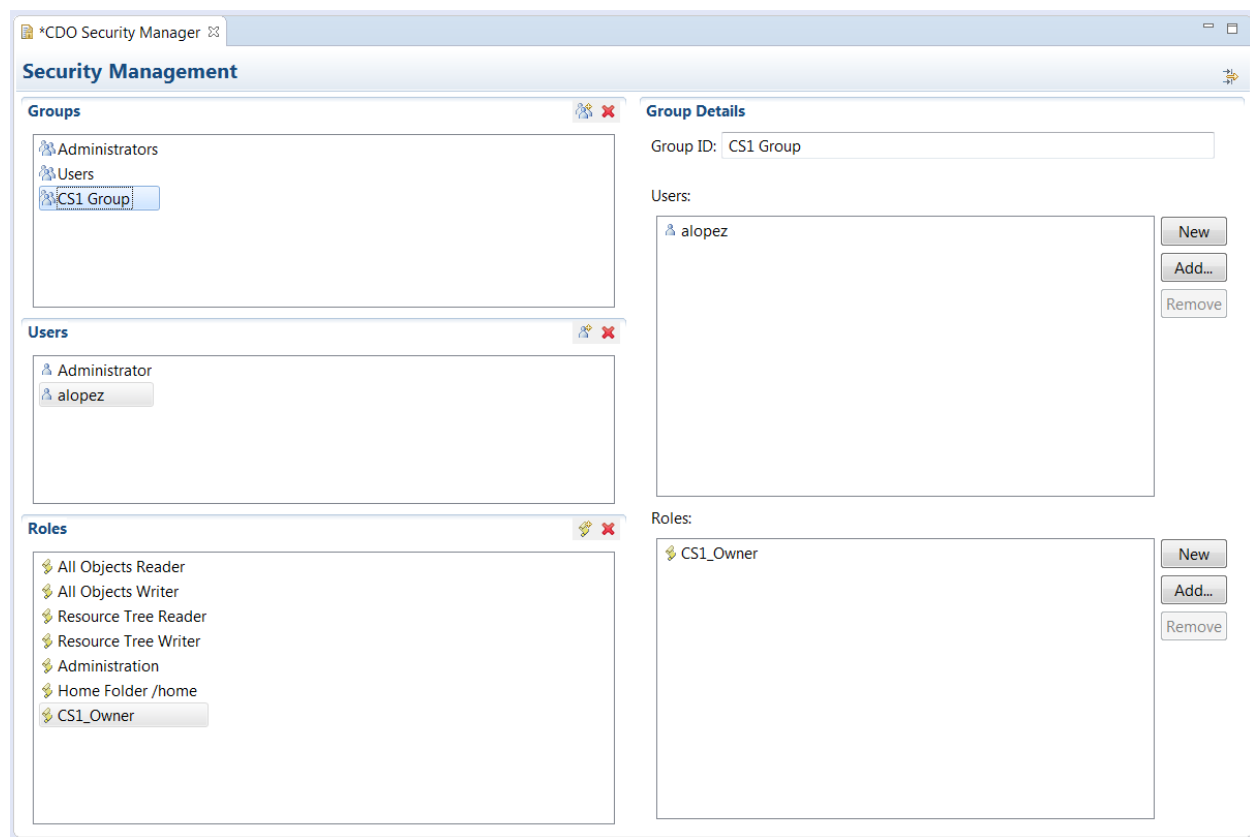


Figure 69. Manage Groups Window

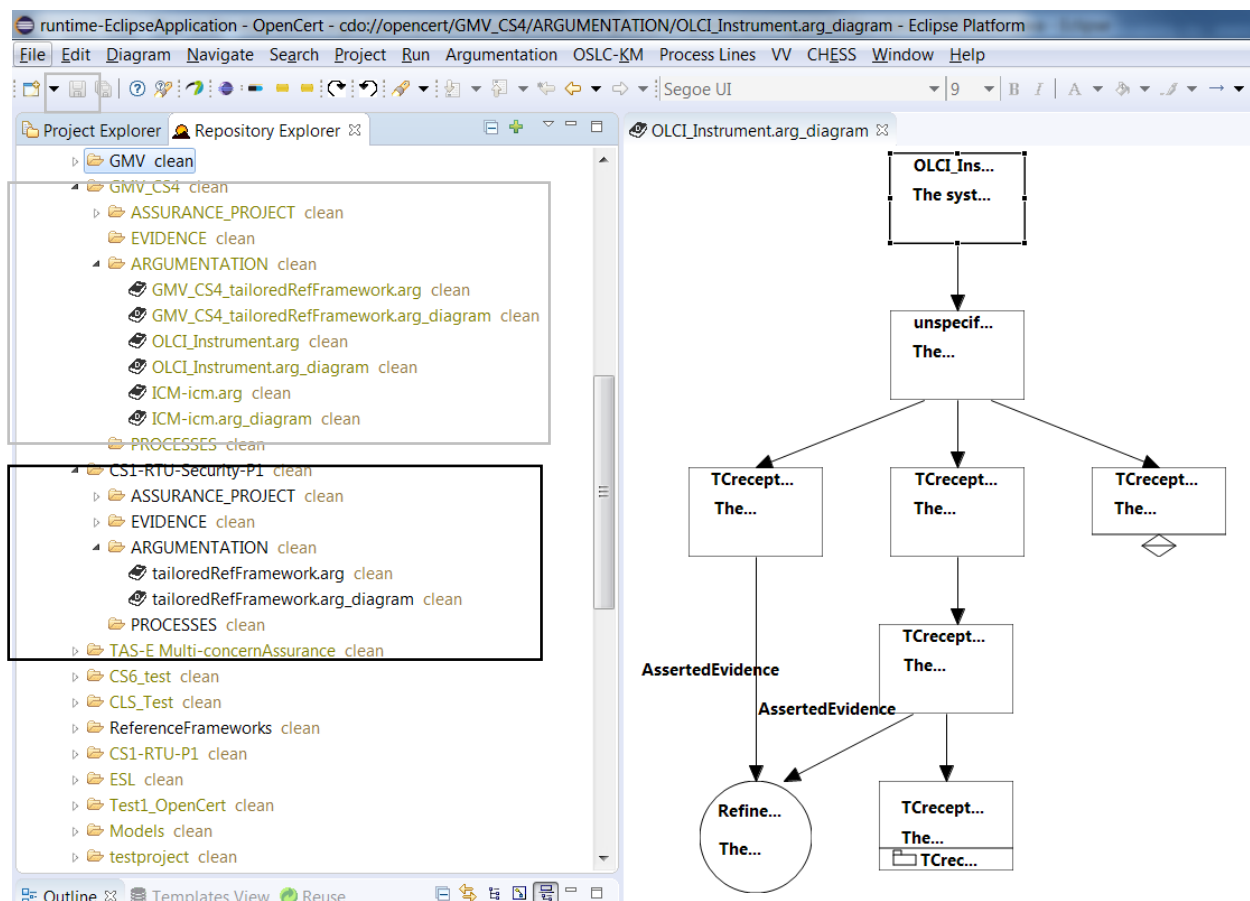


Figure 70. Repository Explorer showing models with different font styles according the access rights

Access permissions will be used by all the OpenCert functionalities that need to create or read assets from the CDO repository (Repository Explorer, Assurance Project Generation, Cross Standard Reuse, Reuse Assistant, the different CACM parts editors, the Arguments importer from EPF Composer...). For example, it is only possible to create an assurance project inside CDO Folders (a kind of CDO Resource) with write permissions (see Figure 71). It's necessary to check the access rights to a concrete CDO Resource before using it or a `org.eclipse.emf.cdo.common.security.NoPermissionException` will be thrown. The following code to check if the logged user has read/write permissions to a concrete CDOResource has been added in several parts of the OpenCert source code:

```
public static boolean hasReadPermission(CDOResourceNode node) {
    node.cdoReload();
    CDOPermission permission = node.cdoRevision().getPermission();
    return permission.isReadable();
}

public static boolean hasWritePermission(CDOResourceNode node) {
    node.cdoReload();
    CDOPermission permission = node.cdoRevision().getPermission();
    return permission.isWritable();
}
```

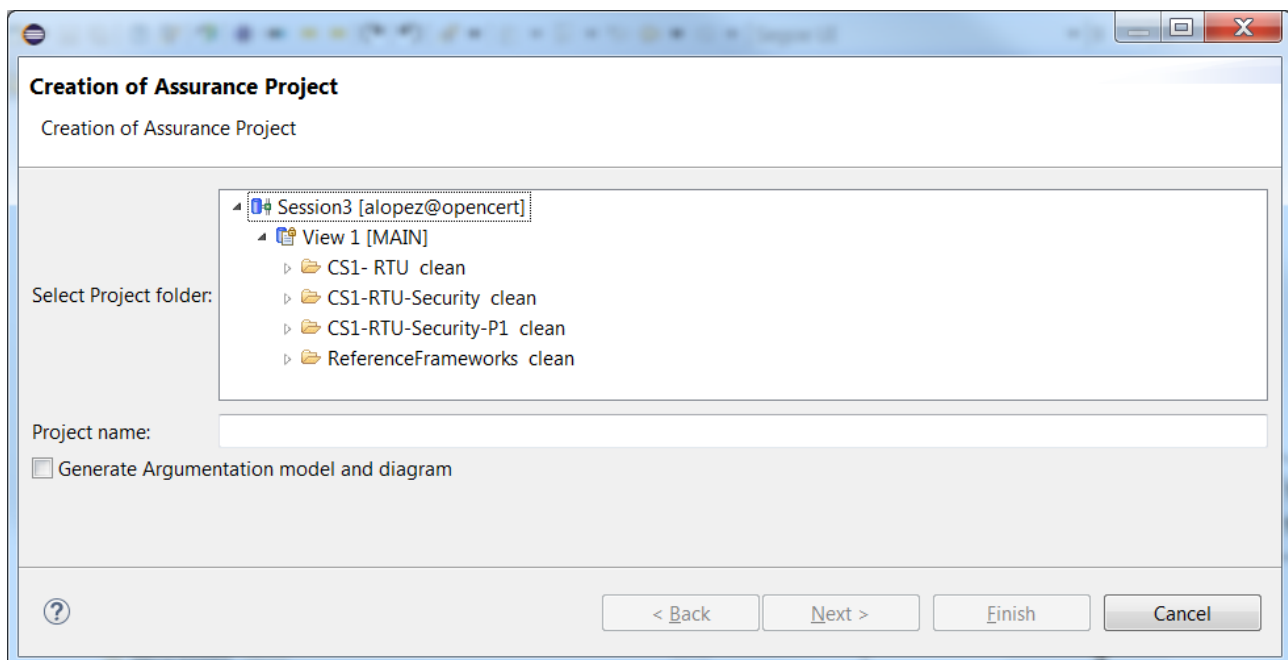


Figure 71. Generation of Assurance Project in Repository folders with write access

The access configuration is also evaluated to allow or deny the use of functionalities that require specific access rights for their proper working. For example, it is not possible to use any feature that imports data to an assurance project without write access to the target assurance project model.

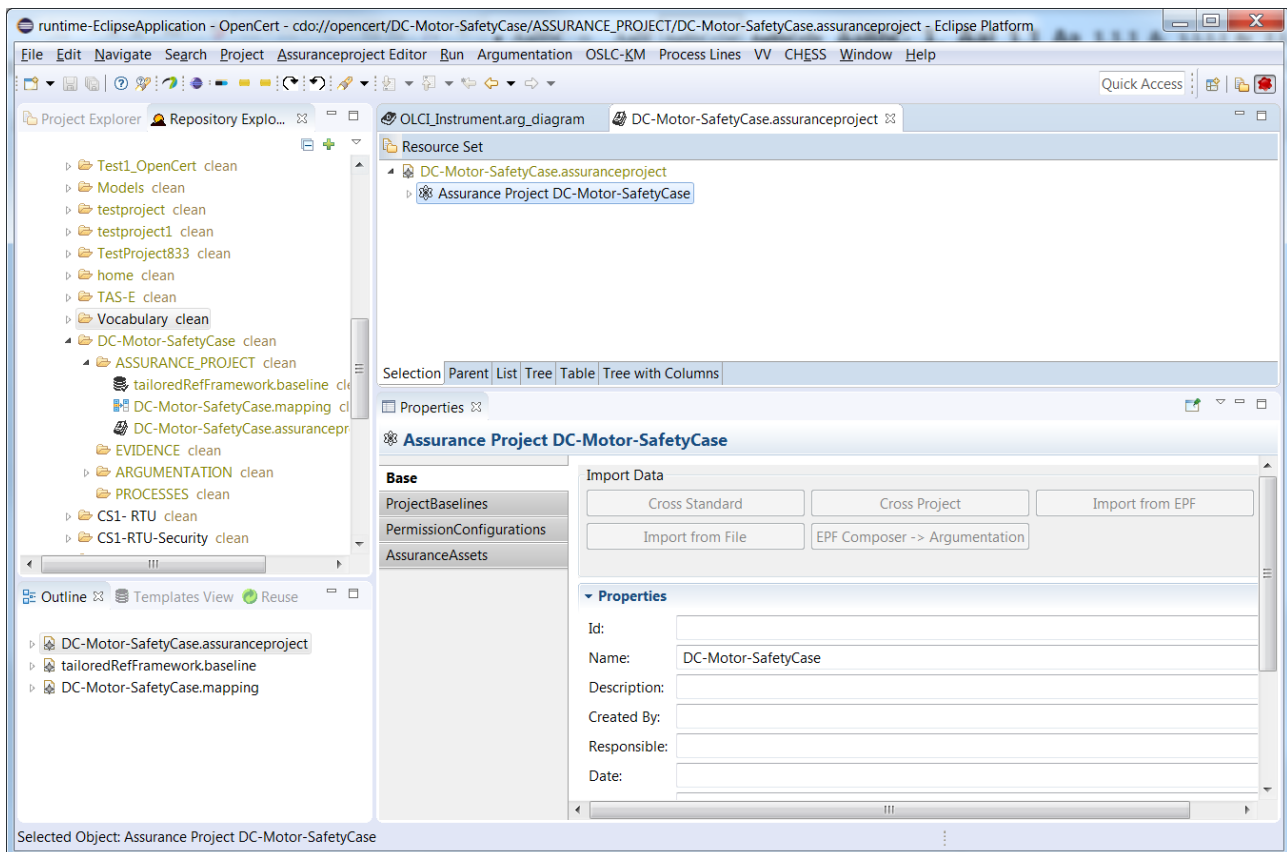


Figure 72. Import data options disabled for an Assurance Project with read only access.

2.2.4.2 'Log in the platform' in OpenCert (**)

The "Repository Explorer" is the Opencert component that opens a session with the CDO server in order to show to the user all the data stored in the CDO Repository. This component is the responsible of requesting the logging credentials to the user that wants to use the OpenCert tool.

The code below, in class CDOConnectionUtil of plugin org.eclipse.emf.cdo.dawn.util, requests the access data to the user and establishes the connection with the CDO repository using the credentials provided by the user.

```
public CDOSession openSession()
{
    ReconnectingCDOSessionConfiguration sessionConfiguration =
    CDOUtil.createReconnectingSessionConfiguration(host, repositoryName,
    IPluginContainer.INSTANCE);

    IPasswordCredentialsProvider credentialsProvider =

    (IPasswordCredentialsProvider)IPluginContainer.INSTANCE.getElement(CredentialsPro
viderFactory.PRODUCT_GROUP, "interactive", null);

    sessionConfiguration.setCredentialsProvider(credentialsProvider);

    currentSession = sessionConfiguration.openNet4jSession();

    IPluginContainer.INSTANCE.putElement(CDOSessionFactory.PRODUCT_GROUP,
"security", currentSession.toString(), currentSession);

    return currentSession;
}
```

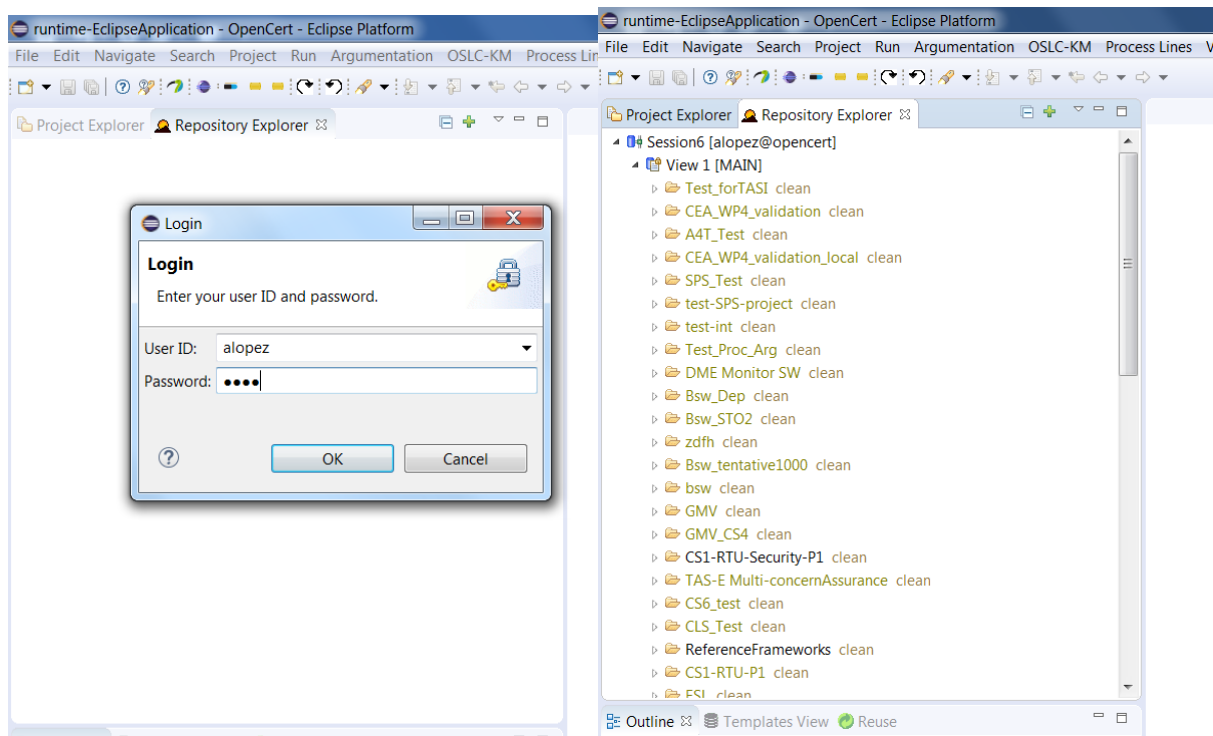



Figure 73. Authentication for Platform Access

The OpenCert platform also allows users to change their password through a contextual menu option.

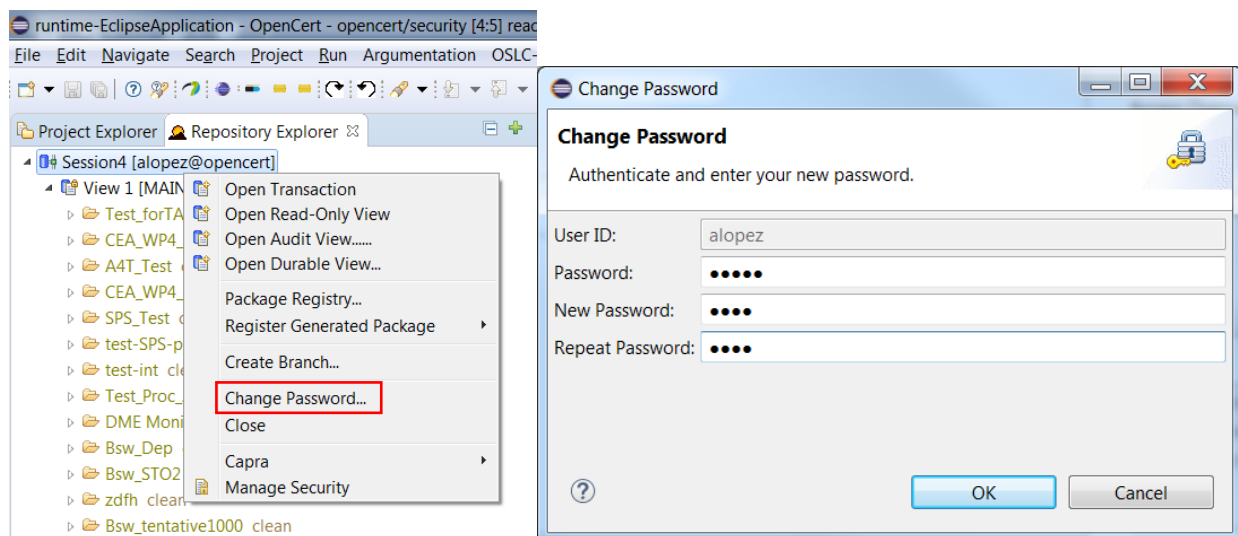


Figure 74. Change password window

2.2.4.3 'Concurrent Assurance Information Edition' with Web-based Technologies

An ultimate goal for the WP5 in AMASS is to provide means to support collaboratively work on the same document or model at the same time without locking. To offer a seamless experience to the user, the editing shall work in both (i) rich client (and tools) based on eclipse, as well as (ii) web-based clients (Figure 75). In this prototype, the collaboration between two web-based clients was the target.

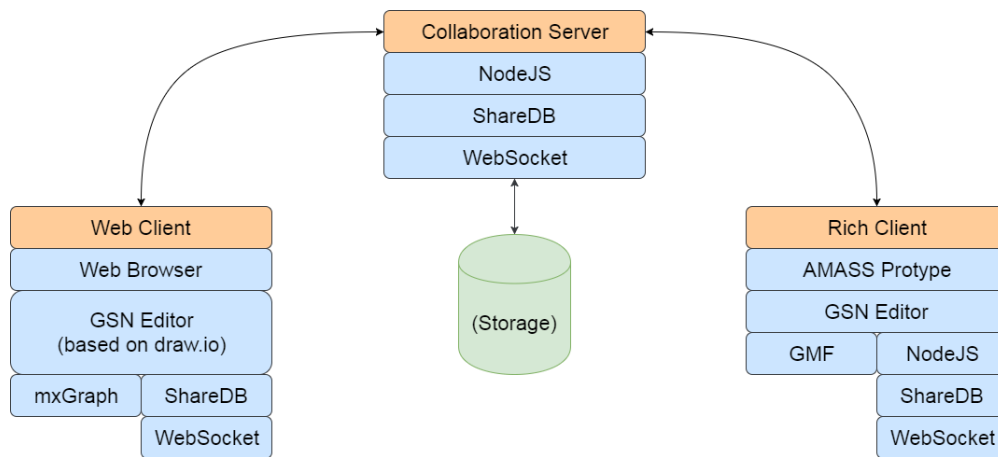


Figure 75. Ultimate picture of collaborative work using rich and web clients

The solution (Figure 76) is based on a NodeJS based server that handles all “Operational Transformations”, i.e. small pieces of change information (so called “mutations”) that are sent by all attached clients and that the server must bring into order, apply them and send them to all attached clients so they can be “eventually” consistent, meaning that all clients are up to date at a given point in time. The server was implemented using purely open-source software as Share DB / Share JS. As a proof of concept, a simple GSN editor was build (again) using open-source software as Draw.IO and mxGraph. Once the server is running, the clients may actively connect and after that all modifications done by any of the clients will appear also at other attached clients. Both, the client, but also the server, were built using the Node.js based build environment and require Node.js installed.

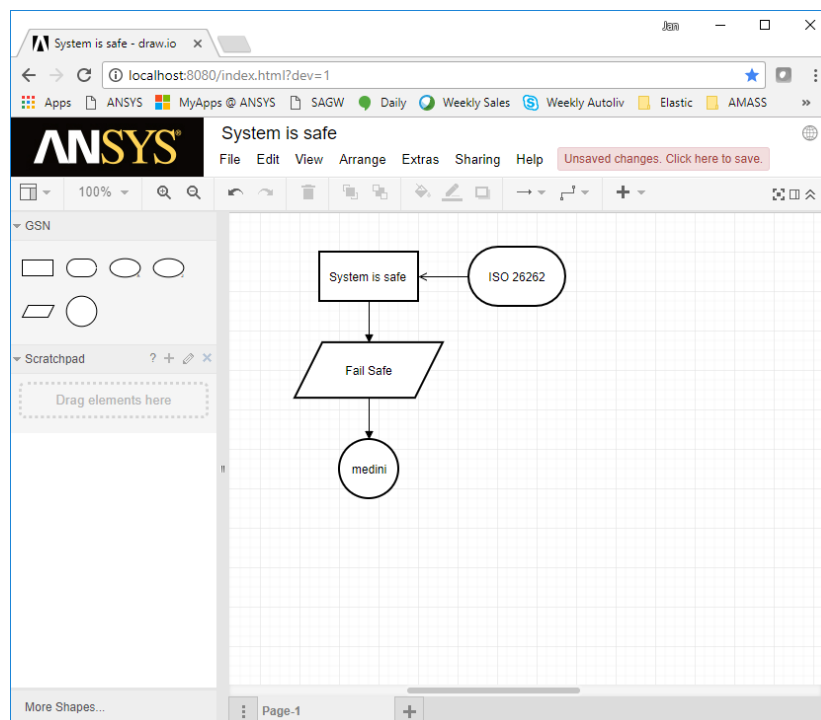


Figure 76. Screenshot of the current version of the web-based tool for collaborative model editing

2.2.4.4 ‘Concurrent Assurance Information Edition’ with Data Mining Technologies

The AMASS tools collect, create and aggregate a lot of data and relationships. It is essential to provide users, but also other functions and modules, a way to quickly search this big-data by means of keywords or other criteria. Based on the Elasticsearch open source software stack [24], a generic indexing feature is available in the platform. In this prototype, it is intentionally kept simple. Arbitrary EMF objects (local or

remote, file or CDO resource) can be indexed via the user interface of the prototype. Attributes and relationships are “crawled” by a generic and reflective indexer. The respective Ecore metamodel is used to decide whether an attribute value is indexed or not. The only pre-requisite is the configuration of the Elastic server (Figure 77).

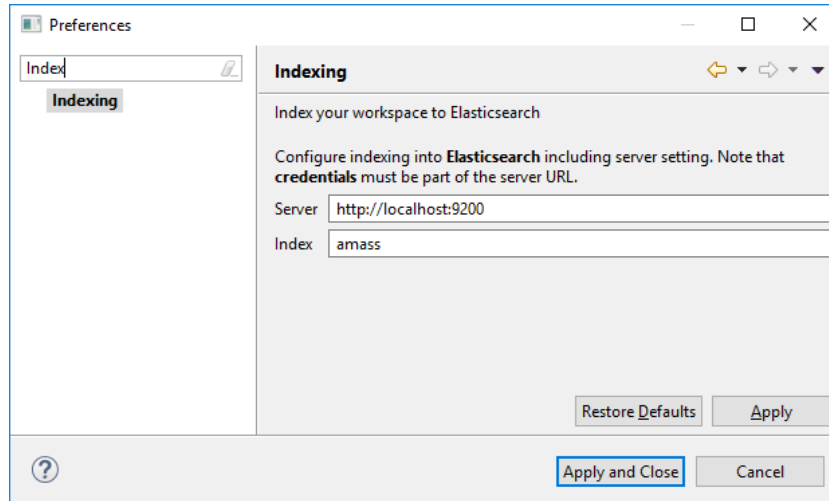


Figure 77. Screenshot of the Indexing configuration preferences

The user can select an arbitrary object and index the object, the object’s resource or the object tree into Elasticsearch (Figure 78).

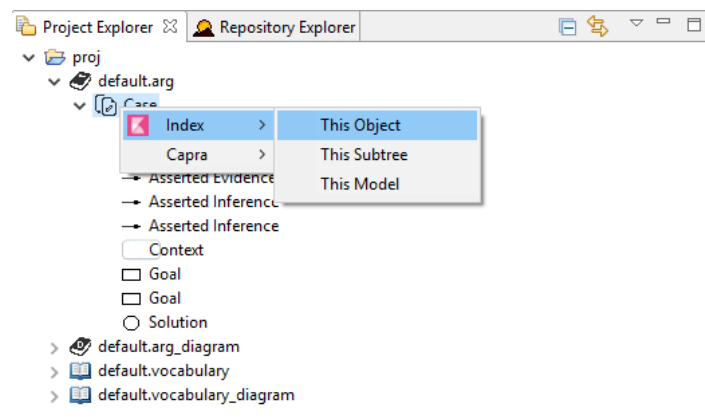


Figure 78. Screenshot of context menu to index data

The prototype further contains a web-based (google like) simple search application (Figure 79). The user may enter arbitrary keywords or other expressions following the Elastic search syntax [25]. The result can be further limited either by document type (here metaclass) or dedicated filters as for example ASIL – which was implemented as an example.

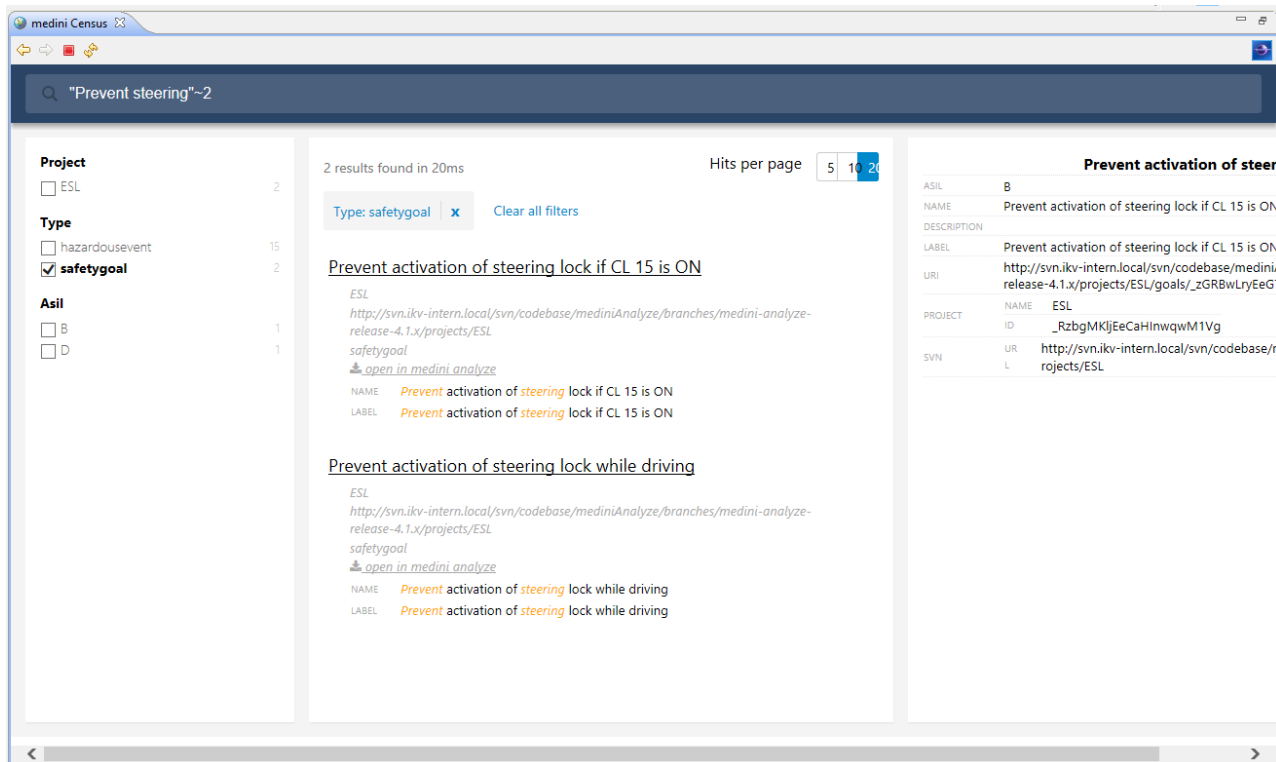


Figure 79. Screenshot of the Data Mining platform for collaborative work

The Kibana Dashboard Software (Figure 80) can be used to visualize all indexed data in a nice and understandable way.

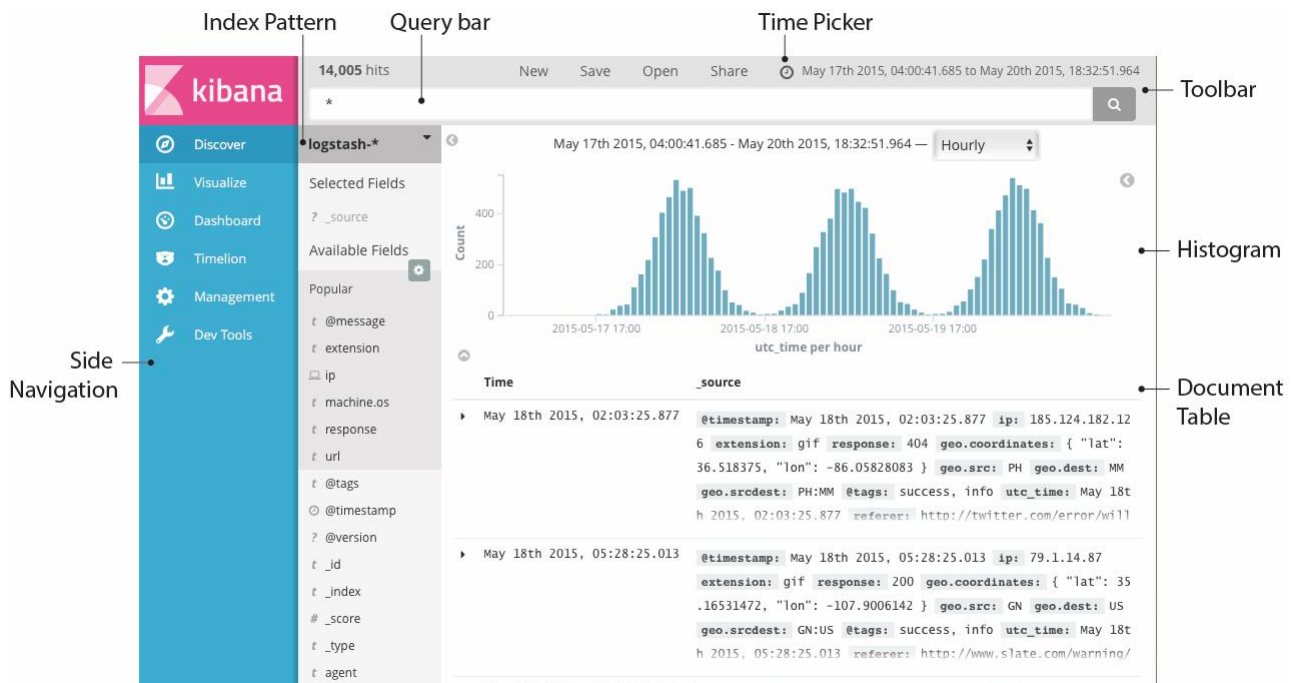


Figure 80. Screenshot of the Kibana Discovery tool

2.2.4.5 ‘Concurrent Assurance Information Edition’ through Automatic Translations (**)

In the frame of the SE Suite tools and the expertise of TRC together with the Seamless interoperability target of the WP5 of AMASS, we have added a new plugin on one of our new tools INTEROPERABILITY

Studio to automatically perform translations of requirements from one specification in one given language, for example, English into another language, such as French. In this way we can ensure the completeness and consistency of projects developed across different teams in different countries and different languages.

The technique is based on having equivalent ontologies for understanding requirements in those languages (Figure 81) and mapping the structures of information needed to understand requirements (also known as requirement-statement template or patterns) from one language to the equivalent in the target language (Figure 82).

Then the idea is quite simple, once this pattern is matched in the source language, the mappings to the target language are traversed and then the process is reversed in the target language, instead of matching patterns in a text, the tool produces languages by trying to instantiate the target pattern with the translations of the source input words.

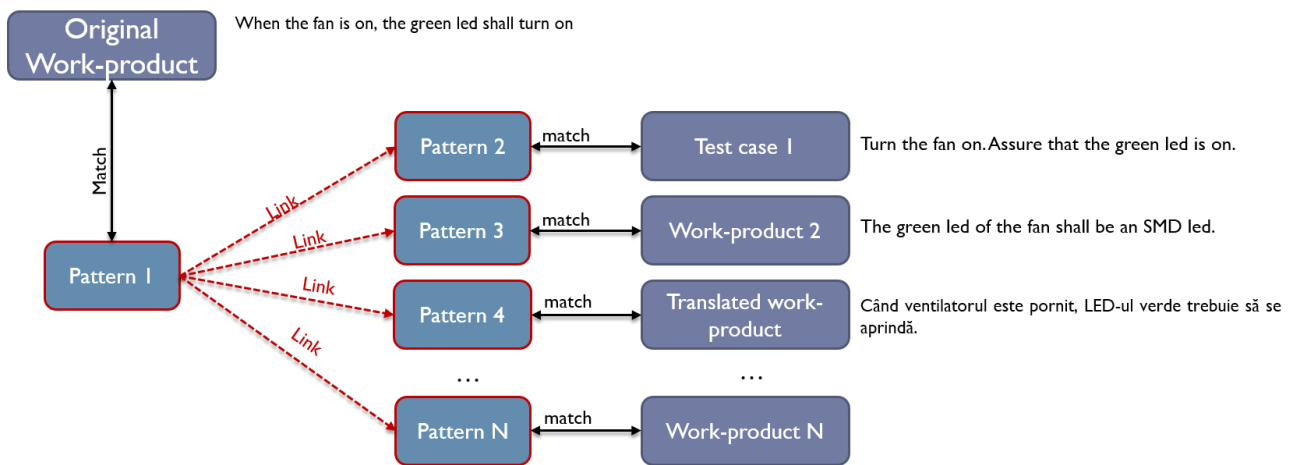


Figure 81. Automatic translations via Ontology patterns

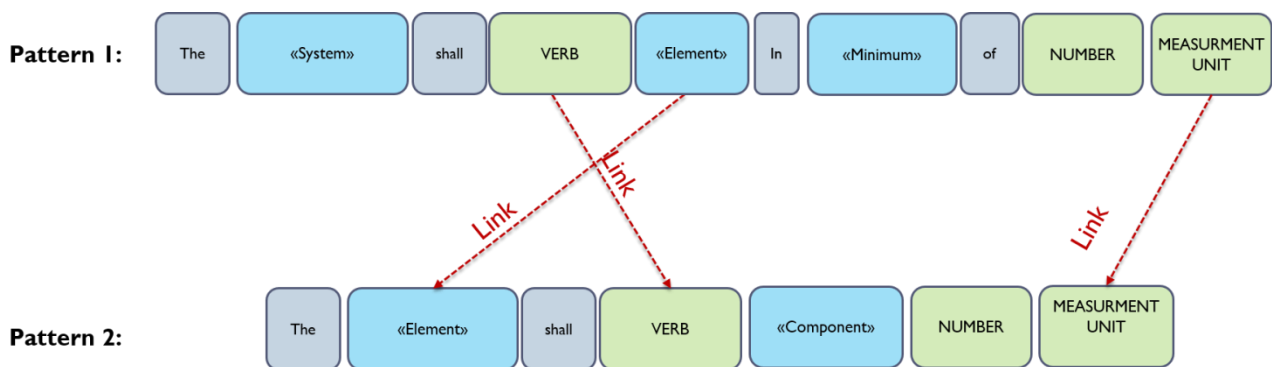


Figure 82. Linking of patterns across different languages

The goal is to generate a readable text in the target language. The main issue arises when coordinating the words with their context, they must match in gender, number and conjugation depending on their syntactical category.

We have reached an initial level of understandable text, but it needs further refinement for the next versions.

Inside INTEROPERABILITY Studio, the user can find the Transformation Manager (or R+ for short) (Figure 83) where there are a set of available transformations that can be parameterized and executed. Among these possible transformations we have included this automatic translation possibility, which, in fact, it's another type of transformation of the source work products.

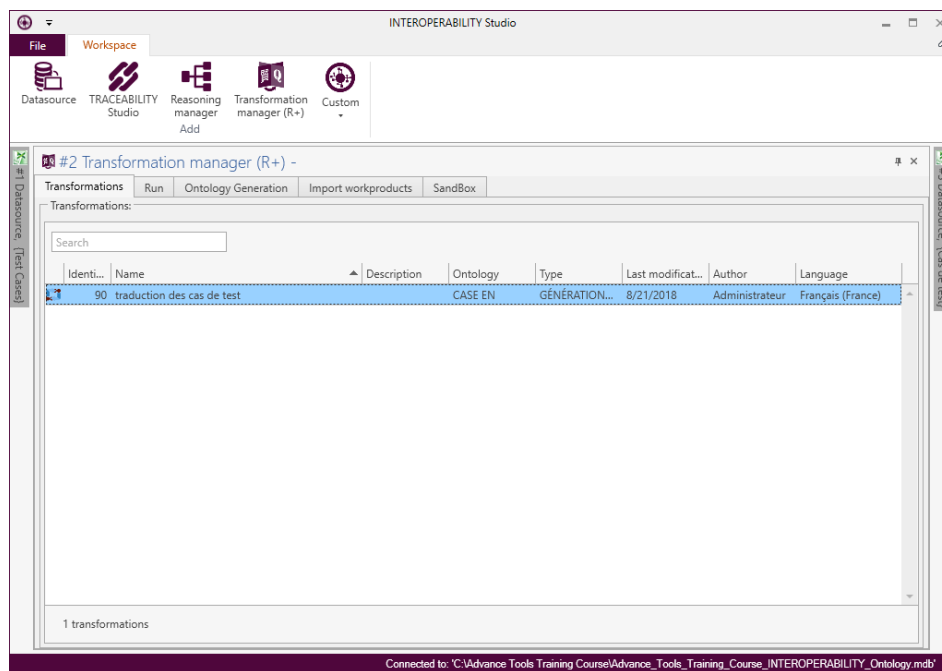


Figure 83. INTEROPERABILITY Studio > Transformation manager

After selecting the desired translation transformation, the source and target specifications must be provided as shown in Figure 84. Then, other parameters of the execution can be managed at the bottom of that window. When all the parameters have been set up, the translation can be performed by clicking in the “Translate” button.

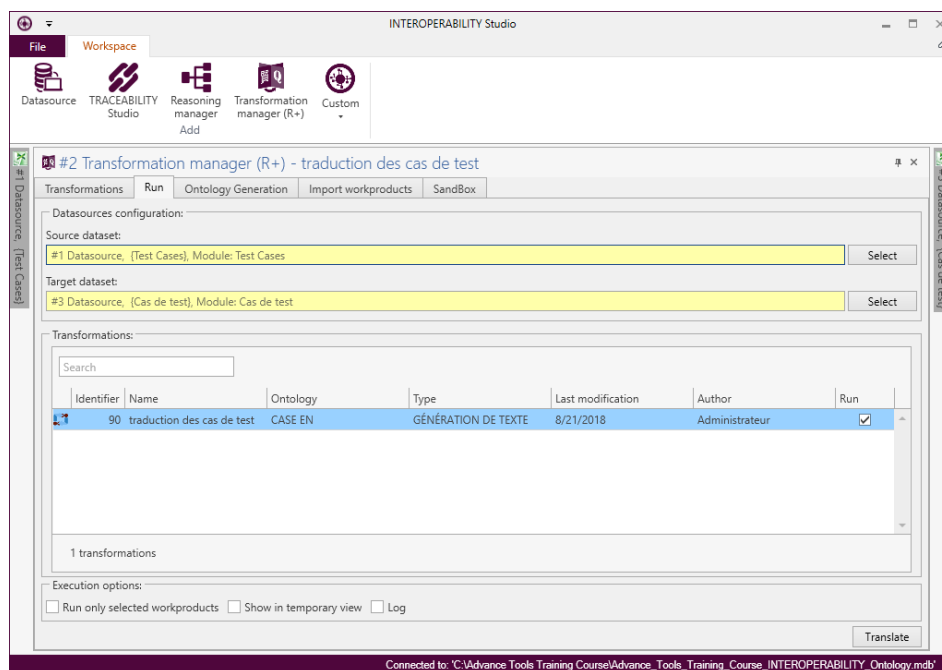


Figure 84. INTEROPERABILTY Studio > Transformation parameterization

Other possibility for the execution of the translation transformation is from the point of view of the source specification (Figure 85), there are contextual menu options displaying the possible transformation available for that source and just by selecting it, the translation is performed automatically.

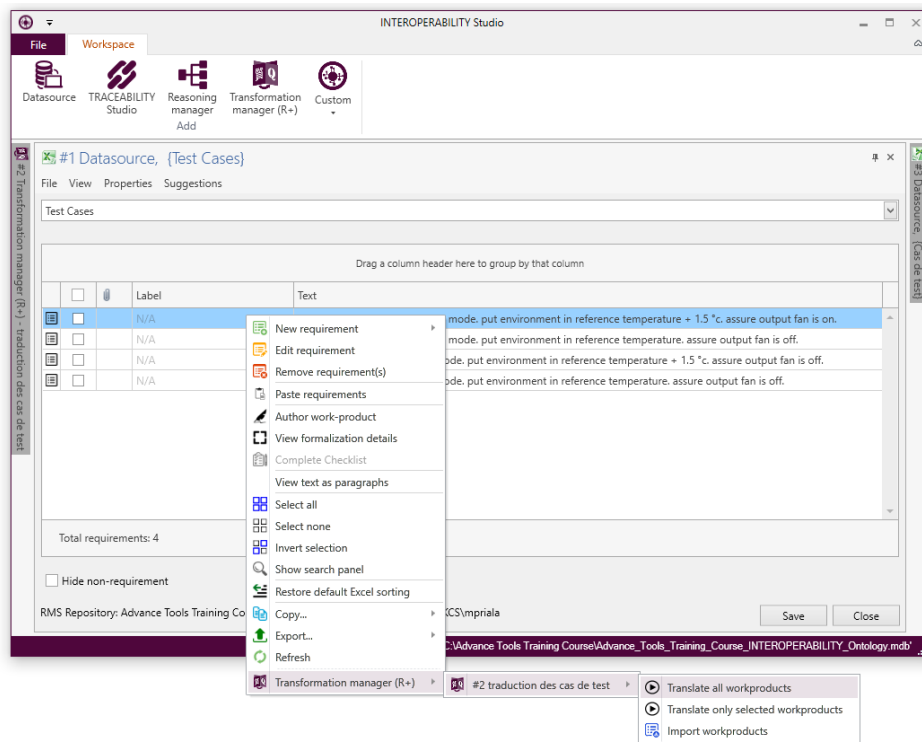


Figure 85. INTEROPERABILITY Studio > Specification point of view

After the translation transformation process has been finished, a new window will appear showing its log information (Figure 86). Then this output window can be closed by clicking in the “OK” button.

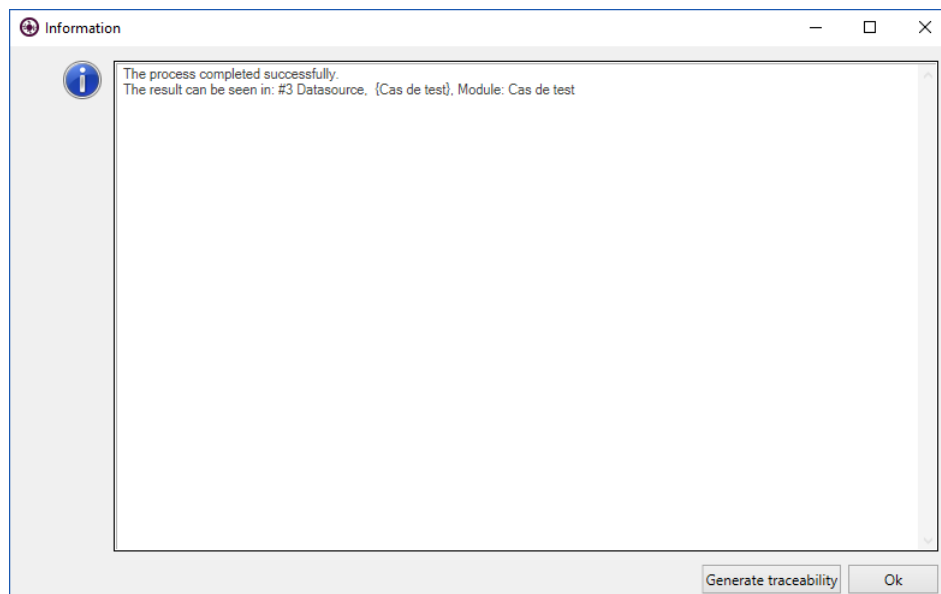


Figure 86. Transformation output log

Then, when navigating in INTEROPERABILITY Studio to the selected target specification (Figure 87) the user can see the new work products (green cell background) that have been added by the process.

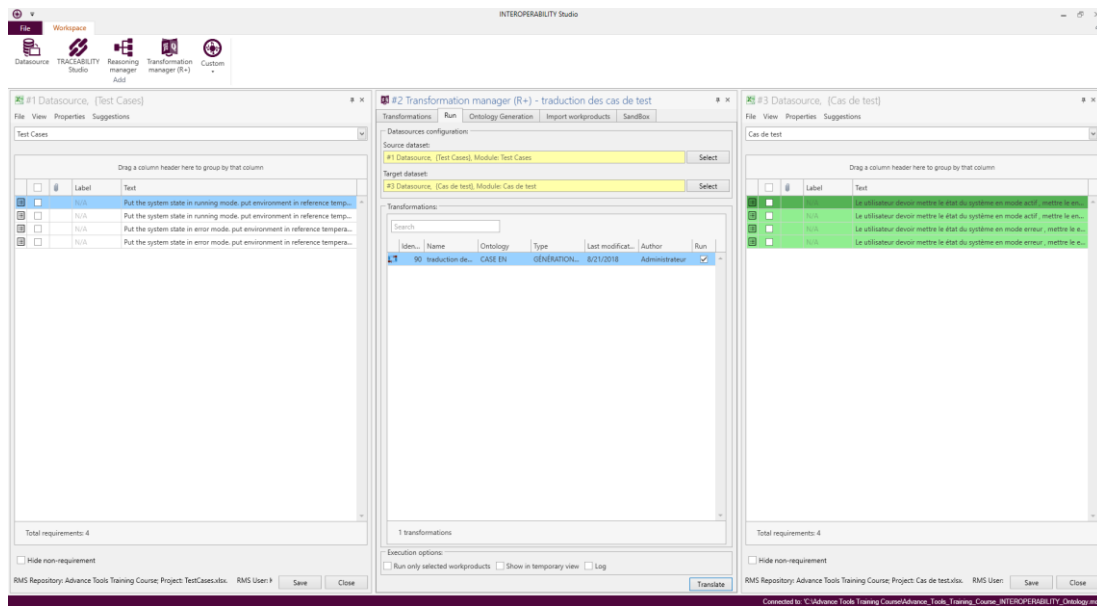


Figure 87. Target specification showing new work products generated automatically

Finally, a SandBox feature (Figure 88) has been included in the tool to allow the user to fine tune the parameters needed to customize the translation transformation.

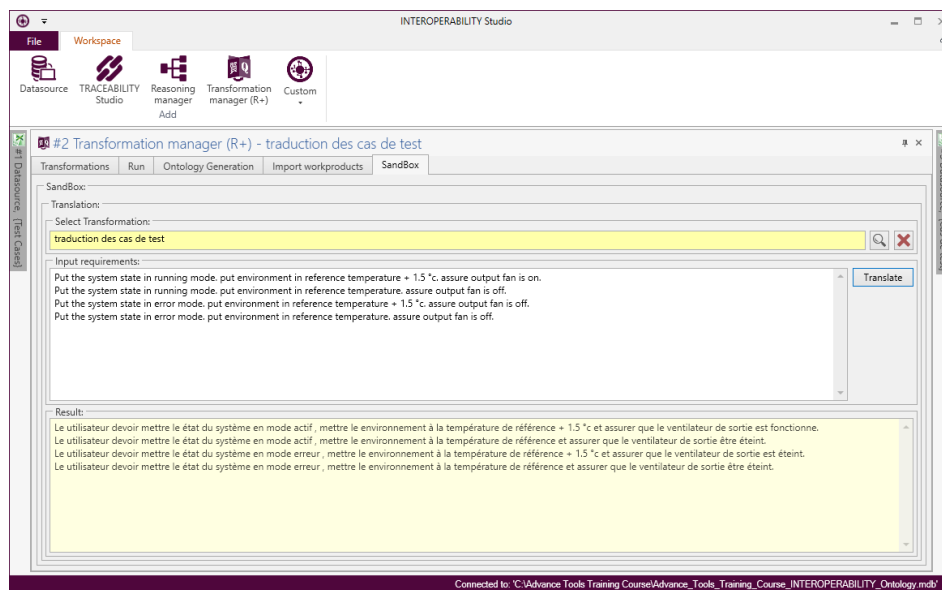


Figure 88. INTEROPERABILITY Studio > SandBox window

2.2.4.6 'Concurrent Assurance Information Edition' in OpenCert (**)

OpenCert supports real-time collaboration on models edition by transferring the changes that one user commits to the repository to all other users connected to the same repository and transparently weaving those changes into their model copies.

By default, model elements are locked optimistically, that is, the CDO server implicitly acquires and releases locks while executing a commit operation the user tries the save the changes made to the model stored in the CDO repository. These implicit locks are not visible to the committing user or any other user of the same repository.

Optimistic locking provides for the highest possible degree of concurrency, but it also comes with a non-zero risk of commit conflicts that are only detected when a commit operation is executed by the CDO

server and, consequently, rejected. The following pop-up message (see Figure 89) is shown to the users when a commit operation is rejected due a conflict.

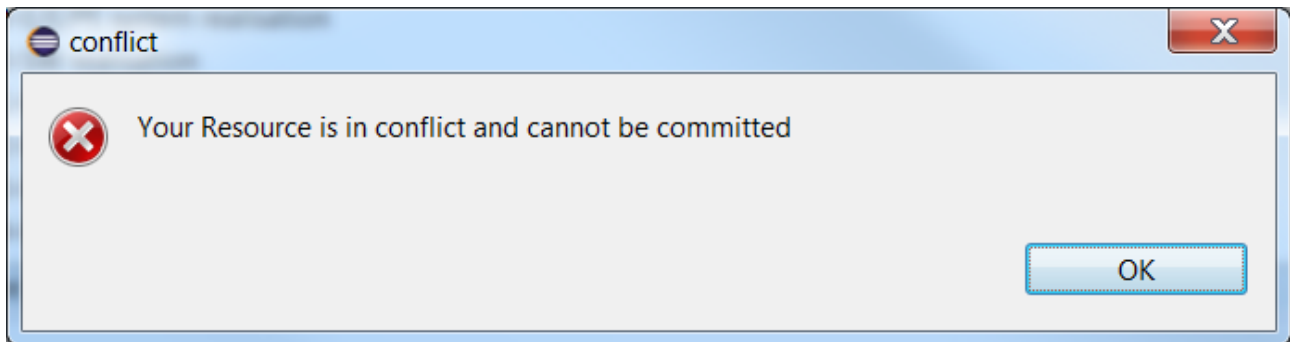


Figure 89. Conflict message when saving a model.

As the local model copies of a user are automatically updated at the time they are changed by other users, CDO can anticipate the potential conflict of the local changes early, in fact, before an attempt to commit those changes is even made. OpenCert marks such conflicting model elements with a red bold font for tree-based editors and with red borders for graphical ones.

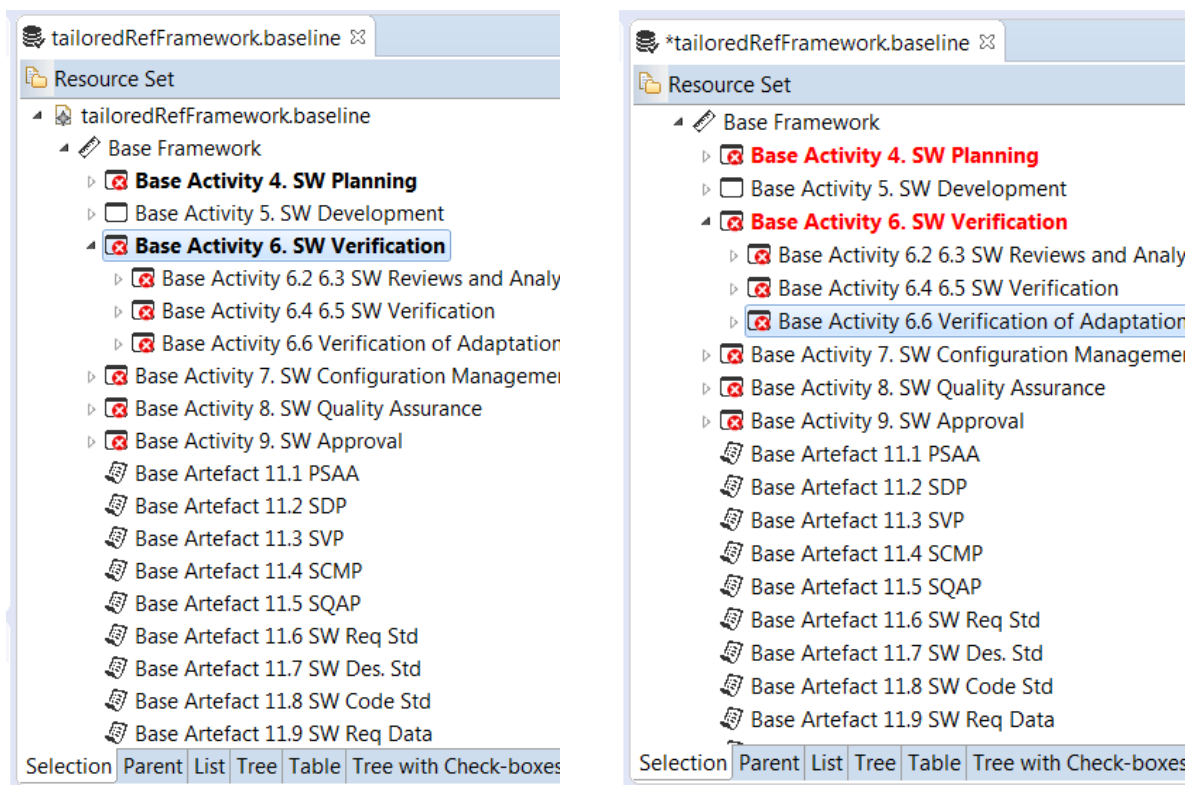


Figure 90. Two users edit the same model elements. When the user 1 (left editor) saves the changes, the conflicted elements are marked in red to the user 2 (right editor)

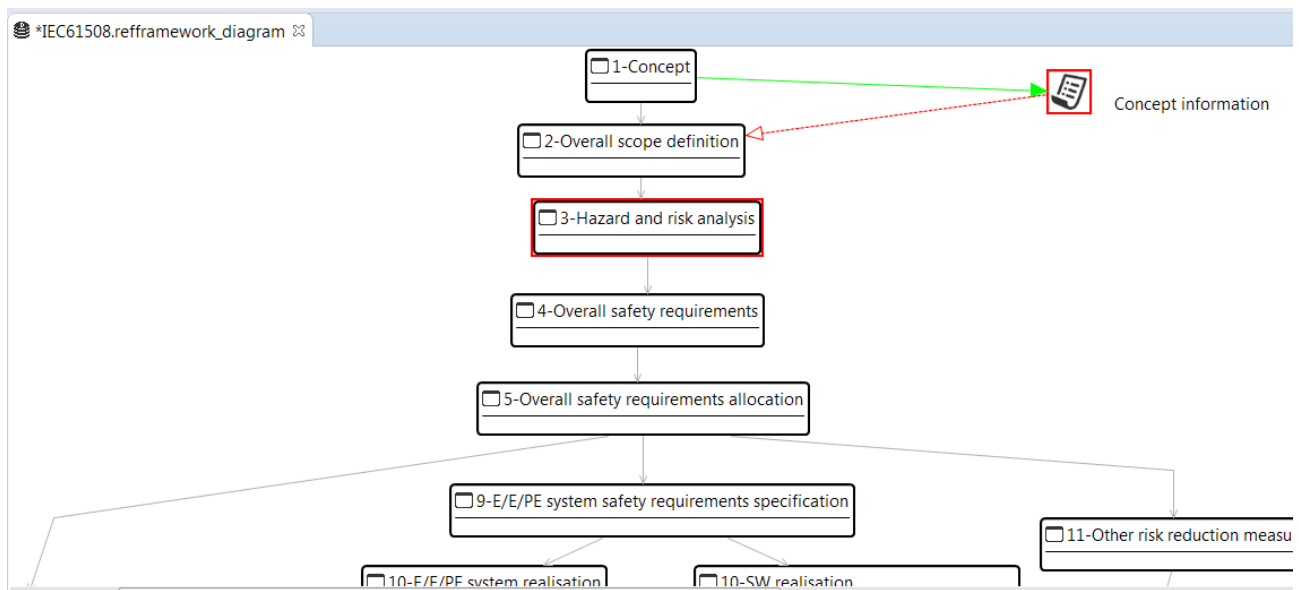


Figure 91. Conflicted elements bordered in red in a graphical editor.

Each time a local transaction is notified of a remote change by the CDO server and local conflicts are detected, those conflicts are categorized as being either trivial conflicts or non-trivial conflicts. Trivial conflicts are:

- Changes to multi-valued features on both sides (local and remote) of the same model element.
- Changes to different single-valued features on both sides (local and remote) of the same model element.

Trivial conflicts are merged automatically into the local transaction; therefore, no user interaction is involved.

When non-trivial changes are detected, i.e., changes to the same single-valued feature on both sides (local and remote) of the same model element, the collaboration of the user to solve the conflict is needed, for that, OpenCert has and Interactive Conflict Resolution feature accessible through a context menu (see Figure 92).

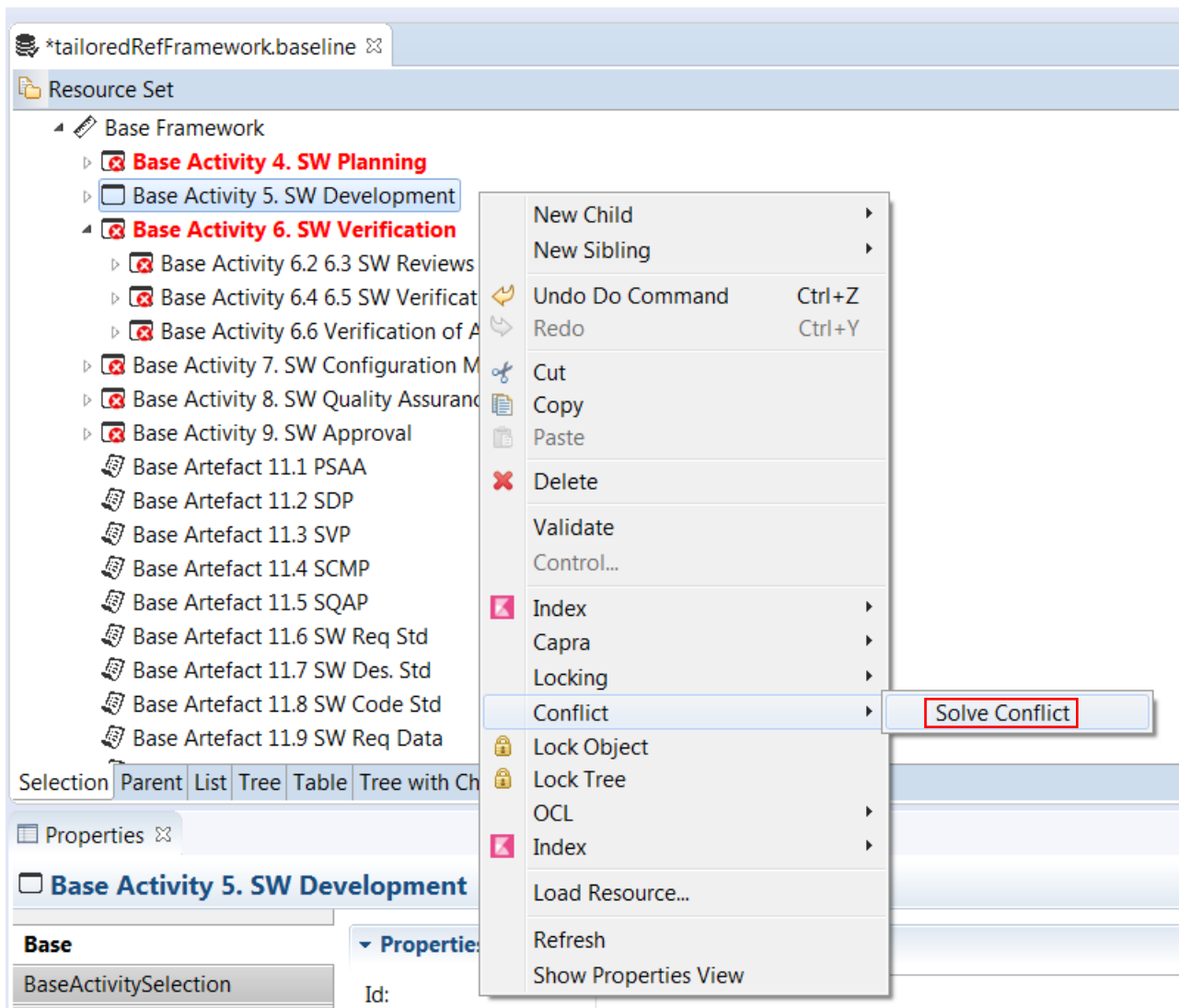


Figure 92. Interactive Conflict Resolution context menu.

The Solve Conflict asks the user if s/he wants to solve the conflict with a rollback operation, it means that the local model copies are automatically updated to their latest remote versions. As a result, all local changes will be lost and need to be re-applied and committed again.

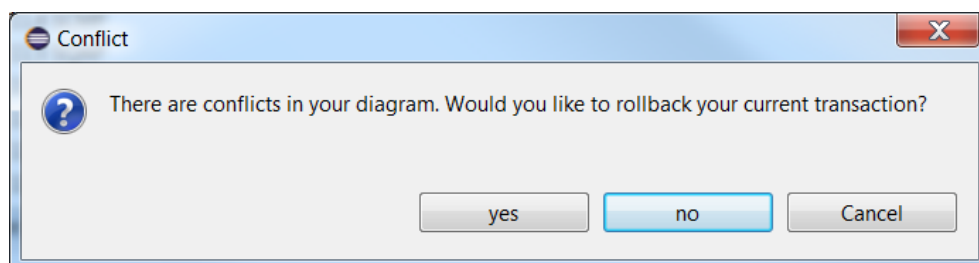


Figure 93. Rollback resolution

OpenCert checks the possibility of creating explicit locks on selected model elements avoiding the modification of those locked parts by other users. It's possible to lock just a single model element ("Lock/Unlock Object" option) or lock the tree of model elements rooted at the selected model element ("Lock/Unlock Tree" option).

This feature is only available for the tree-based model editor using the context menu over the desired model element to Lock or Unlock.

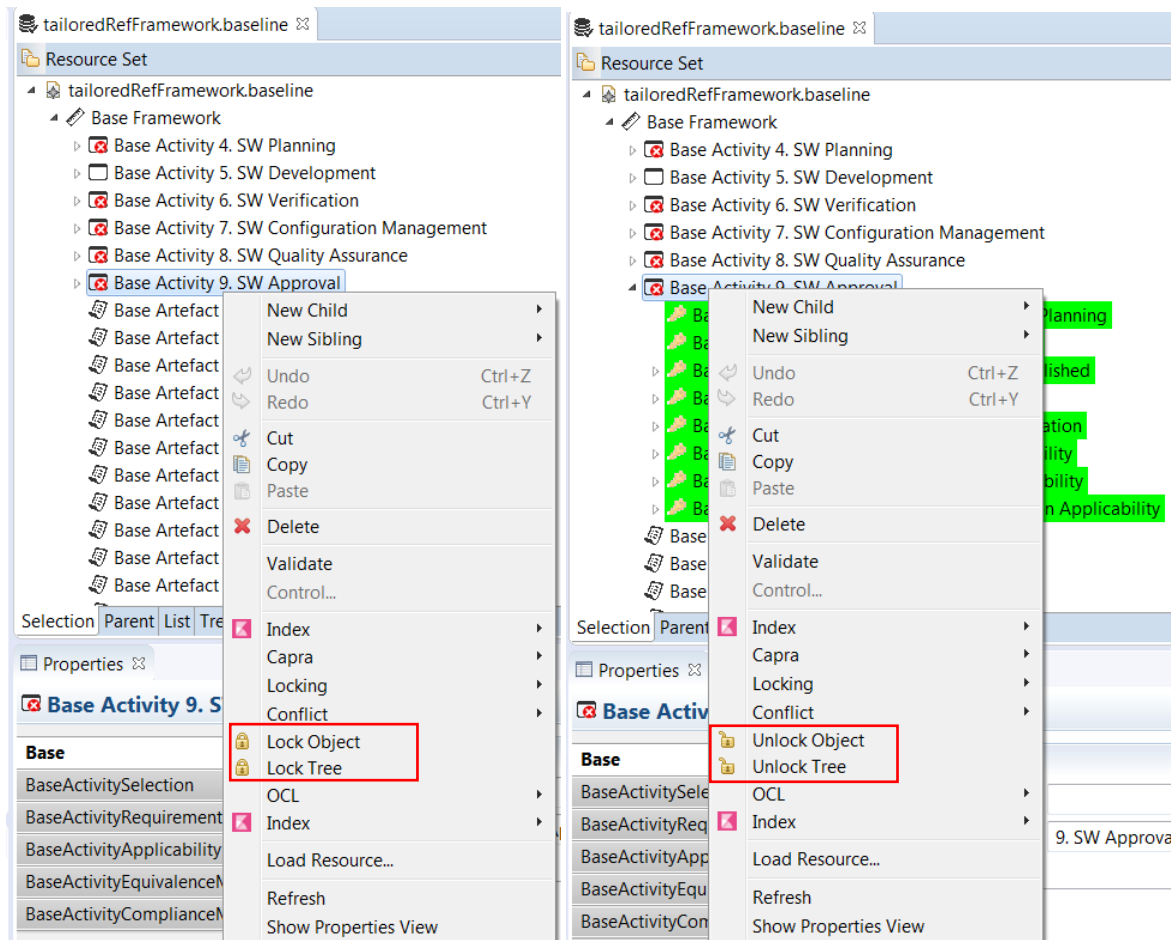


Figure 94. Options to Lock/Unlock model elements

The locked elements are shown differently to the locker user, highlighted in green with a key icon, and to the rest of users, highlighted in yellow with a closed padlock icon (see Figure 95).

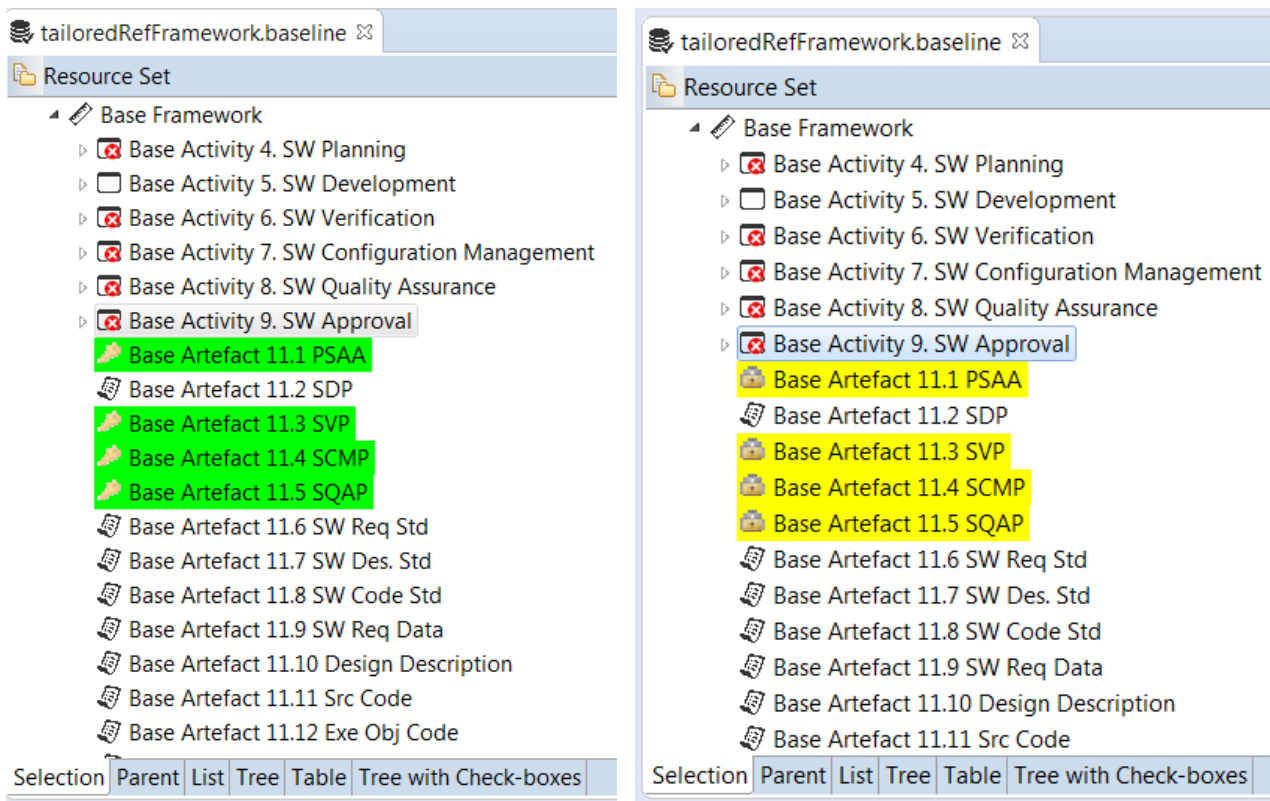


Figure 95. Lock state visualization in editors (locker user editor in the left)

2.2.4.7 Concurrent System Architecture Edition in OpenCert (**)

Collaborative work related to system architecture modelling is made available in AMASS by using the Papyrus support for CDO; moreover, in the context of AMASS, CHES has been extended to allow the usage of the CHES modelling language (e.g. the contract-based extension) and the Papyrus editor extensions while using CDO as model repository.

While creating a new CHES project in a CDO repository, the wizard creates a folder that contains the new model and notation resources (i.e. the .di, .notation and .uml resources), as well as a text file. project that it is used to specify the CHES *nature*⁵ of the created project.

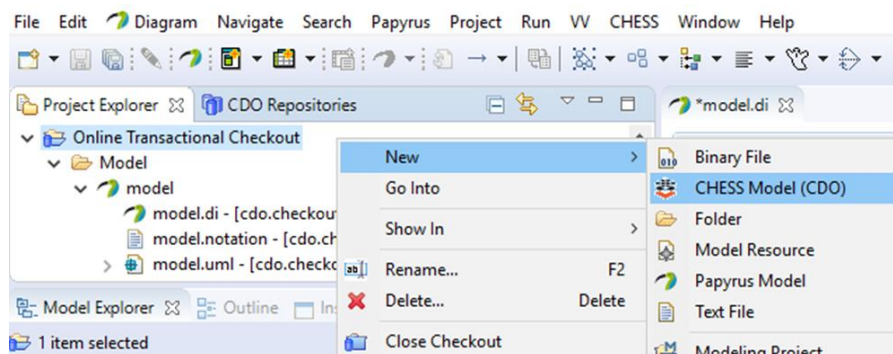


Figure 96. Creating a new CHES Model in CDO

⁵ In Eclipse, project natures allow to tag a project as a specific kind of project. Project natures allow to indicate that a certain tool is used to operate on that project. They can also be used to distinguish projects that plug-in is interested in from the rest of the projects in the workspace.

As further feature, the import/export of CHESS models from workspace file-based projects to CDO projects have been implemented.

Concerning CHESS support for model-based analysis, at the time of writing only the Concerto FLA analysis [7] is supported while working within CDO; to enable the other kind of analysis it is necessary to switch the CHESS CDO project to the file-based version by using the aforementioned CDO export feature.

2.3 Installation and User Manuals (*)

The steps necessary to install the Prototype P2 are exhaustively described in the AMASS User Manual [13] (currently under elaboration for all the AMASS building blocks), thus they are not repeated in this deliverable. In the user manual of the Prototype P2 the users can find the installation instructions, the tool environment description, and the functionality for the specification of evidence-related assurance project information (artefact repository preferences, artefact definitions, artefacts, artefact resources, artefact property values, artefact events, artefact evaluations, impact analysis, executed processes, and property models) and tool integration, among other features.

3. Implementation Description (*)

This section presents the modules that have been implemented, the underlying metamodel, and the source code created.

3.1 Implemented Modules (*)

The modules implemented for the AMASS Prototype P2 in the scope of WP5 are as follows:

- **Platform Management** (Figure 97)
 - *Access Management*
This module is integrated in OpenCert and uses CDO [16] as the main base technology.
 - *Data Management*
This module is integrated in OpenCert and uses CDO [16] as the main base technology.
 - *Collaborative Work*
In addition to some basic support for collaborative work provided by OpenCert (e.g. through CDO features for concurrent data access), the tools that currently implement collaborative work functionality are: Capra, the web-based approach for concurrent assurance information edition, Elasticsearch, and Kibana.
- **Evidence Management** (Figure 98)
 - *Evidence Characterization Editor*
OpenCert implements this module. It is an Eclipse-Based editor for artefact and executed-process information of an assurance project. It contains plugins for edition of artefact models and of process models, and to provide services for evidence storage (determination, specification, and structuring of evidence), and for traceability-related aspects. The editors have been mostly generated with the EMF [18] and EEF [17] Eclipse technologies, in addition to the implementation of some tailored functionality, e.g. for integration with SVN and for impact analysis.
- **Assurance Traceability** (Figure 99)
 - *Traceability Management*
The Evidence Characterisation Editor provides support for evidence traceability. This functionality is complemented with the use of Capra. Further support is provided by Traceability Studio.
 - *Impact Analysis*
The impact analysis support is currently embedded in the Evidence Characterisation Editor. Further support is provided by the Traceability Studio.
- **Tool Integration** (Figure 100)
 - *Toolchain Management*
The current support for Toolchain Management is integrated in OpenCert and CHESS. Each tool integration technology has a dedicated user interface.
 - *Tool Connector*
Each tool integration technology described above has its own Tool Connector component: connector for SVN, connector based on OSLC-KM, etc.

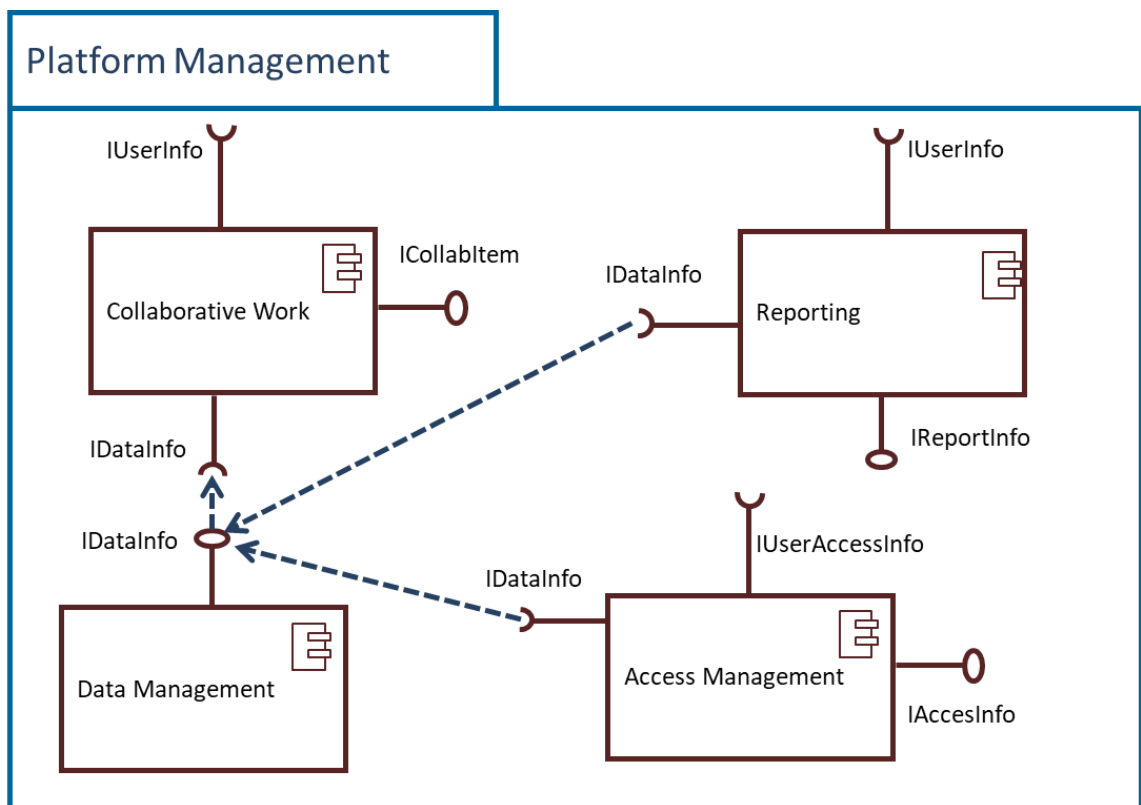


Figure 97. Platform management modules

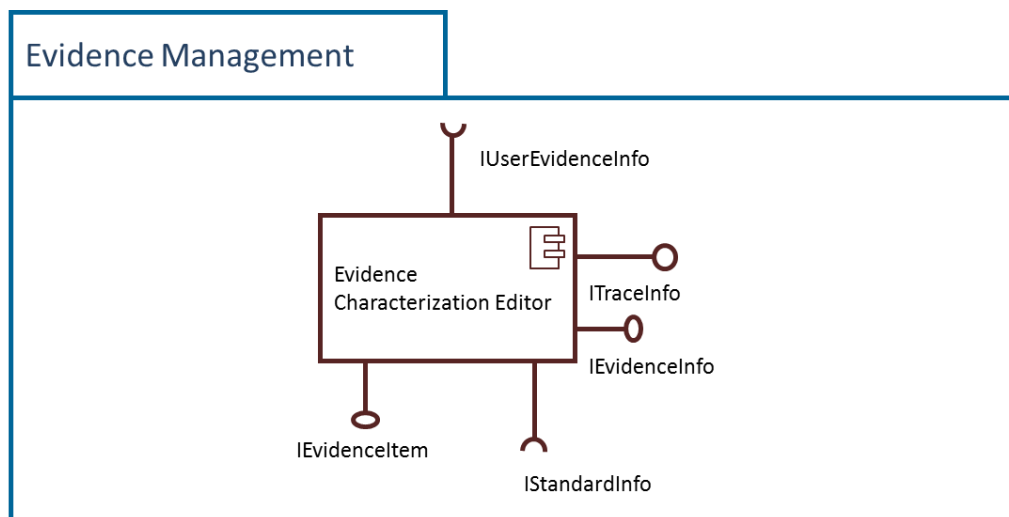


Figure 98. Evidence management module

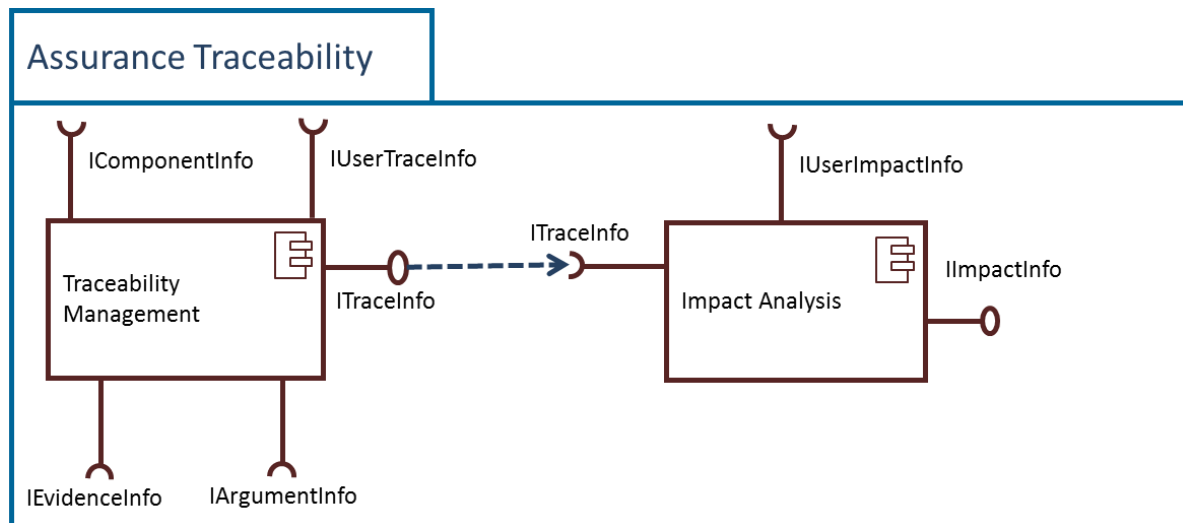


Figure 99. Assurance traceability modules

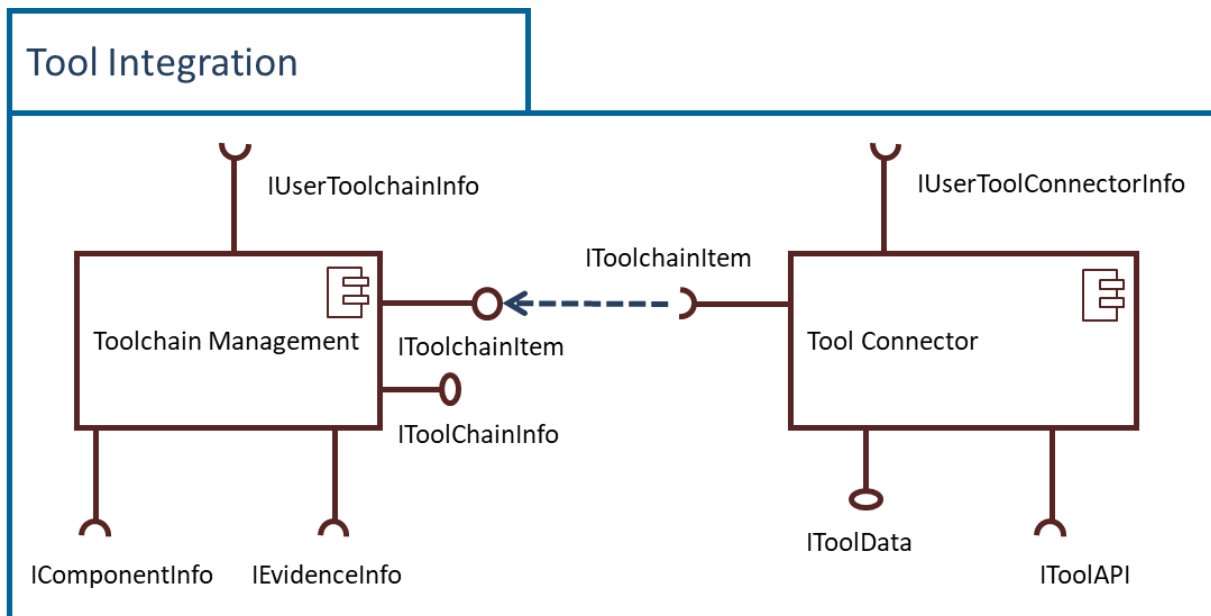


Figure 100. Tool integration modules

3.2 Implemented Metamodel

AMASS D2.4 [4] presents the CACM, including evidence management metamodels. These metamodels correspond to the envisioned, conceptual data structure necessary in AMASS to provide the reuse-oriented holistic approach for architecture-driven assurance, multi-concern assurance, and seamless interoperability. However, the metamodel implemented for the Evidence Management modules does not exactly correspond to the CACM, but to the metamodel implemented in OpenCert.

Such metamodel is the CCL created in the OPENCOS project. The CCL can be regarded as compliant with the CACM because it supports all the evidence information specification needs represented in the CACM. However, the specification of information can be a bit different. For example, traceability information is not specified in the CCL based on a specific metamodel, but this information type is embedded in the CCL artefact metamodel.

Figure 101 shows an excerpt of the CCL to specify evidence information. Further information about the CCL can be found in [20].

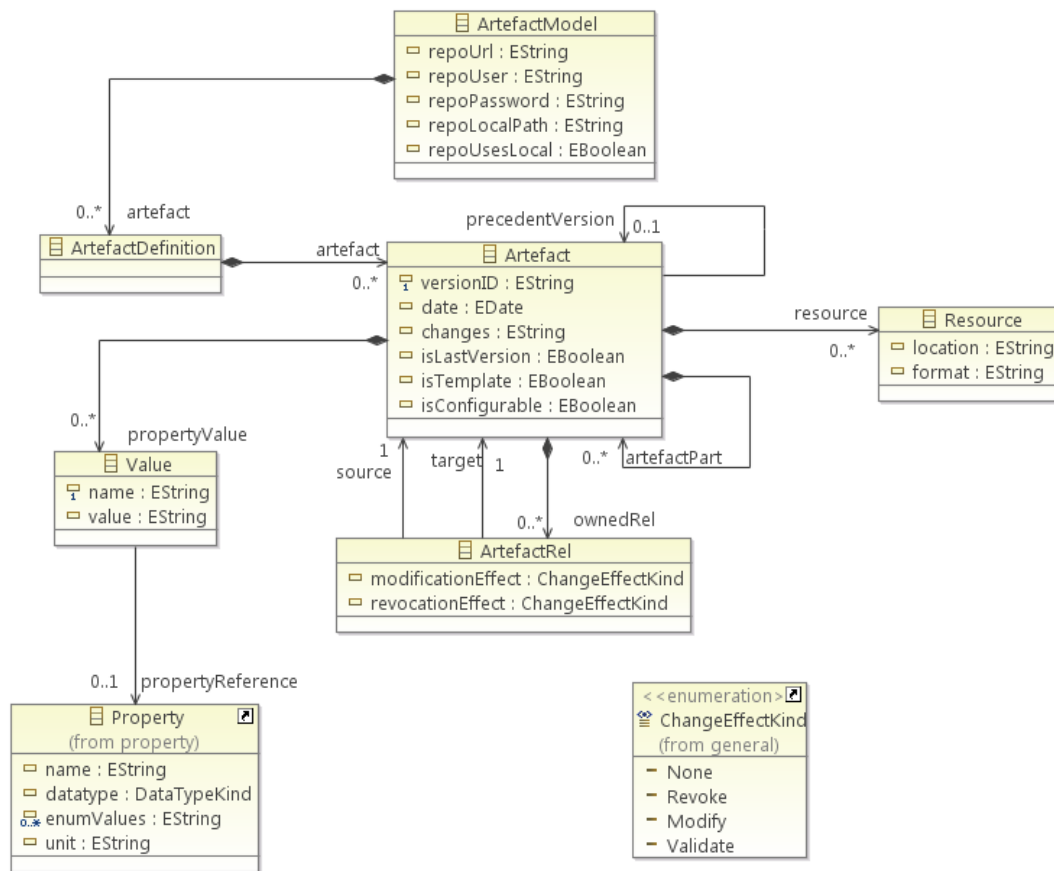


Figure 101. Excerpt of artefact information in the CCL

3.3 Source Code Description for the AMASS Tool Platform (*)

The source code of the second AMASS prototype can be found in the source code SVN repository [15]. The code for Prototype P2 evidence management and system management modules are stored together with the other basic building blocks in the repository under “tag” to distinguish the state of the code at the time of the integrated release.

The necessary plugins for Seamless Interoperability (Figure 102) are:

- **org.eclipse.opencert.evm.evidspes**
In this plugin, the evidence metamodel is defined and stored, and the Java implementation classes for this model are generated.
- **org.eclipse.opencert.evm.evidspes.edit**
This plugin contains a provider to display evidence models in a user interface.
- **org.eclipse.opencert.evm.evidspes.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.
- **org.eclipse.opencert.evm.evidspes.editor.dawn**
This plugin is an extension of the previous one. It aims to communicate with the CDO Server to store the generated model.
- **org.eclipse.opencert.evm.evidspec.preferences**

This plugin defines the default preferences for the communication with the SVN repository, thus it defines the type of repository (local or remote) and a user and password to connect with the remote repository.

- **org.eclipse.opencert.evm.oslc.km.importevid**

This plugin contains all the classes needed to:

1. Request all the parameters needed to insert the structured content of a file as evidences in the Evidence Manager: type of file, file, additional transformation options and destination in the AMASS repository: Assurance Project and evidence name.
2. Perform a request to the SE Suite web-service sending the selected file content and receiving the OSLC-KM model in form of JSON.
3. Build from the JSON string the OSLC-KM model.
4. Parsing the OSLC-KM model into an ArtefactModel from the Evidence Manager.
5. Store the ArtefactModel in the AMASS repository (CDO database).

Regarding the jar file needed to perform step 2, the source code is publicly available at <https://github.com/trc-research/oslc-km>

- **org.eclipse.opencert.evm.oslc.km.importevid**

This plugin contains all the classes needed to add another option in the Preferences window to select the URL of the OSLC-KM Web Service used in the OSLC-KM importer.

- **org.eclipse.opencert.impactanalysis**

This plugin contains the implementation of the change impact analysis module. This module is used by AMASS Tool Platform clients to call and execute change impact analysis.

- **org.eclipse.opencert.infra.properties**

This plugin contains the definition of the Property metamodel, and the Java implementation classes for this model.

- **org.eclipse.opencert.infra.properties.edit**

As the edit plugin for evidence, this plugin contains a provider to display the model in a user interface.

- **org.eclipse.opencert.infra.properties.editor**

As the edit plugin for evidence, this plugin is an editor to create and modify instances of the model.

- **org.eclipse.opencert.infra.svnkit**

In this plugin, the functionalities necessary for the communication with the SVN repository to export and import artefacts are defined.

- **org.eclipse.opencert.pam.procspec**

In this plugin, the process execution 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 execution models in a user interface.

- **org.eclipse.opencert.pam.procspec.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.

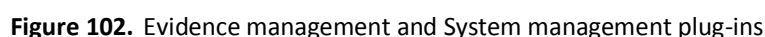
- **org.eclipse.opencert.pam.procspec.editor.dawn**

This plugin is an extension of the previous one. It aims to communicate with the CDO Server to store the generated model.

- **org.eclipse.opencert.storage.cdo**

This plugin contains classes for using the CDO server in the AMASS Tool Platform. This server provides a common storage for all AMASS Tool Platform clients and a server. It accesses PostgreSQL database as its data backend. In addition to common storage implementation, this

This plugin contains classes that implement the user interface and control for the management of the traceability links between CHESS and the other parts of the OpenCert models (as presented in section 2.2.2.2). These classes use the API provided by Capra and use the different kind of trace links provided by the `org.eclipse.opencert.chess.tracemodel` plugin.



- **org.polarsys.chess.cdo**

This plugin contains classes that implement the support for CDO while working with the CHESSE projects. It supports the creation of the CHESSE model, with its structure, in the selected CDO repository and the usage of the CHESSE model profile and Papyrus editor customization. In addition, it provides capabilities to import and export CHESSE projects from CDO to workspace file-based repositories.

3.4 Source Code Description for External Tools (*)

This section describes the source code for features that have been implemented for Seamless Interoperability but have not been integrated into the AMASS Tool Platform.

3.4.1 Seamless Interoperability Features in Systems Engineering Suite by TRC (**)

3.4.1.1 OSLC-KM standard and OSLC-KM implementation (**)

The approach for Seamless Interoperability has been divided into two different steps.

- 1) The first step has been defining a standard to represent all kinds of knowledge, which has been named OSLC Knowledge Management (OSLC-KM). From this standard, all the operations inside the SE Suite by TRC have been defined using it as input instead of defining a connection for each possible different model source.

Once this OSLC-KM model has been introduced as input to SE Suite, the tool creates what is called a specification composed of work products, for example, the requirements found in the model in the Papyrus file and exposes it to the rest of functionality of SE Suite. This will allow to assess its quality in many different perspectives:

- Correct: in the scope of the individual work product
- Complete: in the scope of the specification
- Consistent: in the scope of the specification

- 2) Then the second step has been to map the contents of the model represented in a file, e.g. a Papyrus model, into an instance of the OSLC-KM standard. This has been achieved by using a technology called Extensible Stylesheet Language Transformations (XSLT). For each possible source of models, a XSLT file has been created to map the entities from that model to the entities of the OSLC-KM model.

The final goal in the long term, and outside the scope of the AMASS project, is that every model tool manufacturer will be able to create their implementation of the OSLC-KM model inside their tools, and exposing it via a web service, so that the AMASS platform or SE Suite can consume it without having to execute this transformation, and the OSLC-KM model can be more complete in the sense that not every piece of information of the model stored in the file can be extracted and mapped in the OSLC-KM model. This will create a better representation of the model, thus better results to analyse it.

The implementation of this functionality has been developed inside the SE Suite by TRC and it's composed of several libraries:

- **Rqa.Face.OslcKm**: it implements the connection of the OSLC-KM model instance with the rest of functionalities of the Requirements Quality Suite (SE Suite).
- **System Repository Language (SRL)**: it is the OSLC-KM implementation inside the SE Suite by TRC.
- **Oslc.Km.Parsers**: several parsers have been implemented in the methodology described in the second step (by using XSLT transformation files to create the OSLC-KM model instance). They can

be seen in Figure 105. A part of this transformation, the file generated for Papyrus to get requirements from the model can be found in Figure 106.

- **Oslc.Km.Parsers.XmlToSrl:** in the same Visual Studio solution a Graphical User Interface (GUI) has been generated in SE Suite to manage this kind of transformations (see Figure 106).

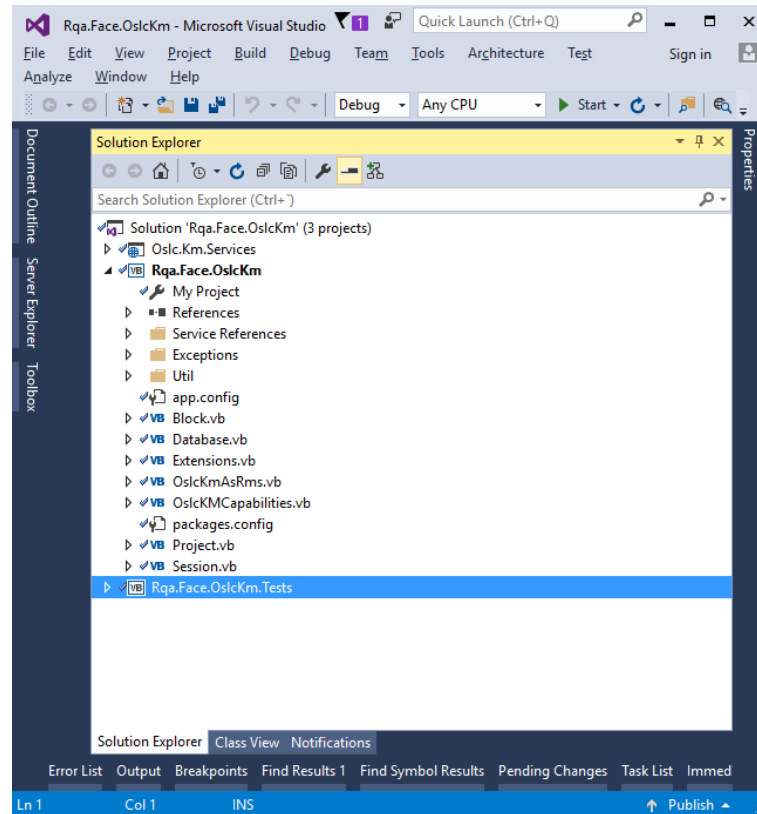


Figure 103. Rqa.Face.OslcKm library

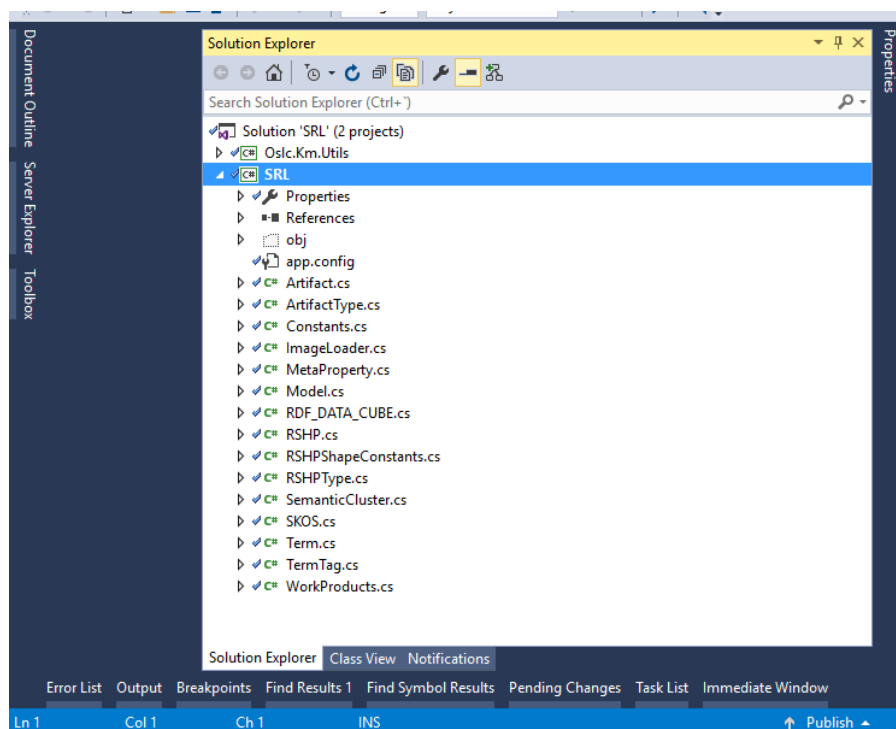


Figure 104. Implementation of the OSLC-KM for SE Suite by TRC

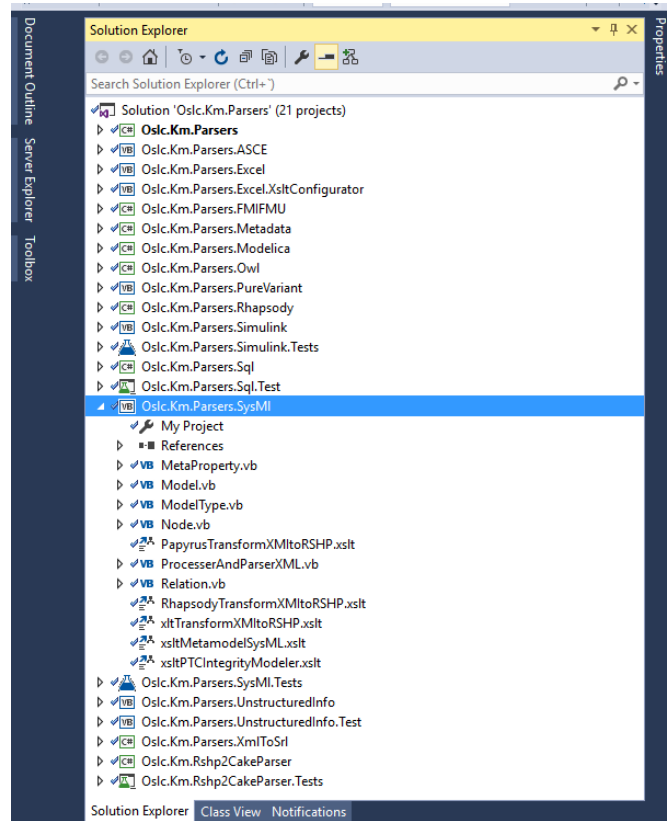


Figure 105. OSLC-KM parsers and XSLT transformation files for Papyrus and Rhapsody

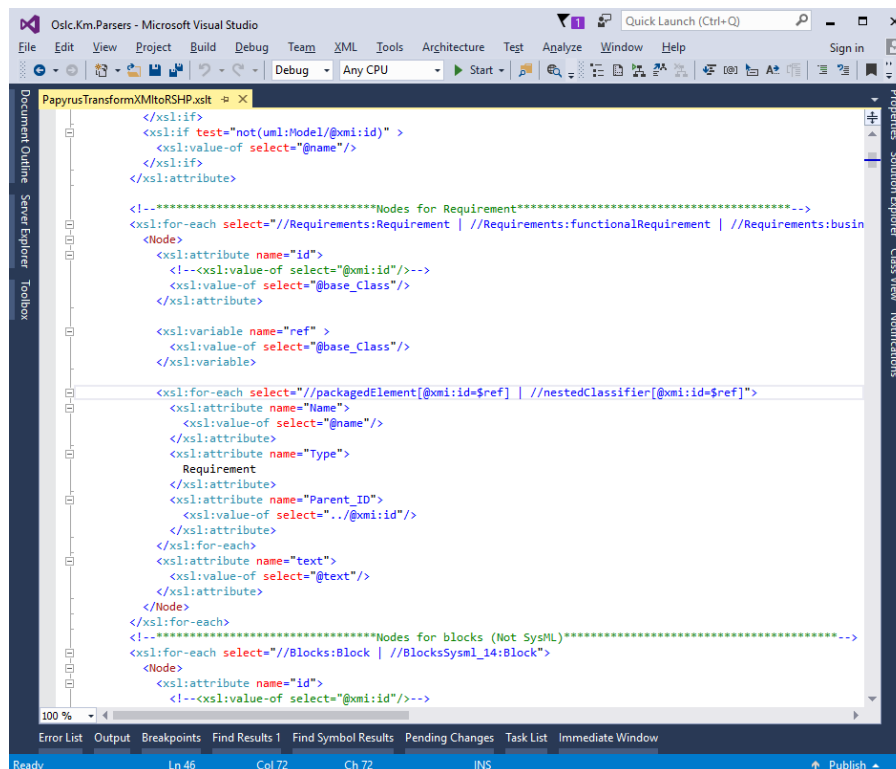
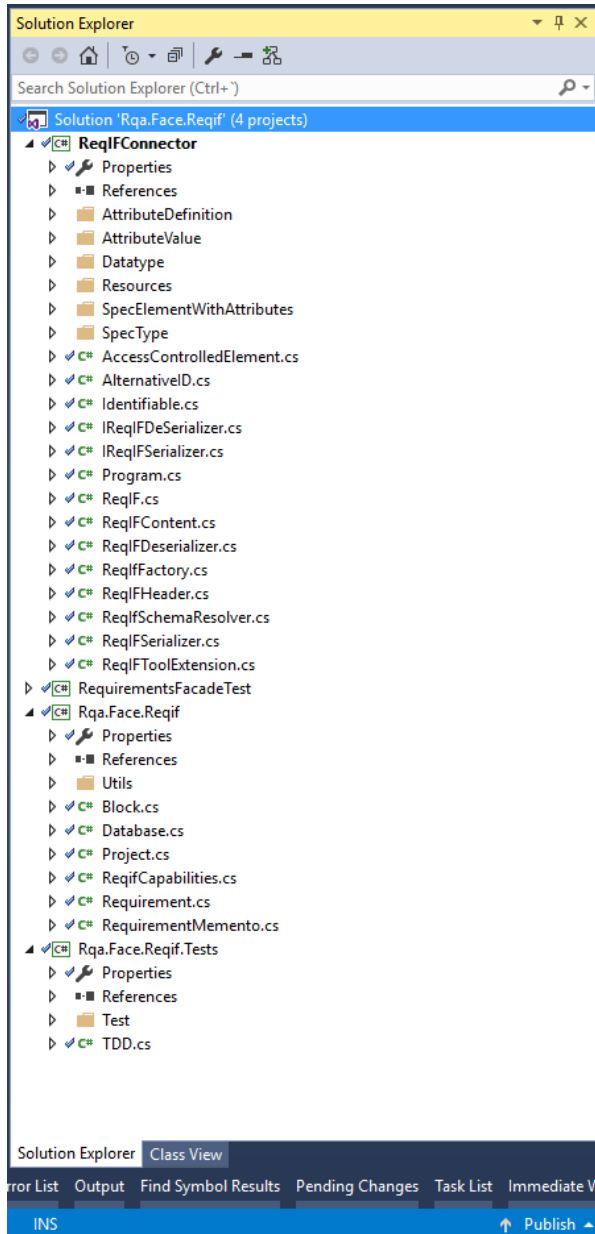


Figure 106. Part of the Papyrus XSLT transformation to map requirements in the model to the OSLC-KM model instance

3.4.1.2 ReqIF Connector (**)

The implementation of this functionality has been developed inside the SE Suite by TRC and it's composed of several libraries:



- **ReqIfConnector**: this library is in charge of providing reading and writing access to the ReqIF file implementing the ReqIF metamodel.
- **RequirementFacadeTest**: this library contains tests to ensure that the functionality developed on the ReqIfConnector works as expected.
- **Rqa.Face.ReqIF**: this library uses the interface exposed by the ReqIfConnector to provide an implementation of the Rqa.Face interface, which is the one used by all the SE Suite tools to perform their operations.
- **Rqa.Face.ReqIF.Tests**: this library contains tests to ensure that the functionality developed on the Rqa.Face.ReqIF library works as expected.

Figure 107. ReqIF connector source code libraries

3.4.1.3 PTC Integrity Connector (**)

The implementation of this functionality has been developed inside the SE Suite by TRC and it's composed of several libraries:

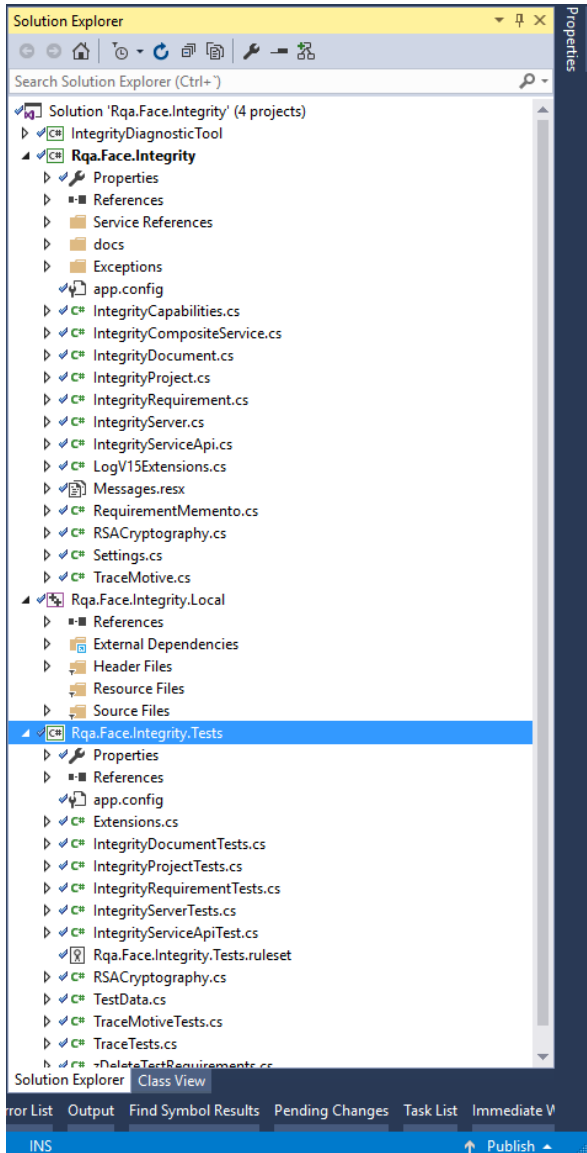


Figure 108. PTC integrity connector source code libraries

- IntegrityDiagnosticTool: this tool contains functionality to provide testing of the connectivity capabilities in several scenarios. This is an internal tool.
- Rqa.Face.Integrity: this library consumes the PTC Integrity Web Service and provides an implementation of the Rqa.Face interface, which is the one used by all the SE Suite tools to perform their operations.
- Rqa.Face.Integrity.Local: this library performs the integration of the output from the RAT plugin for Integrity with the Integrity application using their COM API.
- Rqa.Face.Integrity.Tests: this library contains tests to ensure that the functionality developed on the Rqa.Face.Integrity library works as expected.

3.4.1.4 RAT for Rhapsody Plugin (**)

The implementation of this functionality has been developed inside the SE Suite by TRC and it's composed of several libraries:

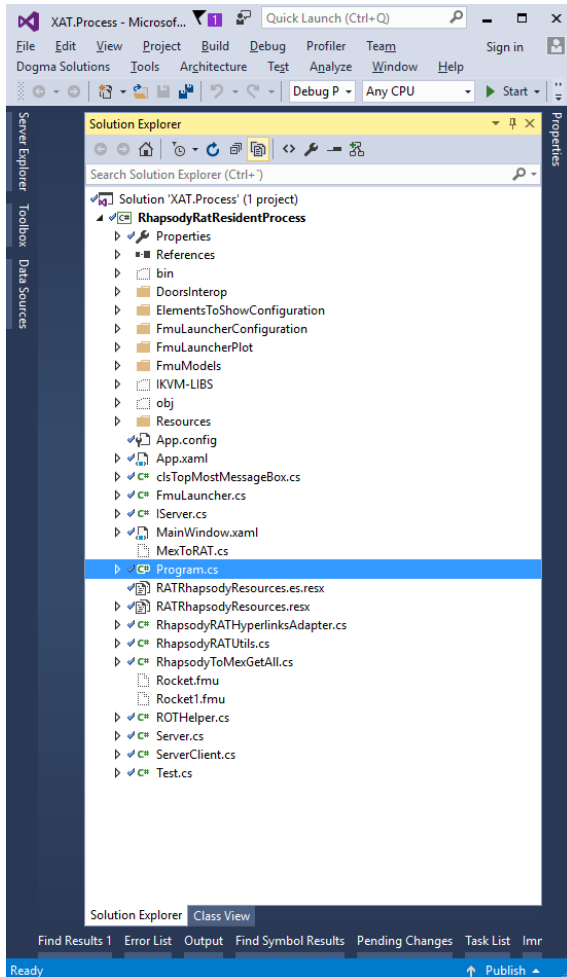


Figure 109. RAT plugin for Rhapsody source code

- **RhapsodyRatResidenProcess:** this tool contains functionality to provide testing of the connectivity capabilities in several scenarios. This is an internal tool.
- **JavaRhapsodyPlugin:** this is the plugin contains an implementation of the custom menu items added Rhapsody. This implementation only launches a the RhapsodyRatResidenProcess functions.
- **JavaPluginHelper.hep:** includes the custom menu options to be added to Rhapsody.

Finally, some major integration points to be mentioned are:

- The requirement format for Rhapsody is HTML and the SE Suite tool works authoring requirements in RTF format, so a conversion process is performed before using the RAT COM object.
- The RAT COM interface has been improved to allow editing requirements having hyperlinks to any other Rhapsody model element at any position of the requirement.
- RAT Edition window is not possible to be modal on top of Rhapsody with this architecture.

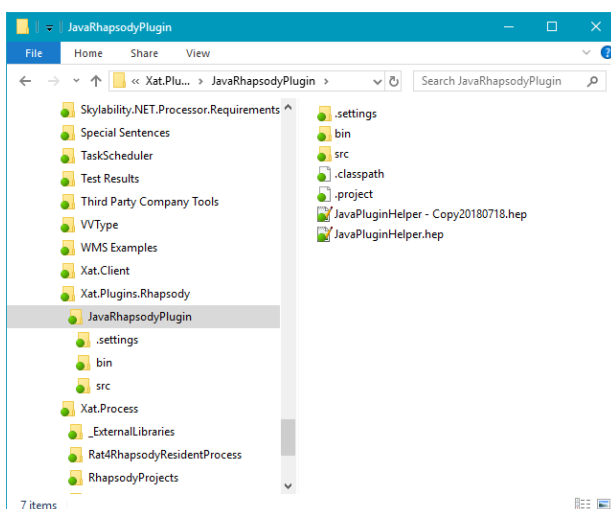
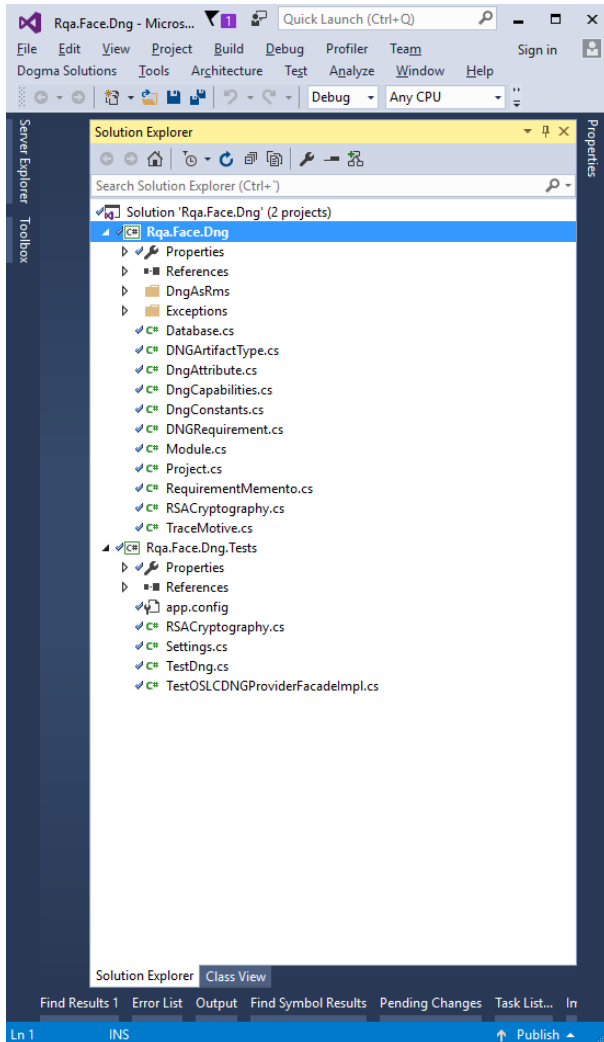


Figure 110. Java plugin for Rhapsody

3.4.1.5 DOORS Next Generation Connector (**)

The implementation of this functionality has been developed inside the SE Suite by TRC and it's composed of several libraries:

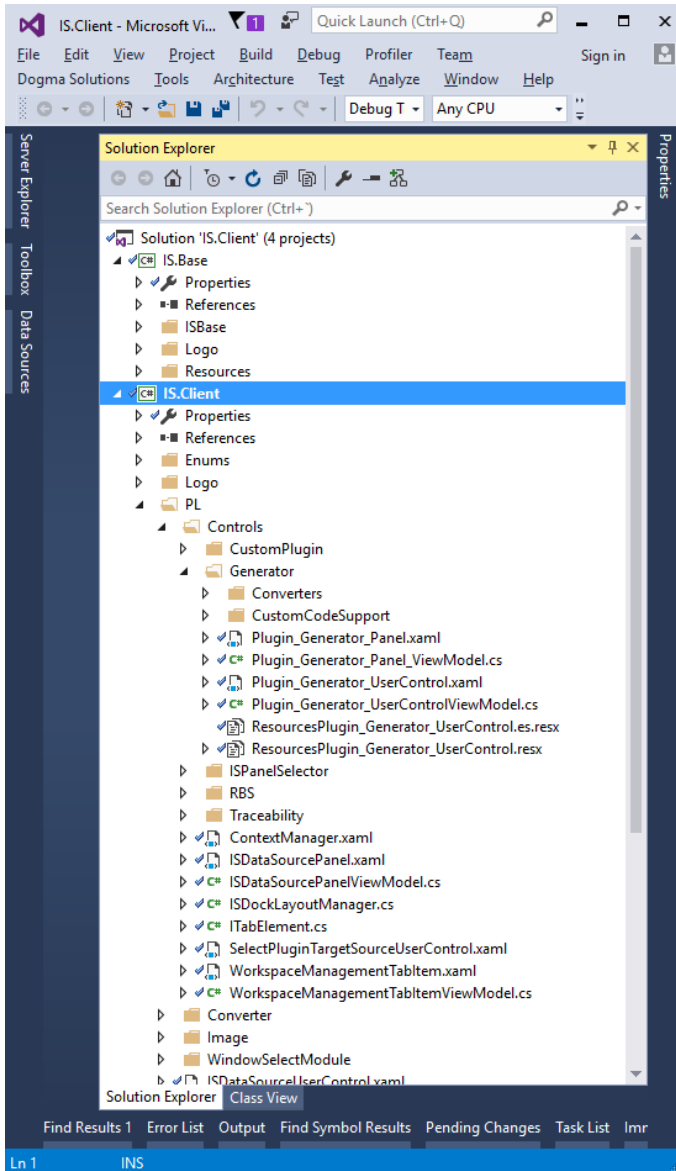


- **Rqa.Face.Dng:** this library consumes the DNG Web Service and provides an implementation of the Rqa.Face interface, which is the one used by all the SE Suite tools to perform their operations.
- **Rqa.Face.Dng.Tests:** this library contains tests to ensure that the functionality developed on the Rqa.Face.Dng library works as expected.

Figure 111. DNG connector source code libraries

3.4.1.6 Automatic translations (**)

The implementation of this functionality has been developed inside the SE Suite by TRC and it's composed of several libraries:



- **IS.Client:** this library is the base for the Interoperability Studio, a new tool in the SE Suite. It has several plugins, but the important one responsible of the automatic translations are the classes defined under the PL\Control\Generator folder. Its name is Generator because it is responsible of generating the target requirements from the input ones.

In the frame of AMASS, we have focused on transforming requirements across different languages (translations)

Figure 112. Automatic translations connector source code libraries

3.4.2 Seamless Interoperability Features for Safety/Cyber Architect tools (**)

Figure 113 illustrates the necessary plugins for the transformation from CHESSE to Safety Architect.

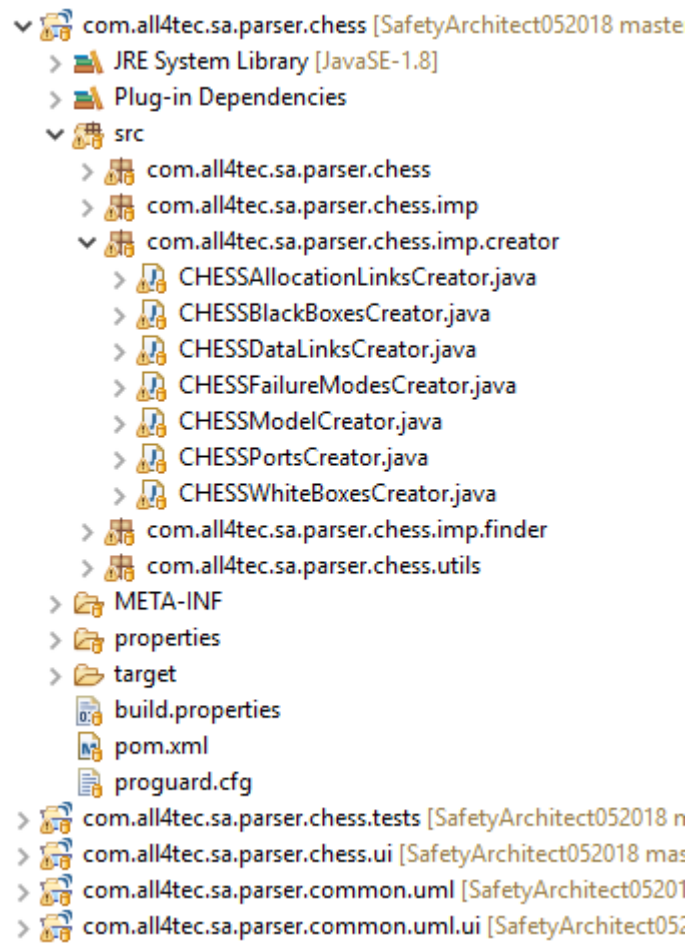


Figure 113. CHESSE to SA plugins

- **Com.all4tec.sa.parser.chess**

This plugin contains the specific strategies to create SA elements (such as: model, block, port, datalink, allocation link ...) from CHESSE elements.

- **Com.all4tec.sa.parser.chess.tests**

This plugin contains the Unit Tests for the transformation CHESSE-SA.

- **Com.all4tec.sa.parser.chess.ui**

This plugin contains the UI Wizard to import a CHESSE model into Safety Architect.

- **Com.all4tec.sa.parser.common.uml**

This plugin contains the common strategies to create SA elements from UML-based elements. This will be reused later by different bridges to convert UML-based model (e.g., Papyrus, CHESSE, MagicDraw etc.) to SA.

- **Com.all4tec.sa.parser.common.uml.ui**

This plugin contains the UI components to be shared by all UML-based bridges.

3.4.3 Integration of CHESSE and V&V Tools (**)

Currently, three V&V tools are integrated in the platform: OCRA, nuXmv and xSAP. As described in the Tool Integration, the integration is performed by adapters that connect the external tools via files or OSLC Automation protocol.

The adapters are composed by two Eclipse plugins and their internal structures are shown below:

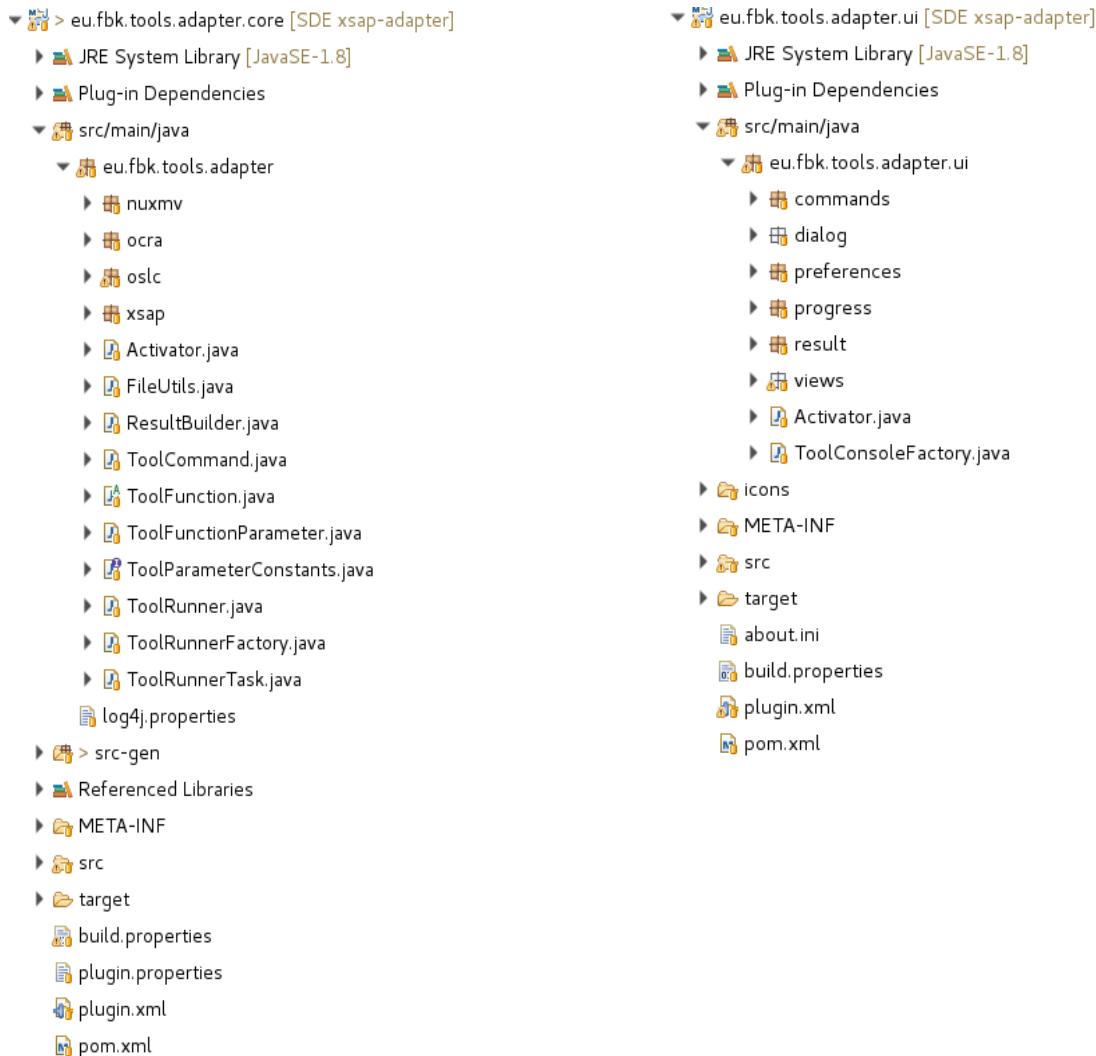


Figure 114. Source project structure of Tool Adapter plugins

The **eu.fbk.tool.adpater.core** plugin contains the classes for the invocation of the tool functions. Each function is mapped to a class (that derives from the *ToolFunction* class) whose attributes are the function parameters. The class itself translates its behaviour and the attributes to the tool invocation command.

The Figure 115 shows the tool functions hierarchy:

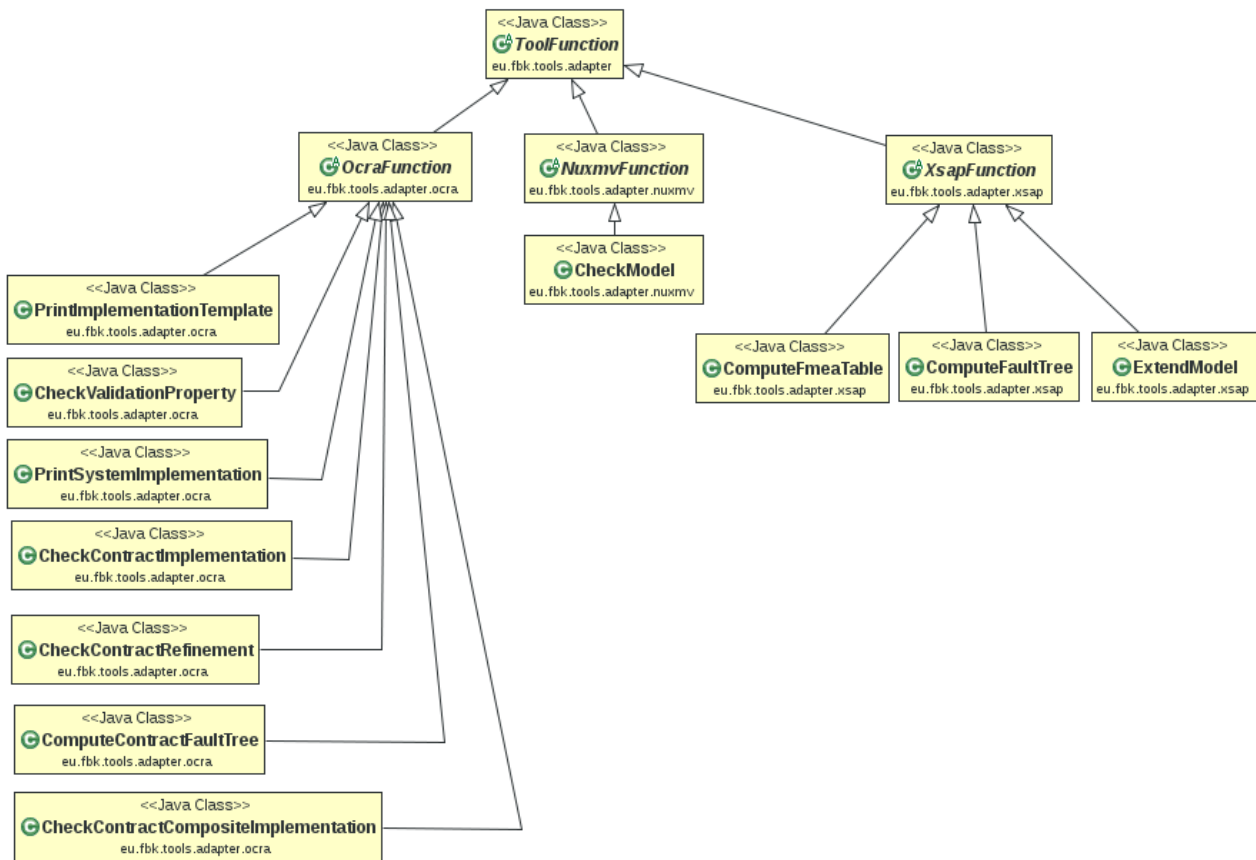


Figure 115. Tool Function Hierarchy

Depending on the plugin runtime configuration (File or OSLC adapter category) and the tool function, the command is executed by the appropriate *ToolRunner* and the result is processed by the *ResultBuilder*. The *ToolFunction* is not aware of the runner which is executed, so we can add a new runner category with no development impact on the tool function classes.

The Figure 116 shows the Tool Runner Hierarchy.

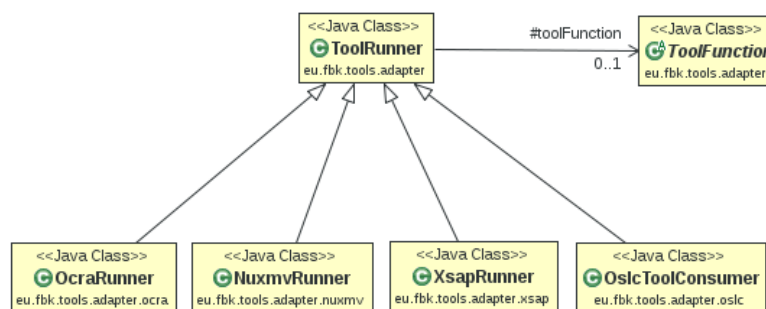


Figure 116. Tool Runner Hierarchy

The **eu.fbk.tool.adapter.ui** allows the invocation of the tool functions as Eclipse commands and presents the command results in some Views. Typically, CHES invokes directly these commands. The command id and the admitted parameters are described in the *plugin.xml* file as depicted by Figure 117.

eu.fbk.tools.adapter.ui

Extensions

All Extensions

Define extensions for this plug-in in the following section.

type filter text

- org.eclipse.ui.commands
 - check category (category)
 - Check Contract Refinement (command)
 - contract_model (commandParameter)
 - algorithm_type (commandParameter)
 - contract_name (commandParameter)
 - time_model (commandParameter)
 - async_execution (commandParameter)
 - result_file (commandParameter)
 - Compute Contract Fault Tree (command)
 - Compute System Implementation (command)
 - Generate Implementation Template (command)
 - Check Contract Implementation (command)
 - Check Contract Composite Implementation (command)
 - Check Validation Property (command)
 - Check Model Behaviour (command)
 - Extend Model (command)
 - Compute Fault Tree (command)
 - Compute FMEA Table (command)

Buttons: Add..., Remove, Up, Down

Extension Element Details

Set the properties of 'command'. Required fields are denoted by '*'. Deprecated fields are denoted by '()'.

id*	eu.fbk.tools.adapter.ui.commands.contract.CheckContractRefinement
name*	Check Contract Refinement
category()	
description	
categoryId	eu.fbk.tools.adapter
defaultHandler	eu.fbk.tools.adapter.ui.commands.contract.CheckContractRefinementCommand
returnTypeId	
helpContextId	

Figure 117. FBK Tool Eclipse Command

4. Conclusion (*)

This deliverable has presented the implementation work performed for Seamless Interoperability in the AMASS Prototype P2, which is the third version of the AMASS Tool Platform. This functionality allows a user to manage evidence artefacts, manage traceability between assurance assets, integrate the Platform with external tools, and collaborate with other users. In addition, some external support is provided for concurrent assurance information editing, e.g. via Kibana. Further support for seamless interoperability is provided by external tools integrated with the AMASS Tool Platform, such as the tools of the SE Suite by TRC.

Prototype P2 has extended the previous versions of the support for Seamless Interoperability in the AMASS Platform new features through the development of security mechanisms for user access management, the enactment of larger toolchains, and the integration of advanced collaborative work functionality.

At its current state, and prior to validation in WP2 and application in WP1, Seamless Interoperability support for Prototype P2 has TRL 3 (experimental proof of concept). The main aspects to address for Seamless Interoperability implementation until the end of AMASS include fixing bugs detected in WP2 activities and enhancing the current features from the feedback provided by users and industry partners in WP1 activities. This will lead to a higher TRL.

Regarding security implications from integration with external tools, the users of the AMASS Tool Platform must take into consideration the security mechanisms that the external tools provided, e.g. authentication. Based on the existence or not of these mechanisms, the degree of confidence in the data exchanged can vary.

References

- [1] AMASS project: D1.1 - Case studies description and business impact. 2016. https://amass-ecsel.eu/sites/amass.drupal.pulsartecnalia.com/files/documents/D1.1_Case-studies-description-and-business-impact_AMASS_Final.pdf
- [2] AMASS project: D2.2 - AMASS reference architecture (a). 2016.
- [3] AMASS project: D2.3 - AMASS reference architecture (b). 2017.
- [4] AMASS project: D2.4 - AMASS reference architecture (c). 2018. https://www.amass-ecsel.eu/sites/amass.drupal.pulsartecnalia.com/files/documents/D2.4_AMASS-reference-architecture-%28c%29_AMASS_Final.pdf
- [5] AMASS project: D3.3 - Design of the AMASS tools and methods for architecture-driven assurance (b). 2018. https://www.amass-ecsel.eu/sites/amass.drupal.pulsartecnalia.com/files/documents/D3.3_Design-of-the-AMASS-tools-and-methods-for-architecture-driven-assurance-%28b%29_AMASS_Final.pdf
- [6] AMASS project: D3.4 - Prototype for architecture-driven assurance (a). 2016. http://amass-ecsel.eu/sites/amass.drupal.pulsartecnalia.com/files/documents/D3.4_Prototype%20for%20architecture-driven%20assurance%20%28a%29_AMASS_final.pdf
- [7] AMASS project: D4.3 - Design of the AMASS tools and methods for multiconcern assurance (b). 2018. https://www.amass-ecsel.eu/sites/amass.drupal.pulsartecnalia.com/files/documents/D4.3_Design-of-the-AMASS-tools-and-methods-for-multiconcern-assurance-%28b%29_AMASS_Final.pdf
- [8] AMASS project: D4.4 - Prototype for multiconcern assurance (a). 2017. http://amass-ecsel.eu/sites/amass.drupal.pulsartecnalia.com/files/documents/D4.4_Prototype-for-multiconcern-assurance-%28a%29_AMASS_final.pdf
- [9] AMASS project: D5.1 - Baseline requirements for seamless interoperability. 2016. http://amass-ecsel.eu/sites/amass.drupal.pulsartecnalia.com/files/documents/D5.1_Baseline-and-Requirements-for-Seamless-Interoperability_AMASS_Final.pdf
- [10] AMASS project: D5.2 - Design of the AMASS tools and methods for seamless interoperability (a). 2017.
- [11] AMASS project: D5.3 - Design of the AMASS tools and methods for seamless interoperability (b). 2018. https://www.amass-ecsel.eu/sites/amass.drupal.pulsartecnalia.com/files/documents/D5.3_Design-of-the-AMASS-tools-and-methods-for-seamless-interoperability-%28b%29_AMASS_Final.pdf
- [12] AMASS project: D6.4 - Prototype for cross/intra-domain reuse (a). 2017. https://amass-ecsel.eu/sites/amass.drupal.pulsartecnalia.com/files/documents/D6.4_Prototype-for-cross-intra-domain-reuse-%28a%29_AMASS_Final.pdf
- [13] AMASS project: Prototype Core User Manual, Version 0.1⁶. 2017. https://services.medini.eu/svn/AMASS_collab/WP-transversal/ImplementationTeam/PrototypeCore/AMASS_Prototype1_UserManual.docx
- [14] AMASS project: Prototype P1 Developers Guide. https://services.medini.eu/svn/AMASS_collab/WP-transversal/ImplementationTeam/PrototypeP1/AMASS_PrototypeP1_DeveloperGuide.doc
- [15] AMASS project: Source code repository. 2017. https://services.medini.eu/svn/AMASS_source/⁷

⁶ 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).

⁷ 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)

- [16] Eclipse: CDO Model Repository. 2017. <https://eclipse.org/cdo/>
- [17] Eclipse: EEF. 2016. <https://eclipse.org/eeef/#/>
- [18] Eclipse: EMF. 2017. <https://eclipse.org/modeling/emf/>
- [19] OPENCROSS project. 2015. <http://www.opencross-project.eu/>
- [20] OPENCROSS project: D4.4 - Common Certification Language: Conceptual Model. 2015. http://www.opencross-project.eu/sites/default/files/D4.4_v1.5_FINAL.pdf
- [21] OSLC community. 2017. <https://open-services.net/>
- [22] PolarSys: OpenCert project. 2017. <https://www.polarsys.org/projects/polarsys.opencert>
- [23] SafeCer Project. 2015. <https://artemis-ia.eu/project/40-nsafecer.html>
- [24] Elasticsearch, <https://www.elastic.co/>
- [25] Elasticsearch Query String Syntax <https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-query-string-query.html#query-string-syntax>
- [26] The REUSE Company: Traceability Studio. <https://www.reusecompany.com/traceability-studio>
- [27] The REUSE Company: Verification Studio. <https://www.reusecompany.com/verification-studio>