Prototype for seamless interoperability (b)

**D5.5**

<table>
<thead>
<tr>
<th>Work Package:</th>
<th>WP5 Seamless Interoperability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dissemination level:</td>
<td>PU = Public</td>
</tr>
<tr>
<td>Status:</td>
<td>Final</td>
</tr>
<tr>
<td>Date:</td>
<td>30 November 2017</td>
</tr>
<tr>
<td>Responsible partner:</td>
<td>Luis M. Alonso (TRC)</td>
</tr>
<tr>
<td>Contact information:</td>
<td><a href="mailto:luis.alonso@reusecompany.com">luis.alonso@reusecompany.com</a></td>
</tr>
<tr>
<td>Document reference:</td>
<td>AMASS_D5.5_WP5_TRC_V1.0</td>
</tr>
</tbody>
</table>

**PROPRIETARY RIGHTS STATEMENT**

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

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

<table>
<thead>
<tr>
<th>Names</th>
<th>Organisation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Luis M. Alonso, Borja López</td>
<td>The REUSE Company</td>
</tr>
<tr>
<td>Jose Luis de la Vara, Jose María Álvarez, Eugenio Parra, Roy Mendieta, Francisco Rodríguez</td>
<td>Universidad Carlos III de Madrid</td>
</tr>
<tr>
<td>Ángel López, Alejandra Ruiz</td>
<td>Tecnalia Research &amp; Innovation</td>
</tr>
<tr>
<td>Pietro Braghieri, Stefano Tonetta, Alberto DeBiasi</td>
<td>Fondazione Bruno Kessler</td>
</tr>
<tr>
<td>Stefano Puri</td>
<td>Intecs</td>
</tr>
<tr>
<td>Tomáš Kratochvila</td>
<td>Honeywell</td>
</tr>
<tr>
<td>Ivana Černá</td>
<td>Masaryk University</td>
</tr>
<tr>
<td>Jan Mauersberger</td>
<td>Ansys medini Technologies</td>
</tr>
</tbody>
</table>

## Reviewers

<table>
<thead>
<tr>
<th>Names</th>
<th>Organisation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frank Badstuebner (Peer reviewer)</td>
<td>Infineon</td>
</tr>
<tr>
<td>Marc Sango (Peer reviewer)</td>
<td>ALL4TEC</td>
</tr>
<tr>
<td>Cristina Martinez (Quality Manager)</td>
<td>Tecnalia Research &amp; Innovation</td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS

Abbreviations and Definitions .......................................................................................................................... 5

Executive Summary ............................................................................................................................................. 6

1. Introduction ................................................................................................................................................... 8

2. Implemented Functionality .......................................................................................................................... 10
   2.1 Scope ..................................................................................................................................................... 10
   2.2 Implemented Requirements .................................................................................................................. 10
      2.2.1 ‘Characterise Artefact’ with OpenCert ......................................................................................... 10
      2.2.2 ‘Link Artefact with External Tool’ with OpenCert ......................................................................... 12
      2.2.3 ‘Specify Artefact Lifecycle’ with OpenCert ............................................................................... 13
      2.2.4 ‘Evaluate Artefact’ with OpenCert ............................................................................................... 14
      2.2.5 ‘Specify Process Information for Artefacts’ with OpenCert ....................................................... 14
      2.2.6 ‘Conduct Impact Analysis of Assurance Asset Change’ with OpenCert .................................... 15
      2.2.7 ‘Specify Traceability between Assurance Assets’ with OpenCert .............................................. 16
      2.2.8 ‘Specify Traceability between Assurance Assets’ with Capra ................................................... 16
      2.2.9 ‘Specify Tool Connection Information’ with OpenCert ............................................................. 18
      2.2.10 ‘Specify Tool Connection Information’ for OSLC-KM-based Integration ................................. 18
      2.2.11 ‘Specify Tool Connection Information’ for Integration with V&V Manager .............................. 21
      2.2.12 ‘Specify Tool Connection Information’ for Integration of CHESS and V&V Tools .................. 22
      2.2.13 ‘Concurrent Assurance Information Edition’ with Web-based Technologies .......................... 25
      2.2.14 ‘Concurrent Assurance Information Edition’ with Data Mining Technologies ..................... 27
   2.3 Installation and User Manuals .................................................................................................................. 28

3. Implementation Description .......................................................................................................................... 30
   3.1 Implemented Modules ............................................................................................................................. 30
   3.2 Implemented Metamodel ......................................................................................................................... 32
   3.3 Source Code Description for the AMASS Tool Platform ..................................................................... 33
   3.4 Source Code Description for External Tools ........................................................................................ 35
      3.4.1 Seamless Interoperability Features in RQA ................................................................................. 35

4. Conclusion ..................................................................................................................................................... 41

References .......................................................................................................................................................... 42
List of Figures

Figure 1. AMASS Building blocks .......................................................................................... 8
Figure 2. Functional decomposition for the AMASS platform .............................................. 10
Figure 3. Artefact definition creation .................................................................................... 11
Figure 4. Artefact data specification ...................................................................................... 11
Figure 5. Use of SVN repository as artefact repository ......................................................... 12
Figure 6. Resource specification for an artefact ................................................................. 12
Figure 7. Resource properties ............................................................................................... 13
Figure 8. Artefact event properties ....................................................................................... 13
Figure 9. Artefact evaluation properties ............................................................................... 14
Figure 10. Process model .................................................................................................... 15
Figure 11. Activity data ........................................................................................................ 15
Figure 12. Modification event of an artefact ....................................................................... 16
Figure 13. Impact analysis information ................................................................................. 16
Figure 14. Advanced CAPRA trace creation view (drop sensitive) ....................................... 17
Figure 15. Tracing a claim to a contract .............................................................................. 18
Figure 16. OSLC-KM Importing an Evidence Model from a model file ............................. 19
Figure 17. Fragment of a Papyrus model to be imported ..................................................... 19
Figure 18. New evidence model from a Papyrus model ....................................................... 19
Figure 19. RQA Connection Window .................................................................................. 20
Figure 20. SysML (Papyrus subtype) .................................................................................. 20
Figure 21. Papyrus mappings .............................................................................................. 21
Figure 22. V&V Manager integration ..................................................................................... 22
Figure 23. FBK Tool Integration via files .............................................................................. 23
Figure 24. FBK Tool Integration via OSLC Automation ....................................................... 23
Figure 25. FBK Tool Adapters Configuration ..................................................................... 24
Figure 26. Contract and Behaviour Verification context menu .......................................... 24
Figure 27. Contract and Behaviour Verification main menu .............................................. 25
Figure 28. FBK Tool Automation Plan example .................................................................. 25
Figure 29. Ultimate picture of collaborative work using rich and web clients ..................... 26
Figure 30. Screenshot of the current version of the web-based tool for collaborative model editing ................................................................. 26
Figure 31. Screenshot of the Indexing configuration preferences ....................................... 27
Figure 32. Screenshot of context menu to index data .......................................................... 27
Figure 33. Screenshot of the Data Mining platform for collaborative work ....................... 28
Figure 34. Screenshot of the Kibana Discovery tool ............................................................ 28
Figure 35. Platform management modules ......................................................................... 31
Figure 36. Evidence management module ......................................................................... 31
Figure 37. Assurance traceability modules ........................................................................ 32
Figure 38. Tool integration modules ................................................................................... 32
Figure 39. Excerpt of artefact information in the CCL ......................................................... 33
Figure 40. Evidence management and System management plug-ins ............................... 34
Figure 41. Rqa.Face.OslocKm library ............................................................................. 37
Figure 42. Implementation of the OSLC-KM for RQS ....................................................... 37
Figure 43. OSLC-KM parsers and XSLT transformation files for Papyrus and Rhapsody .................................................................................................................. 38
Figure 44. Part of the Papyrus XSLT transformation to map requirements in the model to the OSLC-KM model instance .......................................................................................... 39
Figure 45. GUI to allow creation of XSLT files to customise the mapping of XML file nodes to elements in the OSLC-KM metamodel ............................................................... 40
### Abbreviations and Definitions

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>ARTA</td>
<td>AMASS Reference Tool Architecture</td>
</tr>
<tr>
<td>ASIL</td>
<td>Automotive Safety Integrity Level</td>
</tr>
<tr>
<td>CACM</td>
<td>Common Assurance and Certification Metamodel</td>
</tr>
<tr>
<td>CDO</td>
<td>Connected Data Objects</td>
</tr>
<tr>
<td>CCL</td>
<td>Common Certification Language</td>
</tr>
<tr>
<td>CPS</td>
<td>Cyber-Physical Systems</td>
</tr>
<tr>
<td>ECSEL</td>
<td>Electronic Components and Systems for European Leadership</td>
</tr>
<tr>
<td>EEF</td>
<td>Extended Editing Framework</td>
</tr>
<tr>
<td>EMF</td>
<td>Eclipse Model Framework</td>
</tr>
<tr>
<td>GSN</td>
<td>Goal Structuring Notation</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td>OPENCOSS</td>
<td>Open Platform for EvolutioNary Certification of Safety-critical Systems</td>
</tr>
<tr>
<td>OSLC</td>
<td>Open Services for Lifecycle Collaboration</td>
</tr>
<tr>
<td>OSLC-KM</td>
<td>OSLC for Knowledge Management</td>
</tr>
<tr>
<td>RQA</td>
<td>Requirements Quality Analyzer</td>
</tr>
<tr>
<td>RQS</td>
<td>Requirements Quality Suite</td>
</tr>
<tr>
<td>SACM</td>
<td>Structured Assurance Case Metamodel</td>
</tr>
<tr>
<td>SafeCer</td>
<td>Safety Certification of Software-Intensive Systems with Reusable Components</td>
</tr>
<tr>
<td>SVN</td>
<td>Apache Subversion</td>
</tr>
<tr>
<td>TRL</td>
<td>Technology Readiness Level</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>V&amp;V</td>
<td>Verification and Validation</td>
</tr>
<tr>
<td>WP</td>
<td>Work Package</td>
</tr>
<tr>
<td>XSLT</td>
<td>eXtensible Stylesheet Language Transformations</td>
</tr>
</tbody>
</table>
Executive Summary

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

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

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

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

The developed tools for the Prototype P1 support the following use cases:

- Characterise Artefact
- Link Artefact with External Tool
- Specify Artefact Lifecycle
- Evaluate Artefact
- Specify Process Information for Artefacts
- Specify Traceability between Assurance Assets
- Conduct Impact Analysis of Assurance Asset Change
- Specify Tool Connection Information
- Concurrent Assurance Information Edition

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

D5.5 relates to other implementation-related AMASS deliverables:

- Installable AMASS Tool Platform for Prototype P1
- User manuals and installation instructions
- Source code description

In addition, D5.5 is related to the following AMASS deliverables:

- D2.1 (Business cases and high-level requirements) includes the requirements that have been implemented in D5.5.
- D2.2 (AMASS reference architecture (a)) and D2.3 (AMASS reference architecture (b)) present the abstract architecture based on which D5.5 has been created.
- D2.7 (Integrated AMASS platform (b)) reports the results from validating the implementation described in D5.5.
D5.1 (Baseline requirements for seamless interoperability) reviews the main background on seamless interoperability for AMASS and proposes a way forward. D5.5 corresponds to the realisation of this way forward as of October 2017.

D5.4 (Prototype for seamless interoperability (a)) describes the first version of the Seamless Interoperability support in the AMASS Tool Platform.

D5.6 (Prototype for seamless interoperability (c)) will describe the third version of the seamless interoperability support in the AMASS Tool Platform.
1. Introduction

The AMASS approach focuses on the development and consolidation of an open and holistic assurance and certification framework for CPS, which constitutes the evolution of the OPENCOSS [15] and SafeCer [19] approaches towards an architecture-driven, multi-concern assurance, reuse-oriented, and seamlessly interoperable tool platform.

The expected tangible AMASS results are:

a) The **AMASS Reference Tool Architecture**, which will extend the OPENCOSS and SafeCer conceptual, modelling and methodological frameworks for architecture-driven and multi-concern assurance, as well as for further cross-domain and intra-domain reuse capabilities and seamless interoperability mechanisms (based on OSLC specifications [17]).

b) The **AMASS Open Tool Platform**, which will correspond to a collaborative tool environment supporting CPS assurance and certification. This platform represents a concrete implementation of the AMASS Reference Tool Architecture, with a capability for evolution and adaptation, which will be released as an open technological solution by the AMASS project. AMASS openness is based on both standard OSLC APIs with external tools (e.g. engineering tools including V&V tools) and on open-source release of the AMASS building blocks.

c) The **Open AMASS Community**, which will manage the project outcomes, for maintenance, evolution and industrialization. The Open Community will be supported by a governance board, and by rules, policies, and quality models. This includes support for AMASS base tools (tool infrastructure for database and access management, among others) and extension tools (enriching AMASS functionality). As Eclipse Foundation is part of the AMASS consortium, the Polarsys/Eclipse community ([www.polarsys.org](http://www.polarsys.org)) is a strong candidate to host AMASS Open Tool Platform.

To achieve the AMASS results, as depicted in Figure 1, the multiple challenges and corresponding scientific and technical project objectives are addressed by different work-packages.
Since AMASS targets high-risk objectives, the AMASS Consortium decided to follow an incremental approach by developing rapid and early prototypes. The benefits of following a prototyping approach are:

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

AMASS has planned three prototype iterations:

1. During the **first prototyping** iteration (Prototype Core), the AMASS Platform Basic Building Blocks (see [2]), will be aligned, merged and consolidated at TRL4\(^1\).
2. During the **second prototyping** iteration (Prototype P1), the AMASS-specific Building Blocks will be developed and benchmarked at TRL4; this comprises the blue basic building blocks as well as the green building blocks (Figure 1). Regarding seamless interoperability, in this second prototype, the specific building blocks will provide advanced functionalities regarding tool integration, collaborative work, and tool quality characterisation and assessment.
3. Finally, at the **third prototyping** iteration (Prototype P2), all AMASS building blocks will be integrated in a comprehensive toolset operating at TRL5. Functionalities specific for seamless interoperability developed for the second prototype will be enhanced and integrated with functionalities from other technical work packages.

Each of these iterations has the following three prototyping dimensions:

- **Conceptual/research development**: development of solutions from a conceptual perspective.
- **Tool development**: development of tools implementing conceptual solutions.
- **Case study development**: development of industrial case studies (see D1.1 [1]) using the tool-supported solutions.

As part of the Prototype Core, WP5 was responsible for consolidating the previous works on specification of evidence characteristics, handling of evidence evolution, and specification of evidence-related information (e.g. process information) in order to design and implement the basic building block called “Evidence Management” (Figure 1). In addition, WP5 was responsible for the implementation of the “Access Manager” and “Data Manager” basic building blocks. Nonetheless, the functionality of these latter blocks is used not only in WP5, but in all the WPs, e.g. for data storage and access (of system components, of assurance cases, of standards’ representations, etc.). For P1, WP5 has refined and extended the existing implementation with support for specific seamless interoperability based on the development of new functionality, and not only the integration of available tools.

This deliverable reports the **tool development results of the “Evidence Management”, “Access Manager”, “Data Manager”, “Tool Integration Management”, and “Collaborative Work Management” building blocks**. It presents in detail the design of the functionality implemented in the AMASS Tool Platform, the building blocks’ software architecture, the technology used, and source code references. The design is based on the investigated state of the art and state of practice approaches presented in D5.1 [8], and on the ARTA specification in D2.2\(^2\) [2], and D2.3 [3]. Their gaps were identified and analysed to determine a way forward for seamless interoperability, enabling the formulation of requirements to achieve the interoperability vision of AMASS. This vision covers tool integration, collaborative work, and tool quality assessment and characterisation.

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

---


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

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

2.1 Scope

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

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

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

Figure 2. Functional decomposition for the AMASS platform

2.2 Implemented Requirements

The implemented requirements correspond to nine use cases specified in D2.3 [3]. The following subsections include a short description of how the implementation performed supports each use case, and the main tools and technologies supporting the use cases. Some use cases are supported by several tools and technologies. For example, there exist several means for tool integration in the AMASS Tool Platform.

Regarding the implemented requirements, an overview of the status of WP5 requirements is available in D5.2 [7]. A detailed analysis will be included in D5.6, as a reference of the final AMASS support for Seamless Interoperability.

2.2.1 ‘Characterise Artefact’ with OpenCert

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

**Figure 3. Artefact definition creation**

**Figure 4. Artefact data specification**
2.2.2 ‘Link Artefact with External Tool’ with OpenCert

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

Figure 5. Use of SVN repository as artefact repository

Figure 6. Resource specification for an artefact
2.2.3 ‘Specify Artefact Lifecycle’ with OpenCert

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

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

![Artefact evaluation properties](image)

**Figure 9.** Artefact evaluation properties

2.2.5 ‘Specify Process Information for Artefacts’ with OpenCert

Process-related artefact information is specified by means of process models (Figure 10). These models can contain information about activities, participants, persons, tools, organizations, and techniques involved in the processes of an assurance project. Artefacts can later be associated to these elements. For example, ‘activity artefacts’ is a set of activity data (Figure 11) with which the input and output artefact of an activity can be specified.
2.2.6 ‘Conduct Impact Analysis of Assurance Asset Change’ with OpenCert

When changes are made to artefacts and these changes result in modification events (Figure 12), the users can determine the impact of such changes in other artefacts and accept it or refuse it (Figure 13).
2.2.7 ‘Specify Traceability between Assurance Assets’ with OpenCert

OpenCert provides support to specify traceability between evidence artefacts as part of its functionality to characterise artefacts (see Section 2.2.1). More concretely, relationships can be created to specify artefact components with ‘ArtefactPart’ and any other type of relationship with ‘OwnedRel’.

2.2.8 ‘Specify Traceability between Assurance Assets’ with Capra

Capra Eclipse project\(^3\) offers a basic support for the creation, management and visualisation of trace links between resources within Eclipse. In the context of WP5, Capra basic support has been extended to support

\(^3\) [https://projects.eclipse.org/projects/modeling.capra](https://projects.eclipse.org/projects/modeling.capra)
reference to resources which are external to the Eclipse environment (e.g. external files, requirements modelled with DOORS, etc.) and to support references to Eclipse resources stored in CDO. The trace model was extended to have trace directions, that means a trace is a directed relationship, not undirected. The direction is required in AMASS to express upstream-downstream relationships and based on that, the calculation of change impact in upstream artefacts. Furthermore, the user interface was extended so that the creation of traces between internal artefacts and external artefacts is much easier. A new view is available now which can be simply filled in with drag & drop (URLs, files, objects) (Figure 14).

![Figure 14. Advanced CAPRA trace creation view (drop sensitive)](image)

In the context of WP3, Capra approach has been selected for the storage of the traceability links between system architectural entities, like components and contracts, and assurance related entities, like claims and evidences. A specific support/user interface is currently under development to assist the architect in the creation of the traceability links; in particular, by using this support, the architect will be allowed to create the kind of traceability links which are allowed by the system architecture (abstract) metamodel (see D3.2 [4], section 3.2.2.4).

Figure 15 shows an example of the aforementioned support: by selecting a contract in the system architecture model, the latter available through the Papyrus/CHESS editor, a dedicate tab (named OpenCert) is enabled in the properties view. The OpenCert tab allows to check the current assurance case entities already traced to the contract itself, and also allows to create new traceability links. For instance, a trace link between the selected contract and the Goal1 claim available in the assurance case model, showed in the left part of the figure, can be created by using the Claim table in the OpenCert tab. The link to be created will be automatically stored in the Capra model, by using the Capra API’s facilities. The possibility to retrieve existing traceability link associated to the selected architectural entity and the possibility to create new links, requires that the location of the Capra model has to be known by the tool (it can set in the CHESS preference page or by using some setting at the assurance project level, the definition of this part is ongoing).
2.2.9 ‘Specify Tool Connection Information’ with OpenCert

The default OpenCert support to specify tool connection information is presented in Section 2.2.2. The new support developed for Prototype P1 and that has been or will be integrated in OpenCert is presented in Sections 2.2.10, 2.2.11, and 2.2.12.

2.2.10 ‘Specify Tool Connection Information’ for OSLC-KM-based Integration

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

An example use case implemented for the AMASS platform is as follows. From the side of the AMASS Tool Platform, in the menu bar just select the “OSLC-KM” menu, and the option “Import evidence model from file” (Figure 16), then select a Papyrus file (Figure 17). As a result, its content is sent to a Requirements Quality Analyzer (RQA) web service that works as an OSLC producer. The web method returns the OSLC-KM instance, then the AMASS platform loads the model by the Java implementation of the OSLC-KM standards and maps its content to an Artefact Model inside the current assurance project (Figure 18).

On the other side, in RQA the creation of the OSLC-KM model can be parameterized by modifying the mapping between the Papyrus metamodel and the OSLC-KM metamodel. This can be done in RQA in the connection window, selecting a new OSLC-KM connection (Figure 19) and then, in the new window, selecting the Papyrus model from the file system (Figure 20). Finally, in the bottom part of this window, if the “Advance” configuration is selected, a new window will appear allowing to customize the mappings from the Papyrus model and the OSLC-KM instance created for it (Figure 21).
Figure 16. OSLC-KM Importing an Evidence Model from a model file

Figure 17. Fragment of a Papyrus model to be imported

Figure 18. New evidence model from a Papyrus model
Figure 19. RQA Connection Window

Figure 20. SysML (Papyrus subtype)
2.2.11 ‘Specify Tool Connection Information’ for Integration with V&V Manager

V&V Manager allows formal verification of the contracts by external V&V tools that are installed on remote verification servers.

The contracts (or individual formal properties) are selected for example from Block Definition Diagram or at the level of components, using the related contracts. The verification or validation is invoked using a contextual menu Validation → V/V Manager, as depicted in Figure 22.

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

**How to integrate a new V&V tool**

Install it on the verification servers and register it on the Proxygen or Apache-Tomcat server application. The server needs to know:

1. **The tool binary name to be executed** – only if it is different from the name stated in the OSLC Automation Plan.
2. **The tool parameters** – only if the parameters have to be handled differently than as command line arguments or as a content of a configuration file parameters.
3. **Artefacts under verification** (requirements, system architecture, system design) – only if the artefacts have to be handled differently than just to be passed as file arguments.

In summary, if the tool binary name, its parameters and the artefacts under verification could be passed to the command line tool in a standard way, the V&V tool does not have to be registered by the verification server application.
2.2.12 ‘Specify Tool Connection Information’ for Integration of CHESS and V&V Tools

CHESS has been extended and integrated in order to perform V&V activities on the models by using the FBK Tools. Currently two kind of tool adapters are available: the first one that invokes the FBK tools locally by passing the artefacts and the command via files. The second that does the same functionalities via the OSLC-Automation adapter.

Adapter to FBK Tool via files

The architecture of the integration towards FBK tools via files is depicted in Figure 23. The tool adapter takes in charge the request from CHESS, converts the model to the Verification tool format, sets up the artifacts and the commands files, sends them to the Verification Tools and in the end returns the result to CHESS, ready to be shown grafically.

Adapter to FBK Tool via OSLC

Figure 24 represents the same functionality using the OSLC approach. As mentioned above, here we choose to use the OSLC Automation Domain for the integration toward the Verification Tools. From the user side, the choice of the adapter is transparent in terms of functionalities, so the user can decide to ask for a specific validation regardless of where this validation is going to be performed (locally or remotely).

Adapters Configuration

The configuration of such adapters is available in the Preferences menu (Figure 25). The Tools Preference Page allows configuring both the local (via files) and OSCL tools adapters by specifying some parameters such as the executable path, the execution timeout, and the OSLC Service Provider catalogue end in the Service Provider instance.

The Verification actions can be executed on CHESS models and that can be invoked from both the main menu and context menu (Figure 26 and Figure 27). There are some functions for the contract based verification and other for behaviour model checking. The same functions can be invoked by selecting the component in the diagram.
In the OSLC approach, all the verification functions have been mapped on *AutomationPlan* instances. The adapter on the client side maps the required function to the corresponding Automation Plan, then instantiates the Automation Request setting up the parameter values in accordance with the plan. Just as an example, the Contract Refinement check is defined in the Service Provider catalogue (Figure 28).

**Figure 23. FBK Tool Integration via files**

**Figure 24. FBK Tool Integration via OSLC Automation**
Figure 25. FBK Tool Adapters Configuration

Figure 26. Contract and Behaviour Verification context menu
Figure 27. Contract and Behaviour Verification main menu

Figure 28. FBK Tool Automation Plan example

2.2.13 ‘Concurrent Assurance Information Edition’ with Web-based Technologies

An ultimate goal for the WP5 in AMASS is to provide means to support collaboratively work on the same document or model at the same time without locking. To offer a seamless experience to the user, the
editing shall work both, in rich client (and tools) based on eclipse as well as in web based clients (Figure 29). In this prototype, the collaboration between two web based clients was the target.

The solution (Figure 30) is based on a NodeJS based server that handles all “Operational Transformations”, i.e. small pieces of change information (so called “mutations”) that are sent by all attached clients and that the server must bring into order, apply them and send them to all attached clients so they can be “eventually” consistent, meaning that all clients are up to date at a given point in time. The server was implemented using purely open-source software as ShareDB / ShareJS. As a proof of concept, a simple GSN editor was build (again) using open-source software as Draw.IO and mxGraph. Once the server is running, the clients may actively connect and after that all modifications done by any of the clients will appear also at other attached clients. Both, the client but also the server were built using the Node.js based build environment and require Node.js installed.

![Figure 29. Ultimate picture of collaborative work using rich and web clients](image)

![Figure 30. Screenshot of the current version of the web-based tool for collaborative model editing](image)
2.2.14 ‘Concurrent Assurance Information Edition’ with Data Mining Technologies

The AMASS tools collect, create and aggregate a lot of data and relationships. It is essential to provide users, but also other functions and modules, a way to quickly search this big-data by means of keywords or other criteria. Based on the Elasticsearch open source software stack [20], a generic indexing features is available in the platform. In this prototype, it is intentionally kept simple. Arbitrary EMF objects (local or remote, file or CDO resource) can be indexed via the user interface of the prototype. Attributes and relationships are “crawled” by a generic and reflective indexer. The respective Ecore metamodel is used to decide whether an attribute value is indexed or not. The only pre-requisite is the configuration of the Elastic server (Figure 31).

The user can select an arbitrary object and index the object, the object’s resource or the object tree into Elasticsearch (Figure 32).

The prototype further contains a web-based (google like) simple search application (Figure 33). The user may enter arbitrary keywords or other expressions following the Elastic search syntax [21]. The result can be further limited either by document type (here metaclass) or dedicated filters as for example ASIL – which was implemented as an example.
The Kibana Dashboard Software (Figure 34) can be used to visualize all indexed data in a nice and understandable way.

![Figure 33. Screenshot of the Data Mining platform for collaborative work](image)

![Figure 34. Screenshot of the Kibana Discovery tool](image)

### 2.3 Installation and User Manuals

The steps necessary to install the Prototype P1 are exhaustively described in the AMASS User Manual [10] (currently under elaboration for all the AMASS building blocks), thus they are not repeated in this deliverable. In the user manual of the Prototype P1 the users can find the installation instructions, the tool environment description, and the functionalities for the specification of evidence-related assurance project...
information: artefact repository preferences, artefact definitions, artefacts, artefact resources, artefact property values, artefact events, artefact evaluations, impact analysis, executed processes, and property models.
3. Implementation Description

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

3.1 Implemented Modules

The modules implemented for AMASS Prototype P1 in the scope of WP5 are as follows:

- **Platform Management** (Figure 35)
  - **Access Management**
    This module is integrated in OpenCert and uses CDO [12] as the main base technology.
  - **Data Management**
    This module is integrated in OpenCert and uses CDO [12] as the main base technology.
  - **Collaborative Work**
    In addition to some basic support for collaborative work provided by OpenCert (e.g. through CDO features for concurrent data access), the tools that currently implement collaborative work functionality are: Capra, the web-based approach for concurrent assurance information edition, Elasticsearch, and Kibana.

- **Evidence Management** (Figure 36)
  - **Evidence Characterization Editor**
    OpenCert implements this module. It is an Eclipse-Based editor for artefact and executed-process information of an assurance project. It contains plugins for edition of artefact models and of process models, and to provide services for evidence storage (determination, specification, and structuring of evidence), and for traceability-related aspects. The editors have been mostly generated with the EMF [14] and EEF [13] Eclipse technologies, in addition to the implementation of some tailored functionality, e.g. for integration with SVN and for impact analysis.

- **Assurance Traceability** (Figure 37)
  - **Traceability Management**
    The Evidence Characterisation Editor provides support for evidence traceability. This functionality is complemented with the use of Capra.
  - **Impact Analysis**
    The impact analysis support is currently embedded in the Evidence Characterisation Editor.

- **Tool Integration** (Figure 38)
  - **Toolchain Management**
    The current support for Toolchain Management is integrated in OpenCert and CHESS. Each tool integration technology has a dedicated user interface.
  - **Tool Connector**
    Each tool integration technology described above has its own Tool Connector component: connector for SVN, connector based on OSLC-KM, etc.
Figure 35. Platform management modules

Figure 36. Evidence management module
3.2 Implemented Metamodel

AMASS D2.2 [2] presents the CACM, including evidence management metamodels. These metamodels correspond to the envisioned, conceptual data structure necessary in AMASS to provide the reuse-oriented holistic approach for architecture-driven assurance, multi-concern assurance, and seamless interoperability. However, the metamodel implemented for the Evidence Management module does not exactly correspond to the CACM, but to the metamodel implemented in OpenCert. This situation will be re-analysed for future AMASS prototypes.

Such metamodel is the CCL created in the OPENCOSS project. The CCL can be regarded as compliant with the CACM because it supports all the evidence information specification needs represented in the CACM. However, the specification of information can be a bit different. For example, traceability information is not specified in the CCL based on a specific metamodel, but this information type is embedded in the CCL artefact metamodel.
Figure 39 shows an excerpt of the CCL to specify evidence information. Further information about the CCL can be found in [16].

![Diagram of artefact information in the CCL]

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

The source code of the second AMASS prototype can be found in the source code SVN repository [11]. The code for Prototype P1 evidence management and system management modules are stored together with the other basic building blocks in the repository under “tag” to distinguish the state of the code at the time of the integrated release.

The necessary plugins for Seamless Interoperability (Figure 40) are:

- **org.eclipse.opencert.evm.evidspes**
  In this plugin, the evidence metamodel is defined and stored, and the Java implementation classes for this model are generated.

- **org.eclipse.opencert.evm.evidspes.edit**
  This plugin contains a provider to display evidence models in a user interface.

- **org.eclipse.opencert.evm.evidspes.editor**
  This plugin provides the user interface to view instances of the model using several common viewers, and to add, remove, cut, copy and paste model objects, or modify the objects in a standard property sheet.

- **org.eclipse.opencert.evm.evidspes.editor.dawn**
  This plugin is an extension of the previous one. It aims to communicate with the CDO Server to store the generated model.
• **org.eclipse.opencert.evm.evidspec.preferences**
  This plugin defines the default preferences for the communication with the SVN repository, thus it defines the type of repository (local or remote) and a user and password to connect with the remote repository.

• **org.eclipse.opencert.evm.oslc.km.importevid**
  This plugin contains all the classes needed to:
  1. Perform a request to the RQA web-service sending the Papyrus file content and receiving the OSLC-KM model in form of JSON.
  2. Build from the JSON string the OSLC-KM model.
  3. Parsing the OSLC-KM model into an ArtefactModel from the Evidence Manager.
  4. Store the ArtefactModel in the CDO database.

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

Figure 40. Evidence management and System management plug-ins

• **org.eclipse.opencert.impactanalysis**
  This plugin contains the implementation of the change impact analysis module. This module is used by AMASS Tool Platform clients to call and execute change impact analysis.

• **org.eclipse.opencert.infra.properties**
This plugin contains the definition of the Property metamodel, and the Java implementation classes for this model.

- **org.eclipse.opencert.infra.properties.edit**
  As the edit plugin for evidence, this plugin contains a provider to display the model in a user interface.

- **org.eclipse.opencert.infra.properties.editor**
  As the edit plugin for evidence, this plugin is an editor to create and modify instances of the model.

- **org.eclipse.opencert.infra.svnkit**
  In this plugin, the functionalities necessary for the communication with the SVN repository to export and import artefacts are defined.

- **org.eclipse.opencert.pam.procspec**
  In this plugin, the process execution metamodel is defined and stored, and the Java implementation classes for this model are generated.

- **org.eclipse.opencert.pam.procspec.edit**
  This plugin contains a provider to display process execution models in a user interface.

- **org.eclipse.opencert.pam.procspec.editor**
  This plugin provides the user interface to view instances of the model using several common viewers, and to add, remove, cut, copy and paste model objects, or modify the objects in a standard property sheet.

- **org.eclipse.opencert.pam.procspec.editor.dawn**
  This plugin is an extension of the previous one. It aims to communicate with the CDO Server to store the generated model.

- **org.eclipse.opencert.storage.cdo**
  This plugin contains classes for using the CDO server in the AMASS Tool Platform. This server provides a common storage for all AMASS Tool Platform clients and a server. It accesses PostgreSQL database as its data backend. In addition to common storage implementation, this package contains utility classes used when accessing the CDO server by its clients.

- **org.eclipse.opencert.chess.tracemodel**
  This plugin provides a dedicated Capra metamodel which is used to create the links between architectural related entities and assurance and evidence ones. Capra extension points are used to register the metamodel at runtime.

- **org.eclipse.opencert.chess.traceability**
  This plugin contains classes that implement the user interface and control for the management of the traceability links between CHESS and the other parts of the OpenCert models (as presented in section 2.2.8). These classes use the API provided by Capra and use the different kind of trace links provided by the org.eclipse.opencert.chess.tracemodel plugin.

### 3.4 Source Code Description for External Tools

This section describes the source code for features that have been implemented for Seamless Interoperability but have not been integrated into the AMASS Tool Platform. The features correspond to functionality that will be integrated for the next prototype or that will be provided by an external tool. For the latter, tool integration mechanisms for communication between the AMASS Tool Platform and the external tools will need to be developed.

#### 3.4.1 Seamless Interoperability Features in RQA

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

Once this OSLC-KM model has been introduced as input to RQS, the tool creates what is called a specification composed of workproducts, for example, the requirements found in the model in the Papyrus file, and exposes it to the rest of functionality of RQS. This will allow to assess its quality in many different perspectives:
- Correct: in the scope of the individual workproduct
- Complete: in the scope of the specification
- Consistent: in the scope of the specification

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

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

The implementation of this functionality has been developed inside the Requirements Quality Suite (RQS) and it's composed of several libraries:
- **Rqa.Face.OslcKm**: it implements the connection of the OSLC-KM model instance with the rest of functionalities of the Requirements Quality Suite (RQS).
- **System Repository Language (SRL)**: it is the OSLC-KM implementation inside the Requirements Quality Suite (RQS).
- **Oslc.Km.Parsers**: several parsers have been implemented in the methodology described in the second step (by using XSLT transformation files to create the OSLC-KM model instance). They can be seen in Figure 43. A part of this transformation, the file generated for Papyrus to get requirements from the model can be found in Figure 44.
- **Oslc.Km.Parsers.XmlToSrl**: in the same Visual Studio solution a Graphical User Interface (GUI) has been generated in RQS to manage this kind of transformations (see Figure 45).
Figure 41. Rqa.Face.OslcKm library

Figure 42. Implementation of the OSLC-KM for RQS
Figure 43. OSLC-KM parsers and XSLT transformation files for Papyrus and Rhapsody
Figure 44. Part of the Papyrus XSLT transformation to map requirements in the model to the OSLC-KM model instance
Figure 45. GUI to allow creation of XSLT files to customise the mapping of XML file nodes to elements in the OSLC-KM metamodel
4. Conclusion

This deliverable has presented the implementation work performed for Seamless Interoperability in AMASS Prototype P1, which is the second version of the AMASS Tool Platform. The current support in the Platform allows a user to manage evidence artefacts, manage traceability between assurance assets, integrate the Platform with external tools, and collaborate with other users. In addition, some external support is already provided for concurrent assurance information editing, e.g. via Kibana. Some further support for tool integration provided by external tools will be integrated in the AMASS Tool Platform for the next prototype, such as advanced RQA support.

At its current state, and prior validation in WP2 and application in WP1, Seamless Interoperability support for Prototype P1 has TRL 3 (experimental proof of concept).

In addition to the implementation of further requirements for further Seamless Interoperability in the AMASS Tool Platform and to the general revision of some implementation for enhancement, the main aspects to address for Prototype P2 include the final decision upon the implementation of new features targeted at tool quality characterisation and assessment, the development of security mechanisms for user access management, a detailed analysis of the security implications from integration with external tools, the enactment of larger toolchains, and the integration of advanced collaborative work functionality.
References


---

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

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