

ECSEL Research and Innovation Actions (RIA)



AMASS

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

Design of the AMASS tools and methods for seamless interoperability (b) D5.3

Work Package:	WP5: Seamless Interoperability
Dissemination level:	PU = Public
Status:	Final
Date:	29 June 2018
Responsible partner:	Jose Luis de la Vara (Universidad Carlos III de Madrid)
Contact information:	jvara@inf.uc3m.es
Document reference:	AMASS_D5.3_WP5_UC3_V1.0

PROPRIETARY RIGHTS STATEMENT

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

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

Contributors¹

Names	Organisation
Jose Luis de la Vara, Jose Maria Alvarez, Eugenio Parra, Pablo Sánchez, Miguel Téllez, Luis Pérez, Roy Mendieta, Manuela Alejandres	Universidad Carlos III de Madrid (UC3)
Luis M. Alonso, Julio Encinas, Borja López	The REUSE Company (TRC)
Pietro Braghieri, Stefano Tonetta, Alberto Debiasi	Fondazione Bruno Kessler (FBK)
Tomáš Kratochvíla, Vít Koksa	Honeywell (HON)
Jan Mauersberger, Sascha Baumgart	ANSYS medini Technologies AG (KMT)
Huáscar Espinoza, Angel López, Estibaliz Amparan, Garazi Juez	TECNALIA Research & Innovation (TEC)
Barbara Gallina, Faiz Ul Muram	Maelardalens Hoegskola (MDH)
Ivana Cerna	Masaryk University (UOM)
Stefano Puri	INTECS (INT)
Morayo Adedjouma	Commissariat a L'énergie Atomique et aux Energies Alternatives (CEA)
Marc Sango	ALL4TEC (A4T)
Markus Grabowski	Assystem Germany (B&M)
Staffan Skogby, Detlef Scholle, Joel Kallin	Alten (ALT)

Reviewers

Names	Organisation
Markus Grabowski (Peer reviewer)	Assystem Germany (B&M)
Detlef Scholle (Peer reviewer)	Alten (ALT)
Staffan Skogby (Peer reviewer)	Alten (ALT)
Cristina Martinez (Quality Manager)	TECNALIA Research & Innovation (TEC)
Stefano Puri (TC reviewer)	Intecs (INT)
Barbara Gallina (TC reviewer)	Maelardalens Hoegskola (MDH)
Garazi Juez (TC reviewer)	TECNALIA Research & Innovation (TEC)

¹ The list includes the contributors to D5.2, which is evolved in D5.3

TABLE OF CONTENTS

Abbreviations and Definitions.....	8
Executive Summary.....	11
1. Introduction.....	12
2. Conceptual Approach	14
2.1 Seamless Interoperability Vision	14
2.2 Evidence Management	15
2.3 Tool Integration	17
2.4 Collaborative Work	19
2.5 Tool Quality Characterisation and Assessment	20
3. Module Specification	23
3.1 Seamless Interoperability Means	23
3.1.1 OSLC KM	23
3.1.2 Automatic Generation of OSLC KM-based Connectors.....	33
3.1.3 Ad-hoc Tool Integration	34
3.1.4 Papyrus Interoperability.....	37
3.1.5 V&V Tool Integration (*)	38
3.1.6 Integration with Safety and Security Analysis Tools (*)	45
3.1.7 Integration with the Sabotage Simulation-Based Fault Injection Tool (*)	50
3.1.8 Integration in the Farkle Tool (*)	51
3.1.9 Generic REST-API Adapter Concept for Seamless Interoperability (*).....	52
3.1.10 Collaborative Real-Time Model Editing.....	53
3.1.11 Seamless Tracing.....	55
3.1.12 Knowledge-Centric Automated Traceability (*)	57
3.1.13 On-Demand Automated Traceability Maintenance and Evolution (*)	57
3.1.14 Automatic Translations for Collaborative Work	59
3.1.15 Evidence Change Impact Analysis.....	61
3.1.16 Management of V&V evidence (*).....	62
3.1.17 Security Management (*).....	63
3.1.18 Data Management (*)	65
3.2 Seamless Interoperability Components	67
3.2.1 Data Manager Component.....	67
3.2.2 Access Manager Component.....	67
3.2.3 Evidence Editor Component.....	69
3.2.4 Traceability Management Component	70
3.2.5 Impact Analysis Component.....	70
3.2.6 Collaborative Work Components (*).....	71
3.2.7 Toolchain Management Component	71
3.2.8 Tool Connector Component	72
3.2.9 Tool Characterisation Component.....	73
4. Data Model.....	74
4.1 Evidence Management	74
4.2 Tool Integration	75
4.3 Data Management	77
4.4 Security Management.....	78
5. Way Forward for the Implementation (*).....	79

6. Conclusion (*)	85
References (*)	86
Appendix A. Tools in the AMASS Case Studies	91
Appendix B. Changes with respect to D5.2 (*)	93

List of Figures

Figure 1.	AMASS building blocks as initially envisioned in the project	13
Figure 2.	Taxonomy of assurance evidence for safety [80]	16
Figure 3.	Evidence classification by Bender et al. [24]	17
Figure 4.	Evidence management process	17
Figure 5.	Integrity connector architecture sequence diagram	20
Figure 6.	Collaborative-work vision with several editors	21
Figure 7.	Semantic Web Architecture (2005) [61] - "The Two Towers"	26
Figure 8.	Semantic Web Architecture (2015)	26
Figure 9.	UML Class Diagram of the OSLC Knowledge Management Resource Shape	30
Figure 10.	XML2SRL workflow process	34
Figure 11.	ReqIF metamodel	35
Figure 12.	Integrity connector architecture	36
Figure 13.	Rhapsody connector architecture	37
Figure 14.	Communication between the AMASS Platform and the V&V tools	38
Figure 15.	Interactions with the V&V tools	39
Figure 16.	Service execution request to V&V tool	39
Figure 17.	OSLC resources for interaction with V&V tools	40
Figure 18.	Communication of AMASS Platform with V&V Tools using OSLC Automation	41
Figure 19.	Use case diagram for the V&V Manager plugin	42
Figure 20.	V&V Manager top-level static structure and correspondence to the realized use cases	42
Figure 21.	The lower part of the screen contains the V&V Result view	43
Figure 22.	Interaction usage scenario for safety and security analysis	45
Figure 23.	Example of CHESS model - Wheel Brake System (WBS) Model	46
Figure 24.	Import from CHESS tool to Safety Architect tool	46
Figure 25.	Safety Architect WBS model from CHESS WBS model	47
Figure 26.	Cyber Architect project initialised with EBIOS knowledge bases	47
Figure 27.	An interface between Safety Architect and Cyber Architect	48
Figure 28.	Safety & Security viewpoint in Safety Architect	48
Figure 29.	Safety & Security viewpoint selection in Safety Architect	49
Figure 30.	Faults and Attacks Propagation Tree in Safety Architect	49
Figure 31.	Fault Tree Viewer in CHESS	50
Figure 32.	A Process for system safety and security analyses	50
Figure 33.	Sabotage integration with the CHESS/SAVONA, model-based safety analysis and Monitors Tools [14]	51
Figure 34.	Overview of interwork between AMASS tools and Machine Learning-based testing tools	52
Figure 35.	Generic REST adapter concept	53
Figure 36.	Multiple clients collaborating on a single document	54
Figure 37.	Concurrent editing operations by two users	54
Figure 38.	Capra architecture	56
Figure 39.	OSLC-based approach for self-assessment and traceability	57
Figure 40.	Model of knowledge-centric traceability	58
Figure 41.	On-demand traceability maintenance and evolution	59
Figure 42.	Stages of the translation process using NLP	60
Figure 43.	Inputs and outputs of the Language Mapping stage	60
Figure 44.	Inputs and outputs of the Reverse Disambiguation stage	60
Figure 45.	Inputs and outputs of the Reverse Normalization stage	61
Figure 46.	Evidence artefact lifecycle from an impact analysis point of view [87]	62
Figure 47.	Assurance Asset Evaluation in an Artefact Model	63

Figure 48. Security Management window managing users	64
Figure 49. Security Management window managing roles.....	65
Figure 50. Data Management related to other modules.....	66
Figure 51. Authentication for Platform Access	66
Figure 52. Permissions control	67
Figure 53. ARTA Platform Management Component	68
Figure 54. Data Management Component	68
Figure 55. Access Management Component	68
Figure 56. ARTA Evidence Management Component.....	69
Figure 57. Evidence Editor Component	69
Figure 58. ARTA Assurance Traceability Component	70
Figure 59. Traceability Management Component.....	70
Figure 60. Impact Analysis Component	71
Figure 61. Collaborative Work Component	71
Figure 62. ARTA Tool Integration Component	72
Figure 63. Toolchain Management Component.....	72
Figure 64. Tool Connector Component.....	73
Figure 65. ARTA Compliance Management Component	73
Figure 66. OPENCROSS evidence metamodel [86]	74
Figure 67. OPENCROSS assurance asset metamodel [86].....	75
Figure 68. OPENCROSS process metamodel [86]	75
Figure 69. Lyo Toolchain metamodel [43]	76
Figure 70. CDO data model [41]: a) checkout elements; b) sessions; c) views; d) transactions	77
Figure 71. CDO security metamodel [42].....	78

List of Tables

Table 1.	Summary of the main approaches to validate RDF-encoded data	25
Table 2.	OSLC Resource Shapes description for OSLC Defined Resources within the Knowledge Management domain	31
Table 3.	Delegated operations for an OSLC KM provider (SKB)	33
Table 4.	Delegated operations for an OSLC KM provider (SAS)	33
Table 5.	Advantages and disadvantages of different cross-tool tracing strategies in ALM tools	55
Table 6.	Seamless Interoperability requirements	79
Table 7.	Tools used in the AMASS Case Studies	91

Abbreviations and Definitions

ALM	Application Lifecycle Management
API	Application Programming Interface
ARTA	AMASS Reference Tool Architecture
ATM	Air Traffic Management
AUTOSAR	AUTomotive Open System ARchitecture
BPMN	Business Process Model and Notation
CACM	Common Assurance and Certification Metamodel
CAKE	Computer-Aided Knowledge Environment
CDO	Connected Data Objects
CESMI	Common Explicit-State Model Interface specification
COM	Component Object Model
CPS	Cyber-Physical Systems
CRUD	Create, Read, Update and Delete
CS	Case Study
CSV	Comma-Separated Values
DB	DataBase
DDL	Data Definition Language
DOLCE	Descriptive Ontology for Linguistic and Cognitive Engineering
DTD	Document Type Definition
DUI	Delegated User Interface
EAI	Enterprise Integration Patterns
EAST-ADL	EAST Architecture Description Language
EBIOS	Expression des Besoins et Identification des Objectifs de Sécurité - Expression of Needs and Identification of Security Objectives
EC	European Commission
EL	Expressions Language
EMF	Eclipse Modelling Framework
EPF	Eclipse Process Framework
FAA	Federal Aviation Administration
FMECA	Failure Mode, Effects and Criticality Analysis
FOAF	Friend Of A Friend
fUML	foundational UML
GMF	Graphical Modelling Framework
GSN	Goal Structuring Notation
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IMA	Integrated Modular Avionics
HW	Hardware
IEC	International Electrotechnical Commission
ISO	International Standardization Organization
JSON	JavaScript Object Notation
LTL	Linear Temporal Logic
KE	Knowledge Element
KM	Knowledge Management
KM	Knowledge Manager
MBSE	Model-Based Systems Engineering
MLT	Machine Learning for Testing
MOF	MetaObject Facility
MPSoC	MultiProcessor System-on-Chip

NLP	Natural Language Processing
OCRA	Othello Contracts Refinement Analysis
ODM	Ontology Definition Metamodel
OMG	Object Management Group
OSLC	Open Services for Lifecycle Collaboration
OT	Operational Transformation
OWL	Web Ontology Language
OWL-S	Semantic Markup for Web Services
PLM	Product Lifecycle Management
PSCS	Precise Semantics of UML Composite Structures
QL	Query Language
RAS	Reusable Asset Specification
RAT	Requirements Authoring Tool
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
Relax NG	REgular LAnguage for XML Next Generation
ReqIF	Requirements Interchange Format
REST	Representational State Transfer
RIF	Rule Interchange Format
RL	Rule Language
RM	Requirements Management
RMS	Requirements Management System
RQA	Requirements Quality Analyzer
RQS	Requirements Quality Suite
RSA	Rational Software Architect
RTCA	Radio Technical Commission for Aeronautics
RTDS	Real Time Digital Power System
RuleML	Rule Modelling Language
SA-WSDL	Semantic Annotations for WSDL
SACM	Structured Assurance Case Metamodel
SAS	System Assets Store
SBVR	Semantics of Business Vocabulary and Rules
SHACL	Shapes Constraint Language
ShEX	Shape EXpressions
SISSP	School of Information Systems Security Prophets
SKB	System Knowledge Base
SKOS	Simple Knowledge Organization System
SOAP	Simple Object Access Protocol
SPARQL	SPARQL Protocol and RDF Query Language
SQL	Structured Query Language
SRL	System Representation Language
SUMO	Suggested Upper Merged Ontology
SUT	System Under Testing
SVN	Apache Subversion
SVP	Subject+Verb+Predicate
SW	Software
SWRL	Semantic Web Rule Language
SysML	Systems Modelling Language
TCL	Tool Confidence Level
TD	Tool error Detection
TI	Tool Impact
TQL	Tool Qualification Level

UML	Unified Modelling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
V&V	Verification & Validation
W3C	World Wide Web Consortium
WBS	Wheel Brake System
WP	Work Package
WSDL	Web Services Description Language
WSDL-S	Web Service Semantics
WWW	World Wide Web
XAT	universal system Authoring Tool
XML	eXtensible Markup Language
XSLT	eXtensible Stylesheet Language Transformations

Executive Summary

This deliverable is the final output of Task 5.2 (Conceptual Approach for Seamless Interoperability). The deliverable reports on the design of the Seamless Interoperability tool support in the AMASS Tool Platform. It contains information about interfaces, format specifications, the tool architecture, and contributions to the CACM (Common Assurance and Certification Metamodel). The design has been developed incrementally, with revisions after implementation validation, and will serve as the main reference for the implementation of Seamless Interoperability support in the third prototype of the AMASS Tool Platform (P2).

As presentation of the conceptual approach, D5.3 introduces the overall vision for Seamless Interoperability, providing specific details about evidence management, tool integration, collaborative work, and tool quality characterisation and assessment.

For Seamless Interoperability, 18 different means are proposed: OSLC KM (Open Services for Lifecycle Collaboration – Knowledge Management), Automatic Generation of OSLC KM-based Connectors, Ad-hoc Tool Integration, Papyrus Interoperability, V&V Tool Integration, Integration with Safety and Security Analysis Tools, Integration in the Farkle Tool, Generic REST-API Adapter Concept for Seamless Interoperability, Collaborative Real-Time Model Editing, Seamless Tracing, Knowledge-Centric Automated Traceability, On-Demand Automated Traceability Maintenance and Evolution, Data Mining, Automatic Translations for Collaborative Work, Evidence Change Impact Analysis, Management of V&V evidence, Security Management, and Data Management.

The version of the ARTA presented in D2.4 has been refined to decompose the ARTA components responsible for data management, access management, evidence management, assurance traceability, collaborative work, and tool integration. Reference data models for Seamless Interoperability are also presented to address data needs for evidence management, tool integration, data management, and security management.

The main relationships of D5.3 with other AMASS deliverables are as follows:

- D2.1 (Business cases and high-level requirements) includes the requirements that the design for Seamless Interoperability must satisfy.
- D2.4 (AMASS reference architecture (c)) presents the high-level architecture of the AMASS Tool Platform that is refined and further developed in D5.3.
- D5.1 (Baseline requirements for seamless interoperability) reviews and consolidates existing work for Seamless Interoperability and proposes a way forward whose design for materialisation is addressed in D5.3.
- D5.2 (Design of the AMASS tools and methods for seamless interoperability (a)) presents the previous version of the design described in D5.3.
- D5.6 (Prototype for seamless interoperability (c)) will report how the design in D5.3 has been implemented in AMASS Prototype P2.
- D5.8 (Methodological guide for seamless interoperability (b)) will describe how users can employ the Seamless Interoperability design presented in D5.3.

Last but not least, the sections whose content has been modified with respect to D5.2 have been marked with an asterisk (*). This includes the new sections added. If some minor changes have been made (e.g. fixing typos or changing the section number), the corresponding section is not marked. The details about the differences and modifications are provided in Appendix B.

1. Introduction

Embedded systems have significantly increased in number, technical complexity, and sophistication toward open, interconnected, networked systems (such as "the connected car" and the cloud). This has brought a "cyber-physical" dimension with it, exacerbating the problem of ensuring safety, security, availability, robustness and reliability in the presence of human, environmental and technological risks. Furthermore, the products into which these Cyber-Physical Systems (CPS) are integrated (e.g. aircrafts) need to respect applicable standards for assurance, and in some areas, they even need certification. The dimension of the certification issue becomes clear if we look at the passenger plane B 787 as a recent example – it has been reported that the certification process lasted 8 years and has consumed 200,000 staff hours at the FAA just for technical work. The staff hours of the manufacturer even exceeded this figure, as more than 1,500 regulations had to be fulfilled, with evidence reflected onto 4,000+ documents. Although aircrafts are an extremely safety-critical product with many of such regulations, the situation in other areas (railway, automotive, medical devices etc.) is similar.

To tackle all these challenges, the AMASS approach focuses on the development and consolidation of an open and holistic assurance and certification framework for CPS that constitutes the evolution of the OPENCROSS [85] and SafeCer [95] approaches towards an architecture-driven, multi-concern assurance, and seamlessly interoperable tool platform.

The AMASS tangible expected results are:

- a) **The AMASS Reference Tool Architecture**, which will extend the OPENCROSS and SafeCer conceptual, modelling and methodological frameworks for architecture-driven and multi-concern assurance, as well as for further cross-domain and intra-domain reuse capabilities and seamless interoperability mechanisms (based on OSLC specifications).
- b) **The AMASS Open Tool Platform**, which will correspond to a collaborative tool environment that supports CPS assurance and certification. This platform represents a concrete implementation of the AMASS Reference Tool Architecture, with a capability for evolution and adaptation, which will be released as an open technological solution by the AMASS project. AMASS openness is based on both standard OSLC APIs with external tools (e.g. engineering tools including V&V tools) and on open-source release of the AMASS building blocks.
- c) **The Open AMASS Community**, which will manage the project outcomes, for maintenance, evolution and industrialization. The Open Community will be supported by governance board, 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 [89] is a strong candidate to host AMASS.

To achieve the AMASS results, and as depicted in Figure 1, the multiple challenges and corresponding project scientific and technical objectives are addressed by different WPs.

WP5 (Seamless Interoperability) roughly aims at tool interoperability. More specifically, with respect to the AMASS goals, the WP deals with the problem and solution related to AMASS goal G4 and the corresponding project objective O3:

- G4: to demonstrate a potential sustainable impact in CPS industry by increasing the harmonization and interoperability of assurance and certification/qualification tool technologies by 60%.
- O3: to develop a fully-fledged open tool platform that will allow developers and other assurance stakeholders to guarantee seamless interoperability of the platform with other tools used in the development of CPSs.

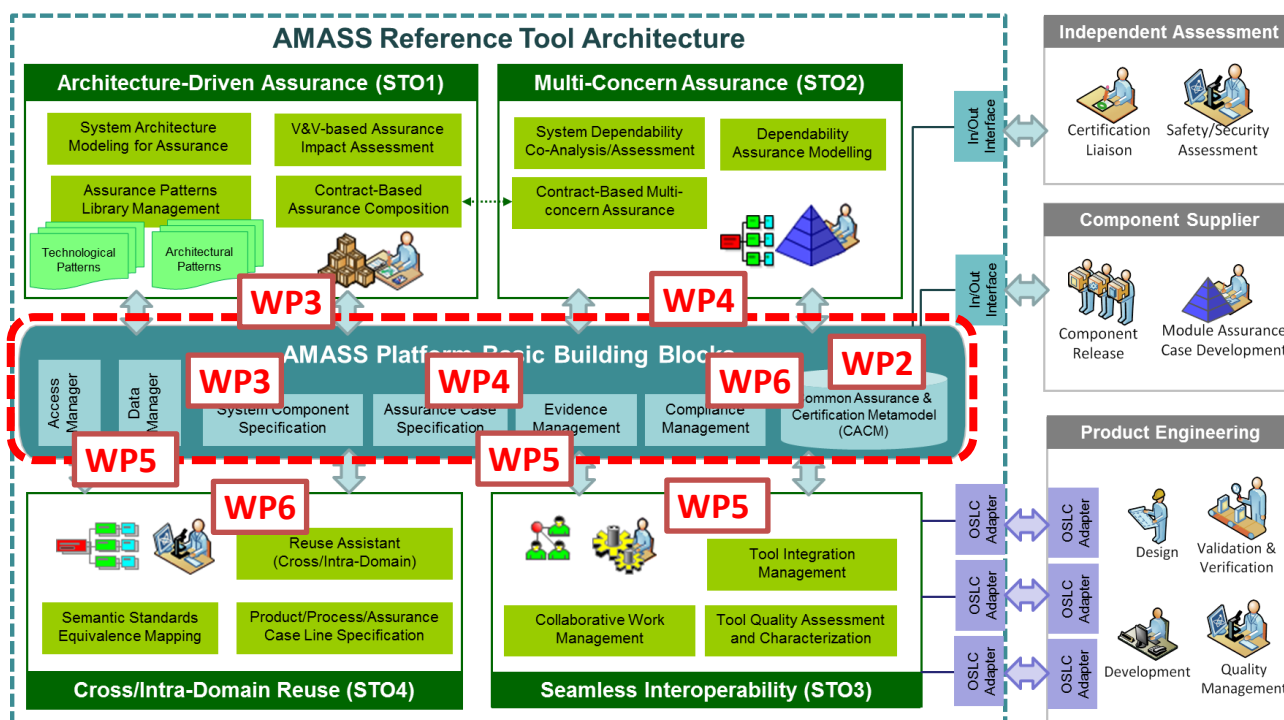


Figure 1. AMASS building blocks as initially envisioned in the project

WP5 shall investigate and provide an open and generically applicable approach to ensure the interoperability between the tools used in the modelling, analysis, and development of CPS, among other possible engineering activities. The WP addresses interoperability from an assurance and certification-specific perspective, and the resulting approach further aims to support collaborative work among the stakeholders of the assurance and certification of CPS. This facilitates the determination of the consequences of the use of a given engineering tool (e.g., based on its available qualification information and documentation), and to ensure that the integrated information makes CPS certification possible. In addition, WP5 is responsible for consolidating previous work on evidence management in order to design and implement the basic building block called ‘Evidence Management’ (Figure 1). WP5 also takes care of the ‘Access Manager’ and ‘Data Manager’ basic building blocks.

This document is the deliverable D5.3, the second deliverable of the Task T5.2 (Conceptual Approach for Seamless Interoperability). The deliverable contributes to the WP5 overall objectives regarding (a) the provision of an extensible tool architecture for Seamless Interoperability, (b) the investigation of suitable generic approaches for tool integration, and (c) the specification of metamodel(s) as a foundation for tool integration. To these ends, the deliverable presents means, components, and data models for Seamless Interoperability. All these elements will result in an open and generically applicable approach to ensure the interoperability between the tools used in the modelling, analysis, and development of CPS, among other possible engineering activities, addressing interoperability from an assurance and certification-specific perspective. The approach further aims to support collaborative work among the stakeholders of the assurance and certification of CPS, to facilitate the determination of the consequences of the use of a given engineering tool (e.g., based on its available qualification information and documentation), and to ensure that the integrated information makes CPS certification possible. In addition, D5.3 proposes a way forward for implementation.

The rest of the deliverable is organised as follows. Section 2 presents the conceptual approach for Seamless Interoperability in AMASS and Section 3 the modules that implement this approach. Section 4 introduces the data model for Seamless Interoperability. Section 5 presents the way forward for implementation, and Section 6 our main conclusions. Finally, the Appendices present additional information that has been used to create the deliverable or that is necessary to understand it.

2. Conceptual Approach

This section presents the main ideas and principles of the AMASS conceptual approach for Seamless Interoperability. First, our vision for Seamless Interoperability is described to introduce the general activities and processes that AMASS aims to support. It corresponds to a description of how we think that users could use the AMASS Tool Platform once the project finishes. Next, the overall work areas are explained: Evidence Management, Tool Integration, Collaborative Work, and Tool Quality Characterisation and Assessment.

The section synthesises and summarises certain aspects from D5.1 [16] that a reader might need to know to understand the Seamless Interoperability design and extends such information with design-specific details and insights gained for D5.2 [17] and D5.3.

2.1 Seamless Interoperability Vision

John is an assurance manager involved in the engineering of light small aircrafts regarded as the new generation of autonomous ‘flying cars’. Such systems, which include advanced features such as coordination and cooperation with road vehicles and other aircrafts, require the demonstration of the fulfilment of stringent regulatory requirements from different assurance standards, as well as the assurance of highly-critical system dependability requirements.

The systems engineering processes need and produce a vast number of artefacts that must be provided as **assurance evidence** for aircraft certification. The AMASS Tool Platform allows John to gather information about all the evidence artefacts generated by assurance engineers and system engineers, recording the whole artefacts’ lifecycles because assessors might want to check them to gain confidence in system dependability. Once John is successfully logged in the Platform according to the **access rights** granted to him, he can indicate the specific parts of a file (e.g. a document) that correspond to the artefacts to manage as assurance evidence, such as the hazard log template presented in the initial system’s safety plan document. John’s profile for using the AMASS Tool Platform allows him to access both specific tool functionality and different information types. The AMASS Tool Platform performs a detailed **data management** of all the actions and data involved in John’s job, keeping track of all the changes made and supporting the continuous analysis, verification, and integration of the data that other users also employ.

Many different tools are used in the lifecycle of the aircraft. From purpose-specific tools such as requirements management and modelling ones to general tools such as Word and Excel, John and the rest of stakeholders of the system (assurance engineers, system engineers, assessors, etc.) need to deal with tools and data of different types. In the past, this required the use of a wide variety of tool environments and data formats. However, the AMASS Tool Platform has enabled the management of assurance information in a much more effective and efficient way thanks to its advanced **tool integration** mechanisms. The application of novel web and standardised technologies (e.g. ModelBus and OSLC) allows John to only need to sign in in the AMASS Tool Platform to retrieve information from different tools. The Platform has simplified the installation procedures by centralising tasks such as tool configuration, as well as data exchange and consistency management through automatic data import and export and the use of non-proprietary data formats. All the assurance information is now available in a single, centralised repository that contains metadata about the specific system artefacts. The stakeholders create and enter data only once, and assessors can easily check data provenance and status for the whole system lifecycle without having to use several tools. All in all, the amount of manual procedures for tool and data integration has been decreased considerably. In addition, the AMASS Tool Platform supports **tool characterisation and quality assessment** for the toolchain used for systems engineering, providing John with information about tool qualification requirements and the associated evidence collection needs.

The AMASS Tool Platform has also enhanced **collaborative work** between John and his colleagues. They all need to cooperate for system assurance and certification in many activities: system analysis, system

specification, system implementation, system V&V, system assessment... The Platform allows them to perform live collaboration on the same data without having to use complicated or cooperation-hindering procedures such as data merge or coarse-grain data locking. Real-time feedback is provided about data access by different users, including concurrent access, about data integrity, and about concurrence modification rules and effects. John can consult metrics and measurements about collaborative creation and management of assurance information.

2.2 Evidence Management

Evidence management is based on several concepts and aspects [39][80]. **Assurance evidence**² corresponds to artefacts that contribute to developing confidence in the dependable operation of a system and that can be used to show the fulfilment of the criteria of an assurance standard. Examples of artefact types that can be used as assurance evidence include risk analysis results, system specifications, reviews, testing results, and source code. Those artefacts that correspond to assurance evidence can be referred to as **evidence artefacts**. The **body of assurance evidence** of an assurance project is the collection of evidence artefacts managed, usually a large set of artefacts that is difficult to overview. A **chain of assurance evidence** is a set of pieces of assurance evidence that are related, e.g. a requirement, the test cases that validate the requirement, and the report where the test results are presented. **Assurance evidence traceability** is the degree to which a relationship can be established to and from evidence artefacts. **Impact analysis of assurance evidence change** is the activity concerned with identifying, in a system's body of assurance evidence, the potential consequences of a change. Finally, according to ISO 26262 [66], a safety case (or assurance case from a more general perspective) is "an argument that the safety requirements for an item are complete and satisfied by evidence compiled from work products of the safety activities during development". Thus, in principle, all work products (evidence) should be traced.

Figure 2 presents a taxonomy of assurance evidence for safety (i.e. of safety evidence). Evidence is divided into process information and product information. More details about the taxonomy, including definitions and examples of evidence artefacts, are available in [80]. Bender et al., in their recently proposed certification framework ([24]; Figure 3), further classify product-based evidence into immediate (system specifications, source code), direct (risk analysis, reviews, verification results...; e.g. testing results) and indirect (e.g. safety plans as well as substantiations of the plans). This further classification is of relevance to distinguish evidence produced while executing activities of the left-hand side of the V-model from evidence produced while executing activities of the right-hand side of the V-model.

Evidence management can be defined as the system assurance and certification area concerned with the collection and handling of the body of assurance evidence of an assurance project, including chains of assurance evidence. Figure 4 shows a general, high-level evidence management process. When managing assurance evidence in an assurance project, the first step is usually to determine what evidence must be provided. Afterwards, the evidence artefacts that conform the body of assurance evidence of the project must be collected, and might also have to be evaluated and traced to other artefacts (creation of chains of evidence). During this process, it might be necessary to make changes in the evidence artefacts, and such changes might impact other items. As a result, issues and problems (e.g., inconsistency) might appear in the body of assurance evidence and would have to be addressed. Otherwise, the body of assurance evidence might not be adequate. Once the body of evidence of the assurance project is regarded as adequate (i.e., it is regarded as complete and no issues exist), the process can be finished.

² The terms Evidence and Assurance Evidence are used indistinctively in this document to denote the same concept

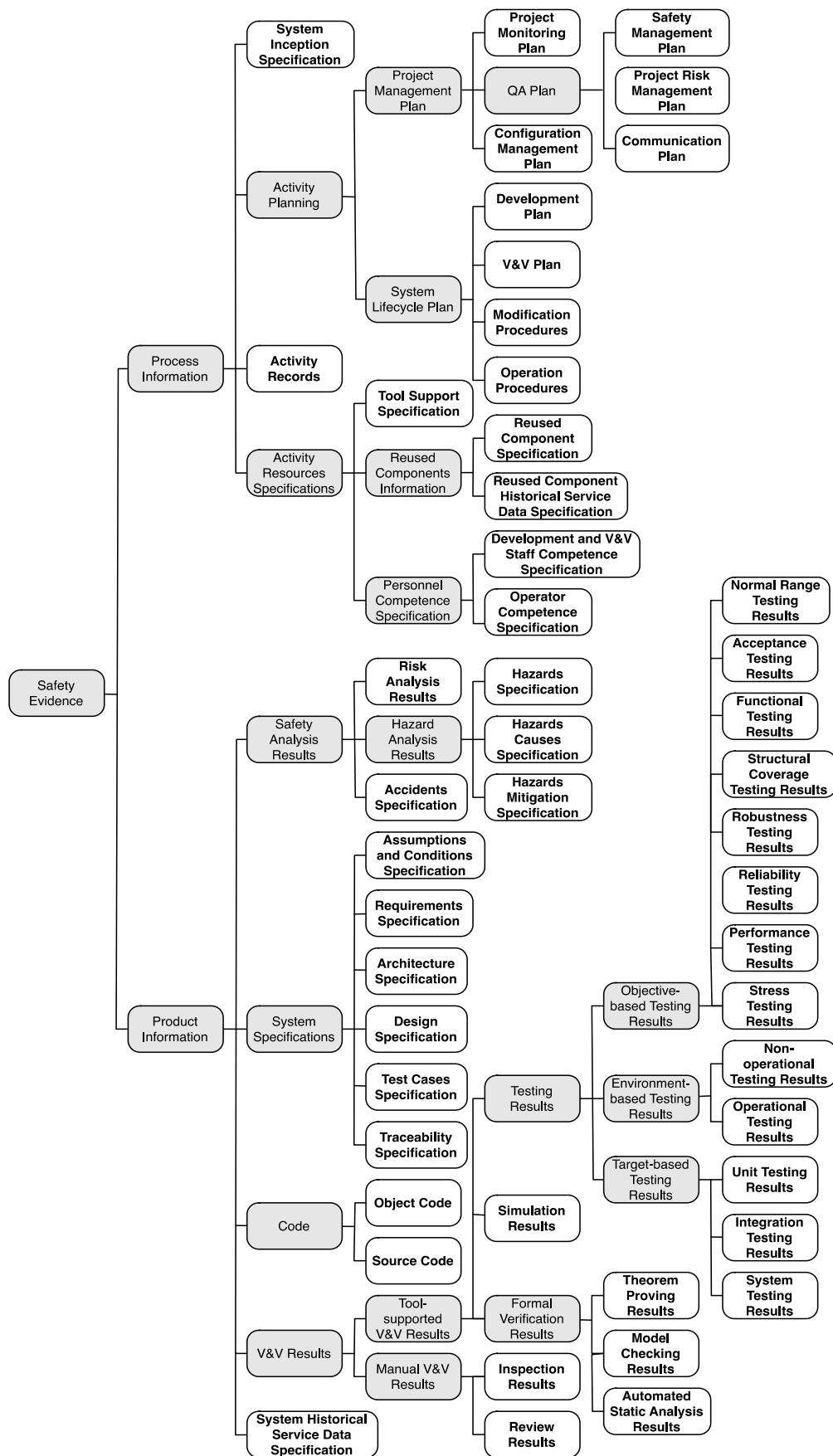


Figure 2. Taxonomy of assurance evidence for safety [80]

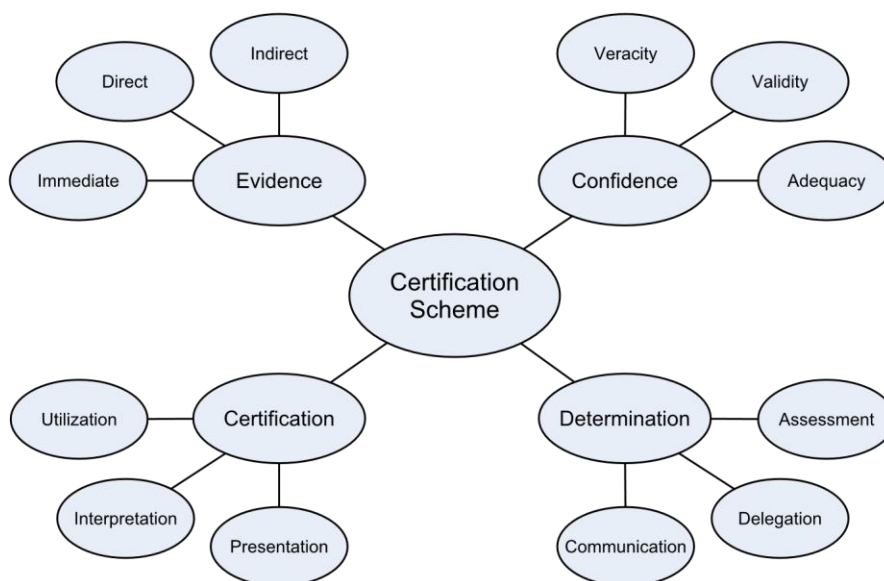


Figure 3. Evidence classification by Bender et al. [24]

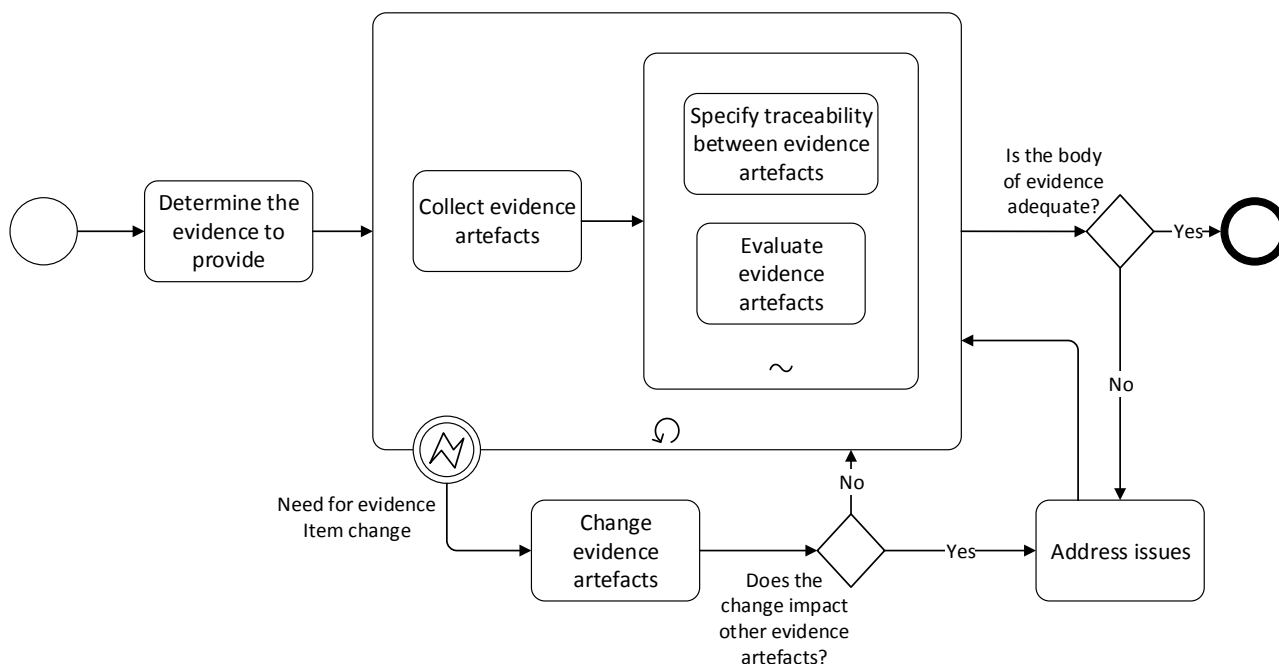


Figure 4. Evidence management process

2.3 Tool Integration

Recent times have seen the deployment of service-oriented computing [68] as a new environment to enable the reuse of software in organizations. In general, a service-oriented architecture comprises an infrastructure (e.g. Enterprise Service Bus) in which services (e.g. software as web services) are deployed under a certain set of policies. A composite application is then implemented by means of a coordinated collection of invocations (e.g. Business Process Execution Language). In this context, Enterprise Integration Patterns (EAI) [60] have played a key role to ease the collaboration among services. Furthermore, existing W3C recommendations such as the Web Services Description Language (WSDL) or the Simple Object Access Protocol (SOAP) have improved integration and interoperability through a clear definition of the input/output interface of a service and communication protocol.

In order to improve the capabilities of this type of web services, semantics was applied to ease some tasks such as discovery, selection, composition, orchestration, grounding and automatic invocation of web services. The Web Services Modelling Ontology (WSMO) [93] represented the main effort to define and to implement semantic web services using formal ontologies. OWL-S (Semantic Markup for Web Services), SA-WSDL (Semantic Annotations for WSDL) or WSDL-S (Web Service Semantics) were other approaches to annotate web services, by merging ontologies and standardizing data models in the web services realm.

However, these semantics-based efforts did not reach the expected outcome of automatically enabling enterprise services collaboration. Formal ontologies were used to model data and logical restrictions that were validated by formal reasoning methods implemented in semantic web reasoners. Although this approach was theoretically very promising, since it included consistency checking or type inference, the reality proved that the supreme effort to create formal ontologies in different domains, to make them interoperable at a semantic level, and to provide functionalities such as data validation, was not efficient. More specifically, it was demonstrated [92] that, in most of cases, data validation, data lifting and data lowering processes were enough to provide an interoperable environment.

That is why the approach based on the W3C recommendations, WSDL+SOAP, fulfilled most of these requirements with a huge industrial and technological support. However, the lack of agreement on the schemas to be shared (any service provider offered their own schema) and the use of a restricted data model such as XML was still present with the result of preventing a paradigm shift.

In the specific case of software engineering and reuse, the application of semantics-based technologies has also focused on the creation of OWL ontologies to e.g. support requirements elicitation [34], and to model development processes [73] or Model Driven Architecture [50], to name just a few. These works leverage ontologies to formally design a meta-model and to meet the requirements of knowledge-based development processes.

Taking advantage of the Linked Data principles and Web standards and protocols, the OSLC effort emerged to create a family of web-based specifications for products, services and tools that support all the phases of the software lifecycle. To do so, OSLC defines several specifications based on the following principles: 1) *Build on the WWW*; 2) *Keep things simple*; 3) *Accommodate different schemes and protocols*; 4) *Accommodate different representations under a common data model (RDF) with different serialization formats (RDF/XML, JSON, etc.)*; and 5) *Align with the W3C Linked Data initiative*.

Similar to OSLC, Agosense Symphony [2] offers an integration platform for application and product lifecycle management, covering all stages and processes in a development lifecycle. It represents a service-based solution with a huge implantation in the industry due to the possibility of connecting existing tools. WSO2 [106] is another middleware platform for service-oriented computing based on standards for business process modelling and management. However, it does not offer standard input/output interfaces based on lightweight data models and software architectures such as RDF and REST. Other industry platforms such as PTC Integrity [91], Siemens Team Center [97], IBM Jazz Platform [67] or HP PLM [62] are now offering OSLC interfaces for different types of artefacts.

In conclusion, it is clear that tool integration and software reuse are active research areas that evolve according to the current trends in development lifecycles. They may have the potential of leveraging new technologies such as the web environment, service-oriented computing, semantics, and Linked Data. That is why current software reuse efforts are focused on providing reuse via software as a service while interoperability is being reached through the agreement on flexible data schemes. Both data schemes and data are being shared using a Linked Data approach (REST services + RDF) with the aim of exchanging any piece of information in a standard environment.

However, data exchange does not necessarily imply knowledge management. From service providers to data items, a knowledge strategy is required to really represent, store, and search software artefacts metadata and content. In this light, the OSLC initiative is currently following this approach, having impact on the main players of software and systems engineering industry. Nevertheless, it only covers a restricted

type of artefacts and some cross-cutting and basic services for reuse, such as indexing or retrieval, must be provided by all third-parties.

Lastly, a system and software knowledge repository to implement a real knowledge strategy for software reuse should be based on the following three requirements:

- 1) A language for representing any artefact's metadata and contents;
- 2) A system for indexing and retrieval, and;
- 3) A standard input/output interface (data shape + REST + RDF) to share and exchange artefact metadata and contents.

Another tool integration perspective is ad-hoc integration, with which connectors are created for specific tools. AMASS aims at Seamless Interoperability, but there are times where the seamless interoperability does not fulfil all the industrial needs from the point of view of some tools.

For example, in terms of performance, the OSLC standard may be used to perform some operations over one evidence artefact. However, if the tools need to do it over a set of evidence artefacts sequentially (also known as batch or bulk mode operations) and these operations have not been defined in the standard to perform them at once, they would force the integration to loop over the evidence artefacts and perform the operation evidence artefact after evidence artefact. This would lead to an unbearable delay for the user of the tool. Then at this point, if the native API of the connected tool allows to perform these operations in bulk mode, it is the most suitable alternative.

Other similar situation can be found in times when the standard allows to perform operations but in reality, these operations are only a subset, powerful but incomplete, of the operations needed to perform all the activities required by a tool from others, which are available in native APIs.

For example, in OSLC RM (OSLC for Requirements Management) the data model is a basic one, but it does not allow to create additional attributes for a requirement, thus other tools such as Requirements Quality Analyzer (by The REUSE Company) will be able to read requirements from this OSLC RM source, but it will not be able to store the quality assessment in any quality attribute created specifically for it by the Requirements Quality Analyzer tool.

Therefore, the tool vendor will automatically create a new ad hoc connector with the native API, or change its integration from the standard to that native API.

The ideal situation would be an extension or revision of the standard which allows these extended and the bulk operations.

2.4 Collaborative Work

In AMASS, collaborative work refers to the situation in which several stakeholders for CPS assurance and certification need to execute some activity together, including execution at the same time: model an assurance case, specify evidence information, etc. The current AMASS vision for collaborative work has two main requirements to fulfil:

- **Consistent data access**, so that when users are accessing data simultaneously, the AMASS Tool Platform manages the possible conflicts;
- **Real-time data access feedback**, so that the AMASS Tool Platform provides users with feedback about how data is being accessed by other user on real time.

Both requirements are linked in the sense that collaborative working will be achieved by (a) keeping consistency on any actions realised by concurrent users accessing at the same database and data models, and (b) informing users about the effects of any action realised by all the users accessing concurrently at the same database and data model. A data model is meant to have the same granularity as model files in a file-based implementation (e.g. argumentation, evidence or process models as well as any graphical models).

We could refine these requirements with the expected functionality by depicting a sequence diagram of a typical concurrent scenario (Figure 5). In this approach, we allow users to modify elements of a model (e.g., instances of a metaclass such as an Artefact of a given Evidence model) one at a time. We inform every user subscribed (having a model opened) about the full list of subscribed users.

If any user starts to modify an element which is under edition by other user, the former user will be blocked for edition of that element. The blocking will last until the latter user starts to edit another element (unlocking the previous edited element) or until saving the model. Modifications of different elements in the same model can be concurrently executed.

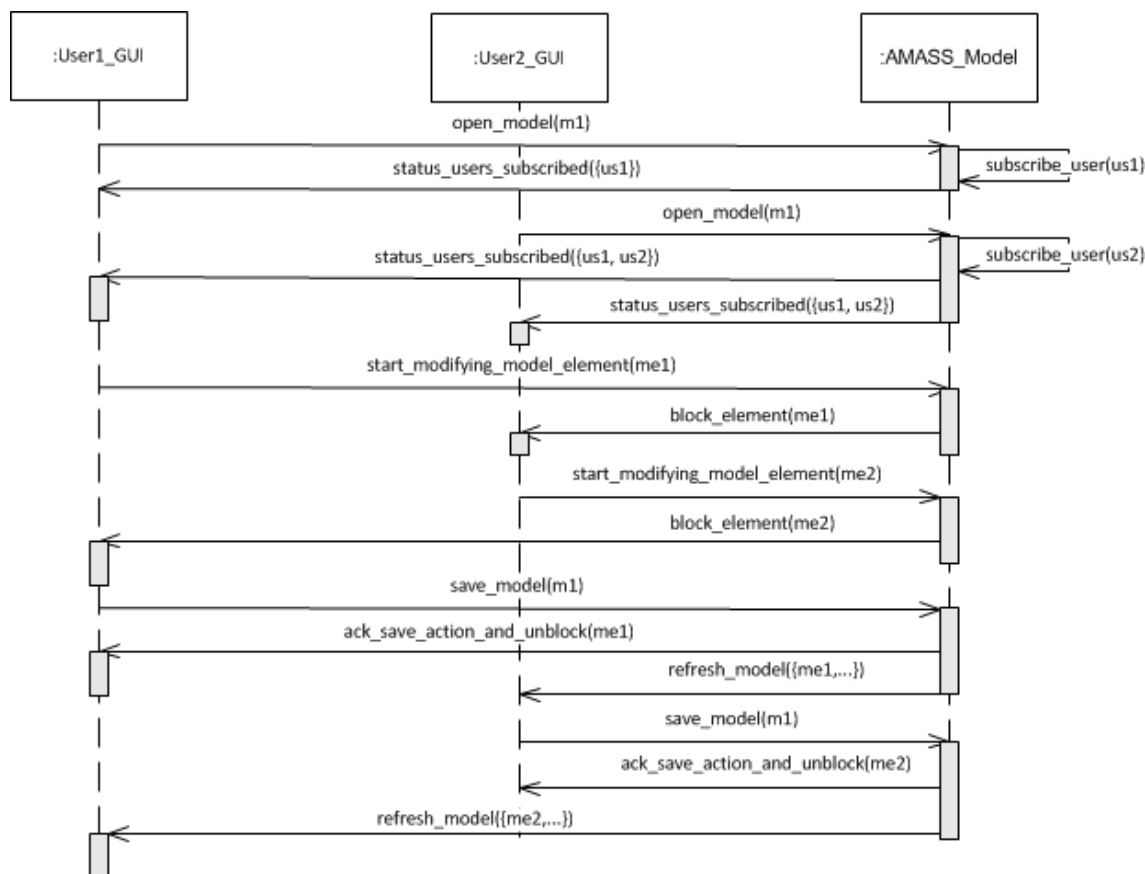


Figure 5. Integrity connector architecture sequence diagram

Another view of the AMASS vision for collaborative work is shown in Figure 6. In the scenario depicted, a user creating an argumentation model (1) with an Eclipse-based desktop editor (e.g. developed with GMF; 2) could lock elements while working on the model (3). The user could decide to start a collaboration to create the model (4) and, based on existing technologies such as Node.js (5), other users could contribute to the creation of the argumentation model with a web-based editor (6a and 6b). The changes in the model with Web-based editor would be notified and reflected in the Eclipse-based one (7a and 7b), and the user could decide to stop de collaboration at some moment (8).

2.5 Tool Quality Characterisation and Assessment

The envisioned needs for Tool Quality Characterisation and Assessment were already presented in D5.1 [16]. They are summarised in this section so that D5.3 is self-contained.

In the context of safety-critical systems engineering, software is increasingly developed and verified (semi)-automatically. Tools for code generation as well as for verification are introduced to automate, replace, or

supplement complex tasks. Since safety might be compromised if such tools fail, safety standards prescribe tool qualification processes.

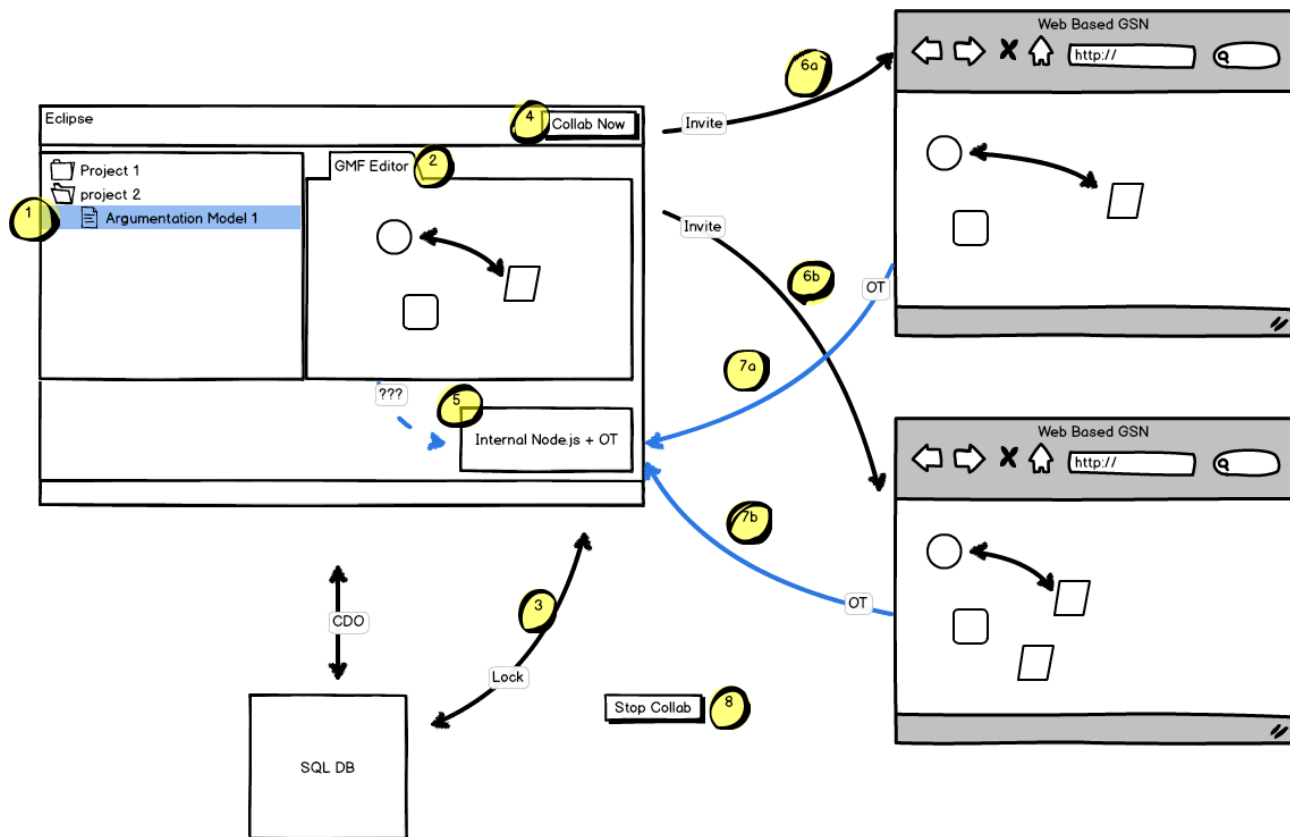


Figure 6. Collaborative-work vision with several editors

Tool qualification can roughly be defined as the provision of formal assurance that a tool's output can be trusted, e.g. the object code that a compiler generates. For AMASS, tool qualification in the scope of WP5 mainly concerns the possible need for tool information, as part of the seamless interoperability, for CPS assurance and certification. For example, if Papyrus is part of a tool chain resulting from the enactment of the seamless interoperability approach, what information about Papyrus should be managed in the corresponding assurance project? On the other hand, the use of a qualified tool, e.g. the GNATcheck tool [1] for static analysis of Ada programs, should lead to the management of its qualification dossier as part of the assurance project. Tool qualification processes deal with two categories of tools: development tools and verification tools.

Tool qualification processes typically consist of three phases: classification, qualification, and usage. During the classification phase, the tools are classified according to the level of confidence that is required to ensure their behaviour is in-line with the safety requirements. Levels are named differently from one standard to another (see standard-specific information below). If a tool is considered to be harmless, it can be used without requiring any qualification. During the qualification phase, the tools that were considered potentially harmful have to be qualified, i.e. manufacturers have to show absence of hazardous events (failures that might lead to accidents). Finally, during the usage phase, tools can be used within the specified restrictions.

It could be an issue for seamless interoperability to keep track of tool qualification status/level and of which tools have had an impact on each artefact. This would be a problem for certification since someone might need to show that the tool qualification level is sufficient for the artefacts. With many tools interacting and manipulating the same data, it can be difficult to keep track of this manually.

It should be noted that tool qualification processes do not tackle toolchains. In the context of AMASS, the qualification of the AMASS Tool Platform (seamless toolchain) is not the primary objective. Within AMASS, the requirements coming from the standards and pertaining to the tool qualification will be taken into consideration during the development in order to point out what should be done.

In automotive, ISO 26262 defines three tool confidence levels (TCL 1-3) that depend on:

- Tool impact, related to the possibility that a malfunction of a particular software tool can introduce or fail to detect errors in a safety-related item or element being developed.
 - TI1 shall be selected when there is an argument that there is no such possibility.
 - TI2 shall be selected in all other cases.
- Tool error detection, related to the confidence in measures that prevent the software tool from malfunctioning and producing corresponding erroneous output, or in measures that detect that the software tool has malfunctioned and has produced corresponding erroneous output.
 - TD1 shall be selected if there is a high degree of confidence that a malfunction and its corresponding erroneous output will be prevented or detected.
 - TD2 shall be selected if there is a medium degree of confidence that a malfunction and its corresponding erroneous output will be prevented or detected.
 - TD3 shall be selected in all other cases.

Tool qualification in avionics currently is governed by the DO-330 standard. It defines five tool qualification levels (TQL-1 to TQL-5), which are assigned to a given tool according to the software assurance level (A-D) and three criteria:

- Criterion 1: A tool whose output is part of airborne software and thus could insert an error.
- Criterion 2: A tool that automates verification processes and could fail to detect an error.
- Criterion 3: A tool that, within the scope of its intended use, could fail to detect an error.

As a rule of thumb, Criterion 3 is related to computer-aided specification, Criterion 2 to V&V tools, and Criterion 1 to compilers.

For railway application, the only relevant standard for tool qualification is EN 50128:2011, which provides engineering assistance tool requirements. Three tool classes are defined:

- T1, which is related to specification assistance (no safety impact in case of errors in this tool).
- T2, which is related to tools that if an error occurs, a safety requirement may be missed.
- T3, which is related to safe data computation.

The classes are therefore similar to those from avionics: T1 is related to computer-aided specification, T2 is related to test automation tools, and T3 to compilers.

3. Module Specification

This section presents the technology-specific solutions that will support and implement Seamless Interoperability in AMASS. The solutions have been divided into two main categories: Seamless Interoperability Means, e.g. specifications for tool data exchange, and Seamless Interoperability Components, e.g. those software components for collaborative work. Information about the state of the art, the state of the practice, and directions for Seamless Interoperability can be found in D5.1 [16].

3.1 Seamless Interoperability Means

This section describes the specific means currently designed for Seamless Interoperability in the AMASS Tool Platform.

3.1.1 OSLC KM

One of the cornerstones in knowledge management lies in the selection of an adequate knowledge representation paradigm. After a long time [63], this problem still persists since a suitable representation format (and syntax) can already be reached in several ways. Obviously, different types of knowledge require different types of representation [38][53]. But on the other hand, knowledge management also implies to the standardization of data and information. Any bit of information must be structured and stored to support other application services such as business analytics or knowledge discovery. This situation also creates an impedance mismatch between the system and the outside world. In this sense, semantic networks, based on concepts and relationships, seem to be a very good candidate to represent any knowledge item whatsoever.

In the context of software and system artefacts reuse, the Open Services for Lifecycle Collaboration (OSLC) initiative [94] is a joint effort between academia and industry to boost data sharing and interoperability among applications by applying the Linked Data principles [28]: *“1) Use URIs as names for things. 2) Use HTTP URIs so that people can look up those names. 3) When someone looks up a URI, provide useful information, using the standards (RDF*, SPARQL) and 4) Include links to other URIs, so that they can discover more things”*. Led by the OASIS OSLC working group [88], OSLC is based on a set of specifications that take advantage of web-based standards such as the Resource Description Framework (RDF) [57] and the Hypertext Transfer Protocol (HTTP) to share data under a common data model (RDF) and protocol (HTTP). Every OSLC specification defines a *shape* for a particular type of resource. For instance, requirements, changes, test cases, models (the OSLC-MBSE specification for Model-Based Systems Engineering by the Object Management Group) or estimation and measurement metrics, to name a few, have already a defined shape (also called OSLC Resource Shape).

Thus, tools for supporting Application Life-cycle Management (ALM) or Product Life-cycle Management (PLM) have now an agreement on what data must be shared, and how. In the knowledge management framework, the *Assets Management* and the *Tracked Resource Set* are the most convenient specifications for the purpose of managing artefacts. However, there are many artefacts generated during the development lifecycle which may not fit to existing shapes or standard vocabularies. Simulation models, business rules or physical circuits are examples of potential artefacts whose OSLC resource shape is not defined yet. Furthermore, some common and useful services such as indexing, naming, retrieval, quality assessment, visualization, or traceability must be provided by all tool vendors, creating a tangled environment of query languages, interfaces, formats and protocols.

As a result, one of the current trends in software development lies in boosting interoperability and collaboration through the sharing of existing artefacts under common data models, formats and protocols. In this context, OSLC is becoming a collaborative software ecosystem [78] for software product lines [105] through the definition of data shapes that serve as a contract to get access to information resources through HTTP-based services.

In particular, the Representational State Transfer (REST) software architecture style is used to manage information resources that are publicly represented and exchanged in RDF. Obviously, OSLC represents a big step towards the integration and interoperability between the agents involved in the development lifecycle.

However, RDF has been also demonstrated [90] to contain some restrictions to represent certain knowledge features such as N-ary relationships [83], and practical issues dealing with reification [82] and blank nodes [76]. On the other hand, some common services such as indexing, retrieval or quality assessment of any kind of information resource are restricted to the internal storage and the query capabilities offered by each particular tool (usually a SPARQL interface).

3.1.1.1 Data Shapes vs Formal Ontologies

In the early days of the Semantic Web, formal ontologies [22] designed in RDFS (Resource Description Framework Schema) or OWL were the key technologies to model and share knowledge. From upper ontologies such as DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering) or SUMO (The Suggested Upper Merged Ontology) to specific vocabularies such FOAF (Friend Of A Friend) or SKOS (Simple Knowledge Organization System), the process to share knowledge consisted in designing a formal ontology for a particular domain and populate data (instances) for that domain. Although the complete reuse of existing ontologies was expected, the reality demonstrated that every party willing to share knowledge and data would create its own ontologies. Thus, the main idea behind web ontologies was partially broken since just a few concepts were really reused.

Once the Linked Data initiative emerged to unleash the power of existing databases, a huge part of the Semantic Web community realized that a formal ontology was not completely necessary to exchange data and knowledge. Taking into account that ontologies were still present, these efforts were based on validating data consistency [22] through the execution of procedures such as: 1) reasoning processes to check consistency, and 2) rules, mainly in SWRL (Semantic Web Rule Language) or SPARQL [27][59]. These procedures are not recommended, due to performance issues, when a huge number of instances are available. As a new evolution, then, the community realized that ontology-based reasoning was not the most appropriate method for data validation.

That is why in recent times the RDF community has seen an emerging interest to manage and validate RDF datasets according to different shapes and schemes. New specifications and methods for data validation are being designed to turn reasoning-based validation into a kind of grammar-based validation (Table 1). These methods take inspiration from existing approaches in other contexts such as DTD (Document Type Definition), XML-Schema or Relax NG (REgular Language for XML Next Generation) for XML, or DDL (Data Definition Language) for SQL (Structured Query Language).

The W3C has recently launched (2014) the RDF Data Shapes Working group (the SHACL- Shapes Constraint Language- is currently being defined) and the ShEX (Shape Expressions) language [29][36][51]. Both are formal languages for expressing constraints on RDF graphs including cardinality constraints as well as logical connectives for disjunction and polymorphism. As other examples of data validation, OSLC Resource Shapes [94], Dublin Core Description Set Profiles [37], and RDF Unit [72] are also constraint languages for domain specific RDF resources.

Following a more classical approach for RDF data validation, the Pellet Integrity Constraints is an extension of the existing semantic web reasoner [99] that interprets OWL ontologies under the Closed World Assumption with the aim of detecting constraint violations in RDF data. These restrictions are also automatically translated into SPARQL queries. This approach has been implemented on top of the Stardog database [101], enabling users to write constraints in SPARQL, in SWRL, or as OWL axioms. Finally, the SPIN language [74] also makes use of SPARQL (mainly its syntax) to define constraints on RDF-based data that can be executed by systems supporting SPIN (SPARQL Inferencing Notation), such as the TopBraid's toolchain.

Table 1. Summary of the main approaches to validate RDF-encoded data

Process	Type	Creation	Scope	Refs.
Consistency check	Vocabulary-based	Semantic Web reasoner	RDF datasets	[22]
Data validation (integrity)	Query-based	Hand-made RDF templates	RDF datasets	[27]
	Vocabulary-based	Hand-made	RDF datasets	[59]
	Vocabulary-based	Hand-made or automatically generated by an OSLC API	OSLC Resource Shape	[94]
	Vocabulary-based	Hand-made	Dublin Core Description Set Profiles	[37]
	Query-based	RDF Unit (test creation)	RDF datasets	[72]
	Query-based (generated from ShEX expressions)	Automatic generation of SPARQL queries	RDF datasets	[29][36] [51][3]
	Vocabulary and Query-based	Automatic generation of SPARQL queries	OWL and RDF under Closed World Assumption	[99]
	Query-based	SPIN language + SPARQL queries	RDF datasets	[74]

In conclusion, the relevance of data validation to exchange RDF-encoded data is clear. RDF Data Shapes in their different flavours, such as OSLC Resource Shapes, are becoming the cornerstone for boosting interoperability among agents. It is also clear that ontologies are becoming less important although a combined approach (data shapes and a formal ontology) can provide important benefits in terms of data validation and knowledge inference (if needed). In the context of software reuse, as it has been outlined above, software artefacts must take advantage of new technologies to enable practitioners the automatic processing of exchanged data.

3.1.1.2 Knowledge Representation Mechanisms

One of the cornerstones to provide knowledge management services for software reuse lies on the selection of an adequate knowledge representation paradigm. After a long time of research [63], this problem still persists, since the choice of a suitable representation format (and syntax) can be reached in several ways.

Obviously, different types of knowledge require different types of representation [38][53], including in some cases inference capabilities. In this light, expressions, rule-based systems, regular grammars, semantic networks, object-oriented representations, frames, intelligent agents or case-based models, to name a few, are some of the main approaches to information and knowledge modelling.

According to the current context for software reuse, it seems that graph approaches based on semantic networks, and deployed under a set of standards in a service-oriented environment (see Figure 7 and Figure 8), are the most appropriate candidates. Taking into account this environment, the next knowledge representation paradigms (focusing on web-oriented technologies) have been selected for comparison:

- The Resource Description Framework [57] (RDF) is a framework for representing information resources in the Web using a directed graph data model. The core structure of the abstract syntax is a set of triples, each consisting of a subject, a predicate and an object. A set of such triples is called an RDF graph. An RDF graph can be visualized as a nodes and directed-arcs diagram, in which each triple is represented as a node-arc-node link. RDF has been used as the underlying data model for building RDFS/OWL ontologies, gaining momentum in the web-based environment due to the explosion of the Semantic Web and Linked Data initiatives that aim to represent and exchange data (and knowledge) between agents and services under the web-based protocols.

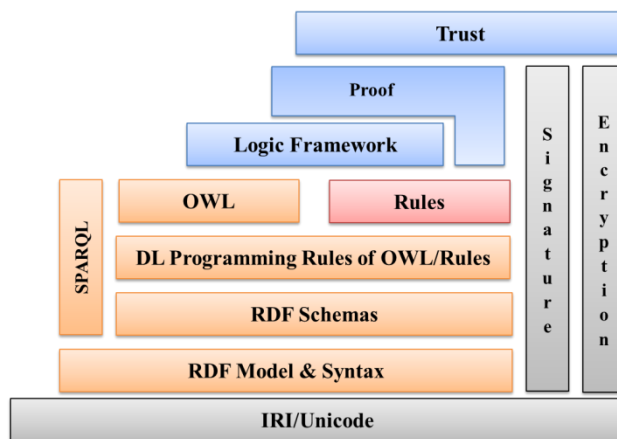


Figure 7. Semantic Web Architecture (2005) [61] - "The Two Towers"

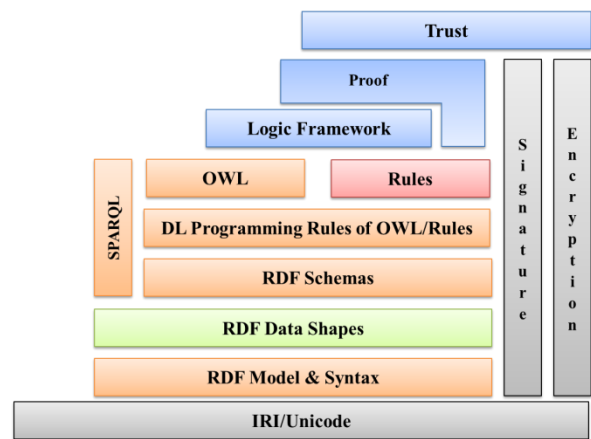


Figure 8. Semantic Web Architecture (2015)

- RDF Schema (RDFS) [31] provides a data-modelling vocabulary for RDF data. It can be seen as a first try to support the creation of formal and simple ontologies with RDF syntax. RDFS is a formal and simple ontology language in which it is possible to define class and property hierarchies, as well as domain and range constraints for properties. One of the benefits of this property-centric approach is that it allows anyone to extend the description of existing resources.
- OWL (Ontology Web Language) [58] is an ontology language for capturing meaningful generalizations about data in the Web. It includes additional constructors for building richer class and property descriptions (vocabulary) and new axioms (constraints), along with a formal semantics. OWL 1.1 consists of three sub-languages with different levels of expressivity: 1) OWL Lite, 2) OWL DL (Description Logic) and 3) OWL Full.

The OWL 2.0 [58] family defines three different profiles: OWL 2 EL (Expressions Language), OWL 2 QL (Query Language) and OWL 2 RL (Rule Language). These profiles can be seen as a syntactic restriction of the *OWL 2 Structural Specification* and more restrictive than OWL DL. The use of profiles is motivated by the needs of different computational processes. OWL EL is designed for enabling reasoning tasks in polynomial time. The main aim of OWL 2 QL is to enable conjunctive queries to be answered in LogSpace using standard relational database technology. Finally, OWL 2 RL is intended to provide a polynomial time reasoning algorithm using rule-extended database technologies operating directly on RDF triples. In conclusion, OWL 2.0 adds new functionalities regarding OWL 1.x. Most of them are syntactic sugar but others offer new expressivity [58]: keys, property chains, richer datatypes, data ranges, qualified cardinality restrictions, asymmetric, reflexive, and disjoint properties; and enhanced annotation capabilities. RIF Core (Rule Interchange Format) [30] comprises a set of dialects to create a standard for exchanging rules among rule systems, in particular among Web rule engines. RIF was designed for exchanging rather than developing a single one-fits-all rule language.

- RIF dialects fall into three broad categories: first-order logic, logic-programming, and action rules. The family of dialects comprises: 1) logic-based dialects (RIF-BLD) including languages that employ some kind of logic such as First Order Logic (usually restricted to Horn Logic) or non-first-order logics; 2) rules-with-actions (RIF-PRD) dialects comprising rule systems such as Jess, Drools and JRules as well as event-condition-action rules such as Reaction RuleML and XChange. RIF also defines compatibility with OWL and serialization using RDF.
- The RSHP universal knowledge representation model [40][75] is based on the ground idea that any information can be described as a group of relationships between concepts. Therefore, the leading element of an information unit is the relationship. For example, Entity/Relationship data models are certainly represented as relationships between entity types; software object models can also

be represented as relationships among objects or classes; in the process modelling area, processes can be represented as causal/sequential relationships between sub-processes. Moreover, UML (Unified Modeling Language) or SysML (Systems Modeling Language) metamodels can also be modelled as a set of relationships between metamodel elements.

RSHP also includes a repository model to store information and relationships with the aim of reusing all kind of knowledge chunks. Furthermore, free text information can certainly be represented as relationships between terms by means of the same structure. Indeed, to represent human language text, a set of well-constructed sentences, including the subject+verb+predicate (SVP), should be used. The SVP structure can be then considered as a relationship typed V between the S and the predicated P. More specifically, the RSHP formal representation model is based on the following principles:

1. The main description element is the relationship since it is the element in charge of linking knowledge elements.
2. A Knowledge Element (KE) is an atomic knowledge component that appears into an artefact and that is linked by one or more relationships with other KEs to build information. It is defined by a concept, and it can also be an artefact (an information container found inside a wider artefact). A concept is represented by a normalized term (a keyword coming from a controlled vocabulary, or domain). Artefacts are knowledge containers of KEs and their relationships.

In RSHP, the simple representation model for describing the content of whatever artefact type (requirements, risks, models, tests, maps, text docs or source code) should be:

RSHP representation for artefact $\alpha = i_\alpha = \{(RSHP_1), (RSHP_2), \dots, (RSHP_n)\}$ where every single RSHP is called RSHP-description and must be described using KE.

One important consequence of this representation model is that there is no restriction to represent a particular type of knowledge. Furthermore, RSHP has been used as the underlying information model to build general-purpose indexing and retrieval systems, domain representation models [40], approaches for quality assessment of requirements and knowledge management tools such as knowledgeMANAGER [103].

Obviously, a plethora of other knowledge representation mechanisms and paradigms can be found as it is presented below. However, we focus here on comparing those that satisfy the three basic requirements of this study: 1) a language for representing any artefact metadata and contents; 2) a system for indexing and retrieval and 3) a standard input/output interface (data shape+REST+RDF) to share and exchange artefact metadata and contents.

- The SBVR (Semantics of Business Vocabulary and Rules) is an OMG standard to define the basis for formal and detailed natural language declarative description of a complex entity.
- The Ontology Definition Metamodel (ODM) is an OMG standard for knowledge representation, conceptual modelling, formal taxonomy development and ontology definition. It enables the use of a variety of enterprise models as starting points for ontology development through mappings to UML and MOF. ODM-based ontologies can be used to support: 1) interchange of knowledge; 2) representation of knowledge in ontologies and knowledge bases; and 3) specification of expressions that are the input to, or output from, inference engines.
- The Reusable Asset Specification (RAS) is an OMG standard that addresses the engineering elements of reuse. It attempts to reduce the friction associated with reuse transactions through consistent, standard packaging.

3.1.1.3 Evaluation and Selection of a Knowledge Representation Mechanism

The previous section has reviewed the main approaches for knowledge representation in a web-oriented environment. In order to select the proper mechanism for knowledge representation of software artefacts, the following points must be considered:

- RDF is based on a directed graph and can only represent binary relationships (unless reification and blank nodes are used). As a representation mechanism, RDF presents some restrictions that have been outlined in several works [90]. For instance, N-ary relationships [83], practical issues dealing with reification [82] and blank nodes [76] are well-known RDF characteristics that do not match the needs of a complete framework for knowledge representation. Furthermore, RDF is built on two main concepts: resources and literals. However, a literal value cannot be used as the subject of an RDF triple. Although this issue can be overcome using a blank node (or even reification) and the property *rdf:value*, it adds extra complexity for RDF users. Finally, RDF has been designed to represent logical statements, constraining also the possibility of representing other widely used paradigms such as objects or entity-relationships models. Due to these facts, it seems clear that RDF can be used for exchanging data but it is not the best candidate for knowledge representation.
- RDFS is a good candidate for modelling lightweight formal ontologies, including some interesting capabilities close to object-oriented models. It can be serialized as RDF, but this can also be a disadvantage due to expressivity restrictions of RDF. RDFS has been designed for expressing logical statements that describe web resources, so its use for other types of information is not advisable.
- Building on the previous discussion, OWL presents a family of logic dialects for knowledge representation. It is based on strong logic formalisms such as Description Logic or F-Logic. It was also designed for asserting facts about web resources although it can be used as a general logic framework for any type of knowledge. One of the main advantages of OWL is the possibility of performing reasoning processes to check consistency or infer types. However, reasoning can be considered harmful in terms of performance and most of times it is not necessary when data is being exchanged. Besides, OWL is not the best candidate for data validation, a key process in knowledge exchange. This situation has been outlined in Section 2.1.
- RIF Core and the family of RIF dialects have been included in this comparison because most of domain knowledge is embedded in rules. Nevertheless, RIF was not designed for data validation and its acceptance is still low (just a few tools export RIF and less are capable of importing RIF files). On the other hand, RIF makes use of the web infrastructure to exchange rules, what means also that this environment is a very good candidate to exchange data, information and knowledge.
- RSHP, based on relationships, allows domain experts to create relationships between terms, concepts or even artefacts (containers). It provides a framework for knowledge representation with capabilities for expressing any kind of cardinality and N-ary relationships. RSHP is based on undirected property graphs, enhancing expressivity. Although it has not been directly designed for data validation, its metamodel allows the possibility of checking cardinality, value, domain and range restrictions. One of the strong points of RSHP is the native support of a tool such as knowledgeMANAGER and the possibility of automatically providing semantic indexing and retrieval mechanisms. Both have generated a strong acceptance and implantation in the industry for requirements authoring and quality checking [5]. As minor drawbacks, this tool was not conceived to be used in a web environment and it is not a standard; nevertheless, it can export/import RDFS and OWL ontologies.
- Finally, as a general comment, there is also a lack of tools working natively on RDF. Generally speaking, RDF was conceived to exchange information over the web. Although some RDF repositories can provide capabilities for indexing and searching RDF resources through an SPARQL interface, the experience has demonstrated that most of times RDF is translated into the native data model of a tool. Thus, the possibility of supporting cross-cutting services such as semantic indexing and retrieval processes is constrained by the native capabilities and data models of third-party tools.

Based on this evaluation and considering the three basic requirements of this study, we conclude that RDF is a good alternative to exchange data. Since formal ontologies and reasoning processes are not completely necessary and, instead, data validation is a key aspect for boosting interoperability, it also seems clear that RSHP fits perfectly to the major objective of knowledge representation. However, we note that the use of formal RDFS or OWL ontologies is not incompatible with RSHP, but possible and enriching. RDFS and OWL are languages for building domain vocabularies, while RSHP is already a domain vocabulary for knowledge representation, so that it is possible to define RSHP through a formal RDFS or OWL ontology.

Due to all of these reasons a combination of RDF and RSHP can create the proper environment for knowledge management: RDF as input/output interface for exchanging data, and RSHP as internal data model to provide advanced services on software reuse. To do so, two main strategies should be designed and implemented:

1. Specify an OSLC Resource Shape of the RSHP metamodel. Besides, and taking into account that the type of information that can be exchanged through OSLC is restricted to just a few artefacts, the RSHP shape can enhance and ease data exchange when providers do not know how to export data.
2. Define a set of mappings [5] to represent any piece of RDF in RSHP (i.e. external to internal representation). Thus, it will be possible to import any kind of existing RDF data source into RSHP (backward compatibility)

3.1.1.4 OSLC-KM: A New OSLC Domain for Knowledge Management

Knowledge management and standard knowledge representation mechanisms have been introduced in the previous sections. Any software or system artefact is now considered a knowledge asset that must be represented, stored, shared and exchanged between tools in a development process. On the other hand, the OSLC initiative is making a strong commitment to apply the principles of Linked Data, RDF and REST to boost interoperability.

Specifications, or more precisely data shapes, have already been defined to model metadata and contents of requirements, assets, test cases, changes and estimation and measurement metrics. In the same way, the OMG group is working on the OSLC-MBSE specification to promote system models to Linked Data. However, there are still some artefacts for which there is no shape, such as an element of a vocabulary, a requirements pattern or a Dynamic System Model. Due to this fact, a common strategy for knowledge management is hard to draw. Moreover, some cross-cutting services such as indexing and retrieval processes are delegated in third-party tools, preventing the implementation of one of the cornerstones of knowledge management for software reuse: selection.

Therefore, we present a data shape for any artefact generated during the development lifecycle. This data shape gives a response to the three basic requirements that have been identified for a modern software knowledge repository, and it accommodates to the processes in a knowledge management strategy, RSHP: the language for representing the metadata and contents of any artefact.

1. knowledgeMANAGER [103]: the basic tool that provides the required services for the software knowledge repository.
2. RDF data shape and OSLC interface: the RSHP language is offered through an input/output OSLC interface, satisfying the need of reusing standards in a web environment.

As it has been previously outlined, and in order to combine RDF and RSHP, it is necessary to provide an RDFS/OWL ontology, i.e. an RDF vocabulary, that defines the entities and relationships in the RSHP representation model to make this specification publicly available and to enable the expression of any piece of knowledge using RSHP.

On the other hand, and due to the fact that a huge amount of data, services and endpoints based on RDF and the Linked Data principles are already publicly available, a mapping between any RDF vocabulary and RSHP is completely necessary to support backward compatibility and to be able to import any piece of RDF data into RSHP.

3.1.1.5 Definition of the Resource Shapes for Knowledge Management

In this case and taking into account the guidelines and definitions of the OSLC Core specification, the data shape for knowledge management will conform the next basic OSLC definitions [94]:

1. “An OSLC Domain is one ALM (Application Lifecycle Management) or PLM (Product Lifecycle Management) topic area”. Each domain defines a specification.

In this case, a new domain is being defined: Knowledge Management (KM).

2. “An OSLC Specification is comprised of a fixed set of OSLC Defined Resources”.

According to the analysis in Section 3.2, the RSHP representation metamodel (Figure 9) will be used as the underlying shape for knowledge items. In order to simplify the external view of RSHP and ease the creation of a set of OSLC Defined Resources, two main changes have been made to the metamodel:

- The classes *Artifact* and *KnowledgeElement* have been merged. However, the model keeps all the semantics since an artefact is considered a KnowledgeElement (just a concept) when it does not contain any relationship.
- The tag and value of a metaproperty now point to a *Term* instead of a *KnowledgeElement*.

The key concepts of this metamodel are the Artifact and RSHP classes. An Artifact is a container of relationships (RHSP) that can have metaproperties (authoring, versioning, visualization features and, in general, provenance information). If an Artifact only represents the apparition of a term, it will contain a reference to the term (element of a controlled vocabulary or taxonomy). This term can have a grammatical category (TermTag) such as name, pronoun, adverb, or verb. In the same manner, a semantic category (SemanticCluster) represented by a term can be assigned to a term for instance the semantics “negative”. Thus, different terms can have different semantics. Finally, a relationship establishes a link between *n* Artifacts and semantics can be also attached to the link, e.g. “part-of”.

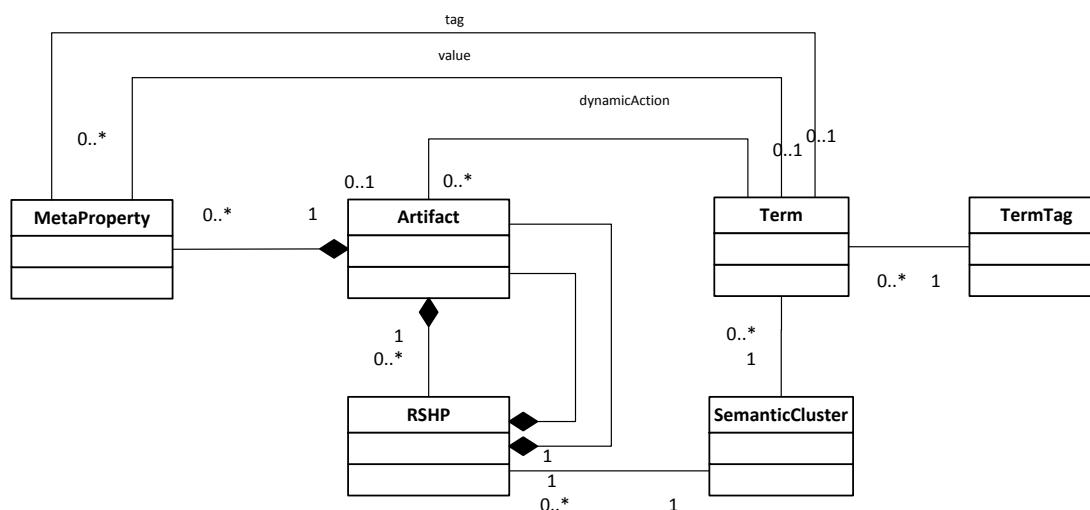


Figure 9. UML Class Diagram of the OSLC Knowledge Management Resource Shape

3. “An OSLC Defined Resource is an entity that is translated into an RDF class with a type”. Every resource consists of a fixed set of defined properties whose values may be set when the resource is created or updated.

In this case and following the previous design, a shape for every class has been defined. Table 2 presents the resource shape links to the official definition (prefix:name, e.g. *oslc_km³:Artifact*) and a brief description of the resource.

³ The prefix *oslc_km* refers to the URI: <https://www.reusecompany.com/oslc/km/>

Taking into account that the Linked Data Initiative has seen in recent times the creation of methodologies, guidelines or recipes [64][55][4] to publish RDF-encoded data, we have paid special attention to follow a similar approach by reusing existing RDF-based vocabularies. More specifically, the following rules have been applied to create the OSLC resource shapes:

- If there is an RDF-based vocabulary that is already a W3C recommendation or it is being promoted by other standards organization, it must be used as it is, by creating an OSLC Resource Shape.
- If there is an RDF-based vocabulary but it is just a *de-facto* standard, it should be used as it is, by including minor changes in the creation of an OSLC Resource Shape.
- If there is not an RDF-based vocabulary, try to take advantage (reusing properties and classes) of existing RDF-based vocabularies to create the OSLC Resource Shape.

In the particular case of knowledge management, we have selected the Simple Knowledge Organization System (SKOS), a W3C recommendation, to define concepts, since it has been designed for promoting controlled vocabularies, thesauri, taxonomies or even simple ontologies to the Linked Data initiative. That is why, in our model, most of the entities can be considered as a *skos:Concept* and we have created the shape of this standard definition of concept in the resource *los_km:Concept*.

4. “An OSLC Defined Property is an entity that is translated into an RDF property”. It may define useful information such as the type of the property, datatypes and values, domain, range, min. and max. cardinality, representation (inline or reference) and readability.
5. An OSLC Service Provider is a tool that offers data implementing an OSLC specification in a REST-fashion.

Table 2. OSLC Resource Shapes description for OSLC Defined Resources within the Knowledge Management domain

RSHP Class	OSLC Resource Shape Name	Description
Artifact	<code>oslc_km:Artifact</code>	A container of relationships between concepts and metaproperties to semantically describe any piece of information. It is the basis for the creation of an underlying semantic network (not based on logic formalisms).
Metaproperty	<code>oslc_km:MetaProperty</code>	A wrapper of a metaproperty containing a tag and a value. Both can be any type of resource or, more specifically, concepts.
RSHP	<code>oslc_km:RSHP</code>	An RSHP is a wrapper to create a relationship between any set of resources. It is possible to add semantics and it can contain any number of elements representing binary, ternary or even n-ary relationships.
Term	<code>oslc_km:Concept</code>	This concept follows the semantics and shape of a <i>skos:Concept</i> [23]. More specifically: “ <i>the notion of a SKOS concept is useful when describing the conceptual or intellectual structure of a knowledge organization system, and when referring to specific ideas or meanings established within a KOS (Knowledge Organization System)</i> ”.
SemanticCluster	<code>oslc_km:Concept</code>	See previous description.
TermTag	<code>oslc_km:Concept</code>	See previous description.

3.1.1.6 Delegated Operations (*)

Although the OSLC approach is perfectly valid for exchanging data resources, there is a huge number of interesting functionalities available in the different tools that should be considered as candidates to be reused through interoperability-based services. As a motivating example, if a model has been created in

Papyrus, the engineer may want to check the quality of such model with the IBM Rhapsody capabilities so the next questions arises: How can we expose functionalities of existing tools in terms of OSLC concepts (enabling operations)?

Actually, this is a topic that has been widely studied in the field of web services where standards such as WSDL (Web Services Description Language) and SOAP were defined to establish a standard way to invoke functionalities via internet protocols (operation-oriented service). The success of web services comes with a lot of APIs already available so an approach for reuse may consider the possibility of invoking existing services but following the principles of an interoperable environment: a common and shared model and a communication protocol. In the first case, WSDL-based services define a metamodel of the data to be exchanged via an XML-Schema. Each service defines their own XML-Schema so, in some cases, the concept of interoperability is hard to reach because mappings between the data generated from the service and the model of the consumer must be aligned. To ease this operation, also known as “grounding”, semantic web services emerged to provide a common model, an ontology, that would be translated into the specific providers. However, the reality showed that the time and effort to transform an abstract model (the ontology) to a specific model (XML-Schema) was not efficient. Furthermore, it has an implicit implication since this kind of transformation occurs under different levels of knowledge representation (logics vs object models). Secondly, the SOAP protocol is basically an HTTP Post request with attachments. In general, this is a standardized protocol that works perfectly. However, the effort to create and consume requests is greater than the mere invocation of an URL via HTTP.

On the other hand, it is also possible to find the notion of “Delegated User Interface” (DUI) in the OSLC specifications. The main objective of the DUI is to provide a better usability experience for third-party consumers of OSLC services. When a user must select or create a resource, s/he can use the native interface of the OSLC provider instead of creating a new one. The unique requirement is that the OSLC provider must have an HTML-based interface. The way of accessing a DUI is natively specified in the OSLC Service description.

Considering the need of keeping backwards compatibility with existing WSDL-SOAP services (even others under protocols such as JSON-RPC) and the notion of DUI in OSLC, we define here the concept of “Delegated Operation” as a function that is exposed by an OSLC service in terms of OSLC resources. It represents a kind of gateway between existing functionality and an OSLC-based environment (Linked Data+REST). In this way, the proposed approach is a hybrid method to expose resource and operation-oriented services.

In the context of the OSLC KM specification, the delegated operations of an OSLC KM provider (Table 3 and Table 4) shall accomplish the following requirements:

- A procedure/function available in a service provider.
- Generalization of the “Delegated User Interface” OSLC concept.
- A delegated operation shall describe its interface like a WSDL service.
 - Host/Port
 - Input parameters
 - Output
- An OSLC KM provider shall implement a system knowledge repository (SKR) comprising a “System Knowledge Base” (SKB) and a “System Assets Store” (SAS).
- A delegated operation shall receive as input an OSLC KM artifact.
- A delegated operation shall generate as output an OSLC KM artifact or a value with a simple data type.
- A delegated operation shall serialize data following the normative OSLC formats (RDF/XML and RDF/JSON) and JSON.

Table 3. Delegated operations for an OSLC KM provider (SKB)

Delegated operation in SKB	
Base URI/prefix	http://www.reusecompany.com/oslc/km/operations
Reuse	<base_uri>/skr Query params: operation = {diff, merge, copy} Body params: content = {srl}
Index	<base_uri>/skb/index Query params: type={text srl table} Body params: content={content}
Trace	<base_uri>/skb/{id}/trace Query params: type={trace type} to = {id}
Visualize	<base_uri>/skb/{id}/visualize
Normalize*	<base_uri>/skb/normalize Query params: type={text srl} Body params: content={content}

Table 4. Delegated operations for an OSLC KM provider (SAS)

Delegated operation in SAS	
Base URI/prefix	http://www.reusecompany.com/oslc/km/operations
Search artifact	<base_uri>/sas/search Query params: query={text} Body params: srl={srl content}
Filter	<base_uri>/sas/filter -Similar to OSLC query capabilities -Similar to LinkedIn API to express filters on attributes: {(key=value,)+}

3.1.2 Automatic Generation of OSLC KM-based Connectors

The OSLC KM connector has been researched and defined in the Knowledge Reuse Group at UC3. One of the additional topics to improve the OSLC KM connector is to generate automatically the specific connectors by only providing one example of the XML file to transform. The work done to solve this problem is the main topic for this section.

The background of the work is as follows:

- RSHP [75] is a universal information representation model based on relationships. It allows a user to handle all kind of artefacts (text, diagrams, code, etc.) using the same representation schema. Therefore, it is possible to generalize the management of these different artefacts.
- CAKE [98] is a framework of tools, applications, and methodologies to identify, classify, organise, and reuse knowledge. The framework aims to allow a user to manage any kind of “knowledge assets”. It uses RSHP as base information representation model. Among its main functionalities, CAKE support information indexing and retrieval. Indexing of an artefact’s information (i.e., the knowledge represented in it) is performed according to the information types and structure in RSHP. CAKE also uses ontologies as reference knowledge bases from which further knowledge can be derived. This is especially important for retrieval. The use ontologies allow CAKE to e.g. exploit synonyms for information search. The terms in a RSHP model are part of the ontologies.
- XML technology provides an ideal representation for the complex structure of models. XML is composed of text and tags that explicitly describe the structure and semantics of the content of a document. The data model for XML is very simple or very abstract, depending on one’s point of view. XML document is a linearization (very simple) of a tree linked structure (very abstract). Among other benefits, XML allows to keep large amounts of complex information in a linear way.

Also, XML represent information through natural language and with an understandable way which makes it a useful technology to manage knowledge.

The main goal is to allow the user to transform any knowledge within a well-formed XML document into the RSHP language. After this transformation, it is possible to use knowledge represented in RSHP in retrieval algorithms and techniques in AMASS.

The input of this process (Figure 10) is a XML file containing a representative model of the information or knowledge to be retrieved later in the AMASS platform. The output of this process will be a XSLT file with all the mappings from the XML document to SRL language.

The process includes the following stages:

1. Load and validate the input XML document.
2. The algorithm shows the inferred information and proposes an initial mapping between this information and the entities in the SRL language (target language; based on RSHP).
3. This mapping can be fine-tuned by the user.
4. The output is stored in the form of a XSLT file.

Once the XSLT resulting from the mapping is generated, the user could index any XML document belonging to the same context of the XML document mapped. This will be done in the same way as described by Mendieta et al. [79].

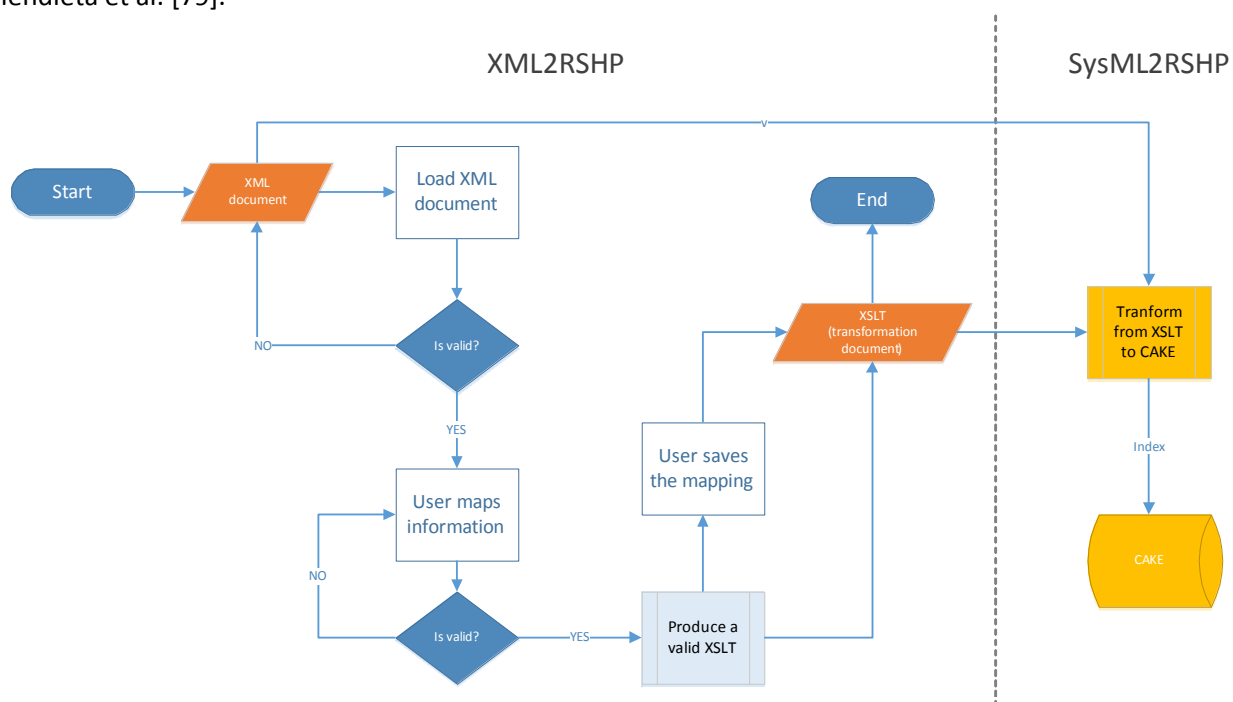


Figure 10. XML2SRL workflow process

3.1.3 Ad-hoc Tool Integration

The TRC toolset is used as a basis of for the study of ad-hoc tool integration. TRC's RQS suite comprises different ad-hoc connectors to enable retrieving requirements from them and run quality assessment processes.

There have been new additions to this set of ad-hoc connectors:

- Integrity: proprietary RMS tool by PTC.
- ReqIF: it is a standard to represent requirements in an XML that can be stored in a textual file.
- Rhapsody: even if Rhapsody is focused on modelling, there are requirements being part of those models, the integration will focus on retrieving them, not the model itself.

3.1.3.1 ReqIF Connector Integration

This is a new ad-hoc connector to retrieve and author requirements from ReqIF specifications (Figure 11).

ReqIF is a well-known standard to represent requirements in XML format. Its structure allows to have several specifications within one project. Indeed, every single ReqIF XML file is considered as a project. Within the project, it may contain from 0 to N blocks or Specifications. Finally, each Specification may contain both, hierarchically-related Specifications and Objects (requirements). In addition to that, ReqIF allows traceability by creating Relations between Objects within the same ReqIF file.

Despite of the fact that ReqIF is a well-known standard, all the information that is contained within the file is meta-defined. It means that it does not contain fixed attributes to contain the different attributes of the Objects but contains meta-definition of attributes that are part of the Objects. So that every single 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 RQA/RAT metamodel. This is compulsory to let the tools know where to extract the Statement, Heading, Author, etc, from each ReqIF Object (requirement).

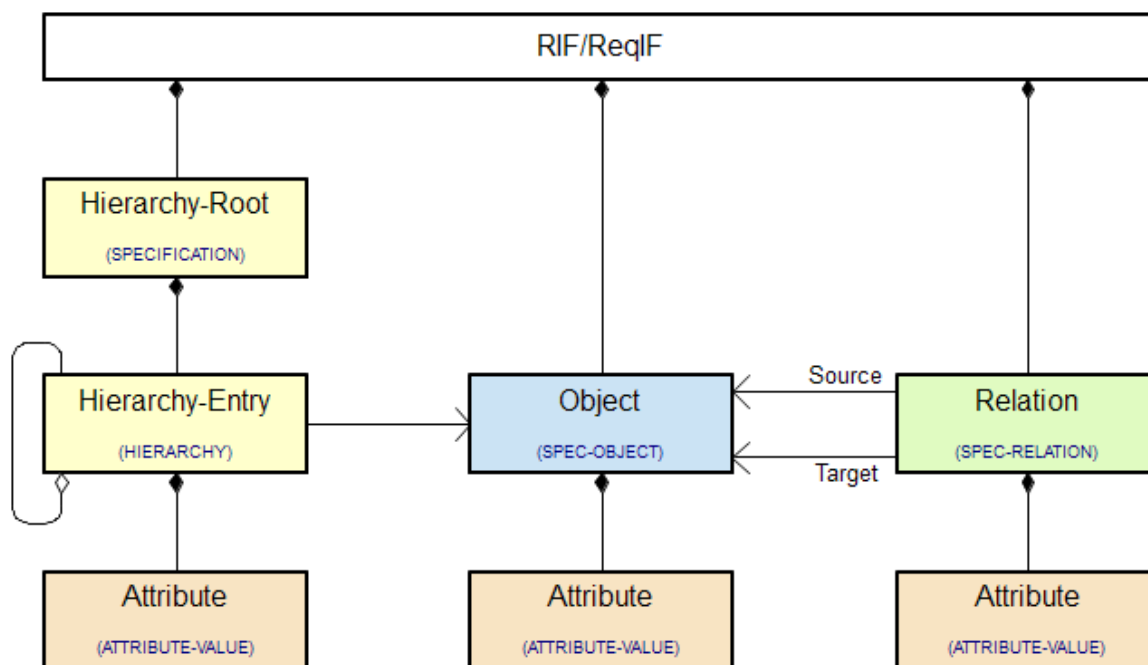


Figure 11. ReqIF metamodel

3.1.3.2 PTC Integrity Integration

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 only specific characteristic that the server is a web server composed of many different interfaces. Furthermore, the client has also some possible interactions via its API and coding it in C language.

The integration has been accomplished (Figure 12) by consuming some of these web service interfaces for RQA and RAT 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.

Finally, the integration for the RAT plugin has not been as seamless as done 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 have the possibility to 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.

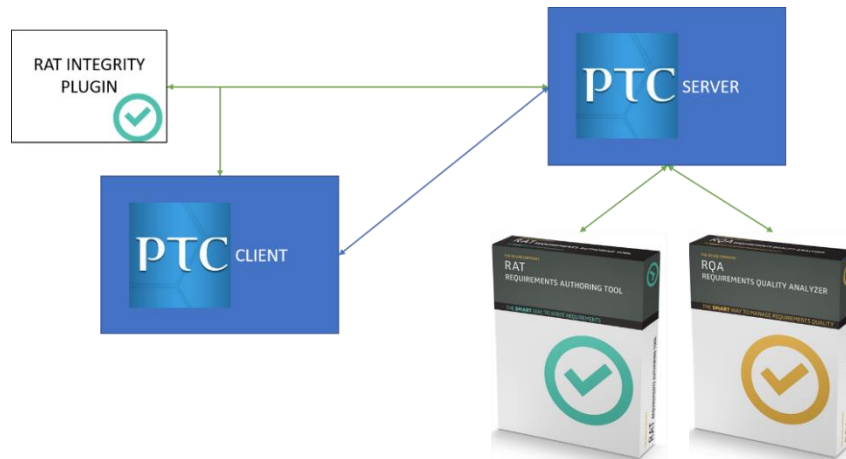


Figure 12. Integrity connector architecture

3.1.3.3 Rhapsody Integration

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 working with files stored either locally in the computer or in a network resource. They could also be under management control using any of the well-known version control management tools, such as Git, Subversion, etc.

Rhapsody allows to interoperate with the content of the project using a Java interface as well as other .NET interface, but the later one is obsolete, so it does not allow us to implement our desired functionalities.

The Java interface allows to subscribe handlers to triggers that are fired inside Rhapsody. Then by creating the suitable Java function and subscribing to the desired trigger, any functionality can be implemented.

The integration between RQS tools and Rhapsody has been done using this Java interface. The architecture (Figure 13) is composed of three different elements:

- RAT Rhapsody plugin: written in Java, subscribes to the 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 different parts:
 - XAT Resident Process: which it is in charge of using the already existing technology provided by TRC to author requirements via a COM object after receiving any trigger handler.
 - Rhapsody COM interface: it is an interface in charge of communicating the XAT Resident Process and Rhapsody. XAT Resident Process commits the changes performed in the RAT COM object back in Rhapsody via using this interface.
- The third element is the RAT COM object that allows to perform any quality assessment and enables guided authoring using patterns, and makes this functionality also available for other RMS tools plugins.

All these three elements must be deployed in the same computer.

Finally, some major integration points to be mentioned are:

- The requirement format for Rhapsody is HTML and the RQS 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.

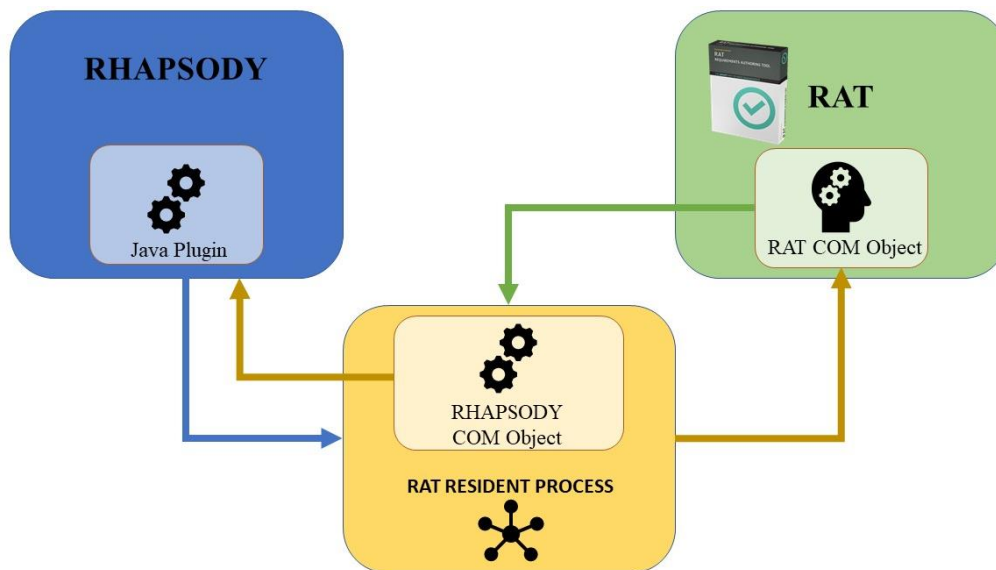


Figure 13. Rhapsody connector architecture

3.1.4 Papyrus Interoperability

Papyrus comprises different ad-hoc connectors to enable exporting and importing model elements from the tool and other modelling tools.

Interoperability through CDO repository

Papyrus provides support for object-level locking, changes to Papyrus's editing behaviour, etc., together with workbench-based server and user administration facilities, with its CDO integration. The collaborative mode integrates Eclipse's read-only and team working mechanisms to offer a locking / unlocking feature that can be extended to work with file-based remote repository (git, svn).

RSA and Rhapsody model importer

Papyrus also provides mechanisms to interact with other tools, e.g., it can import models from RSA and Rhapsody. To be able to use them, you have to load some additional Papyrus components: "RSA model importer" and "Rhapsody model importer".

There are other integrated gateways to enable model-to-model (M2M) transformations from UML/SysML models to Simulink, Autosar, East-ADL, and vice-versa, within the tool.

ReqIF support

Papyrus has a component to retrieve and import information from ReqIF specifications, Excel and CSV files. The ReqIF connector enables to be interoperable with external tools, e.g. DOORS.

Papyrus for Simulation (MOKA plugin)

Particularly, there is a Papyrus component for execution of UML models, which natively includes an execution engine complying with OMG standards fUML and PSCS. Moka is integrated with the Eclipse

debug framework to provide control, observation and animation facilities over executions. It can be easily extended to support alternative execution semantics, e.g. BPMN, Simulink, and also used for co-simulation of UML and others execution semantics.

When considering other additional components, Papyrus supports interoperability features with more tools, e.g. NuSMV and XFTA tools for safety analyses purpose, RTDS tool for testing purpose, Simulink (simulation and testing), etc.

In the context of AMASS project, we will be interested in studying the development of a connector to retrieve/exchange model elements with Prossurance tool.

3.1.5 V&V Tool Integration (*)

A possible scenario envisages the integration between the model editor(s) of the AMASS Tool Platform and tools that validate some requirements/properties and verify models against these requirements (Figure 14). In the end, the results of the V&V activities is returned and can be included as part of the V&V evidence.

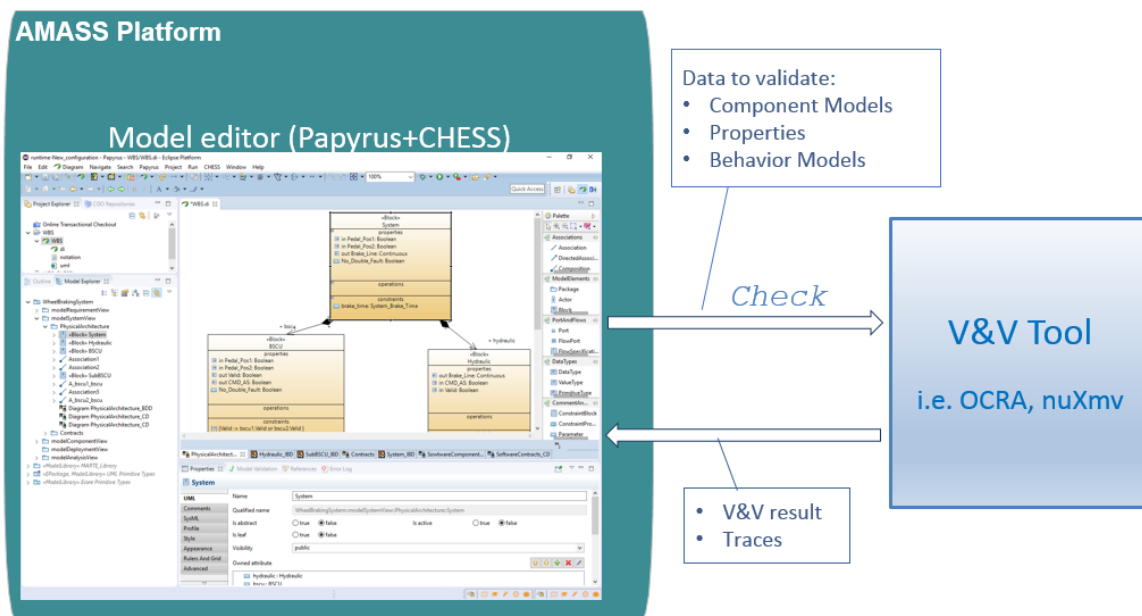


Figure 14. Communication between the AMASS Platform and the V&V tools

In order to foster the interoperability (Figure 15), we need:

- A neutral format (with respect to the tools) of the data exchanged between the platform and the tools.
- API and communication protocol for the tool interaction.

The AMASS Platform defines and uses a tool neutral format to represent the data to be verified/validated and the results. In addition, the platform defines for the V&V domain the API to perform the required V&V functionalities. The tool adapter performs the translation of data and API toward the specific tool.

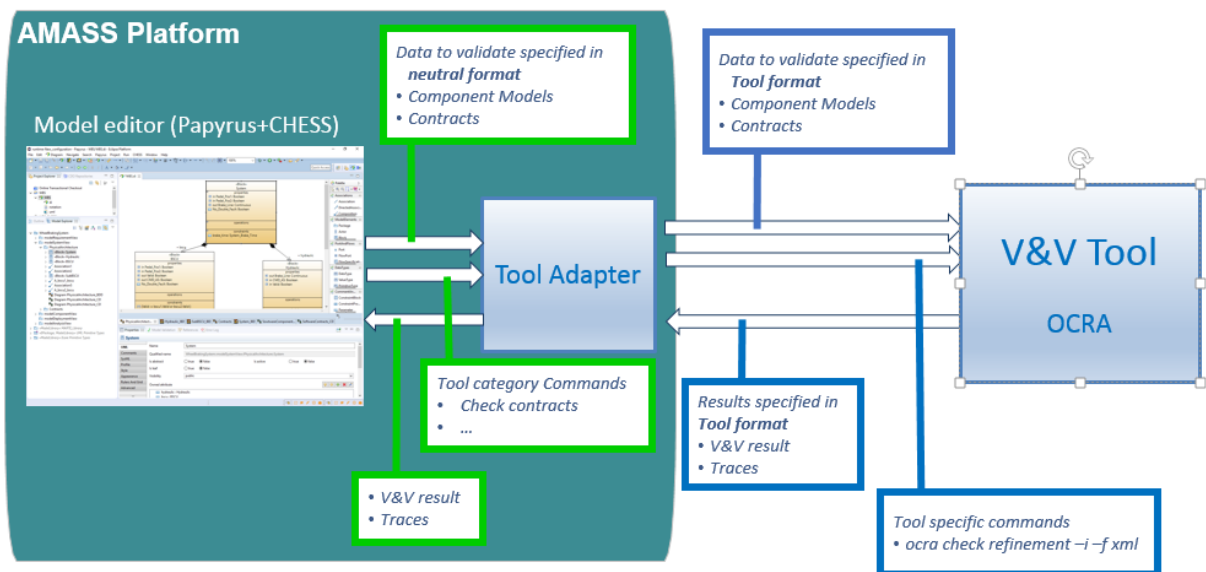


Figure 15. Interactions with the V&V tools

The tool adapter can implement different interoperability technologies (Figure 16), such as by File or a more advanced approach as the OSLC standard. In case of File, the tool adapter translates the model and the API to some tool specific files, invokes the tool and gets back / translates the result.

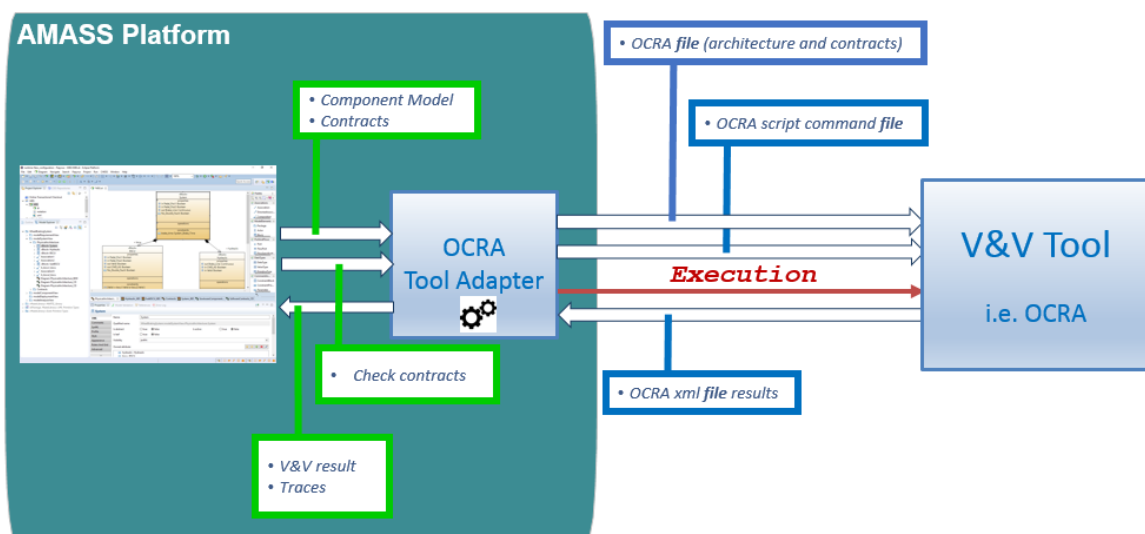


Figure 16. Service execution request to V&V tool

In the case of OSLC (Figure 17), similar to the above case, the tool adapter translates the model and the API, invokes the tool by means of the OSLC framework (for example by using the OSLC Automation Domain) and gets back the result.

With the OSLC approach, the communication between the AMASS Platform and the V&V tools can be synchronous or asynchronous. In the first case, when the request is sent, the user needs to wait for the result of the external tool. In the second case, when the request is sent, the tool adapter receives an ID that represents the request. Then the user, sending the ID to the OSLC service, can ask the state of the request without having to wait for the result.

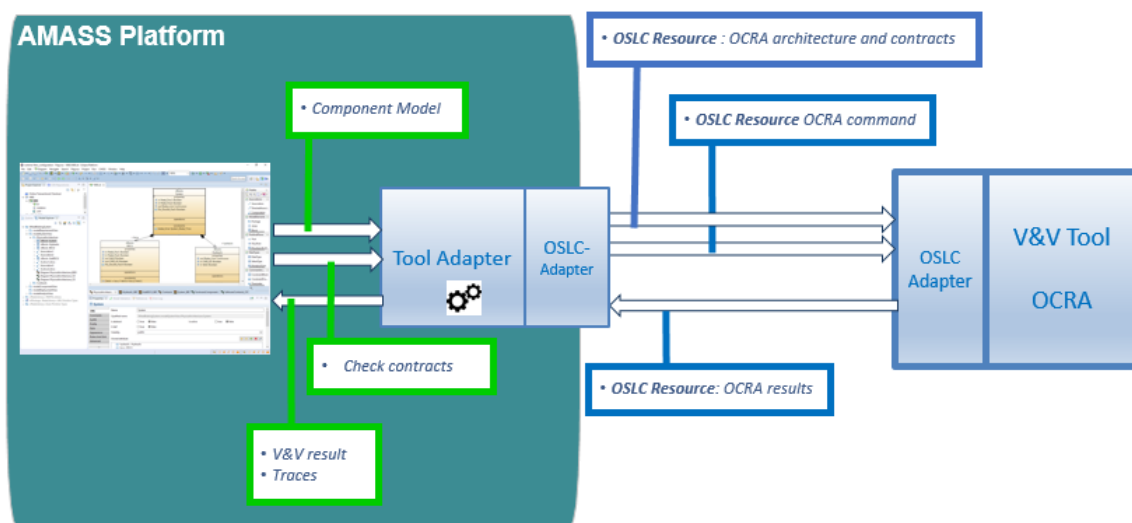


Figure 17. OSLC resources for interaction with V&V tools

Figure 18 shows the interconnection of the AMASS Platform with the V&V tools using OSLC Automation specification and multiple automation servers running in parallel that host the V&V tools. This approach allows to seamlessly integrate for example any command line V&V tool such as DIVINE model checker from Masaryk University.

V&V Manager [13] would encapsulate the following artefacts to OSLC Automation Plan:

- **Formal requirements** (LTL, inputs and outputs) (=contracts, properties) for both requirement semantic analysis and formal verification. Linear Temporal Logic is extended with arithmetical equations instead of Boolean properties and with Metric Temporal Logic bounded operators.
- **Behavioural model** (Simulink, C, C++) for formal verification is used by the tools DIVINE, NuSMV, and nuXmv.
- **System architecture** (SysML) (=component model) for component verification and safety analysis is used by the tool OCRA.

Verification servers will provide V&V automation and service for AMASS Platform.

As an example of the interaction between the AMASS Tool Platform and the Automation servers, we provide an overview of the corresponding communication during formal verification. **Formal verification** in this case verifies if the software design or architecture comply with the high-level requirements or not (see the objectives A-4.1 and A-4.8 in RTCA DO178-C):

- **V&V Manager** sends **formal contracts** and **behavioural model** contained in or linked from **OSLC Automation Plan and Request** to the **Automation Server**.
- **V&V Manager** periodically sends **OSLC Performance Monitoring Requests** to **Automation Server**.
- **Automation Server** periodically sends **OSLC Performance Monitoring Responses** to **V&V Manager**.
- **Automation Server** sends **OSLC Verification Result** to **V&V Manager**.

Similarly, **Requirement Semantic Analysis** identifies problems with high-level requirements, e.g. their possible inconsistency, redundancy, non-realizability (see objectives A-3.2 and A-3.5 in RTCA DO178-C). The communication between the V&V Manager and the Automation Server differs from the above algorithm for the verification just in the absence of the behavioural model in the case of requirement semantic analysis. The only inputs for the analysis are the formal contracts. For realizability and completeness analysis, also input and output ports and their data types are provided.

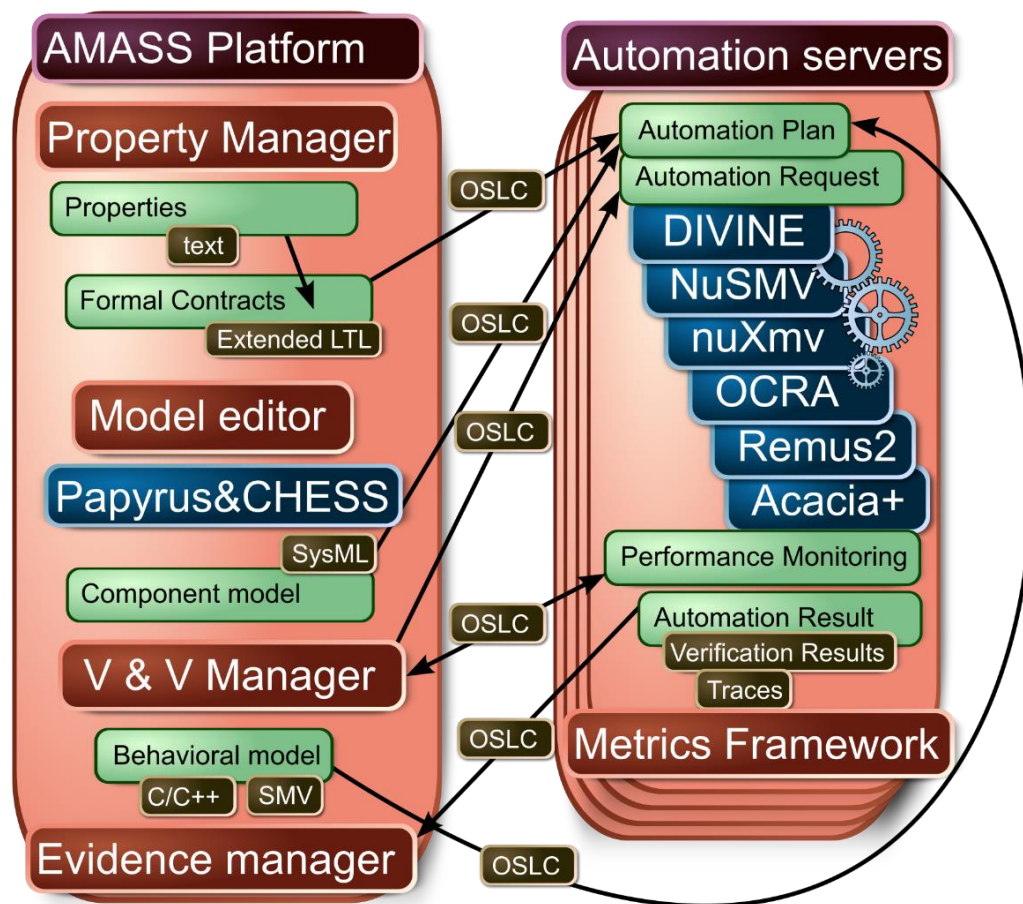


Figure 18. Communication of AMASS Platform with V&V Tools using OSLC Automation

The use case diagram in Figure 19 depicts the main functionalities of the V&V Manager plugin.

From the user's point of view the AMASS Platform and the V&V Manager provide these related services:

- Manual selection of those contracts that should be checked.
- Selection of the appropriate verification technologies that should be executed.
- Selection of the verification servers that provide the best performance for given verification task.
- Sending the selected inputs to the chosen verification tools using OSLC Automation.
- Presenting the received verification results.
- Generating assurance-related information.

The translation from contracts or requirements to LTL should be performed by Property Manager. Therefore, the V&V Manager only need to get the LTL attribute from the contract.

Figure 20 shows the high-level static structure of the V&V Manager and it also depicts which classes realize which of the use cases mentioned above.

The *MessageWriter* class retrieves all the information necessary to create the requests for verification, it combines the information and encodes it into the format expected by the Verification Server or by the verification/validation tools.

The *Sender* takes care for submitting the composed message to the Verification Server.

The *Receiver* monitors the status of verification and, most importantly, it collects the results of verification and validation.

The *ResultBrowser* presents the verification/validation results to the user. An example of the V&V result is shown in the Figure 21.

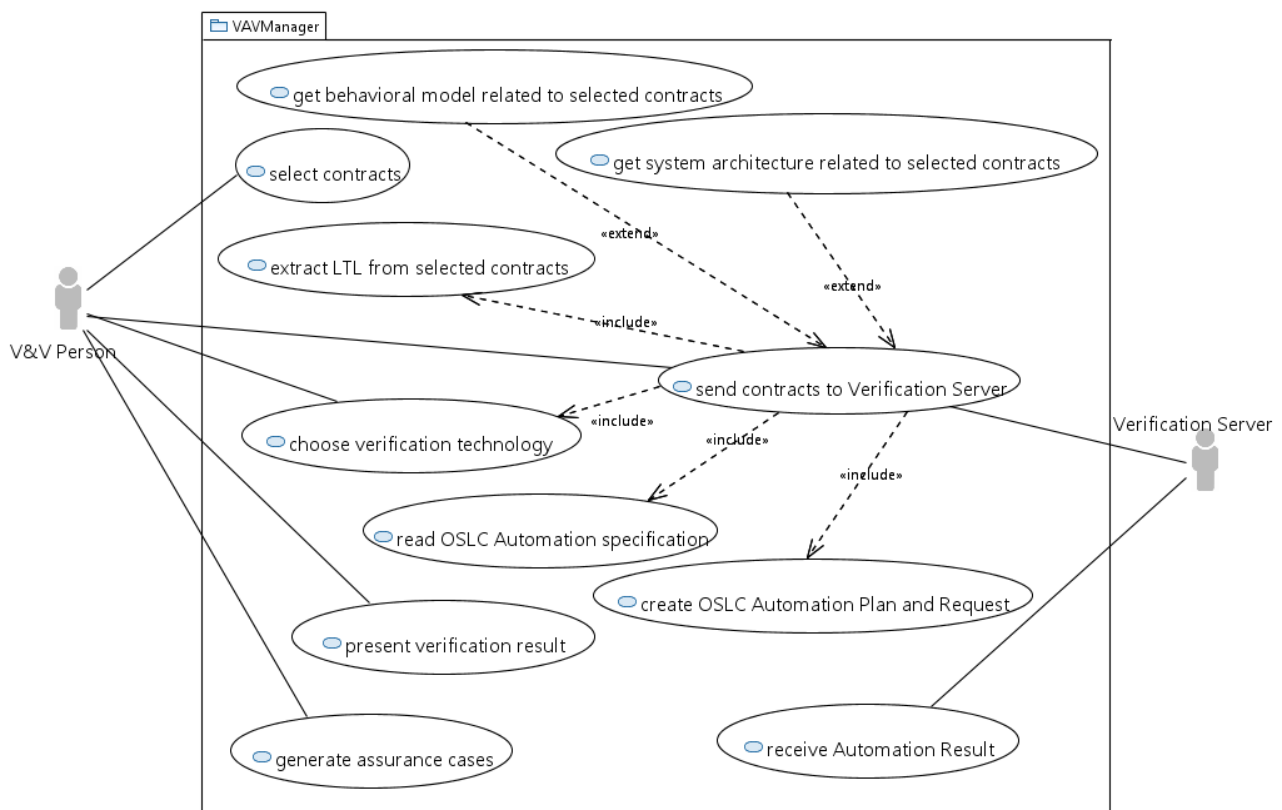


Figure 19. Use case diagram for the V&V Manager plugin

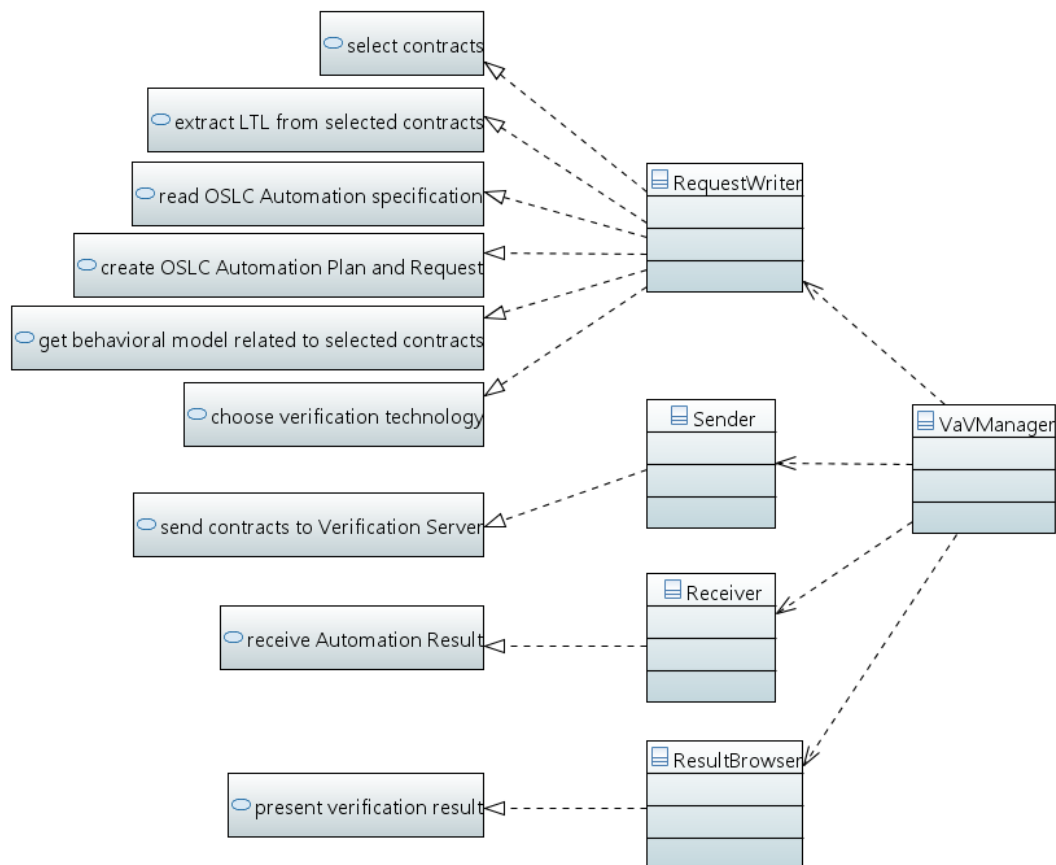
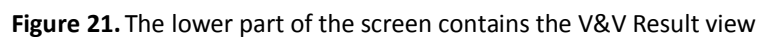


Figure 20. V&V Manager top-level static structure and correspondence to the realized use cases



```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dcterms="http://purl.org/dc/terms/"
  xmlns:oslc="http://open-services.net/ns/core#"
  xmlns:oslc_auto="http://open-services.net/ns/auto#"
  xmlns:oslc_rm="http://open-services.net/ns/rm/">
  <oslc_auto:AutomationPlan
rdf:about="http://158.138.138.152/verificationPlanAndRequest135843703.xml">
  <dcterms:title>Verification plan for the SafetyInterlock</dcterms:title>
  <dcterms:created>4/22/2017 06:21:42</dcterms:created>
  <dcterms:identifier>"135843703"</dcterms:identifier>
  <dcterms:creator rdf:resource="Honeywell International" />
    <oslc_rm:VerifSMV rdf:about="http://158.138.138.152/SafetyInterlock.smv"
oslc:shortTitle="SafetyInterlock" dcterms:identifier="1">
  <dcterms:title>SMV model</dcterms:title>
  <oslc_rm:validatedBy dcterms:identifier="1" dcterms:description="G ( G<math>\wedge</math> P -&gt;
F=3 R )" />
    <dcterms:description />
  </oslc_rm:VerifSMV>
```

```

        <oslc_rm:VerifModel      rdf:about="http://158.138.138.152/SafetyInterlock.cpp"
oslc:shortTitle="SafetyInterlock" dcterms:identifier="1">
    <dcterms:title>CESMI model</dcterms:title>
</oslc_rm:VerifModel>
    <oslc_rm:VerifModelSup      rdf:about="http://158.138.138.152/SafetyInterlock.inc"
oslc:shortTitle="SafetyInterlock" dcterms:identifier="1">
    <dcterms:title>Propositions</dcterms:title>
    <dcterms:parameters rdf:string="-r" />
</oslc_rm:VerifModelSup>
</oslc_auto:AutomationPlan>
<oslc_auto:AutomationRequest>
    <dcterms:title>Verification of the SafetyInterlock</dcterms:title>
    <dcterms:identifier>"135843703"</dcterms:identifier>
    <oslc_auto:state rdf:resource="http://open-services.net/ns/auto#new" />
    <oslc_auto:executesAutomationPlan
http://158.138.138.152/verificationPlanAndRequest135843703.xml" />
    </oslc_auto:AutomationRequest>
</rdf:RDF>

```

An abbreviated example of the OSLC Automation Result is here:

```

<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:oslc_auto="http://open-services.net/ns/auto/"
  xmlns:dcterms="http://purl.org/dc/terms/"
  xmlns:oslc="http://open-services.net/ns/core/"
  xmlns:oslc_qm="http://open-services.net/ns/qm#">
    <oslc_auto:AutomationResult
rdf:about="http://158.138.138.152/verificationResult135852478.xml">
    <dcterms:title>Verification result for the SafetyInterlock</dcterms:title>
    <oslc_qm:status>complete</oslc_qm:status>
    <dcterms:identifier>21</dcterms:identifier>
    <oslc_auto:reportsOnAutomationPlan
rdf:resource="http://158.138.138.152/verificationPlanAndRequest135843703.xml" />
    <dcterms:description>
    ...
    </dcterms:description>
    <dcterms:creator rdf:resource="Honeywell International"/>
    <dcterms:created>05/05/2017 17:48:40</dcterms:created>
    </oslc_auto:AutomationResult>
</rdf:RDF>

```

Another example of the interaction between the AMASS Platform and the Automation Servers except for the verification would be the **sanity checking** of a set of requirements. In this case the **OSLC Automation Plan and Request** contains the **formal contracts** related to a component, but does not contain any behavioural model. The employed tools for semantic requirements analysis (namely the open source tools Looney, Acacia, and ReMUS) check the sanity of the examined set of contracts, i.e. whether:

- the whole set of contracts is satisfiable (consistent),
- there is no redundancy of the requirements,
- there are no vacuously satisfiable requirements,
- the requirements are realizable.

In the case that the given set of contracts is inconsistent, the requirements semantic analysis tools also identify all the sources of the inconsistency, called minimal unsatisfiable subsets, shortly MUSes (for more detailed description of the concept of MUSes, see deliverable D3.3 [12]). The minimal unsatisfiable subsets can be subsequently used by the user to refine the inconsistent set of requirements/contracts.

In addition, an integration of Honeywell internal V&V tools with the AMASS Tool Platform would help us to:

- Allow requirement authoring based on Honeywell internal requirement standards.
- Integrate requirement semantic analysis.

- Integrate requirement-based test generation.
- Translate Simulink to models input language of model checkers.
- Compare different V&V tools for the same base line.
- Evaluate baseline development process with proposed AMASS development process.

3.1.6 Integration with Safety and Security Analysis Tools (*)

A possible integration scenario between AMASS Platform/CHESS tool and safety and security analysis tools for safety and security co-analysis is presented in Figure 22. Safety Architect tool [96] is used for safety analysis and Cyber Architect tool [35] is used for security analysis.

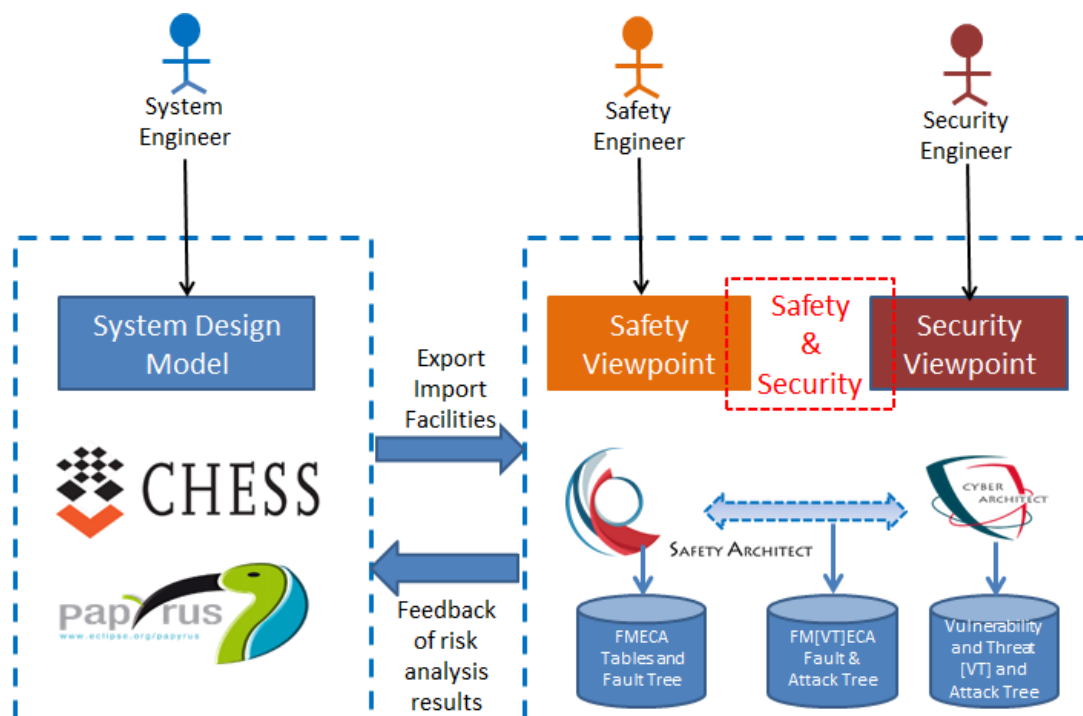


Figure 22. Interaction usage scenario for safety and security analysis

Firstly, the seamless interoperability between tools allows decoupling the system architecture model from safety or security views, so each engineer (system, safety or security engineer) can solely focus on his concerns, with dedicated tools and terminology. For example, safety engineers could use the interface between CHES tool and Safety Architect tool to generate fault trees or FMECA tables from CHES model.

Secondly, the development of a specific-concern viewpoint in a tool dedicated to other concerns allows co-engineering between these concerns. For instance, with the security viewpoint in the Safety Architect tool, safety engineers can use the results of security analysis realized in Cyber Architect (e.g., attack tree) to generate a merged fault tree and attack tree for assessing the influence of security-relevant events on safety top event.

A possible usage scenario is composed by the following steps:

- **Step 1:** System engineers can design its system architecture model with CHES tool. An example of CHES model for the sample Wheel Brake System (WBS) is shown in Figure 23.

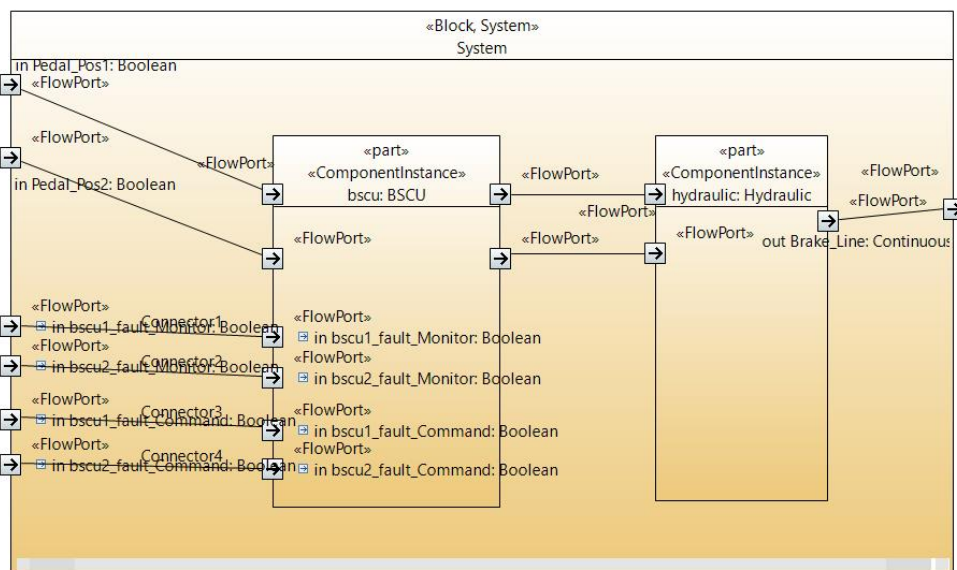


Figure 23. Example of CHES model - Wheel Brake System (WBS) Model

- **Step 2:** Safety Engineers can import system model from AMASS platform – CHES tool to Safety Architect tool for safety analysis thanks to the connector between CHES and Safety Architect, as illustrated in Figure 24 and Figure 25.

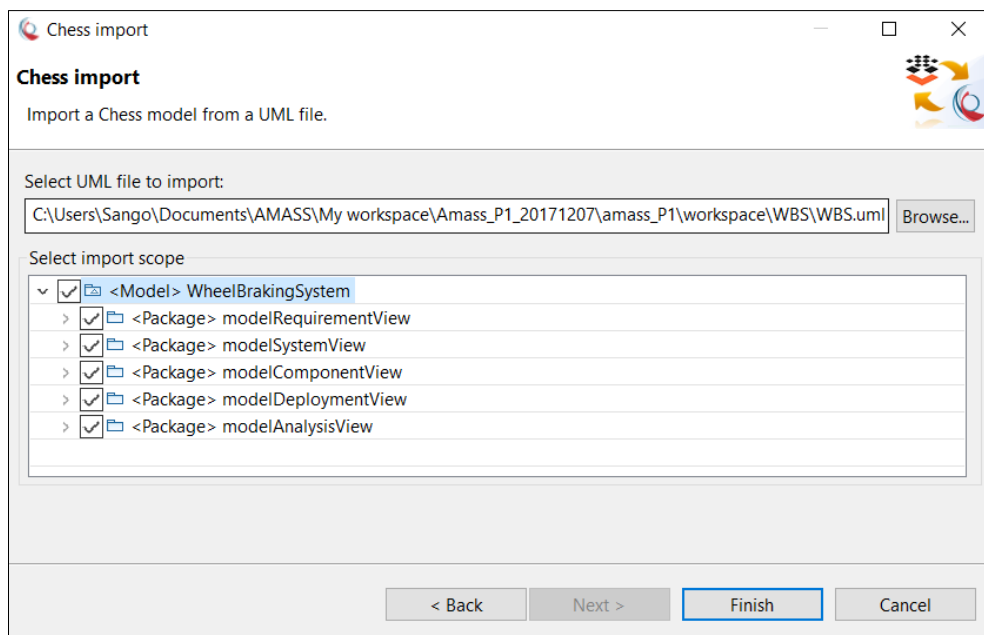


Figure 24. Import from CHES tool to Safety Architect tool

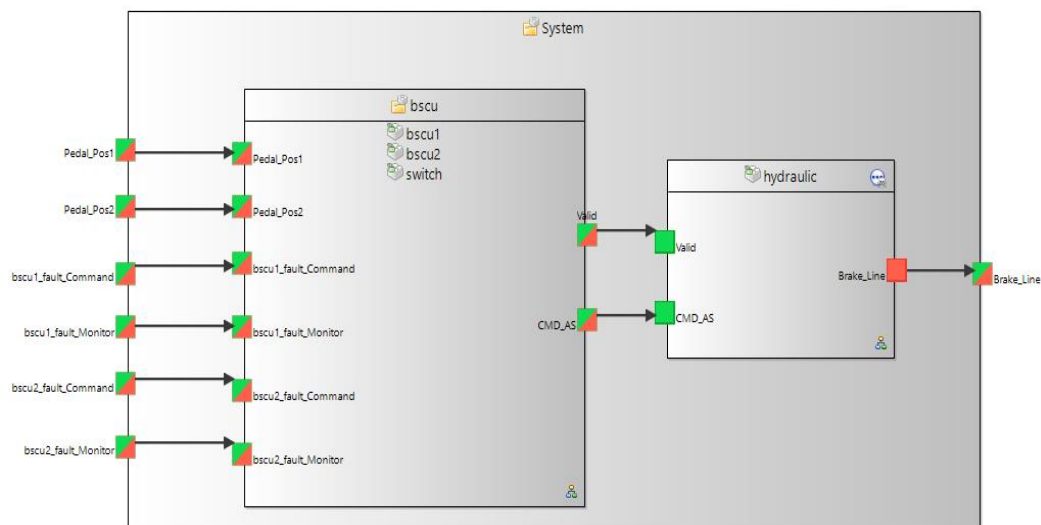


Figure 25. Safety Architect WBS model from CHES WBS model

- **Step 3:** Security engineers (in parallel to safety analysis or based on the process defined in other AMASS platform tools, such as EPF or external tools, such as WEFAC) can perform its security analysis according to the knowledge bases of security analysis method (e.g., EBIOS method). The Cyber Architect tool integrates the EBIOS or SISSP security knowledge bases, as illustrated in Figure 26.

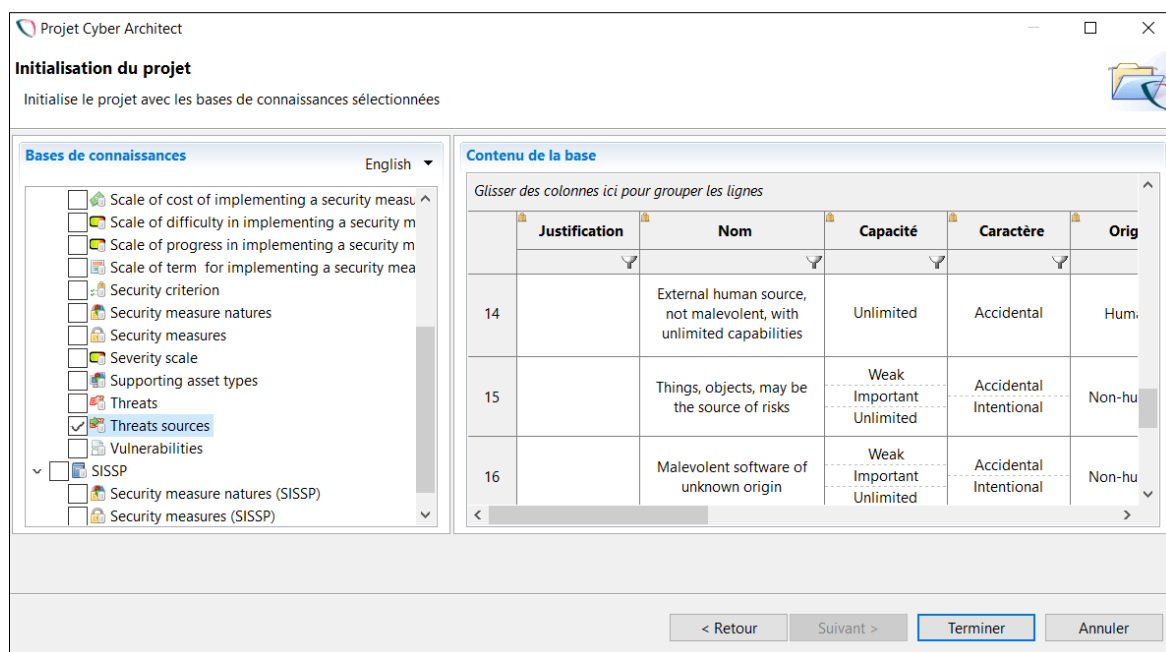


Figure 26. Cyber Architect project initialised with EBIOS knowledge bases

- **Step 4:** Assurance Engineers (safety and security expert), can exploit the bridge between Safety Architect and Cyber Architect to perform it co-analysis or assessment. For example, to analyse the impact of security into safety, assurance engineers can import threats or vulnerabilities from Cyber Architect into Safety Architect for safety and security co-analysis. The interface between Safety Architect tool and Cyber Architect tool is presented in Figure 27.

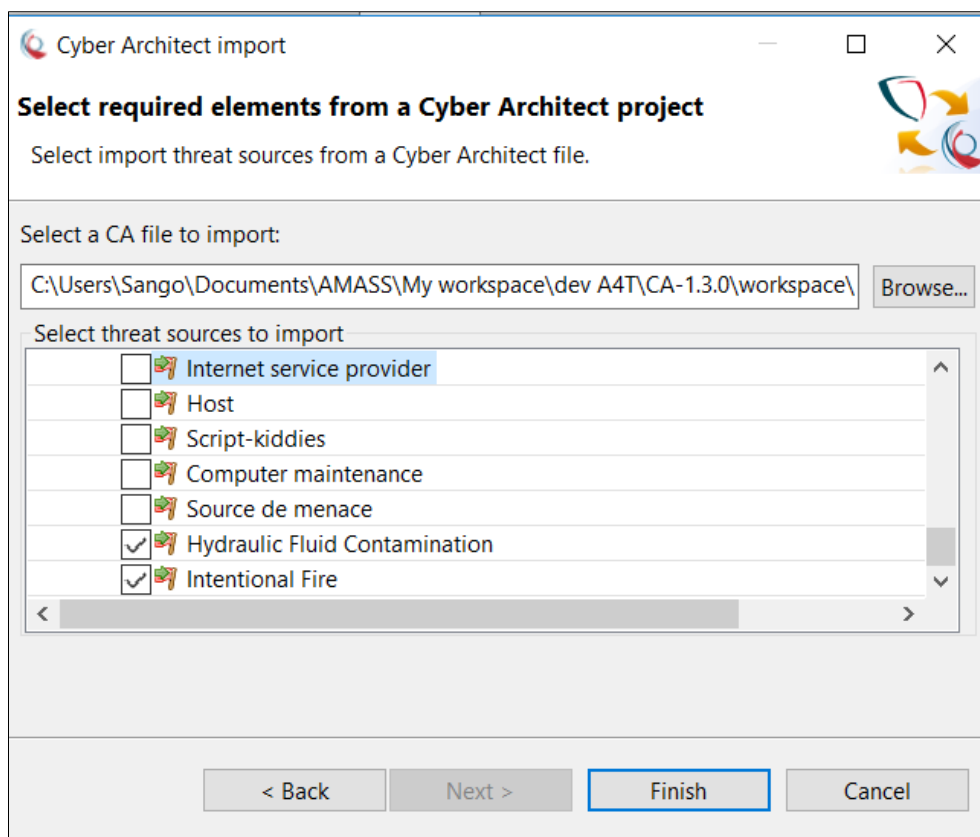


Figure 27. An interface between Safety Architect and Cyber Architect

- **Step 5:** Assurance Engineers can activate the Security Viewpoint in Safety Architect tool for Safety and Security Co-analysis. The activation of Safety & Security viewpoint in Safety Architect allows the annotation of input and output ports of system components with Security Thread Modes (e.g., hydraulic fluid contamination or intentional fire) imported in previous step. The co-analysis is realized thanks to these threat modes, failure modes (internal failure, erroneous) and logical gates, as illustrated in Figure 28.

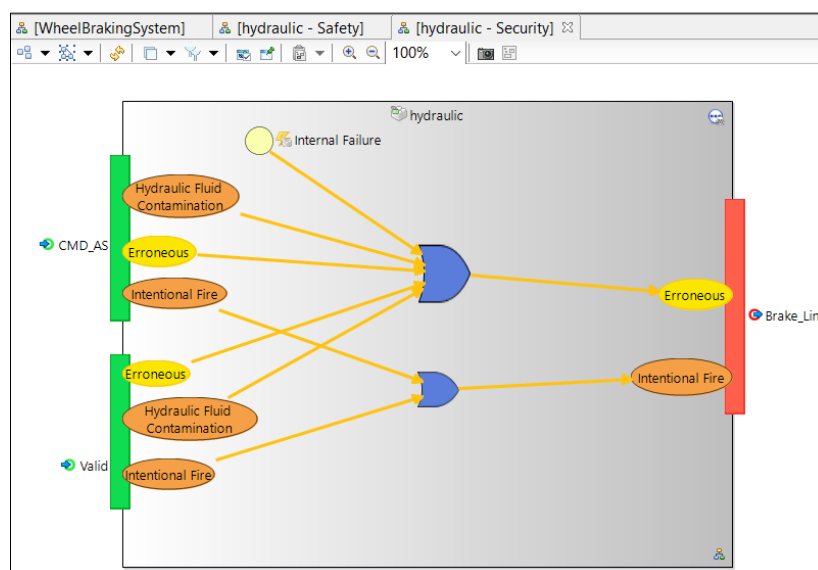
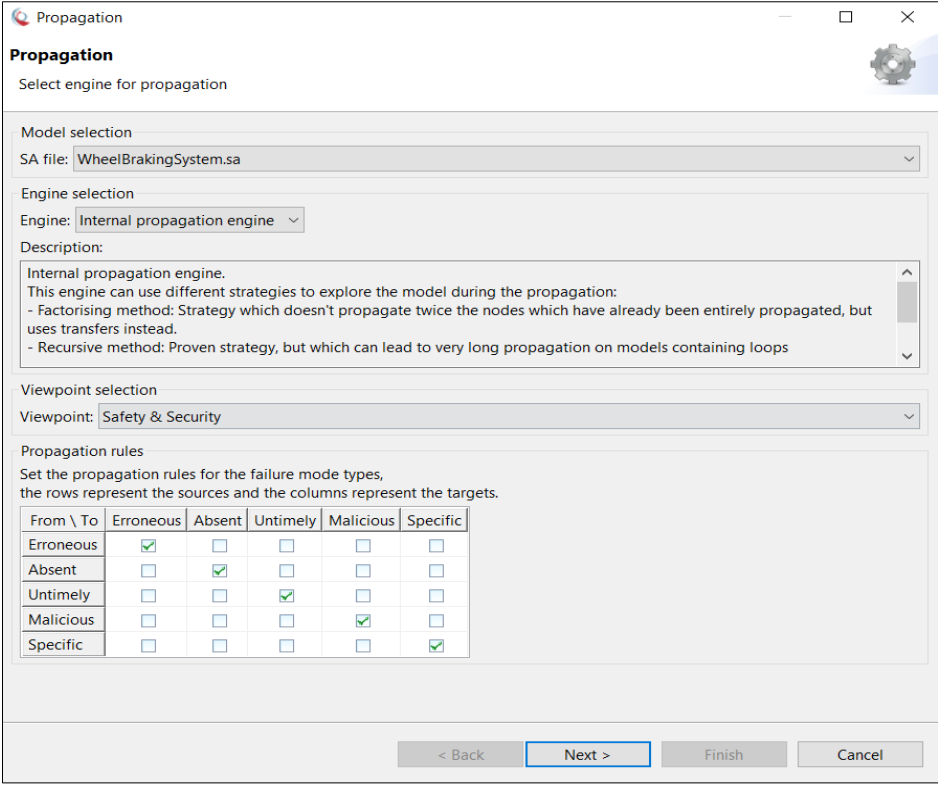


Figure 28. Safety & Security viewpoint in Safety Architect

- **Step 5:** Assurance Engineers can generate the Safety & Security artefacts (e.g., Faults and Attacks Propagation Tree) thanks to the previous Safety & Security co-analysis and the Safety Architect propagation engine with the “Safety & Security” viewpoint selection shown in Figure 29.



Propagation

Select engine for propagation

Model selection
SA file: WheelBrakingSystem.sa

Engine selection
Engine: Internal propagation engine

Description:
Internal propagation engine.
This engine can use different strategies to explore the model during the propagation:
- Factorising method: Strategy which doesn't propagate twice the nodes which have already been entirely propagated, but uses transfers instead.
- Recursive method: Proven strategy, but which can lead to very long propagation on models containing loops

Viewpoint selection
Viewpoint: Safety & Security

Propagation rules
Set the propagation rules for the failure mode types, the rows represent the sources and the columns represent the targets.

From \ To	Erroneous	Absent	Untimely	Malicious	Specific
Erroneous	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Absent	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Untimely	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Malicious	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Specific	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

< Back Next > Finish Cancel

Figure 29. Safety & Security viewpoint selection in Safety Architect

The failure and threats propagation tree generated in Safety Architect can be exported in OpenPSA format (.xml files) and can be read by OpenPSA based tools, such as Arbore Analyste, as illustrated in Figure 30.

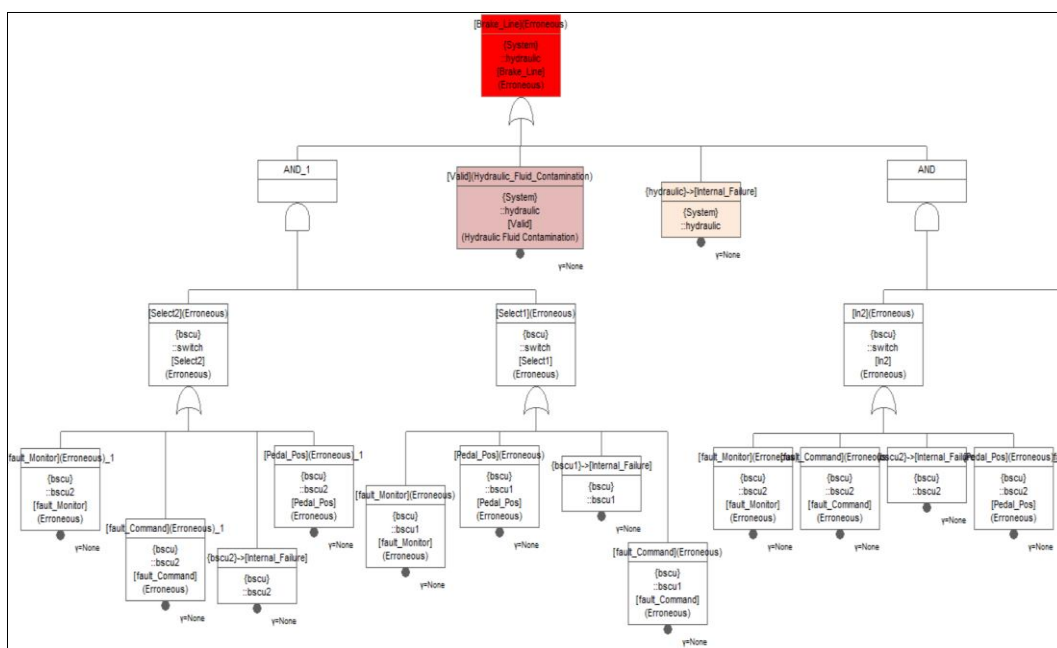


Figure 30. Faults and Attacks Propagation Tree in Safety Architect

- **Step 6:** The last step is the visualization of propagation tree (the .xml files) in the AMASS Platform – CHES tools. To do that go the AMASS Platform (the eclipse bundle) then in “CHES – Fault Tree Viewer – view fault tree diagram from .xml file”, as illustrated in Figure 31.

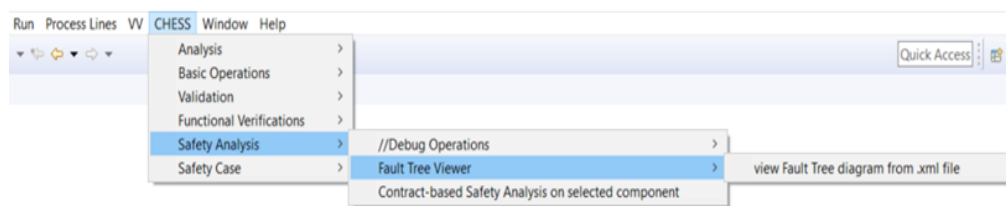


Figure 31. Fault Tree Viewer in CHES

One of the benefits of this integration of system modelling tool (CHES) with safety/security analysis tools (Safety Architect and Cyber Architect) is the identification of safety, security and safety/security items. These items can be highlighted in the system architecture model as presented in Figure 32.

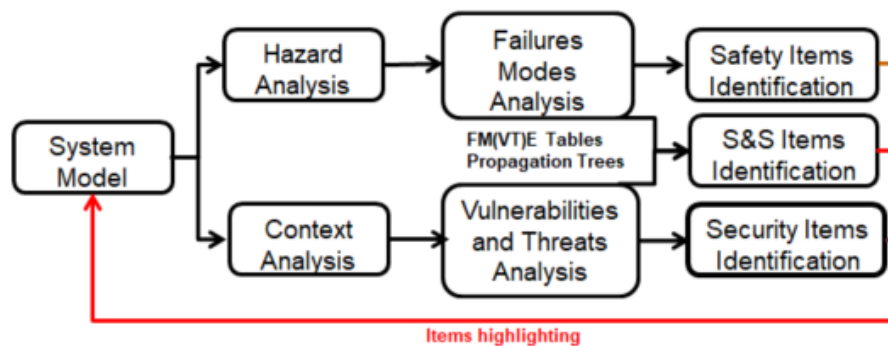


Figure 32. A Process for system safety and security analyses

3.1.7 Integration with the Sabotage Simulation-Based Fault Injection Tool (*)

As described in the deliverable D3.3 [12] and in recent publications [70], model-based design combined with a simulation-based fault injection technique and a virtual vehicle poses as a promising solution for an early safety assessment of automotive systems. Initially, a fault forecasting process takes place. The design with no safety consideration is stimulated with a set of fault injection simulations. By doing so, different safety strategies can be evaluated during early development phases estimating the relationship of an individual failure to the degree of misbehaviour on vehicle level. After including the proper safety mechanisms, a second set of fault injection experiments is performed in order to early validate the safety concept.

A possible integration scenario between CHES/SAVONA, model-based safety analysis tools, AMT 2.0 monitoring tool and Sabotage is presented in Figure 33. The main objective is to integrate the contract-based approach with the inclusion of fault injection blocks and monitors. For further information, please refer to the D3.3 [12], D3.5 [13] and D3.7 [14] deliverables.

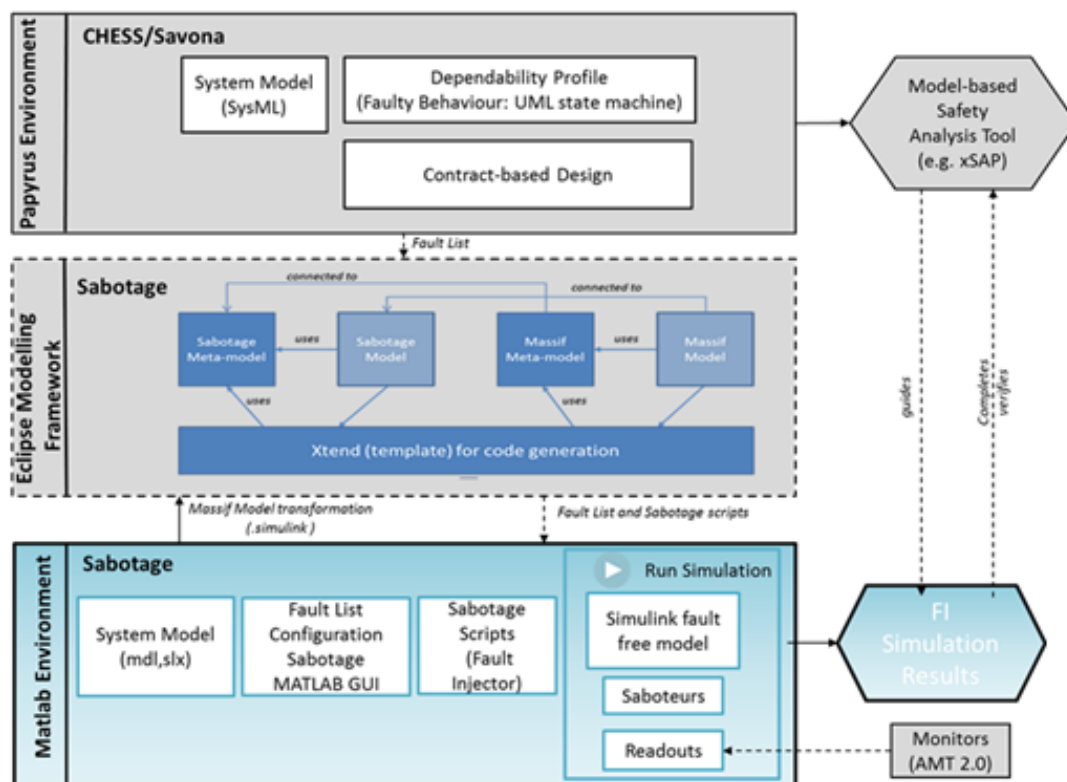


Figure 33. Sabotage integration with the CHESS/SAVONA, model-based safety analysis and Monitors Tools [14]

3.1.8 Integration in the Farkle Tool (*)

As explained in D4.3 [15], Extended Farkle is a tool developed by Alten for model-based V&V. It is a combination of open source (Ruby and Python) and Alten software. Originally, Farkle was a test execution tool. By reading test scripts in Python code and a Ruby-parsing tool, that generates an interface software module for communication to the test object. Later the tool received an extension with model-based test features, the Extended Farkle. Extended Farkle provides test generation and test execution of a SUT. However, Extended Farkle requires a model of the SUT as an input in order to work.

To provide this model, an in-house tool has been developed based on Machine Learning. Earlier another machine learning system was used, LBTest. This tool shows that the concept holds but the architecture of the software component was not acceptable. The new tool we call MLT (from machine learning for testing) supports the generation of the models needed for Extended Farkle. MLT uses machine learning to create tests that Extended Farkle executes, and then based on the results of these tests, MLT can improve the model such that after some time the model will be an accurate representation of the SUT. MLT uses NuSMV and a model-checking component, which generates the aforementioned tests. NuSMV needs LTL requirements in order to create test cases. Extended Farkle executes these test cases and the results are provided to MLT that then uses them to improve the model.

Another component in the tool chain is a model transformation tool called NuSMV2UML. This tool takes the models generated by MLT as an input and generate UML models.

All these tools communicate using OSLC (see Figure 34), thus they are implemented independently from one another giving a loose coupling allowing us to replace components as long as they can communicate over OSLC.

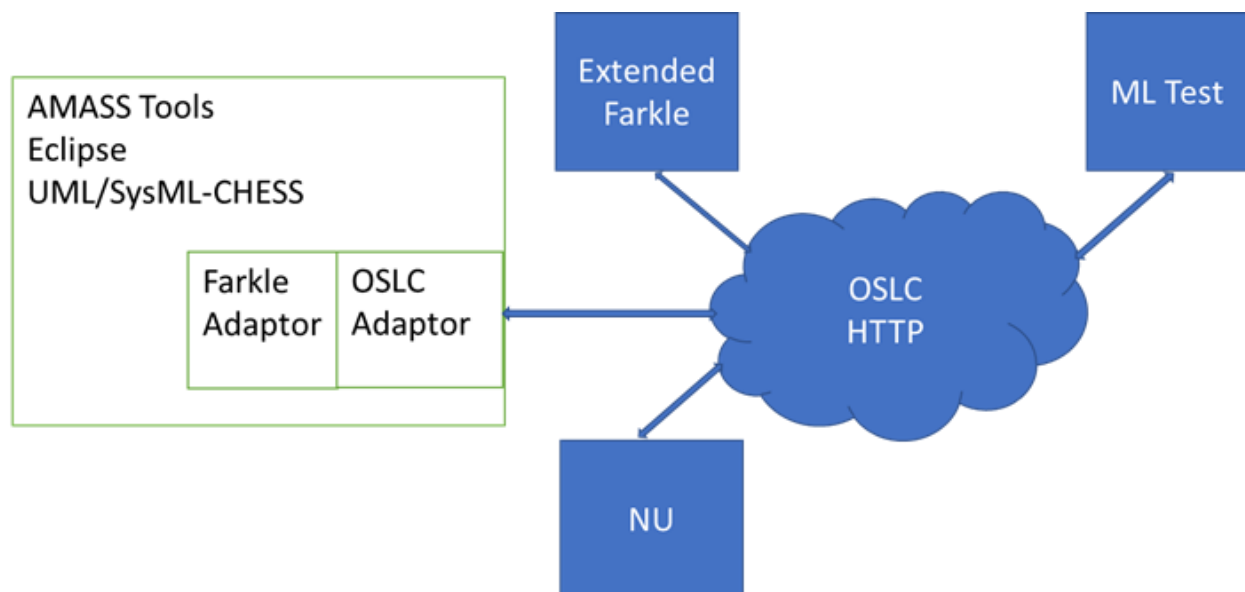


Figure 34. Overview of interwork between AMASS tools and Machine Learning-based testing tools

3.1.9 Generic REST-API Adapter Concept for Seamless Interoperability (*)

Today's development processes include many different tools. Due to a lack of compatibility between these tools, a simple exchange of data is usually not given. Therefore, a number of adapters have to be implemented. A big advantage is that more and more tools are provided as a web service. Web services, in particular data management services, often provide a REST interface, so clients are able to easily create, read, update or delete data according to the CRUD principle.

Project management services like JIRA, Redmine or HP ALM provide also a REST interface to read and modify data. Because of the similar functionality, the three services can be a basis for an implementation of an abstract and generic REST adapter. This adapter can be extended and configured to function for another specific REST API.

Figure 35 shows the concept of a generic REST adapter and function as an interface between REST interfaces and an example for a shared data model interface.

- **RestCommunicator:** The RestCommunicator class communicates with a given REST interface. The returned JSON or XML is being processed and provided to the RestDatabase, RestTable and RestTableEntry classes
- **RestDatabase:** The RestDatabase class represents the base URI of a given REST API. Since the base URI cannot be addressed in the sense of a resource, the RestDatabase is just a container for all the available resources, which are represented by RestTables.
- **RestTable:** The RestTable class represents a specific resource of a REST interface. E.g. in the context of JIRA a list of all available projects will be returned using the URI: <http://localhost:8080/rest/api/2/project>. All resources returning a list of objects can be represented by a RestTable instance.
- **RestTableEntry:** The RestTableEntry class represents a specific item that can be accessed by a specific id or key. For example in the context of JIRA a project with the id {id} can be returned using the URI: <http://localhost:8080/rest/api/2/project/{id}>. The attributes function as columns for the RestTable.

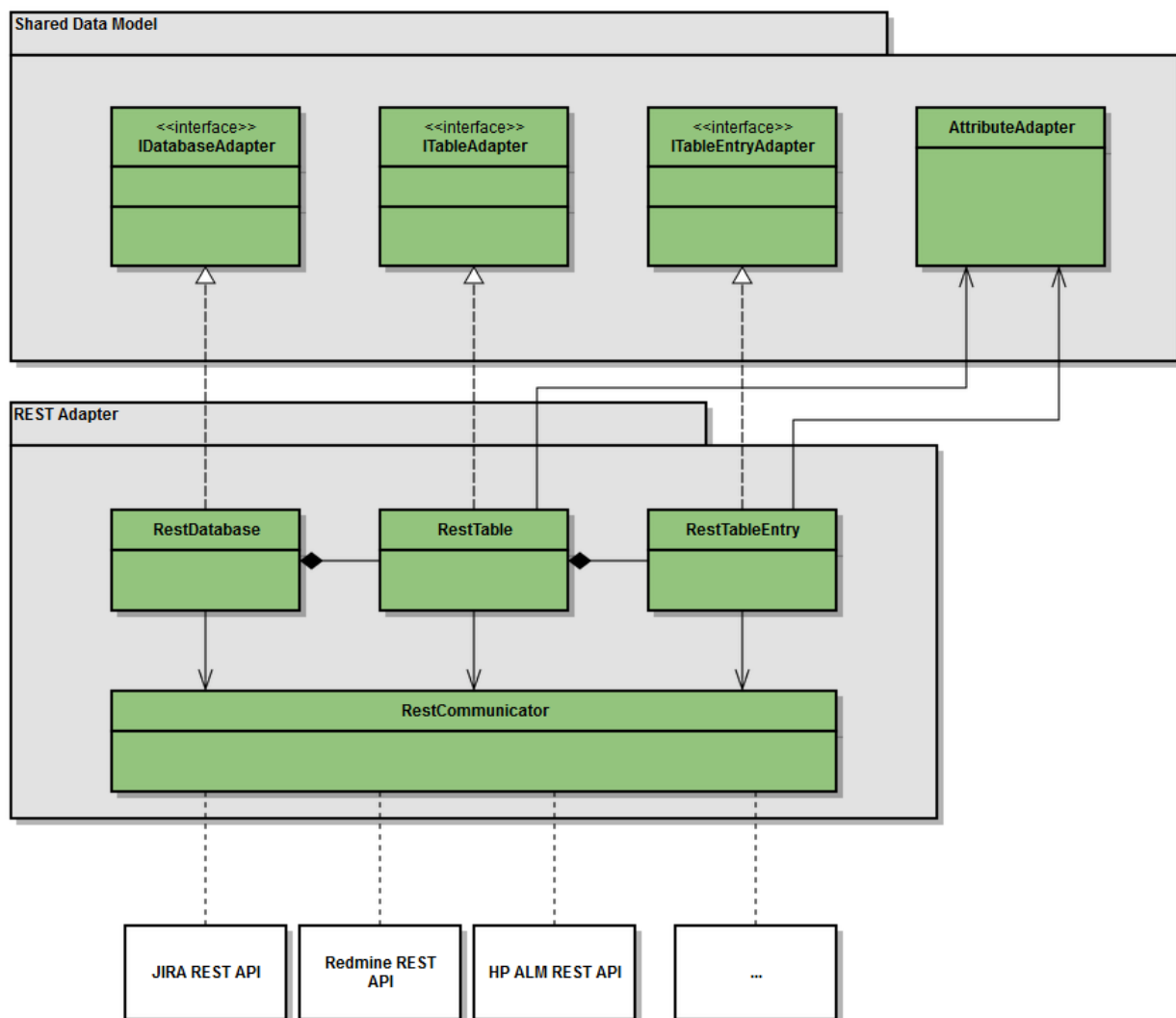


Figure 35. Generic REST adapter concept

3.1.10 Collaborative Real-Time Model Editing

In contrast to many source control systems, which provide checkout and commit cycles, real-time collaboration allows multiple users on a network to edit the same set of data in real-time (Figure 36).

In Figure 36, a client sends its changes to the server, which propagates the change to all other clients. There is usually some kind of database behind the server to store data. Additionally, many implementations offer a chat window and highlight changes done by other users inside the editor window.

Communication over the network is not instantaneous. That introduces the risk of race conditions, which may result in corrupt data or data being inconsistent across clients. To prevent these issues, some implementations like CDO use optimistic and pessimistic locking mechanisms. Optimistic locking means that the server acquires and releases locks while executing a commit operation. That technique provides a good degree of concurrency but it bears the risk of committing conflicts. The risk increases with the number of users and on networks with high latency. Pessimistic locking creates locks on the model parts before editing that model parts. This is easily possible in tree-like data structures but more difficult in network structures. It also completely avoids the issues of true concurrent editing. Therefore, pessimistic locking is usually not considered as real-time collaboration.

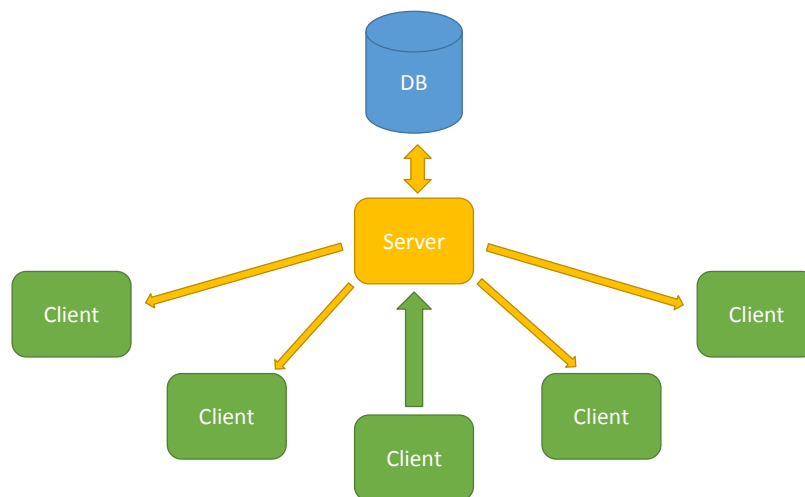


Figure 36. Multiple clients collaborating on a single document

An alternative approach is operational transformation (OT). Instead of serializing the order of editing operations on the server, OT attempts make the order of incoming editing operations interchangeable.

Figure 37 shows a possible path of merging editing operations to reach a final state. The editing operation is a very short description of what a user changed in a document. Each client sends its editing operations to the server which applies them to its own local data model, then sends it to the other clients. As the version on the server may have been changed by other clients, the new incoming editing operation has to be transformed so it does not just override changes from other clients.

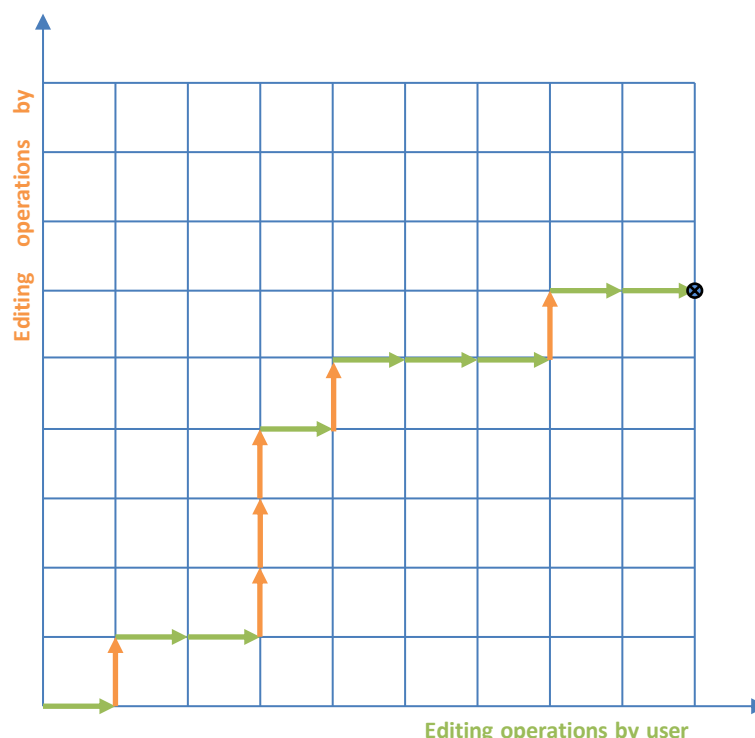


Figure 37. Concurrent editing operations by two users

When the server sends the new editing operation to the other clients, they do the same transformation and merging step on their local data but they do not send their changes back to the server.

With OT, the order in which the operations arrive does **not** matter. Once the server has merged the operations and propagated them to the clients, each client will have the same data. This is called eventual consistency. To guarantee eventual consistency, OT algorithms need to prove the convergence of data.

Besides consistency, it is desirable that the OT algorithm preserves causality. The transformations need to take into account that some operation cannot be reordered, such as an item has to be created before it is modified. If operations are not causally dependent, they are regarded as concurrent.

Lastly, intention preservation of the editing operation is a main feature of OT algorithms. It denotes the concept that the execution effect (set of pre-conditions, the execution itself and post-conditions) is the same for different editing operation orderings.

The first real-time collaboration systems with OT only supported linear data models such as text with just insert and delete operations [44]. Since then, more operations have been introduced to support more complex data models [102] or application specific operations.

3.1.11 Seamless Tracing

During the safety engineering lifecycle, many different tools are typically being used. That has the benefit of tools having the right size for a set of specialized tasks, but there is an implicit cost in that only the tool that created the data can fully interpret it. Together with the broad range of incompatible tool integration mechanisms employed by tool vendors, this may lead to data silos. As a result, data does not flow freely from one tool to another, hindering agile processes, and also reducing data visibility throughout the safety project.

Traceability, i.e. the relationships between artefacts during the safety engineering lifecycle, is prescribed by many safety standards such as ISO 26262. In a diverse tool landscape and in distributed teams, safety engineers face the challenge of having to create and manage traces within tools but also across different tools. There are several use cases revolving around tracing such as change impact analysis, coverage analysis, and project status analysis. Additionally, it is desirable that traceability tooling supports the safety assessment [66]. Therefore, manual approaches using spreadsheets are not effective enough because they do not support trace management or automated evaluation of trace chains.

To provide seamless tracing capabilities for the AMASS platform, we need to overcome the gaps between different types of safety engineering tools and general-purpose tools (Table 5). We identified three main approaches on how to bridge the gaps.

Our investigations showed that all the investigated tools have one primary integration mechanism such as XML files, a COM bridge, REST, OSLC, relational databases, and others. Using that integration mechanism has not been difficult in most cases, with OSLC being somewhat of an exception.

The Eclipse project Capra [32] follows the approach of point-to-point integration with only partial data import. It originally started as part of the AMALTHEA4 public research project [6] and is now in its incubation phase at the time of writing. Capra aims to provide a modular framework for tracing (Figure 38). Everything besides a small generic core is interchangeable and any number of new trace target types can be defined in new Eclipse plugins.

Table 5. Advantages and disadvantages of different cross-tool tracing strategies in ALM tools

Approach Description	Advantages	Disadvantages
OSLC	<ul style="list-style-type: none"> One implementation for many tools 	<ul style="list-style-type: none"> Low OSLC support among external tools, general reluctance by vendors to publish data to OSLC consumers
Point-to-point, Full data import	<ul style="list-style-type: none"> Full understanding of data Only internal traces 	<ul style="list-style-type: none"> Need full data metamodel support Large number of tools
Point-to-point, only partial import	<ul style="list-style-type: none"> Only partial support for external metamodels Use the primary integration mechanism from external tools 	<ul style="list-style-type: none"> Large number of tools need to be integrated

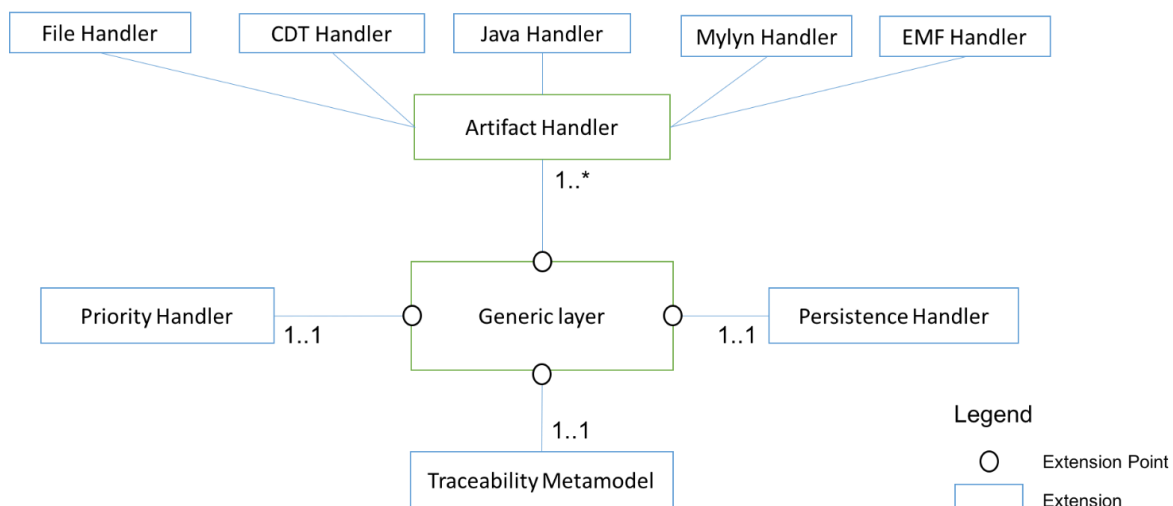


Figure 38. Capra architecture

There is a conceptual overlap between tracing to external sources and evidence management as the items being traced in a safety case will often (but not always) be used as evidence. It is recommended to trace the appropriate evidence objects, if those are available, for several reasons. Firstly, tracing with evidence gives the traced artefact its semantics, which is important for safety assessments. Secondly, evidence can be updated to a new version by repeating the process that created the artefact. This can be helpful for impact analysis. Finally, it removes redundancies from the model as only one type of artefact or evidence wrapper needs to be created.

Capra is built on top of EMF. When a non-EMF object is the trace target, instead a trace to a proxy EMF object is created. That proxy may contain information such as the URI of the original object. That approach is very similar to evidence management in CACM, so there may be opportunities to consolidate the trace metamodel with the evidence metamodel.

Another approach for seamless tracing can be the use of OSLC for the purpose of automatically generate safety cases, and for enabling continuous self-assessment. It is pioneered by Gallina et al. [45][46][47][48][49]. Figure 39 provides an overview of the approach.

Self-assessment can continuously semi-automatically be performed by compiling the different types of evidence. More specifically, Figure 39 limits its focus to the compilation of the evidence related to a limited portion of the ISO 26262 V-model regarding architecture management and quality management (testing). A user (in this case, the safety architect, software component developer, and the safety manager) may preliminary execute SPARQL queries of type “ASK” to make certain that the rationale (reasoning steps) have been documented. If this is the case, then, the safety case generator, embraced by the dotted-line in Figure 39, can execute SPARQL query of type “CONSTRUCT” in order to create and populate argumentation-related RDF-graphs, which are expected to be compliant to a specific argumentation meta-model. Finally, this model can be queried (via SPARQL query of type “SELECT”) and via a model-transformation the retrieved information, can be used to build e.g. GSN-goal structures in compliance with SACM/CACM (argumentation-related subset).

This vision-oriented investigation is still in its early stages and a proper evaluation of the approach has not been performed yet. Evaluating the approach is indeed challenging as the resources (time and workforce) are not available to develop the OSLC adaptors as well as other tools. The evaluation also requires the creation of a complex, real world safety case using it. Therefore, a phased evaluation is envisioned.

In this pioneering and conceptual work, no issue was taken into consideration concerning e.g. maturity of OSLC or scalability when performing complex queries. As surveyed in D5.1 [16], OSLC is still unstable to offer a solution spanning the entire ALM-tool chain. However, given its potential, it is worth investigating

this technological domain and in parallel contribute to its development since findings could be translated to other similar technological domains.

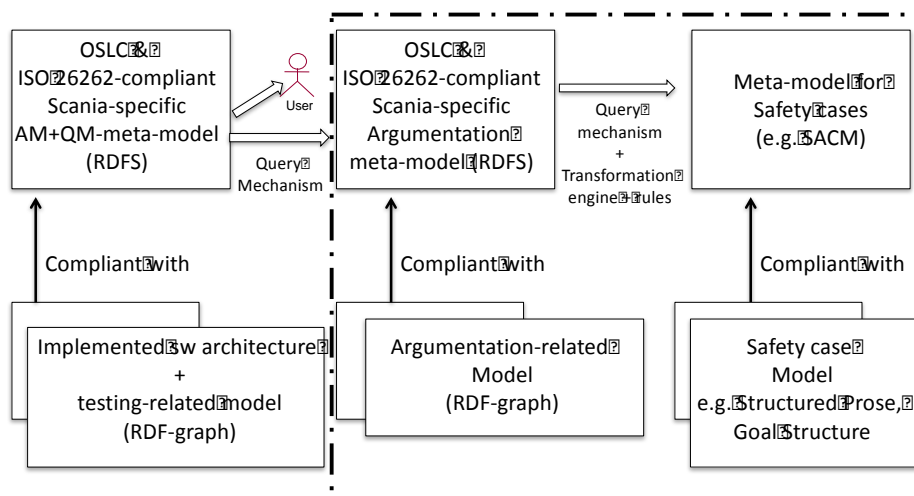


Figure 39. OSLC-based approach for self-assessment and traceability

3.1.12 Knowledge-Centric Automated Traceability (*)

By knowledge-centric automated traceability, we refer to the possibility of exploiting knowledge-centric techniques to support and automate traceability activities. Such activities are essential for CPS assurance and certification, as they aim to manage the relationships between artefacts of a CPS lifecycle, e.g. the relationships between a requirement and the test case that validates the requirement. These relationships show that an appropriate process has been followed during the lifecycle to guarantee system dependability.

In AMASS, a novel approach for knowledge-centric automated traceability has been envisioned. It is based on the RSHP model (see Figure 9), which allows the representation of any type of information or knowledge about a domain, including the semantics of the element under consideration. As outlined in Figure 40, traceability is managed in the approach through Traceability Projects that contain Traceability Modules, which in turn consist of traces. A traceability module is a special kind on RSHP artifact whose traces link elements of two blocks (e.g. requirements in an Excel file and the elements of a Simulink model). The source and target elements of a trace will be part of the source and target block, respectively, of its traceability module. Impact analysis could be performed based on the information of a traceability module. Finally, this RSHP-based approach enables the evaluation of the adequacy of a trace. First, two elements might be linked but a relationship might actually not exist between them, or it might not seem so. Second, a linked element could be modified and its traces might become valid. Tools implementing the RSHP model could support such actions because the aims to represent not only metadata about artefacts but also artefact content.

3.1.13 On-Demand Automated Traceability Maintenance and Evolution (*)

The constructed traceability links need to be maintained continuously or on-demand as a project evolves so that up-to-date traceability links are available when needed. The need for continuous traceability maintenance is triggered by changes to any of the software artefacts (like any architecture component) that, in turn, can be triggered by changes to artefacts within a traceability chain (e.g., underlying requirements and the code classes that implement the component). The semi-automatic support for such maintenance is achieved in some existing approaches [80][55][32]. However, continuous traceability maintenance might not be a feasible solution in case of a substantial evolution of a software system, such as a new major version in large real-world project in which hundreds of developers are involved, maybe even at different, distributed locations. For such new major versions, traceability links are subject to and undergo reconstruction in general. The automated approaches and tools, although helpful to a certain

extent, do not completely prevent insufficient understanding and/or misunderstanding of the traceability links, resulting in unconscious violations during the construction of traceability links [76][107].

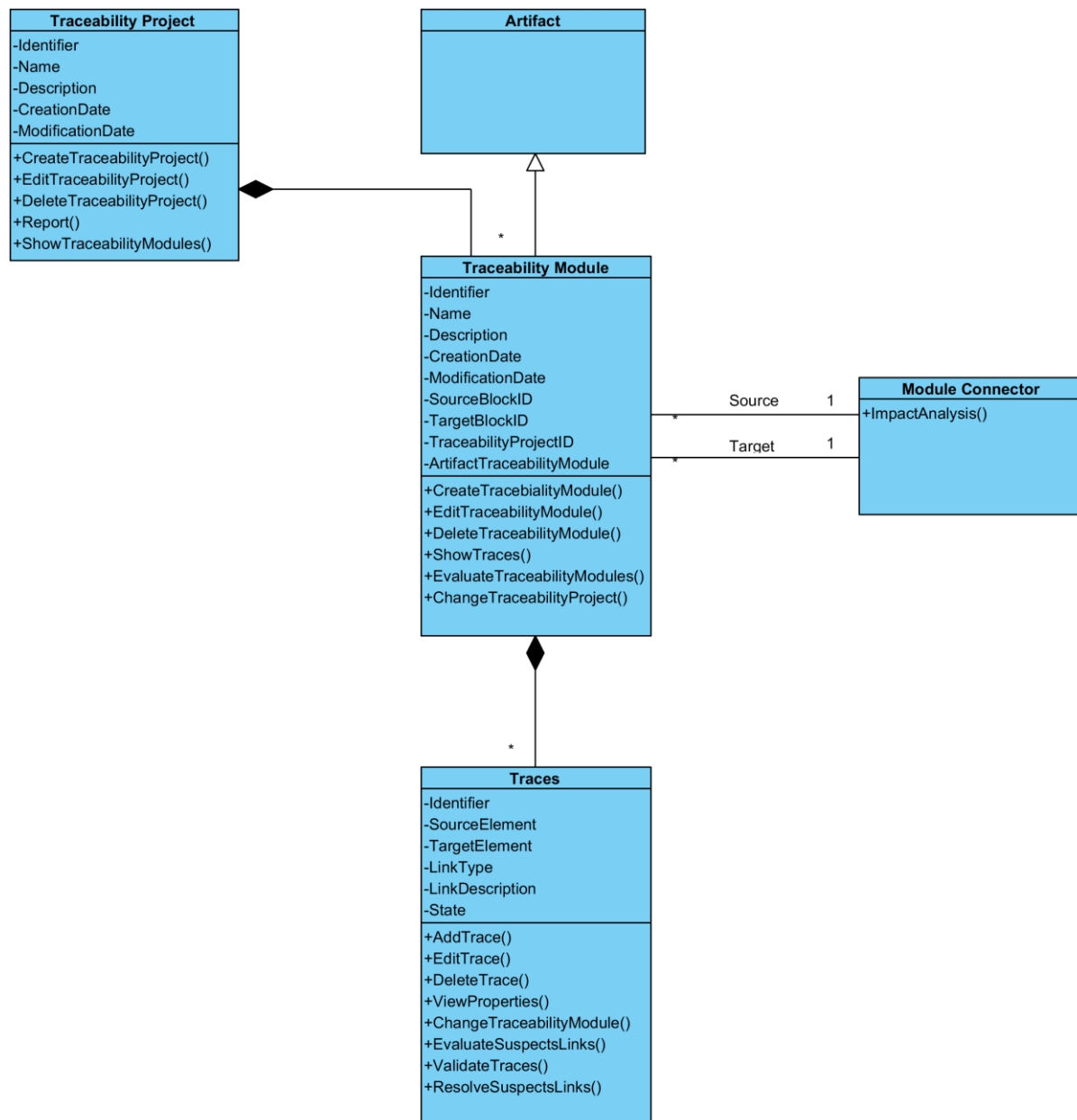


Figure 40. Model of knowledge-centric traceability

We have identified that a main driver for trusted and cost-effective traceability construction is the achievement of highest precision for the initial traceability links [68]. It is therefore decided to use the unchanged traceability links, if available, as an active countermeasure to arbitrarily make traceability decisions and to maintain and preserve the trust in further traceability construction. Accordingly, we have developed an on-demand automated approach for case-based maintenance and evolution of traceability links in the context of different versions of a software project [69]. The approach focuses on the component-to-component features for identification and prioritization of previous traceability cases, which are then used to perform reuse and adaptation of traceability links based on the matches and mismatches, respectively. The adapted (i.e., newly constructed) traceability links can then be verified by a human analyst and stored in a case base for future problem-solving situation. The focus on reuse and adaptation of traceability links would also reduce the burden on a human analyst, in particularly the already verified traceability links from the past (i.e., reused links) in both matched and partially matched cases might be

omitted from validation. However, only adapted traceability links for evolutionary changes would be made available to human analyst for validation. An overview of the on-demand traceability maintenance and evolution approach is shown in Figure 41.

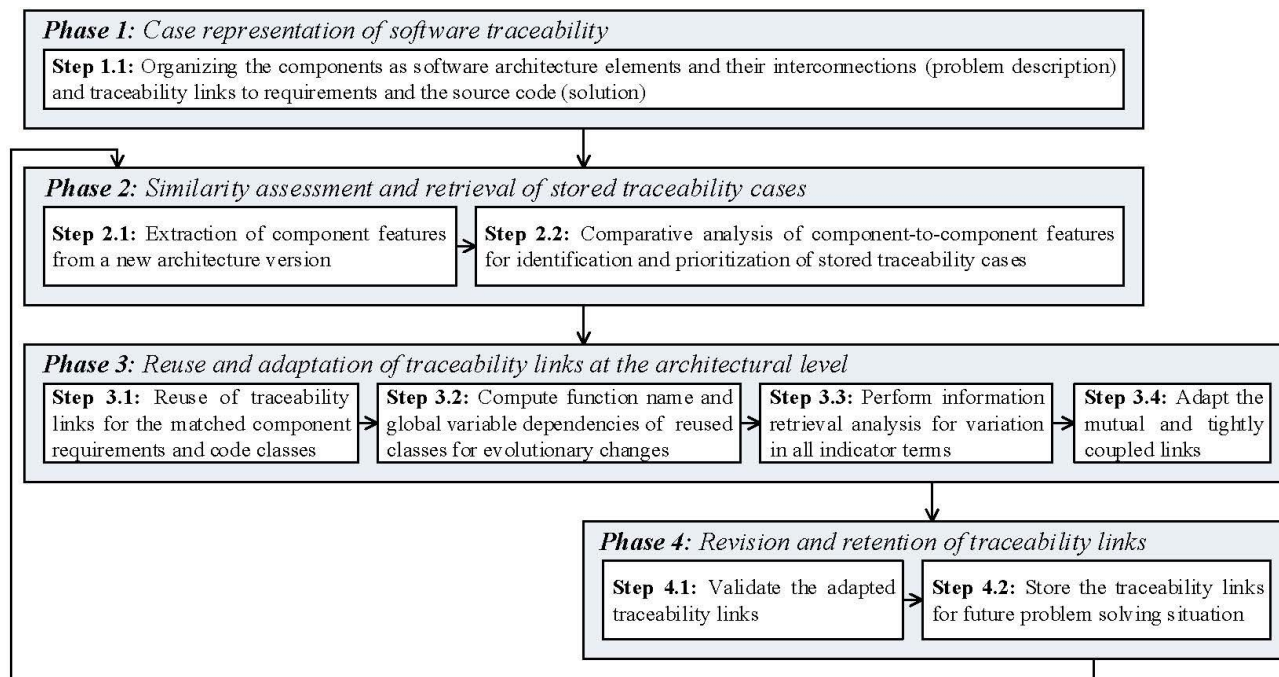


Figure 41. On-demand traceability maintenance and evolution

3.1.14 Automatic Translations for Collaborative Work

Another possible collaborative work is to automatic translate requirements of a specification from one source language to another target language in the frame of multinational teams.

Using the NLP technology provided by TRC as background of the AMASS process (NLP indexing process and the ontology approach including the pattern structures), this work has produced a process to perform this translation automatically.

The patterns are defined of sequences of slots, which are generic linguistic elements defined in the ontology, e.g., words, term tags, clusters or even other patterns.

The translation process is divided into two phases:

- The indexing process of the requirement in the source language, and;
- The transformation of the indexing output into a requirement in the target language.

The indexing phase is used without modification; however, the transformation phase is a completely new development.

The translation process (Figure 42) works as follows:

- The original requirement is analysed by the NLP process (indexing) taking into account the ontology from the source language, producing an output consisting of a sequence of terms that may match some patterns:
 - They define the structure of the translation by allowing to know where to locate each word in the target indexing output. (as described in the next point of the process).
- Then, this output is transformed into the output of a theoretically NLP process (indexing) of another equivalent requirement using the ontology from the target language.
- Then a reverse indexing process has been developed to transform this target language indexing output into the target language requirement.
- Finally, the output of this second phase is the translated requirement.



Figure 42. Stages of the translation process using NLP

Now we will focus on the research done in the reverse indexing process (also known as transformation or translation phase). This process is composed of the following stages:

- **Language mapping:**

This transformation (Figure 43) is accomplished by:

1. Using the source indexing output.
2. Navigating the links from words in source ontology to words in the target ontology.
3. Navigating links created for this purpose (to know where and how to locate the pieces of the output) in the ontology at a pattern slot level from source patterns in the source language ontology to target patterns in the target language ontology.

By using this technique, we can consider that the translated requirement will be grammatically correct.

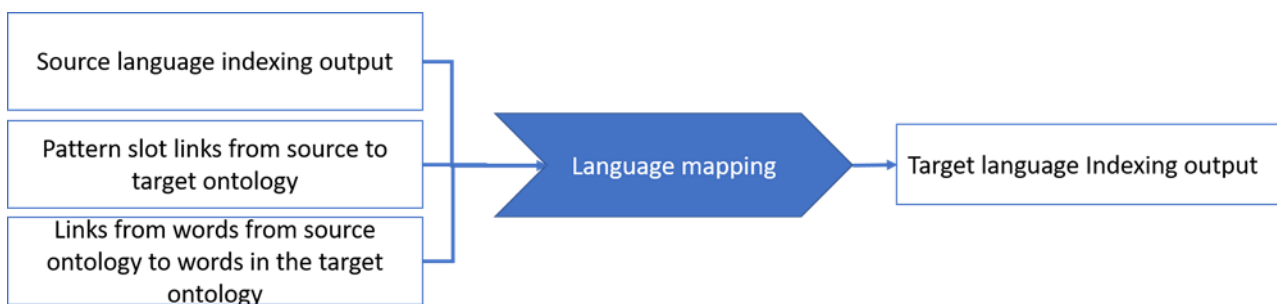


Figure 43. Inputs and outputs of the Language Mapping stage

- **Reverse disambiguation:**

It is responsible for choosing which conjugated word fits best. E.g. if the stage for a verb is going to generate two options for its form in past and future tense, this stage is going to evaluate all the possible options and select the best one. This is achieved (Figure 44) by automatically generating semantic clusters from normalization rules (explained in the next stage) along with a user-specified list of exceptions. If no suitable information is provided as input for this stage, it will generate both options for the next stage.



Figure 44. Inputs and outputs of the Reverse Disambiguation stage

- **Reverse normalization:**

The normalization rules allow get the root form of a conjugated word. But it is also possible to use them in the opposite direction. Thus, in the frame of this process (Figure 45), the ontology structure has been modified to be able to map normalization rules with slots belonging to those patterns used in the translation process. This new reverse normalization stage uses the information of the indexing output already transformed in the target ontology and this

normalization rules links to pattern slots, the get the requirement translated into the target language.



Figure 45. Inputs and outputs of the Reverse Normalization stage

3.1.15 Evidence Change Impact Analysis

Evidence change impact analysis can be defined as the assurance activity that attempts to identify, in the body of safety evidence, the potential consequences of a change. Possible consequences can be the need for adding, modifying, or revoking some evidence artefact. This activity can be necessary in several situations, such as:

- Modification of a new system during its development
- Modification of a new system as a result of its V&V
- Reuse of existing components in a new system
- Re-certification of an existing system after some modification
- Modification of a system during its maintenance
- New safety-related request from an assessor or a certification authority
- Re-certification of an existing system for a different operational context
- Re-certification of an existing system for a different standard
- Re-certification of an existing system for a different application domain
- Changes in system criticality level
- Independent assessment of the risk management process
- Hazards identified after the fact
- Re-certification for temporary works
- Accident analysis

In AMASS, the lifecycle of an artefact according to its status from an impact analysis has been adopted from the OPENCROSS project ([87]; Figure 46). It is supported by OpenCert. When created, an artefact is valid. Then, as a result of changes in the artefact or in related artefacts, it can be necessary to validate or to modify the artefact. It can also happen that the artefact is revoked, i.e. that the artefact is not valid anymore in an assurance project. For example, testing results should be revoked when the source code whose validation the results report changes.

It can be necessary to propagate changes when an artefact (or assurance asset in general) is modified or revoked. Depending on the change effect kind of a relationship between two artefacts (see D2.2 [9]), the effect can be:

- None: A change has no effect
- ToValidate: A validation is required a consequence of a change
- ToModify: A modification is required a consequence of a change
- Modification: A change causes a modification
- Revocation: A change causes a revocation

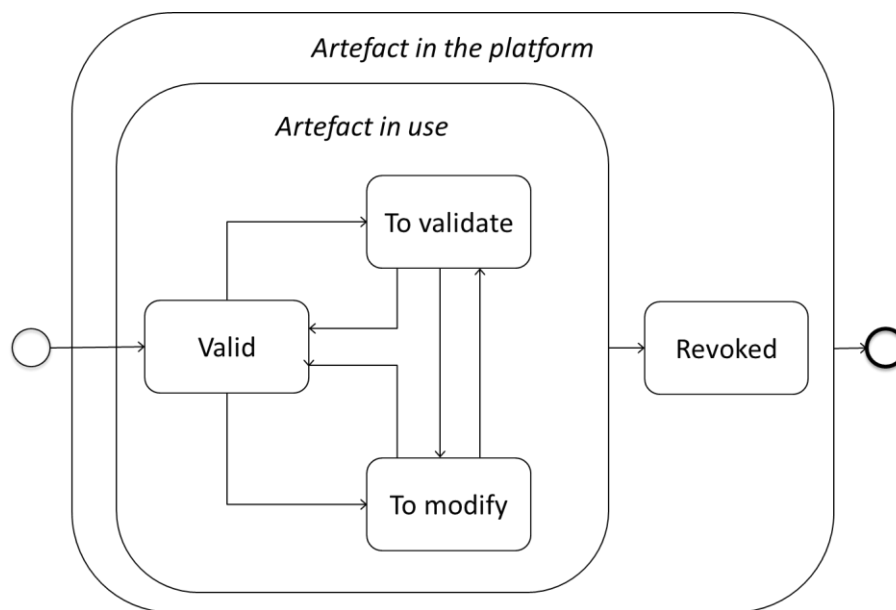


Figure 46. Evidence artefact lifecycle from an impact analysis point of view [87]

3.1.16 Management of V&V evidence (*)

According to IEC 61508 [65], verification can be defined as a confirmation by examination and provision of objective evidence that certain requirements have been fulfilled, and validation as a confirmation by examination and provision of objective evidence the particular requirements for a specific intended use are fulfilled. Examples of artefact types that can be used as V&V evidence include results from testing, simulation, formal verification, inspections, and reviews (see Figure 2 above).

When we talk about V&V evidence, people typically think of evidence that aims to confirm some characteristics of a system, e.g. the activation of a certain safety mechanism in a vehicle under certain circumstances. This evidence must obviously be managed for CPS assurance and certification. The body of V&V evidence to manage must include further artefacts, and more concretely it must include V&V evidence of the characteristics of evidence artefacts. For example, in DO-178C, the Software Verification Results must provide evidence that high-level requirements are accurate and consistent. In this case, the V&V evidence is about the specified requirements, not about the system itself.

The management of the latter V&V evidence has some specific characteristics that must be taken into account. Using as an example the need for confirming that a requirements specification must be correct, consistent, and complete, a process for managing this V&V evidence can be defined as follows:

1. Determine the aspect to take into account (e.g. correctness, consistency, and completeness)
2. Determine how the aspect will be assessed (e.g. through a metric that assess correctness of individual requirements according to its length).
3. Determine the target levels (e.g. the length of a requirement must be between 5 and 20 words)
4. Evaluate the aspect (e.g. measurement of the number of words of a requirements)
5. Record the evaluation result
6. Analyse the result
7. Link the result to its associated artefact under evaluation (i.e. a requirement in the example)

This process can be regarded as a specialisation of the process presented in Section 2.2. Subsequent steps would be the revision of the artefact under evaluation according to the evaluation results.

Since this V&V evidence must be managed, its information would need to be recorded in the AMASS Tool Platform through the Evidence Management building block. Using the CACM version presented in D2.2 as a reference [9], a Managed Artifact (e.g. a requirement) would include a Managed Artifact Evaluation (e.g. to record the correctness evaluation based on the requirement length). In OpenCert, an Assurance Asset

Evaluation should be associated to an Artefact of an Artefact Definition, in an Artefact Model (Figure 47). An Assurance Asset Event could be associated to the Evaluation. An Artefact Value should not be used because they are aimed at recording information about properties that are part of an Artefact itself (e.g. the result of a test case). V&V results for evidence artefacts are usually part of other artefacts (e.g. a verification report).

Last but not least, it must be noted that V&V evidence does not only encompass artefacts from automated activities such as testing, but also manual V&V results. These results could be based on checklists that manually filled according to the criteria and items defined.

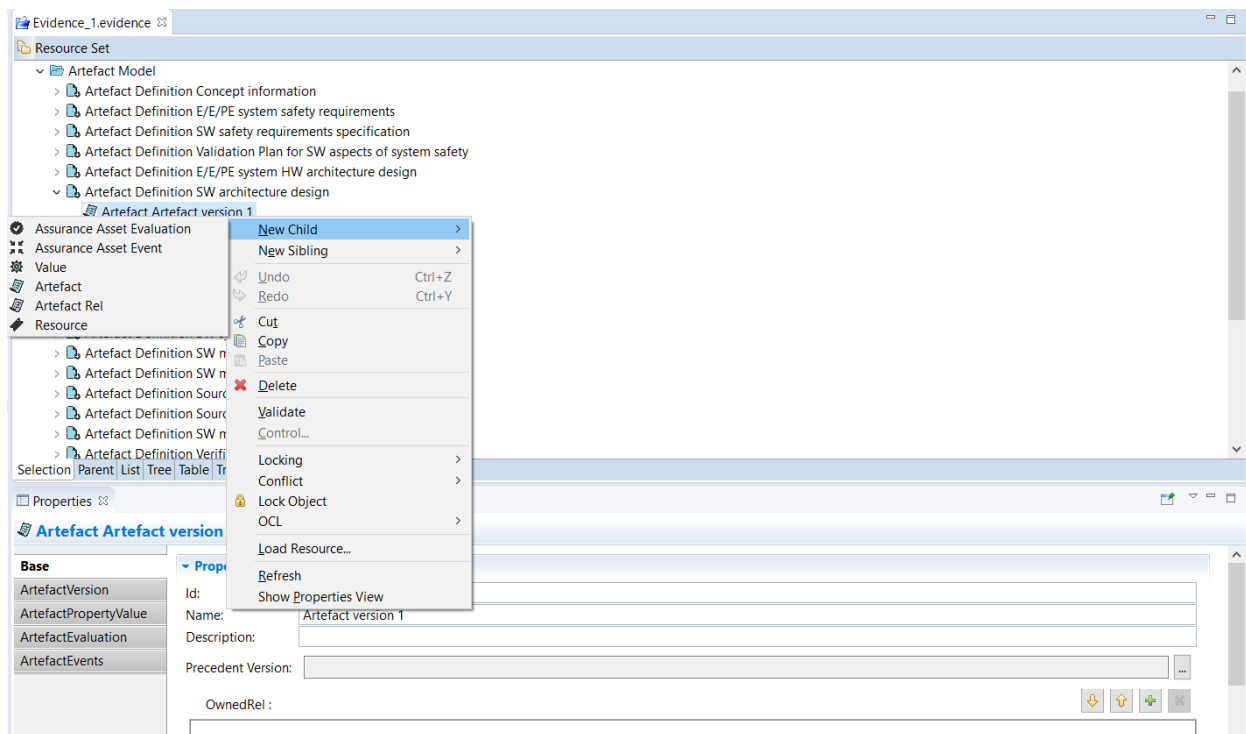


Figure 47. Assurance Asset Evaluation in an Artefact Model

3.1.17 Security Management (*)

CDO includes a security model to specify access rights to the objects of CDO repositories, this model is supported by an API that will be used to develop this component. Only the access to models stored in the AMASS CDO Centralised Repository could be controlled, any other data used by AMASS Platform but not stored in it could not be secured.

The Security Management module should allow to users with the Administrator role creating access policies to AMASS CDO Centralised Repository. This access policies will consist in managing users, groups of users and roles:

- **Users** represent a user of the AMASS Platform.
- Users may be grouped together into **Groups of Users**. One user can belong to several groups.
- **Roles** restrict the access and the access rights over the existing data stored by the Data Module in the CDO Repository. A role can be assigned to a user or to a groups or users, that is, to all the members of the group.

The Figure 48 shows a mock-up with the design of the interface to manage those three concepts. The window presents the existing groups of users, users and roles in selectable lists with buttons over them for create and delete operations. Controls to edit the corresponding properties of the selected element of the lists on the left side will be displayed on the right side of the window.

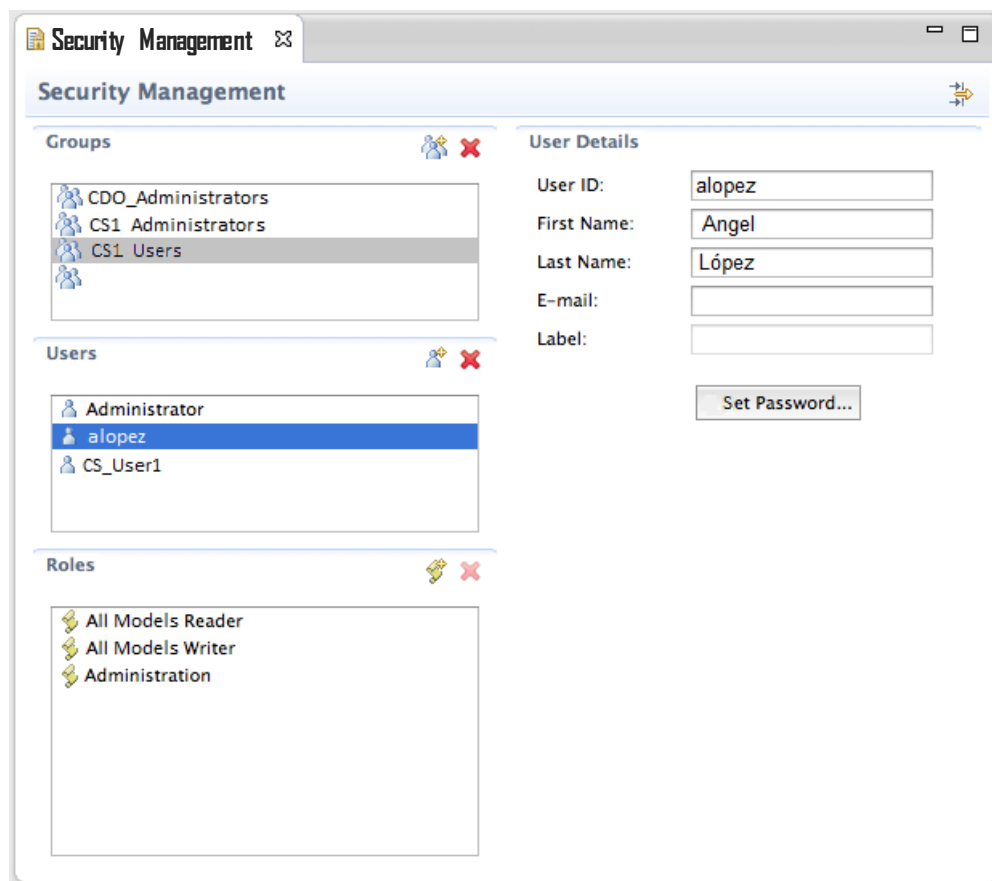


Figure 48. Security Management window managing users

The Figure 49 shows another mock-up of the same window but managing a role. For a role, besides entering a name, it is necessary to select the folder or model stored in the CDO Repository that can be accessed by the user with the role and the access permission to the resource (READ or WRITE). The data of the CDO repository will not be accessible by default, therefore, resources whose access has not been specified in any role will not be accessible.

An access policy example applicable to the AMASS Project could be:

- A **Main_Admin** role with permissions to create users and to access with WRITE right to all AMASS CDO Centralised Repository data.
- A **General_Reader** role with permission to READ any data stored in the CDO Repository.
- One **CSX_Admin** role for each Case Study with permissions to manage the Case Study X users.
- One **CSX_User** role for each Case Study with WRITE rights to the data (Standards, assurance projects, evidences, Argumentations...) related to the Case Study X.
- One **CSX_Reader** role for each Case Study with READ permission to Case Study X data.
- **CSX_Users_Group**, a group for each Case Study with all the user that will work in the CSX generation. The corresponding CSX_User role will be assigned to this group.
- **CSX_Readers_Group**, a group with the users that are not working in the CSX but needs to see the generated data in the CS. The CSX_Reader role will be assigned to this group.
- **One user for each AMASS person** working in any of the 11 AMASS Case Studies. Each user will belong to his/her corresponding CSX_Users_Group group or to several groups in case he/she is involved in several Case Studies. If the user is the Case Study owner will have also the related CSX_Admin role. Besides, if the user needs access to other CS data, he will be included to the corresponding CSX_Readers_Group/s or assigned the corresponding CSX_Reader role/s.
- A **Guest user** with the General_Reader role for any person not involved in the AMASS Project that needs to access to the AMASS Platform.

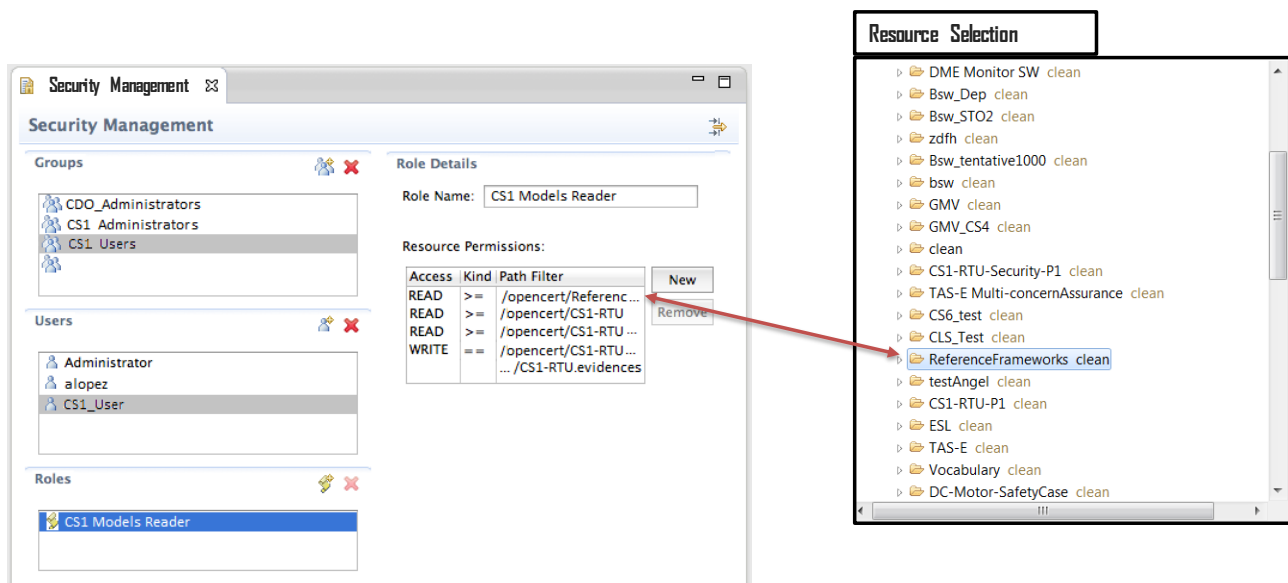


Figure 49. Security Management window managing roles

3.1.17.1 Perspectives on empowering the AMASS Tool Platform with respect to cybersecurity vulnerabilities (*)

Guaranteeing the security of the AMASS Tool Platform is not in focus in any of the AMASS scientific and technical objectives. However, within WP5, a plan for future development has been elaborated and is briefly sketched in this section.

The plan consists in exploiting state of the art solutions for monitoring known vulnerabilities (<https://cve.mitre.org>). Given the relevance and criticality of the growing number of vulnerabilities, ongoing research is aiming for better mining the information contained in vulnerability repositories in order to be able to react more promptly and allocate resources to the right mitigation actions.

In future releases of the AMASS Tool Platform, the plan is to connect to state of the art tools (and eventual empowered tools) currently used by vulnerability assessors and get reports on vulnerabilities that could affect the Platform. More concretely, the plan is to build on top of work presented in e.g. [21][54][100].

3.1.18 Data Management (*)

The data management is responsible of providing data to other components but considering the existing access policies established by the Security Manager. Using the CDO security API, it has to ensure that only the accessible data is provided to other components and only the permitted operations over the data are executed (see Figure 50).

The data management module will require users to be authenticated for AMASS Platform access and if they are authorized users, only the accessible data according to their role/s will be shown to them. As example (see Figure 51), a CS1 user logs into to the AMASS Platform and only the data related to the CS1 is shown.

The data management also needs to control the permissions over the accessible data, avoiding write operation performed by not authorized users, as well as the permissions over the accessible data, avoiding write operation performed by not authorized users. Continuing with the previous example, the CS1 user tries to save the changes made to a model having only the READ permission over it (see Figure 52).

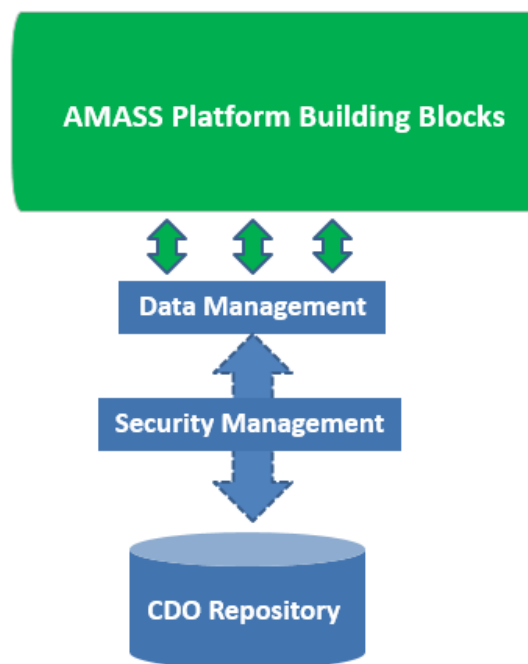


Figure 50. Data Management related to other modules.

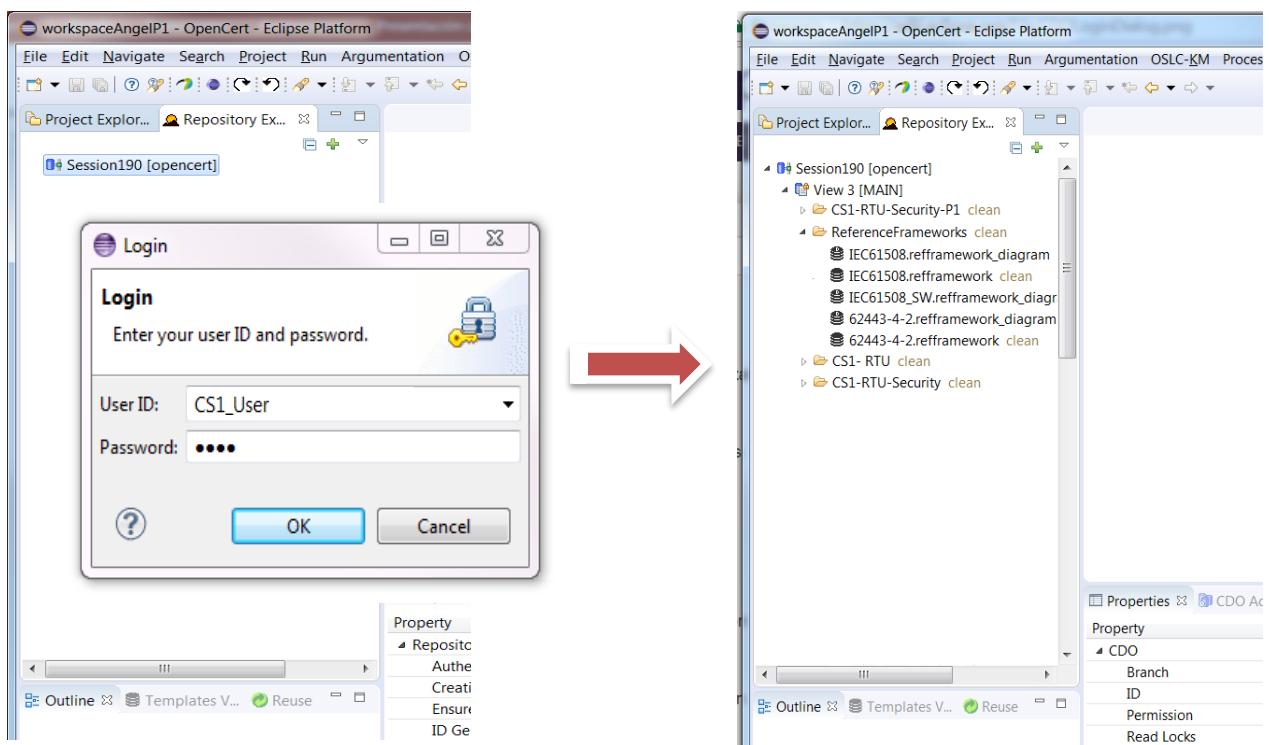


Figure 51. Authentication for Platform Access

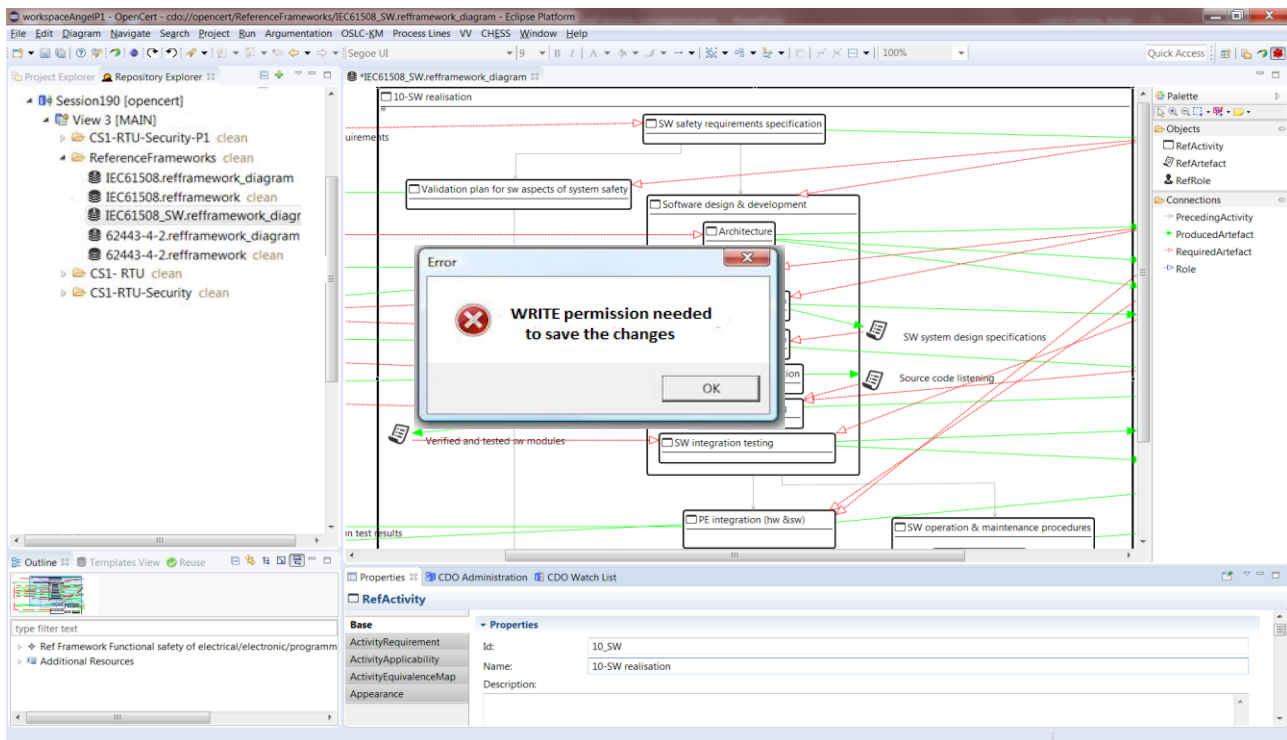


Figure 52. Permissions control

3.2 Seamless Interoperability Components

This section presents the tool components that will be included in the AMASS Tool Platform for Seamless Interoperability. The specification of the components corresponds to a refinement of the ARTA information presented in deliverable D2.3 [10] and D2.4 [11]. There are eight components for Seamless Interoperability, and each has been decomposed into several sub-components. The corresponding conceptual architecture will be materialised in specific components in D5.5.

3.2.1 Data Manager Component

Data Manager is part of the Platform Management in the ARTA (Figure 53). It has been decomposed into two components (Figure 54):

- Data Access is responsible for providing data to other components.
- Data Control is responsible for ensuring data validity.

3.2.2 Access Manager Component

Access Manager is part of Platform Management in the ARTA (Figure 53). It has been decomposed into two components (Figure 55):

- Permission Configuration deals with those aspects related to need for specifying user groups and how each different user can access the data and functionality of the AMASS Tool Platform.
- Permission Enactment takes care of ensuring that the permission configurations are properly used in the Platform.

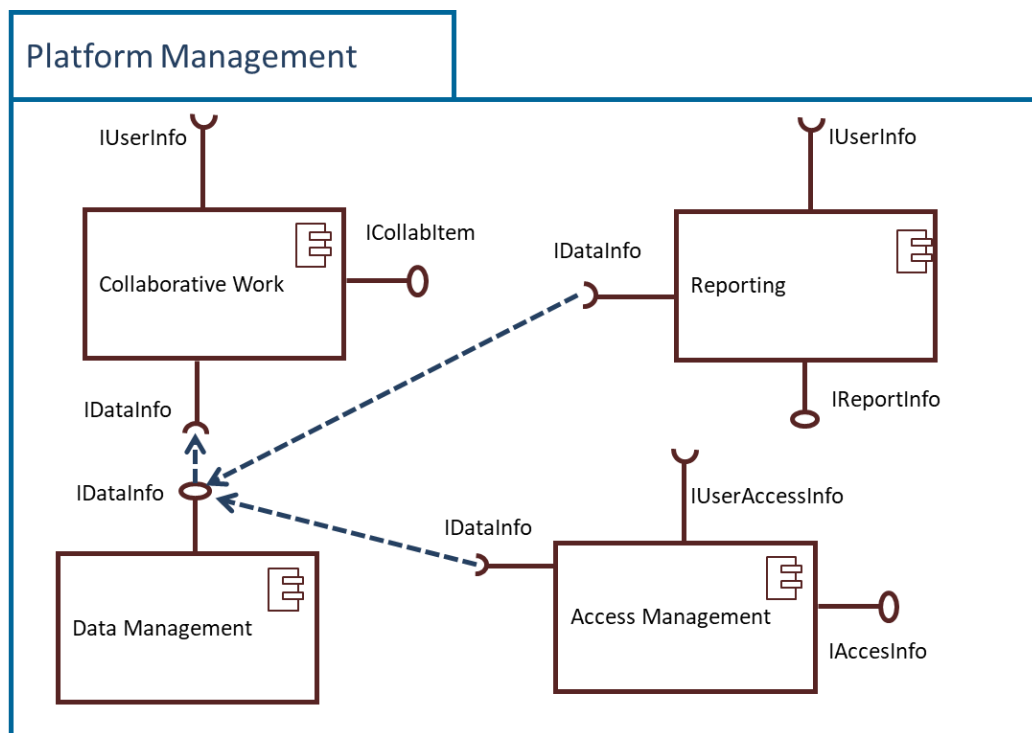


Figure 53. ARTA Platform Management Component

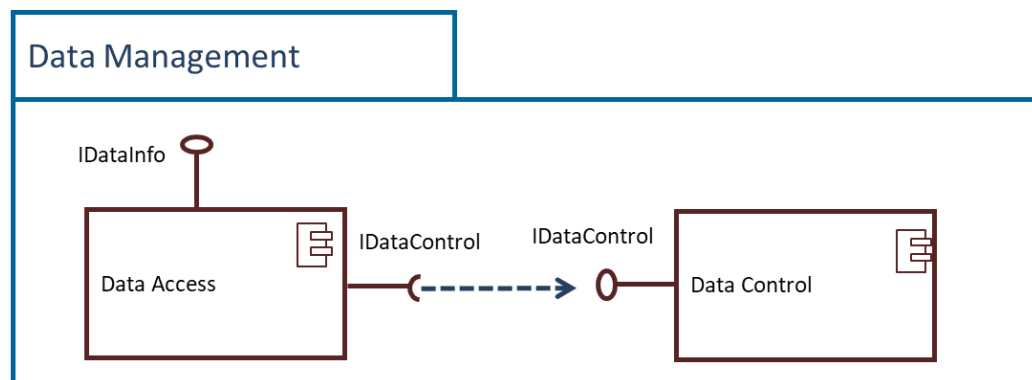


Figure 54. Data Management Component

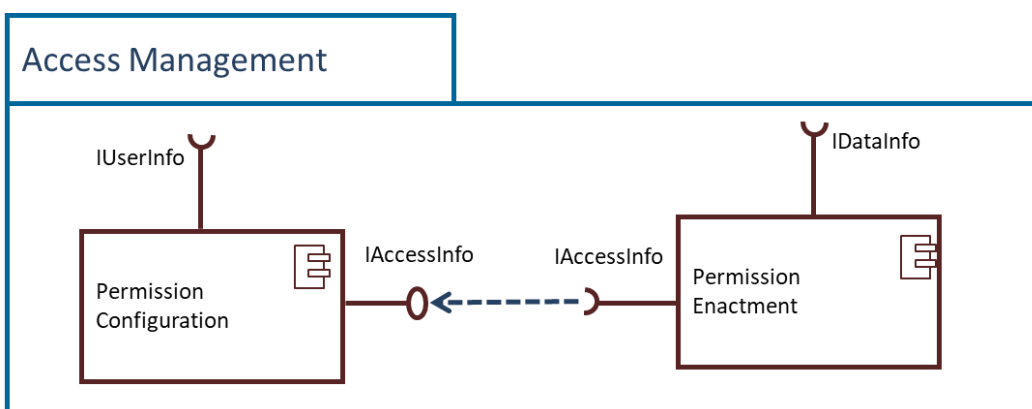


Figure 55. Access Management Component

3.2.3 Evidence Editor Component

Data Manager is part of Evidence Management in the ARTA (Figure 56). It has been decomposed into three components (Figure 57):

- Evidence Characterisation Manager is used for those general actions necessary to define the main characteristics of evidence artefacts at a given state.
- Evidence Evaluation Manager is responsible for the functionality required to allow users, or other tools, to specify assessments about evidence artefacts.
- Evidence Lifecycle Manager deals with the events that occur during an evidence artefact's lifecycle.

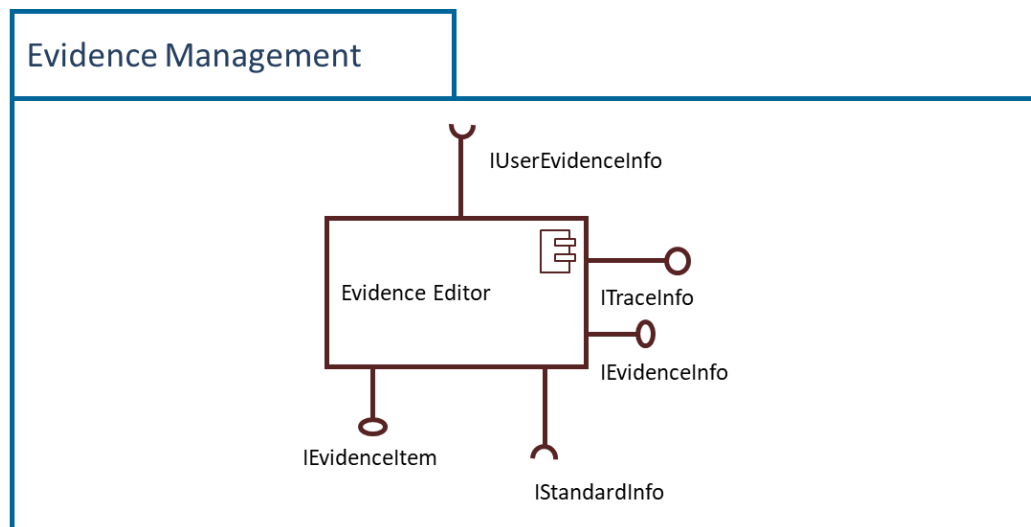


Figure 56. ARTA Evidence Management Component

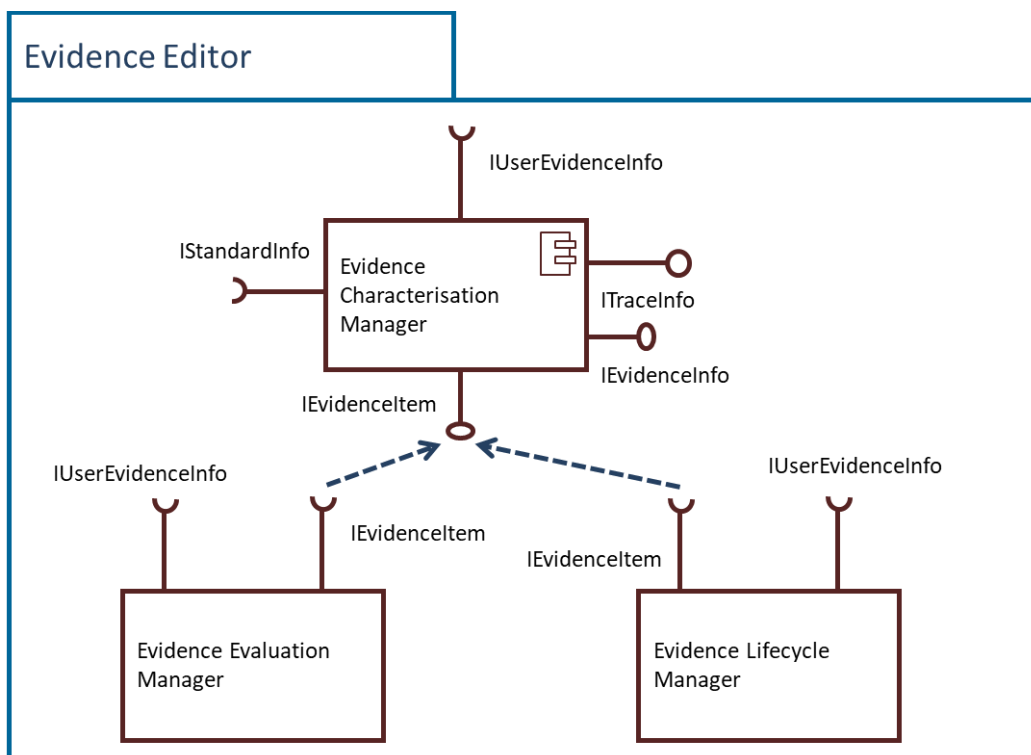


Figure 57. Evidence Editor Component

3.2.4 Traceability Management Component

Traceability Management is part of Assurance Traceability in the ARTA (Figure 58). It has been decomposed into two components (Figure 59):

- Trace Link Manager provides services for the specification and evolution of trace links between different assurance assets.
- Trace Rule Manager handles all the rules that constrain how traceability needs to be managed in a given assurance project.

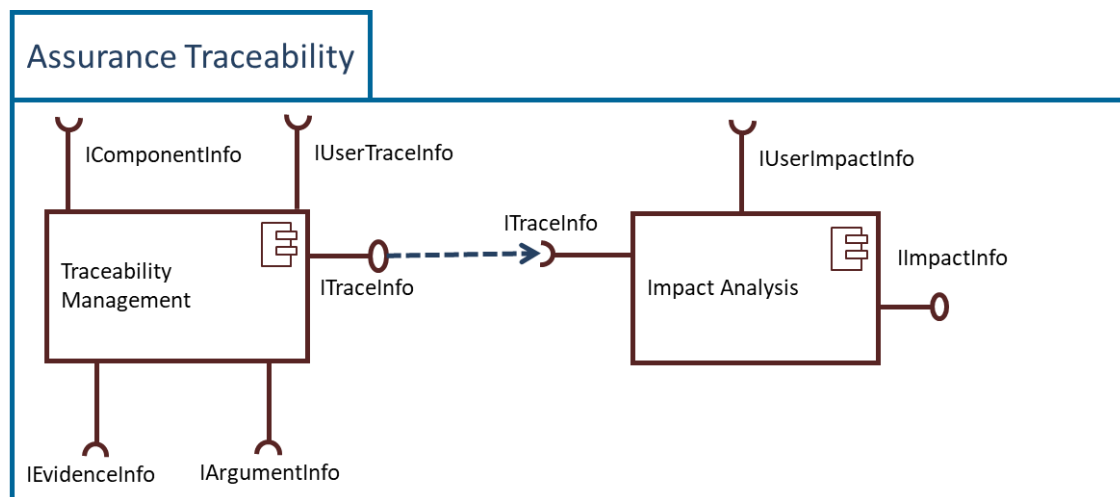


Figure 58. ARTA Assurance Traceability Component

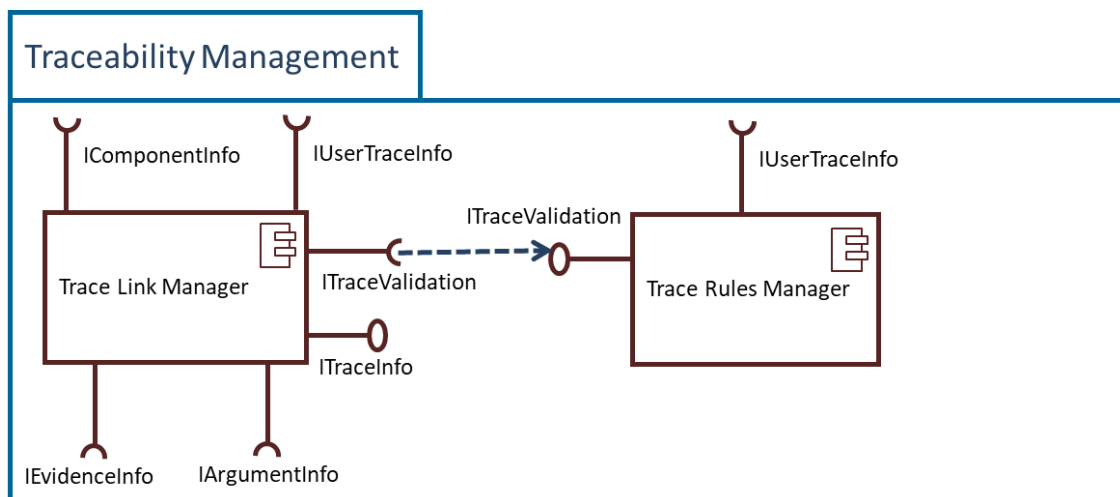


Figure 59. Traceability Management Component

3.2.5 Impact Analysis Component

Impact Analysis is part of Assurance Traceability in the ARTA (Figure 58). It has been decomposed into two components (Figure 60):

- Impact Analyser determines the consequences, and possible consequences, of the change of some assurance asset; i.e. how the change impacts or could impact other assurance assets.
- Impact Propagation is responsible for the actual recording of the impact on different assurance assets.

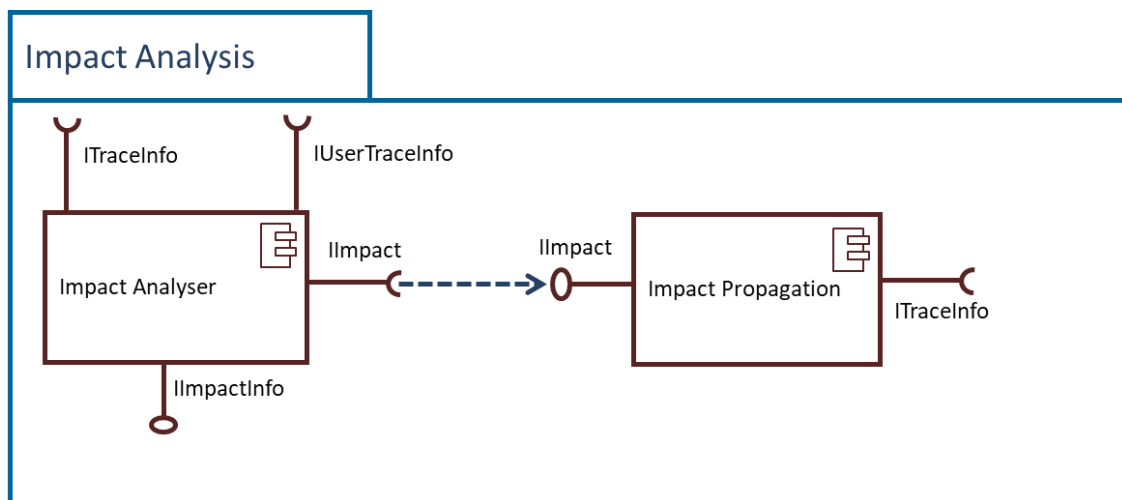


Figure 60. Impact Analysis Component

3.2.6 Collaborative Work Components (*)

Collaborative Work is part of Platform Management in the ARTA (Figure 53). It has been decomposed into two components (Figure 61):

- Synchroniser ensures that changes while accessing data concurrently are informed and reflected to the users accessing the data.
- Concurrent Access Manager supervises how different users use the same data at the same data, ensuring valid and consistent concurrent access. It also supports data mining.

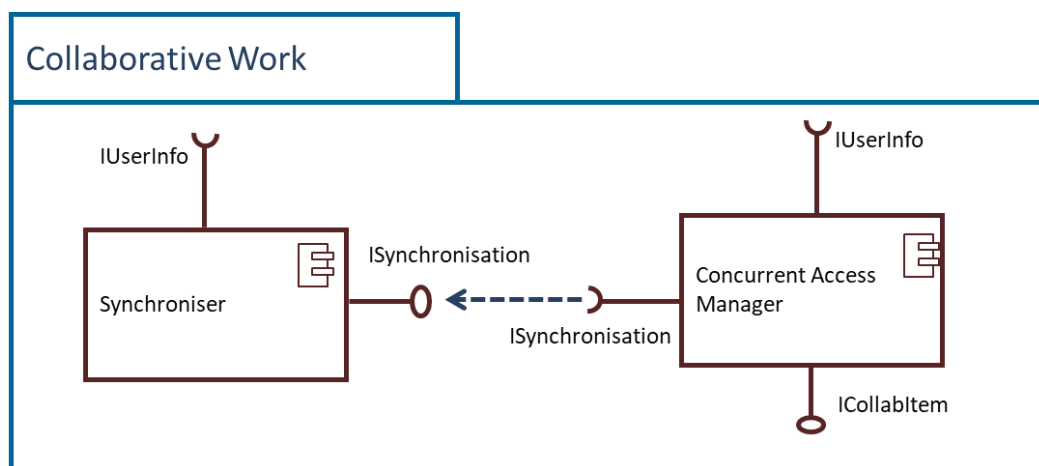


Figure 61. Collaborative Work Component

3.2.7 Toolchain Management Component

Toolchain Management is part of Tool Interoperability in the ARTA (Figure 62). It has been decomposed into two components (Figure 63):

- Toolchain Configuration is responsible for all those aspects necessary to set up a specific toolchain in an assurance project.
- Toolchain Enactment deals with the realisation and use of a specific toolchain in an assurance project.

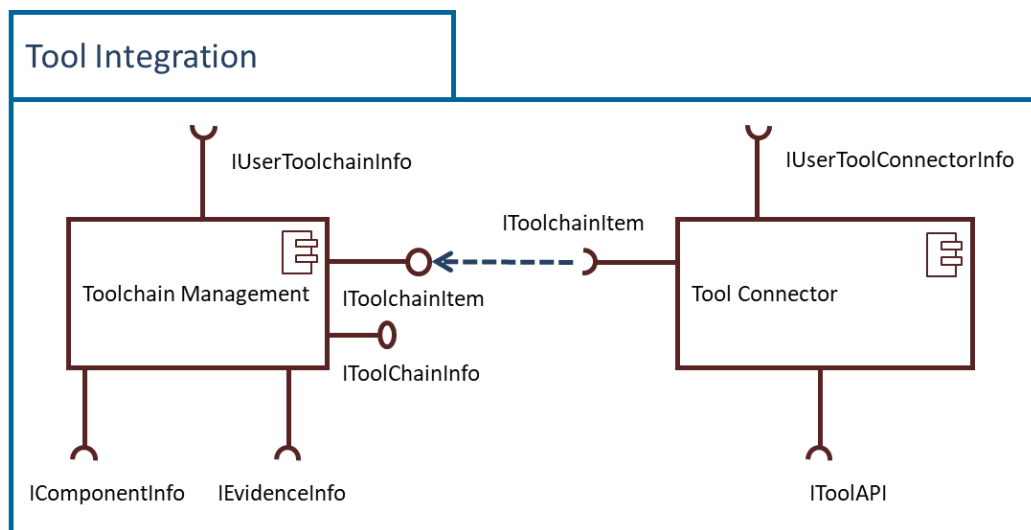


Figure 62. ARTA Tool Integration Component

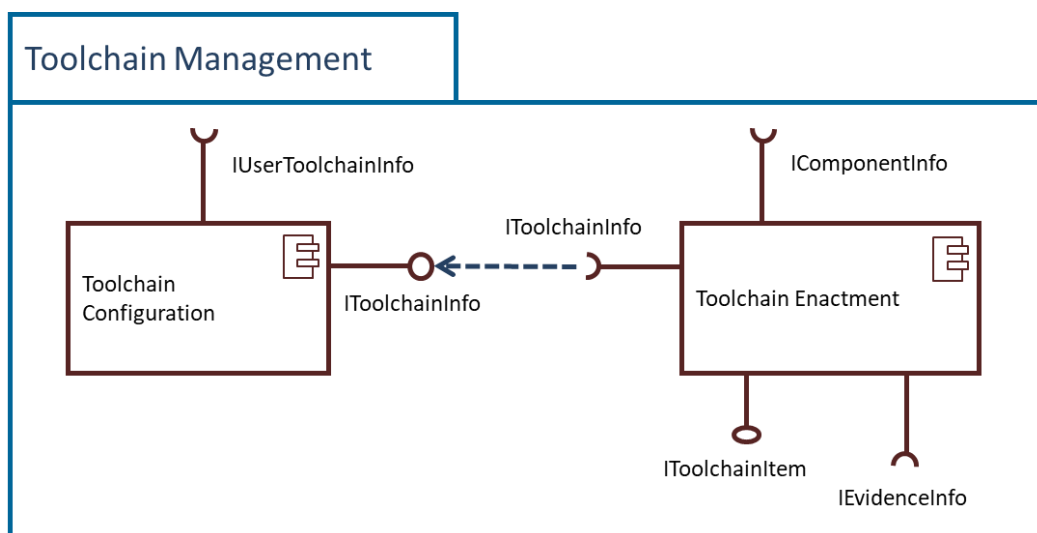
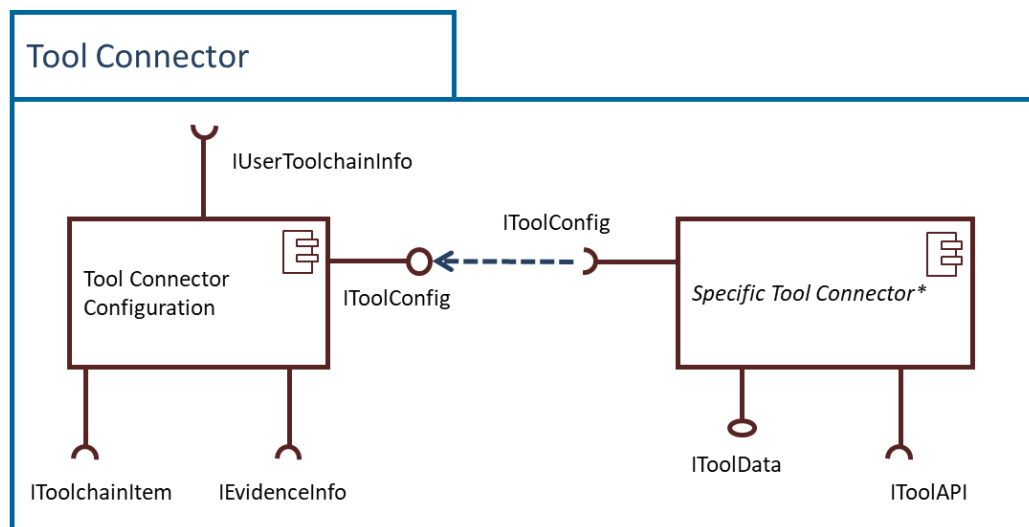


Figure 63. Toolchain Management Component

3.2.8 Tool Connector Component

Toolchain Connector is part of Tool Interoperability in the ARTA (Figure 62). It has been decomposed into two components (Figure 64):

- Tool Connector Configuration provides the services necessary to set up a connection between the AMASS Tool Platform and a specific tool.
- Specific Tool Connector generalises all the specific tool connector that will be used in the AMASS Tool Platform. Some could be associated to the overall Platform and some to one of the main Platform constituents, e.g. Papyrus. The envisioned specific tool connectors include OSLC KM, OSLC V&V, Simulink, Ad-hoc Connector, SVN, RSA, Rhapsody, ReqIF, Assurance Standard-based OSLC Extension, and OSLC Lyo.



*OSLC KM, OSLC V&V, Simulink, Ad-hoc Connector, SVN, RSA, Rhapsody, ReqIF, Assurance Standard-based OSLC Extension, OSLC Lyo

Figure 64. Tool Connector Component

3.2.9 Tool Characterisation Component

No specific components have been designed yet for Tool Characterisation. At this moment, the Compliance Management component of the ARTA (Figure 65) is the available support for Tool Characterisation.

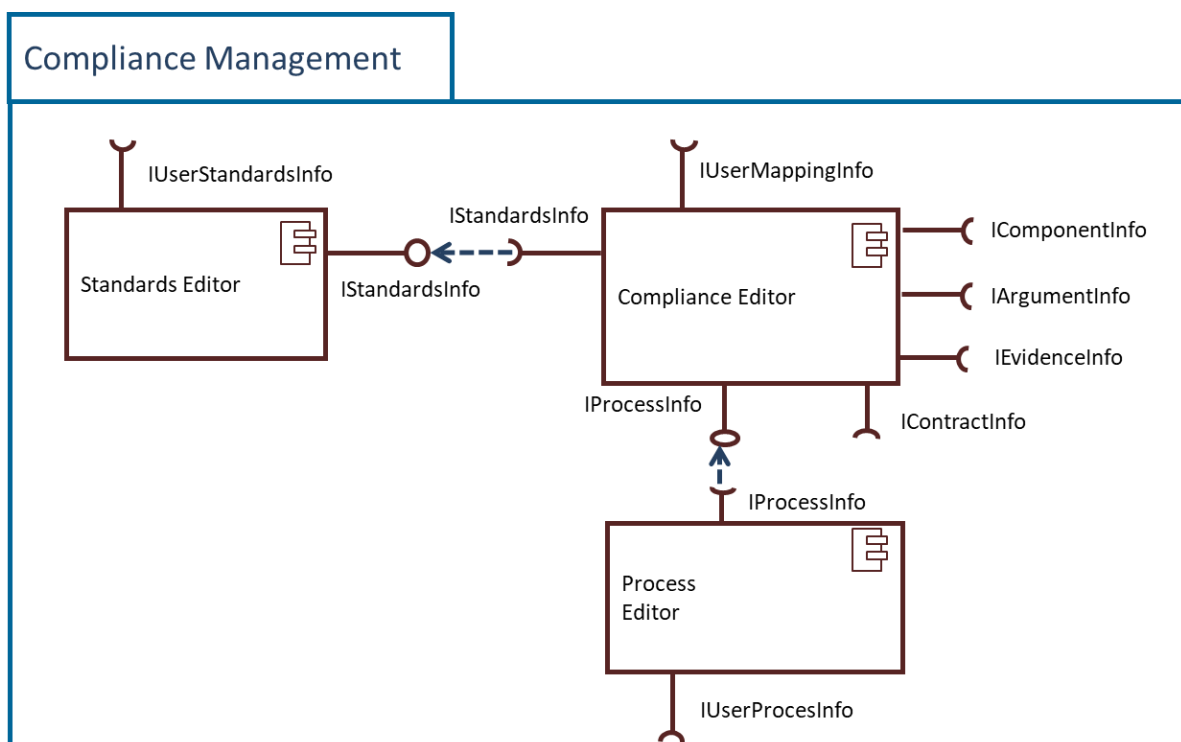


Figure 65. ARTA Compliance Management Component

4. Data Model

This section presents the data models that will be used for Seamless Interoperability, or as a basis for it, in the AMASS Tool Platform.

4.1 Evidence Management

For evidence management, the reference data model corresponds to the metamodel underlying the OpenCert tool [84]. This metamodel was developed in the OPENCROSS project [85].

Two sub-metamodels are the main ones for evidence management:

- Evidence metamodel (Figure 66), which includes the concepts, attributes, and relationship between concepts that characterise evidence artefacts, such as the resources where the artefacts are actually located or their properties and values. Artefacts are also manageable assurance assets (Figure 67), thus they can have events and evaluations associated to specify their lifecycle.
- Process metamodel (Figure 68), with which a user can specify how an assurance process has been performed: the participants, the activities executed, the techniques used, etc. These information items can be associated to evidence artefacts.

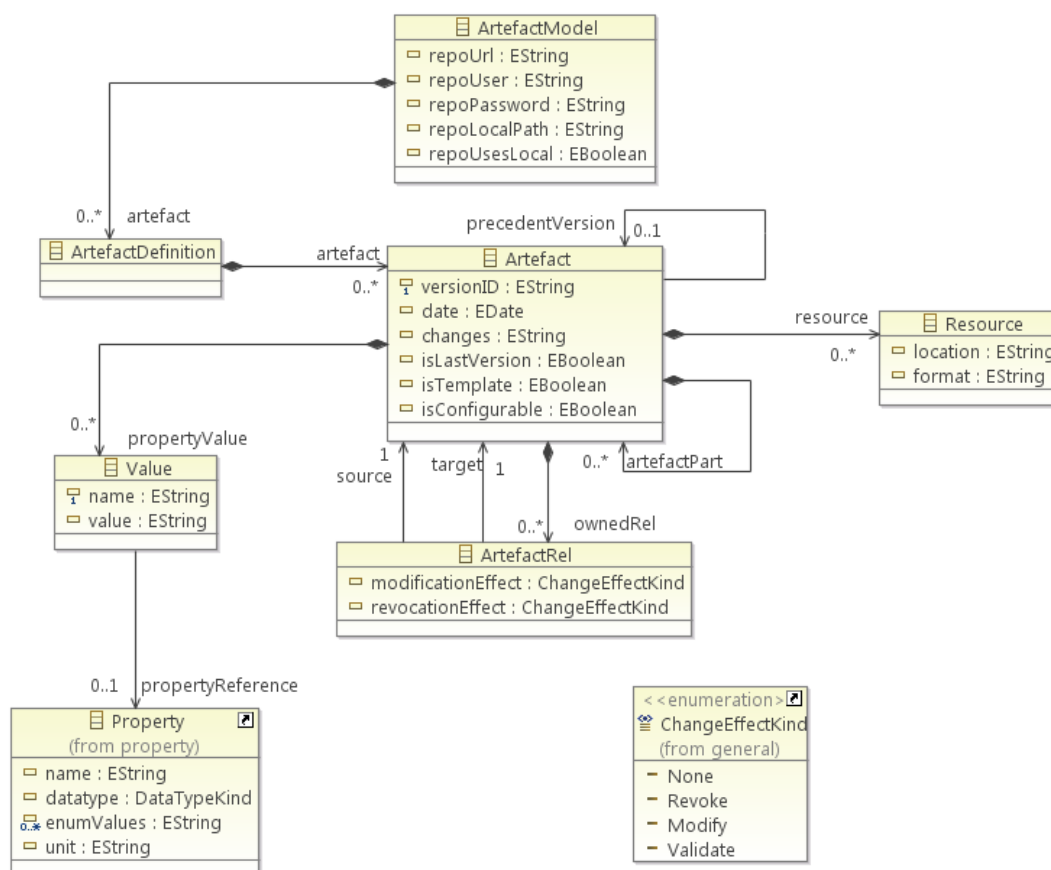


Figure 66. OPENCROSS evidence metamodel [86]

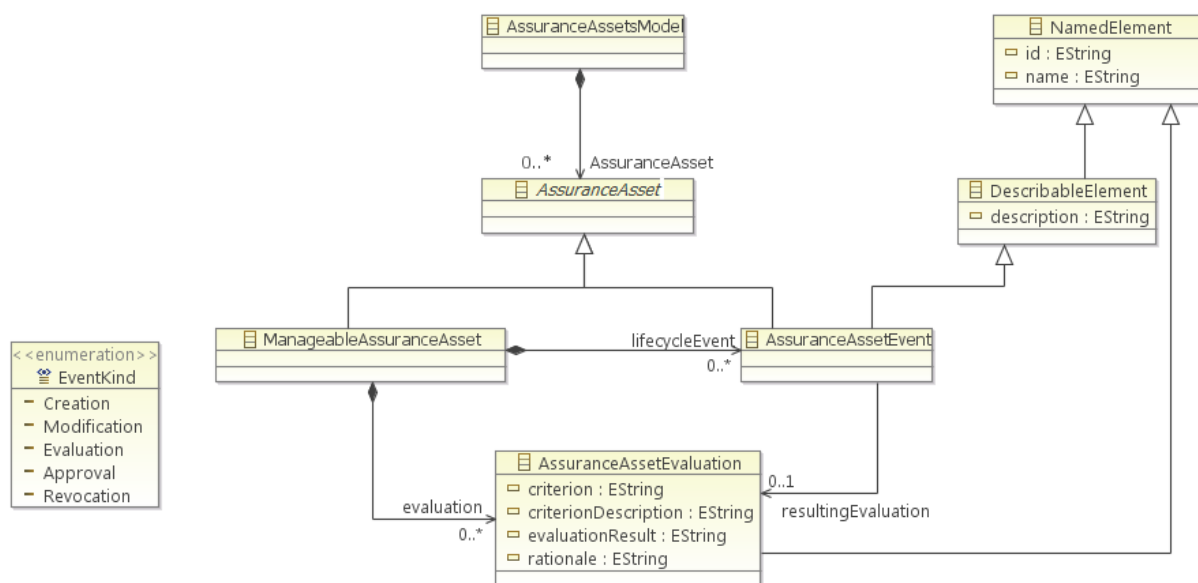


Figure 67. OPENCOS assurance asset metamodel [86]

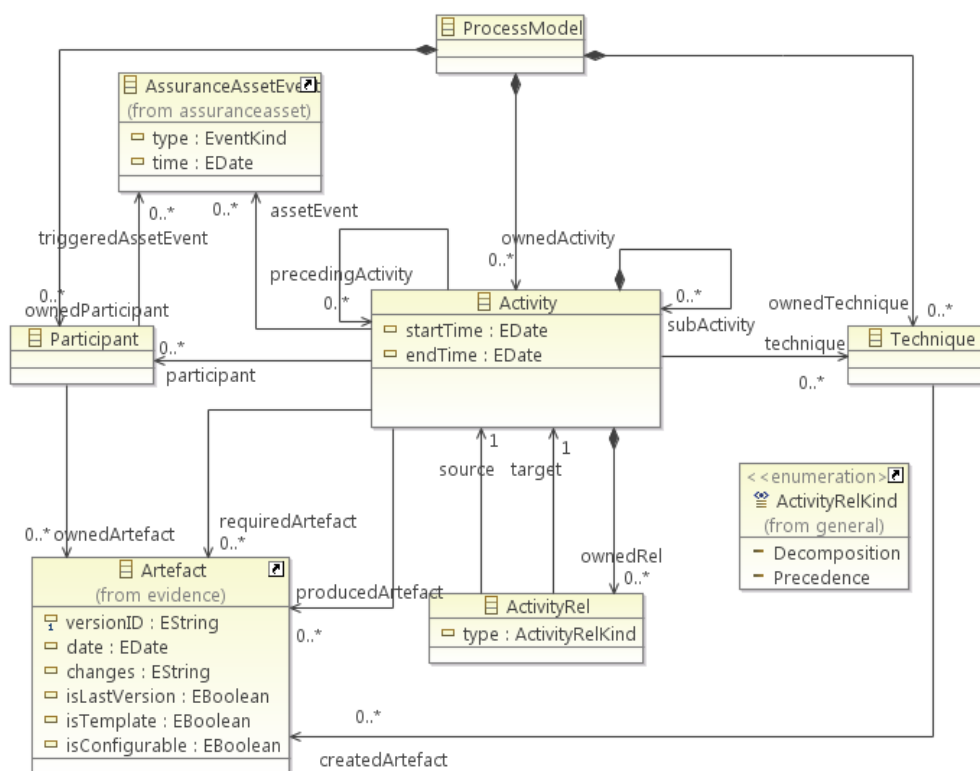


Figure 68. OPENCOS process metamodel [86]

4.2 Tool Integration

For tool integration, we use a metamodel proposed in the scope of the Eclipse Lyo project for toolchain specification [43] (Figure 69). The metamodel consists of three main parts:

- Domain Specification, to model the OSLC domain specification(s) to be exposed and/or consumed by an adaptor, as defined in the OSLC Core Specification.

- [illegible]

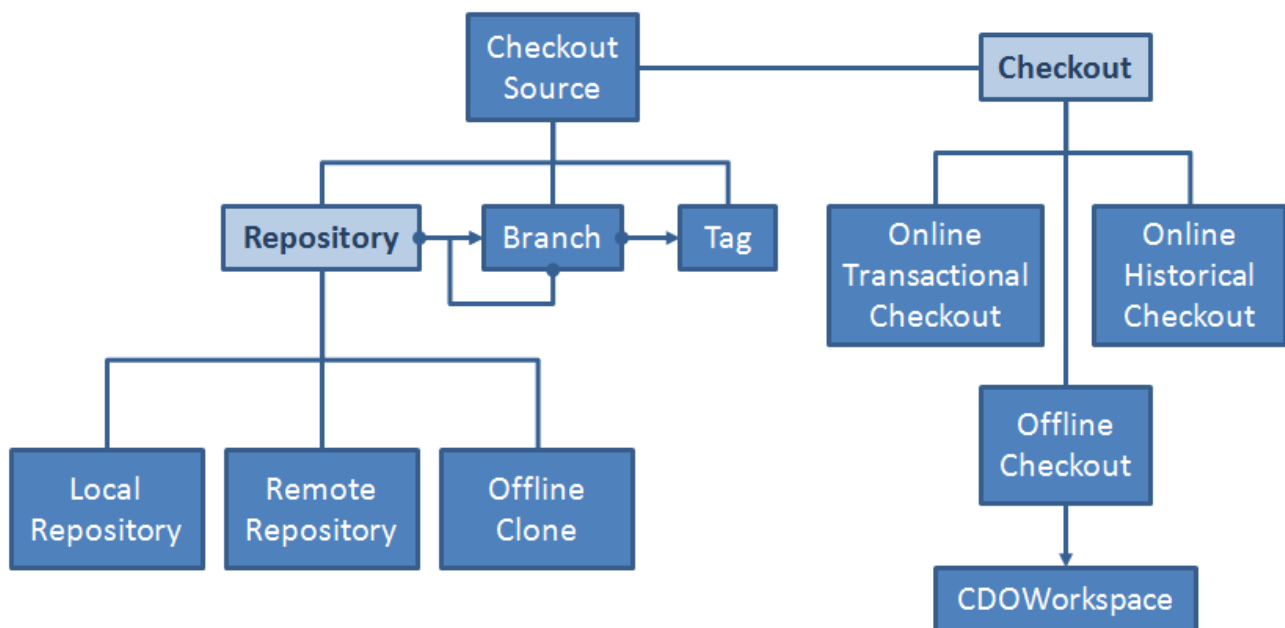
Page 76 of 93

4.3 Data Management

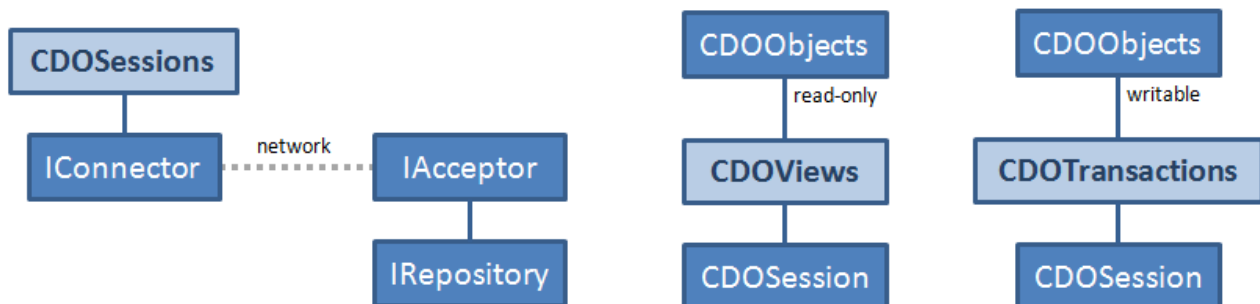
To present the data model for data management, we use the official information about the relationship between the main concepts that are exposed in the CDO User Interface and their underlying technical core concepts [41] (Figure 70).

Four parts can be distinguished:

- Checkout elements. A CDO Checkout is not necessarily a physical copy of a repository on the local disk, but checkouts generally represent the following two aspects:
 - (1) a reference to a configured repository as a way to use the internal session of that repository, and;
 - (2) a branch point information, i.e., branch and time stamp, that is needed to open views and transactions on the shared session of the referenced repository.
- Sessions, which are the technical representation of a protocol connection to a repository. On the transport level, this connection is provided by a Connector/Acceptor pair.
- Views, which provide a client application with all the models and model elements in a repository for a specific point in time and in a specific branch.
- Transactions, which provide a client application with all the latest models and model elements in a repository in a specific branch.



(a)



(b)

(c)

(d)

Figure 70. CDO data model [41]: a) checkout elements; b) sessions; c) views; d) transactions

4.4 Security Management

CDO includes a security model to specify access rights to the objects of repositories [42]. The default CDO security model is implemented on the server side as an Ecore model (Figure 71).

The root element of the security model is the realm, which contains directories, users, groups, and roles. Permissions grant NONE, READ or WRITE (includes READ) access to sets of objects.

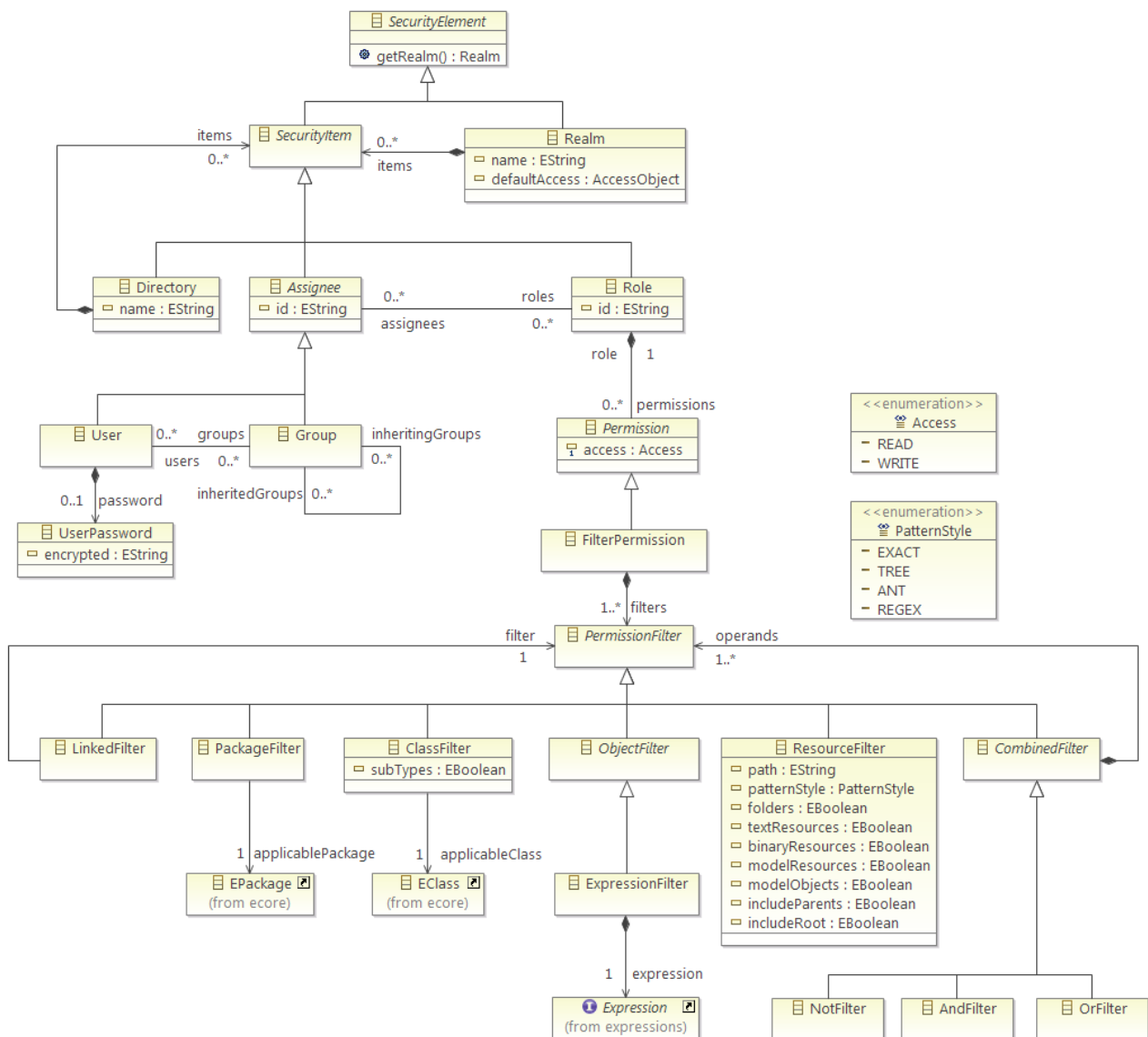


Figure 71. CDO security metamodel [42]

5. Way Forward for the Implementation (*)

As a summary of the way forward for implementation for seamless interoperability in AMASS, Table 6 includes the requirements to be implemented, extracted from D2.1 [8]. The table indicates the prototype in which each requirement is expected to be implemented. Extensions and improvements on some requirements in later prototype versions are possible.

D5.4 [18], D5.5 [19], and D5.6 [20] present the specific details about the tools implemented for Seamless Interoperability. As an overview:

- OpenCert [84] will be the main base tool from and on which seamless interoperability features will be implemented throughout the project.
- Capra [32] will be the main base tool from which traceability features will be implemented for the 2nd prototype.
- The development of mechanisms for tool integration will first not be developed on the AMASS Tool Platform, but with external, more stable and mature tools (e.g. with RQA [104]). The corresponding features will later be integrated in the AMASS Platform, or connectors to integrate the platform with the external tools will be provided.
- As a follow-up point to the previous one, it is possible that some features are not included finally in the AMASS Tool Platform. This can be a consequence of different aspects. Two examples are resource unavailability to complete the integration and commercial decisions of the partners (e.g. to provide a feature as an extension of the AMASS Tool Platform, not as part of the platform itself).

Table 6. Seamless Interoperability requirements

ID	Short Description	Description	Prototype	Priority	Status
WP5_EM_001	Evidence characteristics specification	The AMASS Tool Platform shall allow an assurance engineer to specify the characteristics of assurance evidence.	Prototype Core	Must	Solved
WP5_EM_002	Evidence traceability	The AMASS Tool Platform shall allow an assurance engineer to specify relationships between evidence artefacts.	Prototype Core	Must	Solved
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.	Prototype Core	Must	Solved
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.	Prototype Core	Must	Solved
WP5_EM_005	Evidence information import	The AMASS Tool Platform shall be able to import information about evidence artefacts.	Prototype Core	Must	Solved
WP5_EM_006	Evidence information export	The AMASS Tool Platform shall be able to export information about evidence artefacts.	Prototype Core	Must	Solved

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.	Prototype Core	Must	Solved
WP5_EM_011	Interactive evidence change impact analysis	The AMASS Tool Platform shall allow an assurance manager to indicate what evidence artefacts are actually impacted by the changes to a given evidence artefact.	Prototype Core	Must	Solved
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.	Prototype Core	Must	Solved
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.	Prototype Core	Must	Solved
WP5_EM_016	Evidence report generation	The AMASS Tool Platform shall be able to automatically generate reports, checklists, and evidence for certification purposes.	Prototype Core	Must	Solved
WP5_AM_003	User action log	The AMASS Tool Platform shall maintain a log with all the actions performed by the users.	Prototype Core	Must	Solved
WP5_DM_002	Simultaneous data access	The AMASS Tool Platform shall allow users to access data simultaneously.	Prototype Core	Must	Solved
WP5_DM_005	System artefact information storage	The AMASS Tool Platform shall be able to store information about any type of system artefact.	Prototype Core	Must	Solved
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, SysML, FMI, FMU...).	Prototype Core	Must	Solved
WP5_DM_007	Data versioning	The AMASS Tool Platform shall support data versioning.	Prototype Core	Must	Solved
WP5_TI_001	Automatic data collection	The AMASS Tool Platform shall automatically collect data from external tools.	Prototype Core	Must	Solved
WP5_TI_002	Automatic data export	The AMASS Tool Platform shall be able to automatically export data to external tools.	Prototype Core	Must	Solved
WP5_CW_003	Collaborative management of compliance with standards and of process	The AMASS Tool Platform shall support the collaboration among systems engineers, assurance managers for management of compliance with standards and of	Prototype Core	Must	Solved

	assurance	process assurance.			
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.	Prototype Core	Must	Solved
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.	Prototype Core	Must	Solved
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.	Prototype Core	Must	Solved
WP5_TQ_002	Tool quality evidence management	The AMASS Tool Platform shall manage evidence of tool quality.	Prototype Core	Must	Solved
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.	Prototype Core	Should	Solved
WP5_DM_001	Multi-platform availability	The AMASS Tool Platform shall be accessible from desktop, Web, and cloud environments.	Prototype Core	Should	Solved
WP5_TI_007	Version management tools interoperability	The AMASS Tool Platform shall be able to interoperate with version management tools.	Prototype Core	Should	Solved
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.	Prototype Core	Should	Solved
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.	Prototype Core	Should	Solved
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.	Prototype Core	Should	Solved
WP5_CW_012	Collaborative compliance assessment	The AMASS Tool Platform shall support the collaboration among assurance managers, assurance engineers, and assurance	Prototype Core	Should	Solved

		assessors for compliance assessment.			
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.	Prototype Core	Should	Solved
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.	Prototype Core	Should	Solved
WP5_CW_010	Collaborative compliance needs specification	The AMASS Tool Platform shall support the collaboration among assurance managers for compliance needs specification.	Prototype Core	Could	Solved
WP5_TQ_003	Tool quality information import	The AMASS Tool Platform shall be to import tool quality information such as tool qualification dossiers.	Prototype Core	Could	Solved
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.	Prototype P1	Must	Solved
WP5_TI_014	Client-server support	The AMASS Tool Platform shall support data and tool integration in client-server architectures.	Prototype P1	Must	Solved
WP5_TI_017	Standards-based interoperability	The AMASS Tool Platform shall support standard mechanisms for tool interoperability.	Prototype P1	Must	Solved
WP5_TI_018	Extended standard-based interoperability	The AMASS Tool Platform shall provide extended means to standard mechanisms for tool interoperability.	Prototype P1	Must	Solved
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).	Prototype P1	Should	Solved
WP5_TI_005	System specification tools interoperability	The AMASS Tool Platform shall be able to interoperate with system specification tools.	Prototype P1	Should	Solved
WP5_TI_006	V&V tools interoperability	The AMASS Tool Platform shall be able to interoperate with V&V tools.	Prototype P1	Should	Solved

WP5_EM_008	Visualization of chains of evidence	The AMASS Tool Platform shall display the chains of evidence to which an evidence artefact belongs.	Prototype P1	Could	Solved
WP5_AM_001	User authentication	The AMASS Tool Platform shall require users to be authenticated for Platform access.	Prototype P2	Must	Pending
WP5_AM_005	Access rights groups	The AMASS Tool Platform shall allow users to belong to different access rights groups.	Prototype P2	Must	Pending
WP5_TI_011	Non-proprietary data exchange	The AMASS Tool Platform shall provide exchange data in non-proprietary formats.	Prototype P2	Must	Solved
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.	Prototype P2	Must	Pending
WP5_AM_002	User access	The AMASS Tool Platform shall provide users with different options for data access and for action permission.	Prototype P2	Should	Pending
WP5_AM_004	User profiles	The AMASS Tool Platform shall allow users to have different profiles for Platform access.	Prototype P2	Should	Pending
WP5_DM_003	Consistent data access	When users are accessing data simultaneously, the AMASS Tool Platform shall manage the possible conflicts.	Prototype P2	Should	Pending
WP5_DM_008	Secure data access	The AMASS Tool Platform shall provide a secure standard API for data access.	Prototype P2	Should	Cancelled
WP5_TI_004	System analysis tools interoperability	The AMASS Tool Platform shall be able to interoperate with system analysis tools.	Prototype P2	Should	Pending
WP5_TI_008	Quality management tools interoperability	The AMASS Tool Platform shall be able to interoperate with quality management tools.	Prototype P2	Should	Solved
WP5_TI_010	Interoperability throughout CPS lifecycle	The AMASS Tool Platform shall be able to interoperate with some tool in all CPS lifecycle phases.	Prototype P2	Should	Pending
WP5_TI_012	Data entry effort	The AMASS Tool Platform shall allow users to create and enter data only once.	Prototype P2	Should	Solved
WP5_TI_013	Continuous data management	The AMASS Tool Platform shall support continuous data analysis, verification, and integration.	Prototype P2	Should	Solved
WP5_TI_015	Service offer and discovery	The AMASS Tool Platform shall allow clients to ask for a server's	Prototype P2	Should	Solved

		services and to discover servers.			
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.	Prototype P2	Should	Pending
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.	Prototype P2	Should	Pending
WP5_CW_013	Metrics & measurements reports	The AMASS Tool Platform shall manage metrics and measurements about collaborative work.	Prototype P2	Should	Pending
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.	Prototype P2	Could	Pending
WP5_EM_012	Evidence trace verification	The AMASS Tool Platform shall analyse the quality of the relationships between evidence artefacts.	Prototype P2	Could	Pending
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.	Prototype P2	Could	Pending
WP5_TI_009	MS Office applications interoperability	The AMASS Tool Platform shall be able to interoperate with MS Office applications (Word, Excel, Visio, etc.).	Prototype P2	Could	Solved
WP5_TI_016	Performance monitoring	The AMASS Tool Platform shall allow continuous performance monitoring of the servers.	Prototype P2	Could	Solved
WP5_CW_005	Collaborative system V&V	The AMASS Tool Platform shall support the collaboration among systems engineers for system V&V.	Prototype P2	Could	Solved

6. Conclusion (*)

This document is the second and final version of the design for seamless interoperability for the AMASS Tool Platform. It contains the information about the vision and conceptual basis for seamless interoperability in AMASS, which includes evidence management, tool integration, collaborative work, and tool quality characterisation. The deliverable also presents the specification of the modules and data models necessary to enable it.

A wide range of seamless interoperability means have been presented, from generic tool integration approaches such as OSLC KM to specific approaches such as those for Papyrus. This is necessary to meet different needs from different stakeholders and usage scenarios for CPS assurance and certification. Generic approaches are necessary to provide real seamless integration, but in some situations ad-hoc integration is the solution because of the constraints that some tools impose currently. Tool integration services for V&V are also envisioned and central for CPS assurance and certification, as well as integration with specific tools (e.g. Safety Architect and Cyber Architect). The means for collaborative work have focused on collaborative real-time model editing and automatic translations. Other important means for seamless interoperability include traceability approaches and evidence change impact analysis.

The components for seamless interoperability refine those in the ARTA related to this specific AMASS area, presented in D2.3 [10] and D2.4 [11]. The components provide services for Data Manager, Access Manager, Evidence Editor, Traceability Management, Impact Analysis, Collaborative Work, Toolchain Management, and Tool Connector. The components will be materialised in concrete implemented modules for D5.6.

The data models present the envisioned necessary information for evidence management, tool integration, data management, and security management. These models will be used as basis and revised for CACM specification.

Regarding the way forward for implementation, most of the requirements have already been implemented in the Core Prototype and Prototype P1. Nonetheless, their implementation could be refined and extended Prototype P2, and some features concerning seamless interoperability are still pending, e.g. the provision of enhanced generic means for tool integration in the AMASS Tool Platform. Most functionality will be provided in the OpenCert environment, but some will be provided in external tools.

References (*)

- [1] AdaCore: GNATcheck tool. <https://www.adacore.com/gnatpro/toolsuite/gnatcheck>
- [2] Agosense: Agosense Symphony. <http://www.agosense.com/english/products/agosensesymphony/agosensesymphony>
- [3] Alvarez-Rodríguez, J.M., Labra-Gayo, J., Ordoñez de Pablos, P.: Leveraging Semantics to Represent and Compute Quantitative Indexes: The RDFIndex Approach. Metadata and Semantics Research, vol. 390, E. Garoufallou and J. Greenberg, Eds. Springer International Publishing, pp. 175–187 (2013)
- [4] Alvarez-Rodríguez, J.M., Gayo, J.E.L., Silva, F. A. C., Alor-Hernández, G., Sánchez, C., Luna, J. A. G.: Towards a Pan-European E-Procurement Platform to Aggregate, Publish and Search Public Procurement Notices Powered by Linked Open Data: the Moldeas Approach. Int. J. Softw. Eng. Knowl. Eng. (IJSEKE) 22(3): 365–384 (2012)
- [5] Alvarez-Rodríguez, J. M., Llorens, J., Alejandres, M., Fuentes, J.: OSLC-KM: A knowledge management specification for OSLC-based resources. INCOSE Int. Symp. 25(1): 16–34 (2015)
- [6] AMALTHEA project: <http://www.amalthea-project.org/>
- [7] AMASS project: D1.1 - Case studies description and business impact. http://www.amass-ecsel.eu/sites/amass.drupal.pulsartecnalia.com/files/documents/D1.1_Case-studies-description-and-business-impact_AMASS_Final.pdf (2017)
- [8] AMASS project: D2.1 - Business cases and high-level requirements. http://www.amass-ecsel.eu/sites/amass.drupal.pulsartecnalia.com/files/documents/D2.1_Business-cases-and-high-level-requirements_AMASS_final.pdf (2017)
- [9] AMASS project: D2.2 - AMASS reference architecture (a) (2016)
- [10] AMASS project: D2.3 - AMASS reference architecture (b) (2017)
- [11] AMASS project: D2.4 - AMASS reference architecture (c) https://www.amass-ecsel.eu/sites/amass.drupal.pulsartecnalia.com/files/documents/D2.4_AMASS-reference-architecture-%28c%29_AMASS_Final.pdf (2018)
- [12] AMASS project: D3.3 - Design of the AMASS tools and methods for architecture-driven assurance (b) 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 (2018)
- [13] AMASS project: D3.5 - Prototype for Architecture-Driven Assurance (b) https://www.amass-ecsel.eu/sites/amass.drupal.pulsartecnalia.com/files/documents/D3.5_Prototype-for-architecture-driven-assurance-%28b%29_AMASS_Final.pdf (2017)
- [14] AMASS Project: D3.7 - Methodological Guide for Architecture-Driven Assurance (a) (2017)
- [15] AMASS project: D4.3 Design of the AMASS tools and methods for multiconcern assurance (b) 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 (2018)
- [16] AMASS project: D5.1 Baseline and Requirements for Seamless Interoperability. http://amass-ecsel.eu/sites/amass.drupal.pulsartecnalia.com/files/documents/D5.1_Baseline-and-Requirements-for-Seamless-Interoperability_AMASS_Final.pdf (2016)
- [17] AMASS project: D5.2 - Design of the AMASS tools and methods for seamless interoperability (a) (2017)
- [18] AMASS project: D5.4 - Prototype for seamless interoperability (a) https://www.amass-ecsel.eu/sites/amass.drupal.pulsartecnalia.com/files/documents/D5.4_Prototype-for-seamless-interoperability-%28a%29_AMASS_Final.pdf (2017)

-
- [19] AMASS project: D5.5 - Prototype for seamless interoperability (b) https://www.amass-ecsel.eu/sites/amass.drupal.pulsartecnalia.com/files/documents/D5.5_Prototype-for-seamless-interoperability-%28b%29_AMASS_Final.pdf (2017)
- [20] AMASS project: D5.6 - Prototype for seamless interoperability (c) (2018)
- [21] Arief, B., Adzmi, M. A. B., Gross, T.: Understanding cybercrime from its stakeholders' perspectives: Part 1—attackers. *IEEE Security Privacy* 13(1): 71–76 (2015)
- [22] Baclawski, K., Kokar, M. M., Waldinger, R. J., Kogut, P. A.: Consistency Checking of Semantic Web Ontologies. *International Semantic Web Conference*, pp. 454–459 (2002)
- [23] Baker, T., Bechhofer, S., Isaac, A., Miles, A., Schreiber, G., Summers, E.: Key choices in the design of Simple Knowledge Organization System (SKOS). *Web Semant. Sci. Serv. Agents World Wide Web* 20: 35–49 (2013)
- [24] Bender, M., Maibaum, T., Lawford, M., Wassyn, A.: Positioning Verification in the Context of Software/System Certification. *Electronic Communications of the EASST* 46 (2012)
- [25] Benjamins, V. R., Fensel, D., Gómez-Pérez, A.: Knowledge Management through Ontologies. *PAKM* 1998
- [26] Berners-Lee, T.: Linked Data (2006)
- [27] Bizer, C., Cyganiak, R.: Quality-driven information filtering using the WIQA policy framework. *Web Semant. Sci. Serv. Agents World Wide Web* 7(1): 1–10 (2009)
- [28] Bizer, C., Heath, T., Berners-Lee, T.: Linked Data - The Story So Far. *Int. J. Semantic Web Inf. Syst.* 5(3): 1–22 (2009)
- [29] Boneva, I., Gayo, J. E. L., Hym, S., Prud'hommeau, E. G., Solbrig, H. R., Staworko, S.: Validating RDF with Shape Expressions. *CoRR*, vol. abs/1404.1270 (2014)
- [30] Boley, H., Hallmark, G., Kifer, M., Paschke, A., Polleres, A., Reynolds, D. (eds.): *RIF Core Dialect (Second Edition)*. W3C Recommendation (2013)
- [31] Brickley, D., Guha, R. V. (eds.): *RDF Schema 1.1*. W3C Recommendation (2014)
- [32] Buchgeher, G., Weinreich, R.: Automatic tracing of decisions to architecture and implementation. 9th Working IEEE/IFIP Conference on Software Architecture, WICSA '11
- [33] Capra tool: <https://projects.eclipse.org/proposals/capra>
- [34] Castañeda, V., Ballejos, L., Caliusco, L., Galli, R.: The Use of Ontologies in Requirements Engineering. *GJRE* 10(6) (2010)
- [35] Cyber Architect tool: <https://www.all4tec.net/cyber-architect>
- [36] Cyganiak, R., Reynolds, D.: The RDF Data Cube Vocabulary. W3C, W3C Recommendation (2014)
- [37] Coyle, K., Baker, T.: Dublin Core Application Profiles Separating Validation from Semantics. W3C, (2013)
- [38] Davis, R., Shrobe, H., Szolovits, P.: What is a knowledge representation? *AI Mag.* 14(1): 17 (1993)
- [39] de la Vara, J.L., Borg, M., Wnuk, K., Moonen, L.: An Industrial Survey on Safety Evidence Change Impact Analysis Practice. *IEEE Transactions on Software Engineering* 42(12): 1095 – 1117 (2016)
- [40] Díaz, I., Llorens, J., Genova, G., Fuentes, J.M.: Generating domain representations using a relationship model. *Inf. Syst.* 30(1): 1–19 (2005)
- [41] Eclipse CDO: CDO Model Repository Documentation. <http://download.eclipse.org/modeling/emf/cdo/drops/R20170614-0227/help/index.html>
- [42] Eclipse CDO: Security Manager. https://wiki.eclipse.org/CDO/Security_Manager
- [43] Eclipse Lyo: Toolchain Modelling and Code Generation Workshop. <https://wiki.eclipse.org/Lyo/ToolchainModellingAndCodeGenerationWorkshop>
- [44] Ellis, C. A., Gibbs, S. J.: Concurrency control in groupware systems. *ACM SIGMOD Conf. on Management of data*, pp. 399 – 407 (1989)

- [45] Gallina, B., Nyberg, M.: Reconciling the ISO 26262-compliant and the Agile Documentation Management in the Swedish Context. Critical Automotive applications: Robustness & Safety (CARS) (2015)
- [46] Gallina, B., Castellanos Ardila, J. P., Nyberg, M.: Towards Shaping ISO 26262-compliant Resources for OSLC-based Safety Case Creation. Critical Automotive applications: Robustness & Safety (CARS) (2016)
- [47] Gallina, B., Padira, K., Nyberg, M.: Towards an ISO 26262-compliant OSLC-based tool chain enabling continuous self-assessment. 10th International Conference on the Quality of Information and Communications Technology- Track: Quality Aspects in Safety Critical Systems (QUATIC) (2016)
- [48] Gallina, B.: Towards an ISO 26262-compliant OSLC-based Tool Chain Enabling Continuous Self-assessment. <http://safety.addalot.se/upload/2017/2-3-1%20Gallina.pdf>
- [49] Gallina, B., Nyberg, M.: Pioneering the Creation of ISO 26262-compliant OSLC-based Safety Cases. WoSoCer 2017
- [50] Gašević, D., Devedžić, V., Djuric, D.: Model Driven Architecture and Ontology Development. Springer (2006)
- [51] Gayo, J. E. L., Prud'hommeaux, E., Boneva, I., Staworko, S., Solbrig, H. R., Hym, S.: Towards an RDF Validation Language Based on Regular Expression Derivatives. Workshops of the EDBT/ICDT 2015 Joint Conference (EDBT/ICDT), pp. 197–204 (2015)
- [52] Génova, G., Fuentes, J. M., Morillo, J. L., Hurtado, O., Moreno, V.: A framework to measure and improve the quality of textual requirements. Requir Eng 18(1): 25–41 (2013)
- [53] Groza, T., Handschuh, S., Clark, T., Buckingham Shum, S., de Waard, A.: A short survey of discourse representation models (2009)
- [54] Guo, M., Wang, J.A.: An ontology-based approach to model common vulnerabilities and exposures in information security. ASEE Southeast Section Conference 2009
- [55] Hammad, M., Collard, M. L., Maletic, J. I.: Automatically identifying changes that impact code-to-design traceability during evolution. Software Quality Control, 19(1): 35–64 (2011)
- [56] Hausenblas, M., Villazón-Terrazas, B., Hyland, B.: GLD Life cycle. W3C, W3C Government Linked Data Group (2011)
- [57] Hayes, P.: RDF Semantics. World Wide Web Consortium (2004)
- [58] Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S.: OWL 2 Web Ontology Language Primer. World Wide Web Consortium, W3C Recommendation (2009)
- [59] Hogan, A., Umbrich, J., Harth, A., Cyganiak, R., Polleres, A., Decker, S.: An empirical survey of Linked Data conformance. Web Semant. Sci. Serv. Agents World Wide Web 14: 14–44 (2012)
- [60] Hohpe, G., Woolf, B.: Enterprise integration patterns: designing, building, and deploying messaging solutions. Addison-Wesley (2004)
- [61] Horrocks, I., Parsia, B., Patel-Schneider, P., Hendler, J.: Semantic web architecture: Stack or two towers? Principles and practice of semantic web reasoning. Springer, pp. 37–41 (2005)
- [62] HP: PLM. <http://www8.hp.com/us/en/business-services/it-services.html?compURI=1830395>
- [63] Hull, R., King, R.: Semantic database modeling: Survey, applications, and research issues. ACM Comput. Surv. 19(3): 201–260 (1987)
- [64] Hyland, B., Terrazas, B.V.: Cookbook for Open Government Linked Data. W3C, W3C Task Force-Government Linked Data Group (2011)
- [65] IEC: Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems (IEC 61508), 2nd ed. (2011)
- [66] ISO: Road vehicles — Functional safety (ISO 26262) (2011)
- [67] Jazz Platform. <https://jazz.net/>

-
- [68] Javed, M. A., Stevanetic, S., Zdun, U.: Towards a pattern language for construction and maintenance of software architecture traceability links. 21st European Conference on Pattern Languages of Programs (EuroPlop 2016)
- [69] Javed, M. A., UI Muram, F., Zdun, U.: On-Demand Automated Traceability Maintenance and Evolution. 17th International Conference on Software Reuse (ICSR 2018)
- [70] Juez, G., Amparan, E., Lattarulo, R., Ruíz, A., Pérez, J., Espinoza, H.: Early safety assessment of automotive systems using Sabotage simulation-based fault injection framework. SAFECOMP 2017
- [71] Krafzig, D., Banke, K., Slama, D.: Enterprise SOA: service-oriented architecture best practices. Prentice Hall Professional (2005)
- [72] Kontokostas, D., et al.: Test-driven evaluation of linked data quality. 23rd International World Wide Web Conference (WWW '14), pp. 747–758 (2014)
- [73] Kossmann, M., Wong, R., Odeh, M., Gillies, A.: Ontology-driven Requirements Engineering: Building the OntoREM Meta Model (2008)
- [74] Knublauch, H., Hendler, J.A., Idehen, K.: SPIN - Overview and Motivation. W3C, Member Submission (2011)
- [75] Llorens, J., Morato, J., Genova, G.: RSHP: an information representation model based on relationships. Soft Computing in Software Engineering 159: 221–253 (2004)
- [76] Lucia, A. D., Fasano, F., Oliveto, R., Tortora, G. Recovering traceability links in software artifact management systems using information retrieval methods. ACM Trans. Softw. Eng. Methodol. 16 (4): 3 (2007)
- [77] Mallea, A., Arenas, M., Hogan, A., Polleres, A.: On blank nodes. The Semantic Web–ISWC 2011. Springer, pp. 421–437 (2011)
- [78] Manikas, K., Hansen, K.M.: Software ecosystems – A systematic literature review. J. Syst. Softw. 86(5): 1294–1306 (2013)
- [79] Mendieta, R., de la Vara, J.L., Llorens, J., Álvarez-Rodríguez, J.M.: Towards Effective SysML Model Reuse. 5th International Conference on Model-Driven Engineering and Software Development, pp. 536–541 (2017)
- [80] Mäder, P., Gotel, O.: Towards automated traceability maintenance. J. Syst. Softw. 85(10): 2205–2227 (2012)
- [81] Nair, S., de la Vara, J.L., Sabetzadeh, M., Briand, L.: An Extended Systematic Literature Review on Provision of Evidence for Safety Certification. Information and Software Technology 56(7): 689–717 (2014)
- [82] Nguyen, N., Bodenreider, O., Sheth, A.: Don't like RDF reification?: making statements about statements using singleton property (2014)
- [83] Noy, N., Rector, A.: Defining N-ary Relations on the Semantic Web. W3C Working Group (2006)
- [84] OpenCert: <https://www.polarsys.org/opencert/>
- [85] OPENCROSS project: <http://www.opencross-project.eu/>
- [86] OPENCROSS project: D4.4 - Common Certification Language: Conceptual Model. http://www.opencross-project.eu/sites/default/files/D4.4_v1.5_FINAL.pdf (2015)
- [87] OPENCROSS project: D6.6 - Implementation of the evidence management service infrastructure. http://www.opencross-project.eu/sites/default/files/D6.6_Implementation_of_the_evidence_management_service_infrastructure_V1.4.pdf (2015)
- [88] OSLC. <http://www.oasis-osl.org/>
- [89] Polarsys: <https://www.polarsys.org/>
- [90] Powers, S.: Practical RDF. O'Reilly (2003)
- [91] PTC: Integrity. <http://www.ptc.com/application-lifecycle-management/integrity>

- [92] Rodríguez, M. G., Alvarez-Rodríguez, J. M., Muñoz, D.B., Paredes, L.P., Gayo, J.E.L., de Pablos, P. O.: Towards a Practical Solution for Data Grounding in a Semantic Web Services Environment. JUCS 18(11):1576–1597 (2012)
- [93] Roman, D., et al.: Web service modeling ontology. Appl. Ontol. 1(1): 77–106 (2005)
- [94] Ryman, A. G., Hors, A. L., Speicher, S.: OSLC Resource Shape: A language for defining constraints on Linked Data. LDOW (2013)
- [95] SafeCer project: <https://artemis-ia.eu/project/40-nsafecer.html>
- [96] Safety Architect tool: <https://www.all4tec.net/safety-architect>
- [97] Siemens: Teamcenter. http://www.plm.automation.siemens.com/en_us/products/teamcenter/
- [98] Silva, A.: Tools Exhibits. UML Modeling Languages and Applications Satellite Activities (2004)
- [99] Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. J Web Sem 5(2): 51–53 (2007)
- [100] Smyth, V.: Software vulnerability management: how intelligence helps reduce the risk. Network Security 3: 10 – 12 (2017)
- [101] Stardog. <http://stardog.com/>
- [102] Sun, C., Wen, H., Fan, H.: Operational Transformation for Orthogonal Conflict Resolution in Real-time Collaborative 2D Editing Systems. ACM Conf. on Computer Supported Cooperative Work (2012)
- [103] The Reuse Company: Knowledge Manager (KM). <https://www.reusecompany.com/knowledgemanager>
- [104] The Reuse Company: Requirements Quality Analyzer (RQA). <https://www.reusecompany.com/requirements-quality-analyzer>
- [105] Thüm, T., Apel, S., Kästner, C., Schaefer I., Saake, G.: A Classification and Survey of Analysis Strategies for Software Product Lines. ACM Comput. Surv. 47(1): 1–45 (2014)
- [106] WSO2. <http://wso2.com/>
- [107] Zou, X., Settini, R., Cleland-Huang, J.: Improving automated requirements trace retrieval: A study of term-based enhancement methods. Empirical Softw. Eng. 15 (2): 119–146 (2010)

Appendix A. Tools in the AMASS Case Studies

This appendix presents the tools that have been reported in D1.1 [7] for the AMASS case studies. There are 11 case studies:

- CS1: Industrial and Automation Control Systems
- CS2: Advanced driver assistance function with electric vehicle
- CS3: Collaborative automated fleet of vehicles
- CS4: Design and safety assessment of on-board software applications in Space Systems
- CS5: Platform Screen Doors Controller
- CS6: Automatic Train Control Formal Verification
- CS7: Safety Assessment of Multi-Modal Interactions in Cockpits
- CS8: Telematics Function
- CS9: Safety-Critical SW Lifecycle of a Monitoring System for NavAid (ATM domain)
- CS10: Certification basis to boost the usage of MPSoC architectures
- CS11: Design and efficiency assessment of model-based Attitude and Orbit Control software development

According to the case study descriptions, the tools with an 'X' in Table 7 are currently used in practice, whereas the tool with and '*' might be used when conducting the case studies in AMASS. All these tools are candidate to interoperate with the AMASS Platform or between them.

Table 7. Tools used in the AMASS Case Studies

Tool	Case Study										
	1	2	3	4	5	6	7	8	9	10	11
Orcad CIS (Cadence)	X										
HDL (Cadence)	X										
Allegro (Cadence)	X										
Tina (Texas Instrument)	X										
Visual Basic .Net (Microsoft)	X										
AVR Studio (Atmel)	X										
CodeWarrior (Freescale)	X										
STM Studio (ST)	X										
Coocox	X										
Quartus (Altera)	X										
MySQL	X										
Eclipse	X			*					*	X	
Visual Studio (Microsoft)	X		X								
Work bench 3.1 (Wind River)	X										
MS Excel	X		X				X	X			
CHESS	*			*	*		*		*	*	*
OCRA	*			*	*		*		*	*	*
MATLAB / Simulink		X	X				X				X
MATLAB Embedded Coder		X	X								X
Medini analyze (AMT)		X	*								
TESTONA		X									
Rhapsody		*	*	*					X	X	
Enterprise Architect		*	*	*						X	
MS Office		*			X				*		
Savona			X								

Visio			X	*						X	
WEFACT			*								
PTC Integrity			*								
Farkle			*								
DOORS				X			X		X	X	X
HRT-UML				X							
Borland Together				X							
Jira				*			X	X		X	
SVN				*			X	X		X	
PIC32 Microchip IDE					X						
Atelier B					X	*					
FIDES					X						
Frama-C					*						
Safety Architect					*						
Why3					*						
MaTeLo					*						
MS Word							X	X			
HAM							X				
HiLiTE							X				
ForReq							*				
DIVINE							*				
NuSMV							*				
NuXmv	*			*	*		*		*	*	*
Acacia							*				
RAT							*				
RQA							*				
xSAP	*			*	*		*		*	*	*
Git								X		X	
Doxygen								X			
CMake								X			
Jenkins								X			
CTest								X			
CUnit								X			
PC-Lint								X			
IVVV Manager									X		
ClearCase									X		
Bugzilla									X		
Melody Advance										X	
Melody CCM										X	
Thales Control										X	
Satsim											X
Alten Code Coverage											*
Sabotage			X								

Appendix B. Changes with respect to D5.2 (*)

New Sections:

Section	Title
3.1.1.6	Delegated Operations
3.1.6	Integration with Safety and Security Analysis Tools
3.1.7	Integration with the Sabotage Simulation-Based Fault Injection Tool
3.1.8	Integration in the Farkle Tool
3.1.9	Generic REST-API Adapter Concept for Seamless Interoperability
3.1.12	Knowledge-Centric Automated Traceability
3.1.13	On-Demand Automated Traceability Maintenance and Evolution
3.1.16	Management of V&V evidence
3.1.17	Security Management
3.1.18	Data Management

Sections whose number has changed:

Former Section No.	New Section No.	Title
3.1.6	3.1.10	Collaborative Real-Time Model Editing
3.1.7	3.1.11	Seamless Tracing
3.1.8	3.1.15	Evidence Change Impact Analysis
3.1.9	3.1.14	Automatic Translations for Collaborative Work

Modified Sections:

Section	Title	Change
3.1.5	V&V Tool Integration	Added paragraph on sync/async requests. In addition, V&V Manager was extended to improve scalability and expressiveness of the requirement semantic analysis technology. This will allow to find more requirement defects even for a larger number of requirements, especially for consistency and non-redundancy checking. Remus2 tool from Masaryk University was integrated.
3.2.6	Collaborative Work Components	Information added about data mining support
5	Way Forward for the Implementation	Implementation status updated
6	Conclusion	Conclusions updated
	References	New references added

Removed Sections:

Section	Title	Change
Appendix B	Stakeholders and Stakeholder Collaborations in the AMASS Case Studies	The information has evolved since the preparation of D5.2 and it is included in WP1 and WP2 deliverables.