

ECSEL Research and Innovation actions (RIA)



AMASS

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

Integrated AMASS platform (b) D2.7

Work Package:	WP2 Reference Architecture and Integration
Dissemination level:	PU = Public
Status:	Final
Date:	31 th January 2018
Responsible partner:	Morayo Adedjouma/ Bernard Botella (CEA)
Contact information:	{morayo.adedjouma, bernard.botella } AT cea.fr
Document reference:	AMASS_D2.7_WP2_CEA_V1.0

PROPRIETARY RIGHTS STATEMENT

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

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

Contributors

Names	Organisation
M. Adedjouma, B. Botella	Commissariat à L'énergie Atomique et aux Energies Alternatives (CEA)
A. Debiasi, L. Cristoforetti	Fondazione Bruno Kessler (FBK)
Bernhard Winkler	Virtual Vehicle (VIF)
Jan Mauersberger	medini Technologies (AMT)
Marc Sagno	Alliance pour les technologies de l' informatique (A4T)
Stefano Puri	Intecs (INT)

Reviewers

Names	Organisation
Ran Bi (Peer-reviewer)	Rapita Systems (RPT)
Silvia Mazzini (Peer-reviewer)	Intecs (INT)
Barbara Gallina (TC review)	Maelardalens Högskola (MDH)
Cristina Martinez (Quality manager)	Tecnalia Research & Innovation (TEC)
Alejandra Ruiz (TC review)	Tecnalia Research & Innovation (TEC)



TABLE OF CONTENTS

Executive Summary	7
1. Introduction.....	8
1.1 Scope	8
1.2 Purpose of the deliverable	9
1.3 Relations to others deliverables	9
1.4 Structure of the document	10
2. AMASS Platform Architecture.....	11
2.1 Conceptual and Implementation Architecture	11
2.2 AMASS Platform prototype P1	12
2.3 Testing and Validation Methodology	13
3. Testing and Validation for WP3-related Blocks.....	15
3.1 Functionalities.....	15
3.2 Test Cases	15
3.3 Test Results.....	23
4. Testing and Validation for WP4-related Blocks.....	26
4.1 Functionalities.....	26
4.2 Test Cases	26
5. Testing and Validation for WP5-related Blocks.....	32
5.1 Functionalities.....	32
5.2 Test Cases	32
5.3 Test Results.....	34
6. Testing and Validation for WP6-related Blocks.....	36
6.1 Functionalities.....	36
6.2 Test Cases	36
6.3 Test Results.....	41
7. Prototype P1 Validation Synthesis.....	44
7.1 Analysis of Test Results	44
7.2 Recommendations	44
8. Conclusion	46
Abbreviations and Definitions.....	47
References	48
Appendix A: Validation status of the AMASS Prototype P1	49



List of Figures

Figure 1. Overall AMASS Platform Architecture	11
Figure 2. Layered structure of AMASS Reference Architecture	12
Figure 3. AMASS prototype P1 testing and validation methodology	13



List of Tables

Table 1. System Component Specification and Architecture driven assurance functionalities.....	15
Table 2. Test Case WP3_TC_001 for WP6_PPA_004	15
Table 3. Test Case WP3_TC_002 for WP3_SC_007	16
Table 4. Test Case WP3_TC_003a for WP3_CAC_001	16
Table 5. Test Case WP3_TC_003b for WP3_CAC_001	17
Table 6. Test Case WP3_TC_004a for WP3_CAC_005	17
Table 7. Test Case WP3_TC_004b for WP3_CAC_005	17
Table 8. Test Case WP3_TC_005a for WP3_CAC_006	18
Table 9. Test Case WP3_TC_005b for WP3_CAC_006	18
Table 10. Test Case WP3_TC_006a for WP3_CAC_007	18
Table 11. Test Case WP3_TC_006b for WP3_CAC_007	19
Table 12. Test Case WP3_TC_007a for WP3_CAC_008	19
Table 13. Test Case WP3_TC_007b for WP3_CAC_008	19
Table 14. Test Case WP3_TC_007c for WP3_CAC_008.....	20
Table 15. Test Case WP3_TC_007d for WP3_CAC_008	20
Table 16. Test Case WP3_TC_008a for WP3_CAC_009	20
Table 17. Test Case WP3_TC_008b for WP3_CAC_009	21
Table 18. Test Case WP3_TC_009a for WP3_CAC_011	21
Table 19. Test Case WP3_TC_009b for WP3_CAC_011	21
Table 20. Test Case WP3_TC_010 for WP3_VVA_005	22
Table 21. Test Case WP3_TC_011 for WP3_VVA_010.....	22
Table 22. Test results for the WP3 implemented functionalities	23
Table 23. Assurance case Specification and multi-concern assurance functionalities.....	26
Table 24. Test Case WP4_TC_001 for WP4_ACS_001 functionality	26
Table 25. Test Case WP4_TC_002 for WP4_ACS_010 functionality	27
Table 26. Test Case WP4_TC_003 for WP4_ACS_005 functionality	27
Table 27. Test Case WP4_TC_004 for WP4_ACS_002 functionality	27
Table 28. Test Case WP4_TC_005 for WP4_ACS_003 functionality	28
Table 29. Test Case WP4_TC_006 for WP4_ACS_002 functionality	28
Table 30. Test Case WP4_TC_007 for WP4_DAM_001 and WP4_DAM_002 functionality.....	28
Table 31. Test Case WP4_TC_008 for WP4_DAM_001 functionality	29
Table 32. Test Case WP4_TC_009 for WP4_ACS_007 functionality	29
Table 33. Test Case WP4_TC_010 for WP4_SDCA_002 functionality	29
Table 34. Test Case WP4_TC_011 for WP4_SDCA_003 functionality.....	30
Table 35. Test Case WP4_TC_012 for WP4_CMA_003 functionality	30
Table 36. Test results for the WP4 implemented functionalities.....	31
Table 37. Evidence Management and seamless interoperability functionalities.....	32
Table 38. Test Case WP5_TC_001 for WP5_EM_006 functionality.....	32
Table 39. Test Case WP5_TC_002 for WP5_TI_006 functionality	33
Table 40. Test Case WP5_TC_003 for WP5_TI_003 functionality	33
Table 41. Test Case WP5_TC_004 for WP5_EM_015 functionality	33
Table 42. Test Case WP5_TC_005 for WP5_TI_017 and WP5_TI_018 functionalities	33
Table 43. Test Case WP5_TC_006 for WP5_TI_005 functionality	34



Table 44. Test results for the WP5 implemented functionalities.....	34
Table 45. Compliance Management and cross and intra-domain reuse functionalities.....	36
Table 46. Test Case WP6_TC_001 for WP6_CM_001 functionality	36
Table 47. Test Case WP6_TC_002 for WP6_RA_006 functionality	37
Table 48. Test Case WP6_TC_003 for WP6_RA_001, WP6_RA_005 functionalities.....	37
Table 49. Test Case WP6_TC_004 for WP6_CM_002 functionality.....	38
Table 50. Test Case WP6_TC_005 for WP6_CM_002, WP6_CM_008 functionalities	38
Table 51. Test Case WP6_TC_006 for WP6_CM_005 functionality.....	38
Table 52. Test Case WP6_TC_007 for WP6_RA_002, WP6_RA_003, WP6_RA_004 functionality.....	38
Table 53. Test Case WP6_TC_008 for WP6_RA_002, WP6_RA_003, WP6_RA_004 functionalities	39
Table 54. Test Case WP6_TC_009 for WP6_RA_001, WP6_RA_002, WP6_RA_003, WP6_RA_004, WP6_RA_005 functionalities.....	39
Table 55. Test Case WP6_TC_010 for WP6_PPA_001 functionality.....	39
Table 56. Test Case WP6_TC_011 for WP6_PPA_002 functionality.....	40
Table 57. Test Case WP6_TC_012 for WP6_PPA_003 functionality.....	40
Table 58. Test Case WP6_TC_013 for WP6_PPA_004 functionality.....	40
Table 59. Test results for the WP6 implemented functionalities.....	41
Table 60. Prototype P1 Implementation Status	44
Table 61. Results of the test cases for prototype P1 implemented functionalities	44
Table 62. Prototype P1 Functionalities Status	49



Executive Summary

The AMASS Open Tool Platform is the main result of the AMASS project. This platform corresponds to a collaborative tool environment supporting Cyber Physical Systems assurance and certification. The development of the AMASS Open Tool Platform follows an incremental approach by developing rapid and early prototypes in three iterations called Core, P1, and P2.

The current deliverable (D2.7) is the second one produced in the Task 2.4 AMASS Platform Validation. It concerns the validation of the prototype P1.

The functionalities of the AMASS platform are described in the AMASS deliverable D2.3 (AMASS Reference Architecture) [6]. The Prototype Core has been built upon three pre-existing toolsets from the OPENCOS project [1], the CHESS project (Polarsys Platform) [12] and the SafeCer project [2] (which built on top of the Eclipse Process Framework project). This prototype P1 extends the prototype Core with specific blocks/functionalities/tools addressing the AMASS STOs: Architecture-Driven Assurance (STO1), Multi-Concern Assurance (STO2), Seamless Interoperability (STO3), and Cross/intra-Domain Reuse (STO4).

The prototype P1 has been released as an Eclipse bundle. Two manuals have been provided with the platform: Developer Guide that is dedicated to the AMASS Platform developers, and User Manual that targets AMASS Platform users.

This deliverable:

- recalls the architecture of the overall AMASS Platform and its building blocks,
- presents the validation activities that have been conducted on the prototype P1:
 - This validation has been based on an analysis of the requirements and corresponding functionalities, planned for prototype P1 and defined in D2.1 [7], and the usage scenarios defined in D2.3 [6]. These items have been refined into test cases that are compatible with the current developments of the AMASS platform. The previous validation results of prototype Core have been revised as well as the functionalities that were postponed for P1.
- summarizes the validation results:
 - Many test cases have been executed with the status *Passed_But*, which means that the functionalities should be enhanced. The verification of many functionalities has been postponed for the next iteration, because some of them were not completely available or no related use cases or guidelines have been identified.
- and gives recommendations for the next prototype iteration, such as traceability among requirements, use cases, developed functionalities, and methodological guidelines for a better validation process.



1. Introduction

1.1 Scope

AMASS will create and consolidate a de-facto European-wide assurance and certification open tool platform, ecosystem and self-sustainable community spanning the largest Cyber-Physical System vertical markets. The ultimate aim is to lower certification costs in face of rapidly changing product features and market needs. This will be achieved by establishing a novel holistic and reuse-oriented approach for:

- architecture-driven assurance fully compatible with standards such as AUTOSAR [25] and Integrated Modular Avionics (IMA) [26];
- multi-concern assurance, for example compliance demonstration, impact analyses, and compositional assurance of security and safety aspects;
- seamless interoperability between assurance/certification and engineering activities along with third-party activities (external assessments, supplier assurance);
- cross/intra-domain re-use of, for instance, semantic standards and product/process assurance.

The AMASS tangible expected results are:

- a) The **AMASS Reference Tool Architecture**, which will extend the OPENCOS [1] and SafeCer [2] 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 (e.g. based on Open Services for Lifecycle Collaboration (OSLC)¹ specifications).
- 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 Application programming interfaces (APIs) [21] 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 [3] is a strong candidate to host AMASS.

To achieve these results, the AMASS Consortium has decided to follow an incremental approach by developing rapid and early prototypes in three iterations:

1. During the **first prototyping** iteration (prototype Core), the AMASS Platform Basic Building Blocks were aligned, merged and consolidated at Technology Readiness Level (TRL) 4 (technology validated in laboratory).
2. During the **second prototyping** iteration (prototype P1), the single AMASS-specific Building Blocks have been developed, integrated with previous prototype and benchmarked at TRL 4.
3. Finally, at the **third prototyping** iteration (Prototype P2), all AMASS building blocks are integrated in a comprehensive toolset operating at TRL 5 (technology validated in relevant environment).

¹ <https://open-services.net>

1.2 Purpose of the deliverable

This deliverable is the second one from the Task 2.4 AMASS Platform Validation. The purpose of this deliverable is to serve as a complement to the prototype P1. It provides a summarised version of the implementation work that has been done related to the AMASS blocks implementation and their integration based on the reference architecture that was envisioned for the platform in deliverable D2.3 [6].

This document presents the platform architecture and its different blocks, and the methodology followed for its validation. It also presents the testing and validation activities of the AMASS platform that correspond to the scope of prototype P1, in order to check the global functionality of the platform according to the requirements defined in WP2, T2.1. For the validation activities, we have performed an analysis of the functionalities planned for building blocks constituting the prototype P1 defined in D2.1 [7] and collected usage scenarios defined in D2.3 [6] in order to refine these items into test cases that are compatible with the current developments of the AMASS platform. We have also analysed the functionalities specified for the basic building blocks of prototype Core that were postponed for the next iteration. Additional test cases have been defined for those functionalities that were implemented during the previous iteration but whose test results were not found satisfactory. The manual execution of the test cases enables us to provide direct feedback regarding implementation status and potential further enhancements for the next iteration.

The testing results together with the validation team feedback will allow WP1 and T1.5 to do an assessment of: 1) how the objectives of the case studies are met, 2) which applications perform best, and consequently, have the biggest market potential, and 3) which aspects can be improved.

1.3 Relations to others deliverables

D2.7 is related to others AMASS deliverables:

- D2.1 [7] (Business cases and high-level requirements) defines the business models of the AMASS solutions as well as the requirements to be met by the WP3, WP4, WP5, and WP6 technical AMASS work packages.
- D2.3 [6] (AMASS Reference Architecture (b)) describes the overall architecture of the AMASS platform including needs from the case studies that must be covered by the platform.
- D3.5 [8] (Prototype for Architecture-Driven Assurance (b)), D4.5 [9] (Prototype for multi-concern assurance (b)), D5.5 [10] (Prototype for seamless interoperability (b)) and D6.5 [11] (Implementation for Cross-Domain and Intra-Domain Reuse (b)) define the development of a tooling framework to support the AMASS platform second prototype. These deliverables describe the tools whose testing is reported in the current document.
- The methodological guides D3.7 [13] (Methodological Guide for Architecture-Driven Assurance (a)), D4.7 [14] (Methodological_Guide_for_Multiconcern_Assurance_(a)), D5.7 [15] (Methodological_Guide_for_Seamless_Interoperability_(a)), and D6.7 [16] (Methodological_Guide_for_Cross_Intra_Domain_Reuse_(a)).
- The AMASS Prototype P1 user manual² [4] provides a guide on how to use the AMASS platform. It is the update of the previous prototype Core user manual that targets AMASS Platform users as the desirable audience.

² User manual available at https://services.medini.eu/svn/AMASS_collab/WP-transversal/ImplementationTeam/PrototypeP1/AMASS_PrototypeP1_UserManual.docx



- The AMASS Prototype P1 developer guide³ [5] provides a guide on how to set up the development environment and the tools integrated in the AMASS platform. The manual targets the AMASS Platform developers. It was created at the same time than the implementation in a collaborative way by the own developers and validated among them.

The D2.3 [6] deliverable and the AMASS user manual [4] have been the main reference documents from which new test cases have been derived, so that the features described there can be validated.

1.4 Structure of the document

The deliverable is structured as follows: Chapter 2 is a brief presentation of the AMASS platform and the tooling architecture and technologies used to implement them, and describes the testing and validation procedure. Chapter 3 to 6 contains the implementation status of the functionalities for the prototype P1, the test cases that have been defined to evaluate them and the results of execution of these test cases. Chapter 7 provides a synthesis of the validation results of the prototype P1 and some recommendations to be considered for the prototype P2. To sum up, some conclusions about validation on prototype P1 have been included in Chapter 8. Finally, Appendix A provides a detailed status of the implementation of prototype P1.

³ Developer guide available at: https://services.medini.eu/svn/AMASS_collab/WP-transversal/ImplementationTeam/PrototypeP1/AMASS_PrototypeP1_DeveloperGuide.doc

2. AMASS Platform Architecture

2.1 Conceptual and Implementation Architecture

A general top-level architecture of the AMASS platform has been designed in the D2.3 deliverable [6]. As part of the overall platform, the AMASS prototype Core was the result of merging existing technologies from OPENCROSS [1] and SafeCer [2], and other related project such as CHESS [12]. The prototype P1 includes building blocks composed of tools to extend the functionalities provided by the basic building blocks of the prototype Core in order to address the following specific concerns: architecture-driven assurance, multi-concern assurance, seamless interoperability and cross/intra-domain reuse.

Figure 1 provides a high-level picture of the AMASS Reference Tool Architecture (ARTA) where the basic building blocks constituting the prototype Core are surrounded by a red dash-line and the building blocks implemented in prototype P1 are depicted in green boxes.

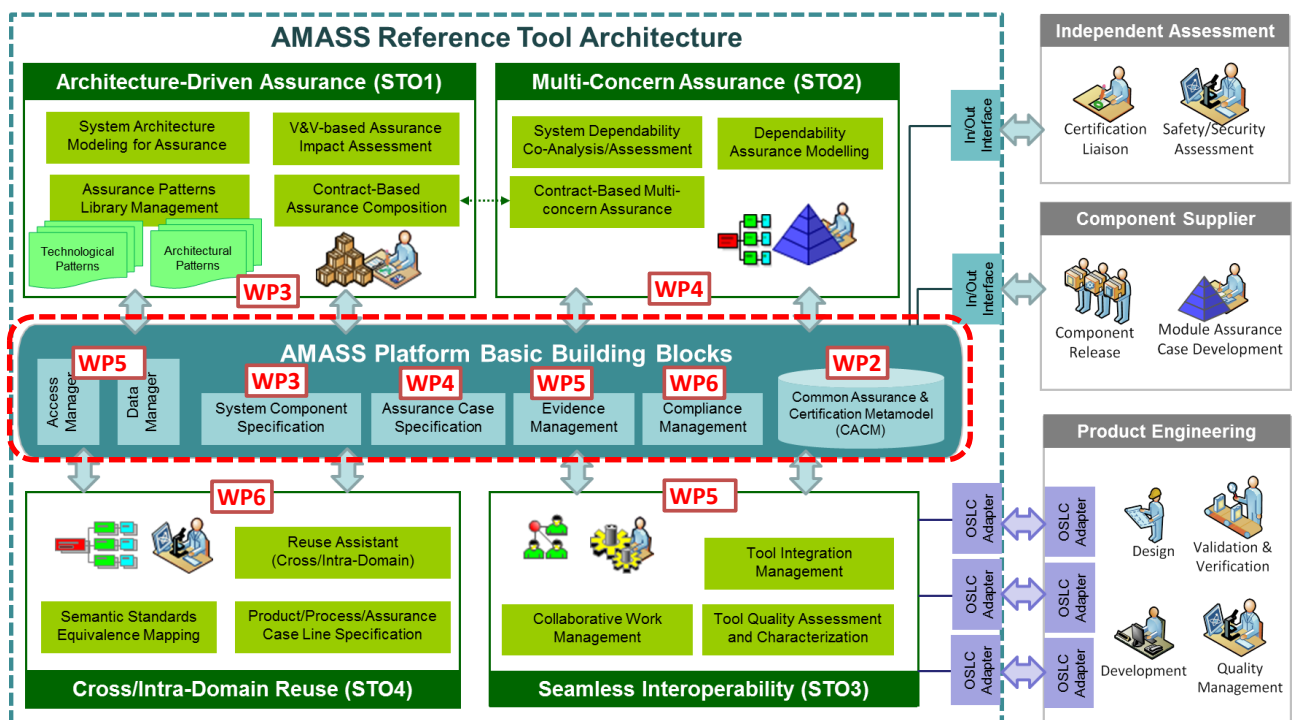


Figure 1. Overall AMASS Platform Architecture

The AMASS platform is composed of a set of tools providing the functionalities described in the AMASS deliverable D2.3 [6] (AMASS Reference Architecture, first prototype). Figure 2 presents an overall picture of the layered structure of the AMASS implemented architecture. This architecture has been implemented in the scope of the T3.3, T4.3, T5.3 and T6.3 tasks.

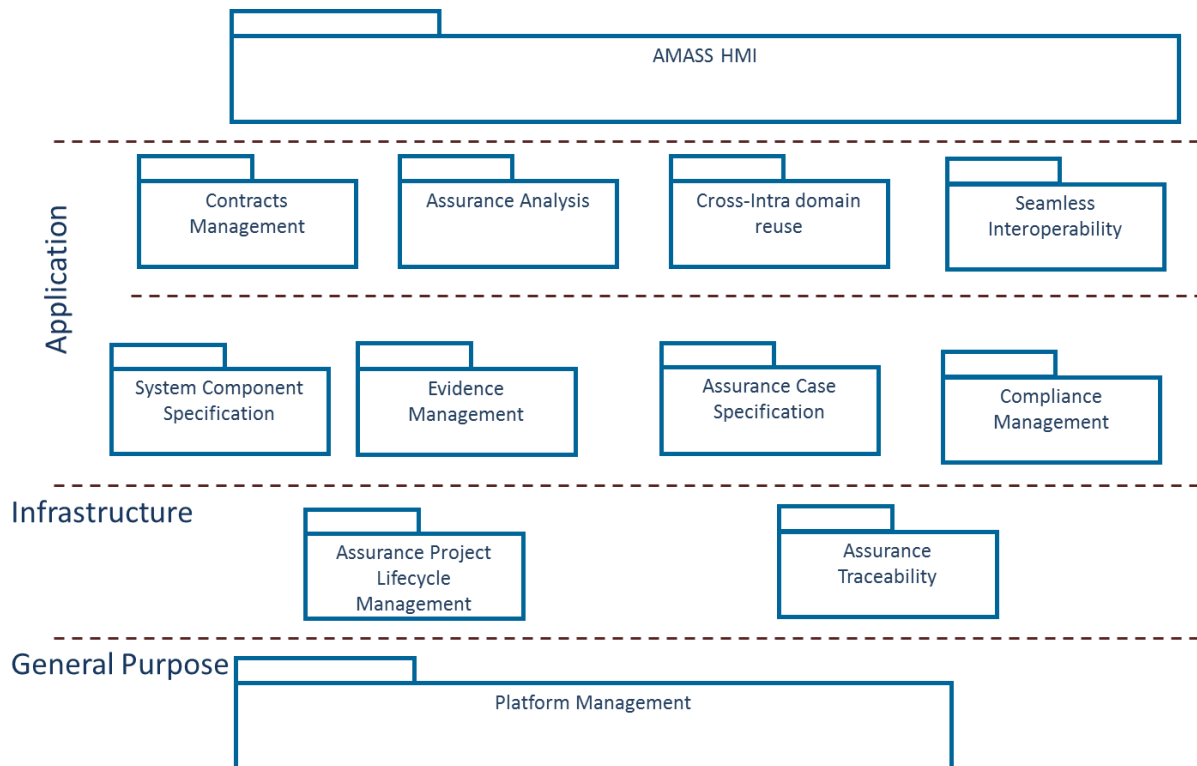


Figure 2. Layered structure of AMASS Reference Architecture

2.2 AMASS Platform prototype P1

The prototype P1, which implements some specific STOX blocks, has been built upon the following baseline technologies and toolsets:

1. Tools from OPENCOS project [1]
2. Papyrus tool and some of its features
3. Tools from the CHESS Project (Polarsys Platform) [12]
4. Tools from the EPF (Eclipse Process Framework) Project [20]
5. The Base Variability Resolution (BVR) tool [18]
6. Tools from the Capra project [24]
7. WEFACT tool [19]
8. Knowledge Manager (KM) toolset [22]
9. Open Service for Lifecycle Collaboration (OSLC) technology [21]

This prototype has been released as an Eclipse bundle, available at:

https://services.medini.eu/svn/AMASS_collab/WP-Transversal/ImplementationTeam/PrototypeP1/Tools/Client_Bundle/20171215_OpenCertCHESSClient_Win_x64.zip

Its source code is available at https://services.medini.eu/svn/AMASS_source/

2.3 Testing and Validation Methodology

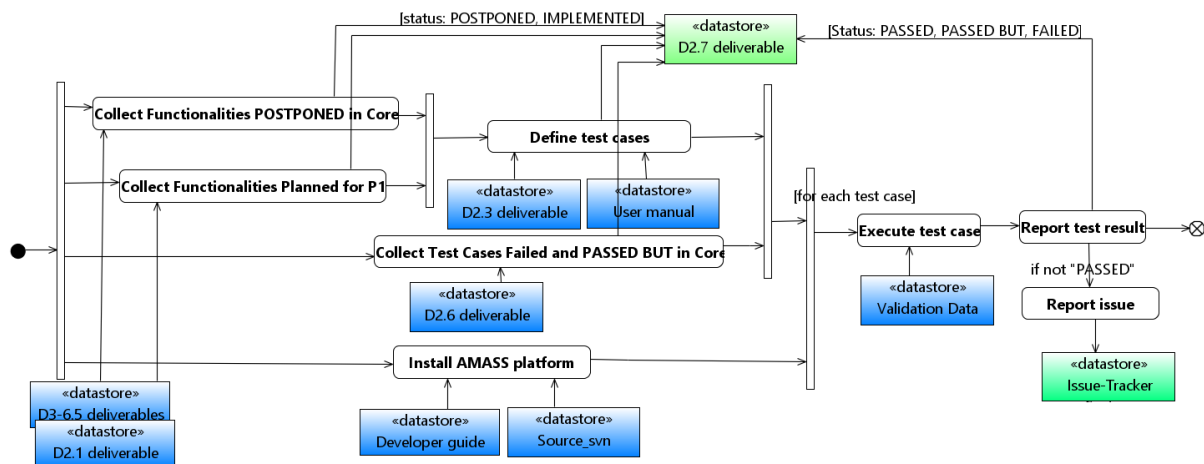


Figure 3. AMASS prototype P1 testing and validation methodology

Figure 3 presents the overall validation and testing methodology that has been defined in task T2.4. The methodology aims to validate that the AMASS prototype P1 platform satisfies its requirements and to check the system behaviour against the users' needs and the case studies (see D2.1 [7] and D2.3 [6] deliverables). It also checks that those functionalities specified for the prototype Core which validation results were not found satisfactory during last validation iteration are now correctly integrated in the platform.

The test cases listed in this document are mainly based on the scenarios defined in the use cases of D2.3 deliverable. These test cases aim to provide concrete scenarios about how AMASS will be used and when such usage can be regarded as successful. The test cases have been also traced to the D2.1 requirements of the AMASS prototype P1 (and some of the prototype Core as well) to ensure their theoretical coverage.

We have also used the AMASS user manual and the methodological guides D3.7 [13], D4.7 [14], D5.7 [15], and D6.7 [16] provided for the technical WP3 to WP6 as a reference document to enhance some test cases input(s), steps, and expected result(s).

Similarly, as for the previous AMASS Platform validation, a test case specification consists of the following information:

- Test Case ID, which uniquely identifies the test case.
- Scope, which provides the context and summarizes the purpose of the test case.
- Functionality ID, which refers to the AMASS related requirements that must be validated.
- Related use cases, which refer to the use case scenarios that are concerned.
- Input, which specifies the necessary input data needed prior to execute the test case.
- Steps are the execution steps to follow in order to run the test case.
- Expected results specify the behaviour or computation results expected from the execution of the test case.
- MoSCoW Priority⁴ as defined for the AMASS requirements in D2.1 deliverable [7].

Dedicated partners have performed the installation of the platform and executed manually the test cases checking its implementation. The testing partners have indicated the material used to run the test cases: machine configuration, validation data, etc. We report the status of the execution of the test cases as:

- PASSED: functionality that works as required

⁴ Must have, Should have, Could have, and Won't have but would like



- PASSED_BUT: functionality that works but could be enhanced
- FAILED: functionality that does not work
- POSTPONED: functionality implemented but that has not been tested

For each “Passed but”, “Failed” or “Postponed” status, a rationale is given to detail the reason of such status. We generate a ticket within the AMASS Issue-Tracker system for such test cases to report the problem to the Implementation Team. The overall validation results are summarized in this document.

3. Testing and Validation for WP3-related Blocks

3.1 Functionalities

The functionalities concerning the System Component Specification and Architecture driven assurance blocks are defined in D2.1 deliverable [7]. Table 1 is an excerpt of the relevant functionalities planned for prototype P1 in addition to the functionalities postponed in prototype Core, their implementation status and the implementation responsible. Among the twelve planned functionalities, eight have been implemented, and four functionalities were pending and then postponed for the next version of the AMASS platform.

Table 1. System Component Specification and Architecture driven assurance functionalities

ID	Functionality	Status	Responsible
WP3_VVA_004	Trace requirements validation checks	Pending	INT
WP3_SC_007	Fault injection (includes faulty behaviour of a component)	Implemented	INT
WP3_CAC_001	Validate composition of components by validating their contracts	Implemented	FBK
WP3_CAC_005	General management of contract-component assignments	Implemented	FBK
WP3_CAC_006	Refinement-based overview	Implemented	INT, FBK
WP3_CAC_007	Overview of check refinements results	Implemented	FBK
WP3_CAC_008	Contract-based validation and verification	Implemented	FBK
WP3_CAC_009	Improvement of Contract definition process	Implemented	FBK
WP3_CAC_011	Overview of contract-based validation for behavioural models	Pending	FBK
WP3_VVA_005	Verify (model checking) state machines	Pending	FBK, HON, UOM
WP3_VVA_010	Model-based safety analysis	Implemented	FBK
WP3_VVA_002	Trace model-to-model transformation	Pending	INT

3.2 Test Cases

In this section, we present the set of test cases defined to validate the implementation of the functionalities implemented for prototype P1. The test cases are based on the use case scenarios defined in the deliverable D2.3 [6] for the concerned functionalities when existing.

Table 2. Test Case WP3_TC_001 for WP6_PPA_004

ID	WP3_TC_001
Scope	Support specification of variability at the component level.
Functionality ID	WP6_PPA_004
Related use cases	“Manage product variability”
Input	The component warehouse (Base Model) has been specified.
Steps	<ol style="list-style-type: none">1. The user manages variability via the Variability, Resolution, and Realization editors.2. The user generates/exports the new component model, obtained as tailoring of the Base Model.
Expected results	New component model tailoring from the Base component model.



Priority	Shall
----------	-------

Table 3. Test Case WP3_TC_002 for WP3_SC_007

ID	WP3_TC_002
Scope	Fault injection (includes faulty behaviour of a component).
Functionality ID	WP3_SC_007
Related use cases	"Specify system architecture at different levels"
Input	A CHESS model with already defined nominal state machines (e.g., the Battery new project).
Steps	<ol style="list-style-type: none"> 1. Browse the model using the "Model Explorer" view and select the "System View" package. 2. Select the component with the state diagram to be enrich. 3. Right click on the selected component, then go to "New Diagram" – "State Machine Diagram". 4. Select the newly created state machine and apply the stereotype "Error Model". 5. Open the related State Machine Diagram in the editor. 6. Create the state machine with the correct stereotypes.
Expected results	The user is able to create fault injection state machines.
Priority	Must

Table 4. Test Case WP3_TC_003a for WP3_CAC_001

ID	WP3_TC_003a
Scope	Validate composition of components by validating their contracts.
Functionality ID	WP3_CAC_001
Related use cases	"Validate components composition through contracts-based design"
Input	A CHESS model with some components and contracts already defined (e.g. the WBS project).
Steps	<ol style="list-style-type: none"> 1. Browse the model using the "Model Explorer" view (e.g. for the WBS project, go inside the "PhysicalArchitecture" package under "modelSystemView" package). 2. Open the Block Definition Diagram inside the package. 3. Select a component (in the "Model Explorer" view) or the corresponding graphical representation (in the diagram editor). The properties available to check will be the assumptions and guarantees of contracts owned by the selected component and by its sub components. This operation includes recursively all the properties from the root to the leaves of the selected component. 4. Right click on the selected component, then go to "CHESS" – "Validation" – "Check Validation Property on selected component". A popup appears. 5. Select the model of time of the system, "Hybrid" or "Discrete" ("Discrete" for the WBS project) and another popup appears. 6. Select the type of check to perform, i.e., consistency, possibility, or entailment. Then select the Component and Properties ID and press OK. 7. When the check is completed, the status of the check is shown in the "Trace" view.
Expected results	The user can validate the composition of the components through their contracts.
Priority	Should

**Table 5.** Test Case WP3_TC_003b for WP3_CAC_001

ID	WP3_TC_003b
Scope	Validate the feature for composition of components by validating their contracts.
Functionality ID	WP3_CAC_001
Related use cases	"Validate components composition through contracts-based design"
Input	A CHESS model including some components and their contracts.
Steps	<ol style="list-style-type: none">1. Select a component (in the "Model Explorer" view) or the corresponding graphical representation (in the diagram editor).2. Right click on the selected component, then go to "CHESS - Validation – Check Validation Property on Selected Component"3. A popup appears to set the parameters of the command.4. Select the property Type (consistency, possibility, and entailment).5. Select the component selected in step 1 or its sub components.6. Select the assumptions and guarantees of contracts.7. Click "ok" to perform the validation and show the result of the verification.
Expected results	A status of the check is shown in the "V&V Result» view.
Priority	Shall

Table 6. Test Case WP3_TC_004a for WP3_CAC_005

ID	WP3_TC_004a
Scope	General management of contract-component assignments.
Functionality ID	WP3_CAC_005
Related use cases	"Browse components and associated contracts"
Input	A CHESS model with some components and contracts already defined (e.g. the WBS project).
Steps	<ol style="list-style-type: none">1. Browse the model using the "Model Explorer" view (e.g. for the WBS project, go inside the "PhysicalArchitecture" package under "modelSystemView" package).2. Open the Block Definition Diagram inside the package.3. Select the "Hierarchical Model View" view to check the status of the components currently defined in the system architecture, together with its associated contracts.
Expected results	The system should enable users to have an overview in terms of components and their associated contracts.
Priority	Should

Table 7. Test Case WP3_TC_004b for WP3_CAC_005

ID	WP3_TC_004b
Scope	General management of contract-component assignments.
Functionality ID	WP3_CAC_005
Related use cases	"Browse components and associated contracts"
Input	System architecture (components with potential associated contracts).
Steps	<ol style="list-style-type: none">1. Go to Window – Show View – Hierarchical Model View
Expected results	Contracts assigned for each component in the "System Architectures" Column of "Hierarchical Model View".
Priority	Should

**Table 8.** Test Case WP3_TC_005a for WP3_CAC_006

ID	WP3_TC_005a
Scope	Refinement-based overview.
Functionality ID	WP3_CAC_006
Related use cases	"Browse Contracts refinement status"
Input	A CHESS model with some components and contracts with refinements already defined (e.g. the WBS project).
Steps	<ol style="list-style-type: none">1. Browse the model using the "Model Explorer" view (e.g. for the WBS project, go inside the "PhysicalArchitecture" package under "modelSystemView" package).2. Open the Block Definition Diagram inside the package.3. Select the "Contract Refinement View" to check all the defined contracts and their refinements.
Expected results	The system should enable users to have a hierarchical view of the contracts and relative refinements along the system architecture.
Priority	Should

Table 9. Test Case WP3_TC_005b for WP3_CAC_006

ID	WP3_TC_005b
Scope	Refinement-based overview.
Functionality ID	WP3_CAC_006
Related use cases	"Browse Contracts refinement status"
Input	Components and their contracts
Steps	<ol style="list-style-type: none">1. Go to Window – Show View – Contract Refinement View
Expected results	Contract refinement for each contract in the "Refined Contract" Column of the "Contract Refinement View".
Priority	Should

Table 10. Test Case WP3_TC_006a for WP3_CAC_007

ID	WP3_TC_006a
Scope	Overview of check refinements results.
Functionality ID	WP3_CAC_007
Related use cases	"Inspect contracts refinement result"
Input	A CHESS model with some components and contracts with refinements already defined (e.g. the WBS project).
Steps	<ol style="list-style-type: none">1. Browse the model using the "Model Explorer" view (e.g., go inside the "PhysicalArchitecture" package under "modelSystemView" package).2. Open the Block Definition Diagram inside the package.3. Ensure that a contract refinement check has already been run or start a check (for instruction steps see use case "Verify contract refinement").4. Open the "V&V Results" view and look for a function called "ocra_check_refinement".5. Right click on the function and select "Show result". The "Contract trace" view will open and show the results.
Expected results	The system should enable users to have an overview in terms of status of check refinement of all the defined contracts.
Priority	Should

**Table 11.** Test Case WP3_TC_006b for WP3_CAC_007

ID	WP3_TC_006b
Scope	Overview of check refinements results.
Functionality ID	WP3_CAC_007
Related use cases	"Inspect contracts refinement result"
Input	Component contracts
Steps	1. Select the "Contract Refinement View" to check the refining contracts for each contract.
Expected results	Number of sub-contracts for each refined contract.
Priority	Should

Table 12. Test Case WP3_TC_007a for WP3_CAC_008

ID	WP3_TC_007a
Scope	Contract-based validation and verification.
Functionality ID	WP3_CAC_008
Related use cases	"Verify contract refinement", "Perform contract-based fault tree generation", and "Validate weak contracts".
Input	A CHESS model with some components and contracts with refinements already defined (e.g. the WBS project).
Steps	<ol style="list-style-type: none">1. Browse the model using the "Model Explorer" view (e.g., go inside the "PhysicalArchitecture" package under "modelSystemView" package).2. Open the Block Definition Diagram inside the package.3. Select a component (in the "Model Explorer" view) or the corresponding graphical representation (in the diagram editor). The contract refinements considered will be the ones associated to the selected component and the ones associated to its sub components. This operation includes recursively all the contracts along the subcomponents, from the root to the leaves of the system.4. Right click on the selected component, then go to "CHESS" – "Functional Verifications" – "Check Contract Refinement on selected component".5. When the analysis is completed, it is possible to see the status of each refinement in the "Contract trace" view. If the check fails, it is possible to see the counter example, i.e., the instances of values to assign to the ports that cause the failure of the contract refinement.
Expected results	The system must provide support for contract-based system validation and verification.
Priority	Must

Table 13. Test Case WP3_TC_007b for WP3_CAC_008

ID	WP3_TC_007b
Scope	Contract-based validation and verification.
Functionality ID	WP3_CAC_008
Related use cases	"Verify contract refinement"
Input	Component contracts and their refinement. Configuration of external tool (OCRA tool) allowing contracts refinement.
Steps	<ol style="list-style-type: none">1. Select a component (in the "Model Explorer" View) or the corresponding graphical representation (in the Diagram Editor).2. Right click on the selected component, then go to CHESS-Functional Verification – Check Contract Refinement on Selected Component.



Expected results	A status of the check is shown in the “V&V Result” view.
Priority	Must

Table 14. Test Case WP3_TC_007c for WP3_CAC_008

ID	WP3_TC_007c
Scope	Contract-based validation and verification.
Functionality ID	WP3_CAC_008
Related use cases	“Perform contract-based fault trees generation”
Input	Component contracts and their refinement. Configuration of external tool (OCRA tool) allowing contracts refinement.
Steps	<ol style="list-style-type: none"> 1. Select a component (in the “Model Explorer” View) or the corresponding graphical representation (in the Diagram Editor). 2. Right click on the selected component, then go to CHESS-Safety Analysis – Contract-based Safety Analysis on selected component.
Expected results	A status of the check is shown in the “V&V result” view.
Priority	Must

Table 15. Test Case WP3_TC_007d for WP3_CAC_008

ID	WP3_TC_007d
Scope	Contract-based validation and verification.
Functionality ID	WP3_CAC_008
Related use cases	“Validate weak contracts”
Input	Component contracts and their refinement. Configuration of external tool (OCRA tool) allowing contracts refinement.
Steps	<ol style="list-style-type: none"> 1. Select a component (in the “Model Explorer” View) or the corresponding graphical representation (in the Diagram Editor). 2. Right click on the selected component, then go to CHESS-Safety Analysis – Contract-based Safety Analysis on selected component.
Expected results	A status of the check is shown in the “V&V result” view.
Priority	Must

Table 16. Test Case WP3_TC_008a for WP3_CAC_009

ID	WP3_TC_008a
Scope	Improvement of Contract definition process.
Functionality ID	WP3_CAC_009
Related use cases	“Assign a contract to the component”, and “Structure properties into contracts (assumptions/guarantees)”
Input	A CHESS model with some components already defined.
Steps	<ol style="list-style-type: none"> 1. Select a component (in the “Model Explorer” view) or the corresponding graphical representation (in the Diagram Editor) and go to the “Properties” view. 2. Go to the “ContractEditor” tab, type the contract name and click on “Add Contract”. A popup appears. 3. Create a new contract and select the pencil icon. A popup appears. 4. Select “No” to avoid the creation of empty formal properties. 5. A new contract is created, along with a contract instance. 6. Select the contract just created in the Contract List drop down menu. 7. Type the Assume and Guarantee properties in the text boxes. Notice how grammar keywords are highlighted and flow ports are suggested as terms.



Expected results	The user is able to add contracts in a simple way and the typing of formal properties is eased by the editor.
Priority	Should

Table 17. Test Case WP3_TC_008b for WP3_CAC_009

ID	WP3_TC_008b
Scope	Improvement of Contract definition process.
Functionality ID	WP3_CAC_009
Related use cases	"Assign a contract to the component", and "Structure properties into contracts"
Input	None
Steps	<ol style="list-style-type: none"> 1. Browse the model using the "Project Explorer" view 2. Select the diagram from the "Model Explorer" view 3. Select Contract from the Palette and click on the diagram 4. Give a proper name to the Contract 5. Create a ContractProperty inside the Block/Component 6. In the "Property" view – Contract Tab, type the just created ContractProperty with the Contract
Expected results	A component updated with a property that represents the contract assignment.
Priority	Should

Table 18. Test Case WP3_TC_009a for WP3_CAC_011

ID	WP3_TC_009a (pending)
Scope	Overview of contract-based validation for behavioural models.
Functionality ID	WP3_CAC_011
Related use cases	"Perform contract-based verification of behavioural models"
Input	A CHESS model with some components, contracts with refinements, and state machines already defined.
Steps	<ol style="list-style-type: none"> 1. Browse the model using the "Model Explorer" view (e.g., go inside the "PhysicalArchitecture" package under "modelSystemView" package). 2. Open the Block Definition Diagram inside the package. 3. Select a component (in the "Model Explorer" view) or the corresponding graphical representation (in the Diagram Editor). The contracts and the state machines considered will be the ones associated to the selected component and the ones associated to its sub components. This operation includes recursively all the contracts and state machines along the subcomponents, from the root to the leaves of the system. 4. Right click on the selected component, then go to "CHESS"–"Functional Verifications"–"Check Contract Implementation on selected component". A popup appears. 5. Select the model of time of the system, "Hybrid" or "Discrete". 6. Receive the results of the analysis.
Expected results	None defined
Priority	Could

Table 19. Test Case WP3_TC_009b for WP3_CAC_011

ID	WP3_TC_009b
Scope	Overview of contract-based validation for behavioural models.
Functionality ID	WP3_CAC_011
Related use cases	"Perform contract-based validation for behavioural models"



Input	Component contracts and their refinement. CHESS finite state machines.
Steps	<ol style="list-style-type: none"> 1. Select a component (in the “Model Explorer” View) or the corresponding graphical representation (in the Diagram editor). 2. Right click on the selected component, then go to CHESS-Functional Verification – Model Checking on Selected Component. 3. A popup appears to set the parameters (Check Type, Algorithm Type and Property) of the command.
Expected results	A component updated with a property that represents the contract assignment
Priority	Could

Table 20. Test Case WP3_TC_010 for WP3_VVA_005

ID	WP3_TC_010 (pending)
Scope	Verify (model checking) state machines.
Functionality ID	WP3_VVA_005
Related use cases	“Perform contract-based verification of behavioural models”
Input	A model with some components and state machines already defined.
Steps	<ol style="list-style-type: none"> 1. Browse the model using the “Model Explorer” view (e.g., go inside the “PhysicalArchitecture” package under “modelSystemView” package). 2. Open the Block Definition Diagram inside the package. 3. Select a component (in the “Model Explorer” view) or the corresponding graphical representation (in the Diagram editor). The components behaviour to check will be the behaviour of the selected component and the behaviour of its sub components. This operation includes recursively all the behaviours from the root to the leaves of the selected component. 4. Right click on the selected component, then go to “CHESS”– “Functional Verifications”–“Model Checking on selected component”. A popup appears. 5. Select the nuXmv parameters and press OK. 6. Receive the results of the analysis.
Expected results	None defined
Priority	Must

Table 21. Test Case WP3_TC_011 for WP3_VVA_010

ID	WP3_TC_011
Scope	Model-based safety analysis.
Functionality ID	WP3_VVA_010
Related use cases	“Generate fault tree”
Input	A CHESS model with already defined nominal and ErrorModel state machines (e.g., the Battery_new project)
Steps	<ol style="list-style-type: none"> 1. Browse the model using the “Model Explorer” view and select the “System View” package. 2. Select the root component of the CHESS system, or the corresponding graphical representation (in the Diagram editor). 3. From the “Properties” view, stereotype the component as CHGaResourcePlatform. 4. In the “Model Explorer” view, go to the “DependabilityAnalysisView” package under the “Analysis View” package. 5. Right click on the package, then go to “New Diagram”–“Class Diagram”. 6. Right click again on the package, then go to “New Child”–“Class”.



	<ol style="list-style-type: none"> 7. Select the class from the “Properties” view, stereotype the component as “GaAnalysisContext”. 8. In the “Properties” view, select the stereotype “GaAnalysisContext”. 9. Set the platform attribute to the CHGaResourcePlatform entity and the context attribute write the top-level condition to be used for the FTA. 10. Go to the menu “CHESS” – “Analysis” – “Dependability” – “FTA with NuSMV (xSAP)”. A popup appears. 11. Select one of the analysis contexts found in the model from the drop-down list and press “OK”. 12. The external tool runs and displays the fault tree.
Expected results	The system shall allow the user to generate and view fault trees.
Priority	Must

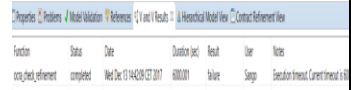
3.3 Test Results

Table 22 presents the results of the execution, the status and the validation responsible for the Test Cases in section 3.2. The instructions for installing the used testing environment are described in the AMASS Developer Guide [5]. The functionalities provided for System component specification and architecture-driven assurance in Prototype P1 are detailed in the AMASS User manual [4].

The test cases have been performed with the following machine configurations: Windows 10 Enterprise (64 bits) Operating system, Intel(R) Core(TM) i7-6700HQ and i7-5600U processors, CPU @ 2.60 GHz, 16 GB of RAM.

Nineteen test cases have been defined: seven test cases were successfully PASSED, eleven test cases were executed with the status PASSED_BUT, and one has not been executed since its implementation was not realized.

Table 22. Test results for the WP3 implemented functionalities

Test Case ID	Execution Results	Status	Rationale	Test Responsible
WP3_TC_001		Postponed	This requirement will be implemented in WP6.	
WP3_TC_002	The user is able to create fault injection state machines	Passed		FBK
WP3_TC_003a	The validation status in the “V&V Results” view	Passed_But	<p>Passed for model with discrete time (e.g., CHESS project “WBS_SM_Single_State”). However, an Execution timeout is triggered for other example such as CHESS project WBS.</p>  <p>An assumption is needed in the related use case “Validate components composition through</p>	A4T



			contracts-based design" specification in D2.3 [6].	
WP3_TC_003b	The validation status in the "V&V Results" view	Passed_ But	An assumption is needed in the related use case "Validate components composition through contracts-based design" specification in D2.3 [6].	A4T
WP3_TC_004a	Contracts assigned for each component in the "System Architectures" Column of "Hierarchical Model" view	Passed_ But	How the "Number of Subcomponent and Contract" is computed for subcomponents and contracts together?	A4T
WP3_TC_004b	Contracts assigned for each component in the "System Architectures" Column of "Hierarchical model" view	Passed_ But	How the "Number of Subcomponent and Contract" is computed for subcomponents and contracts together?	A4T
WP3_TC_005a	Contract refinement for each contract in the "Refined Contract" Column of "Contract Refinement" view	Passed		A4T
WP3_TC_005b	Contract refinement for each contract in the "Refined Contract" Column of "Contract Refinement" View	Passed		A4T
WP3_TC_006a	Number of sub-contracts for each refined contract	Passed		A4T
WP3_TC_006b	Number of sub-contracts for each refined contract	Passed		A4T
WP3_TC_007a	A status of the check is shown in the "V&V result" view	Passed_ But	An assumption is needed in the related use case specification in D2.3 [6]	A4T
WP3_TC_007b	A status of the check is shown in the "V&V result" view	Passed_ But	An assumption is needed in the related use case specification in D2.3 [6]	A4T
WP3_TC_007c	A status of the check is shown in the "V&V result" view	Passed_ But	An assumption is needed in the related use case specification in D2.3 [6]	A4T
WP3_TC_007d	A status of the check is shown in the "V&V result" view	Passed_ But	An assumption is needed in the related use case specification in D2.3 [6]	A4T
WP3_TC_008a	A component updated with a property that	Passed		A4T



	represents the contract assignment			
WP3_TC_008b	A component updated with a property that represents the contract assignment	Passed		A4T
WP3_TC_009a	A status of the check is shown in the "V&V result" view	Passed_ But	An assumption is needed in the related use case specification in D2.3 [6]	A4T
WP3_TC_009b	A status of the check is shown in the "V&V result" view	Passed_ But	An assumption is needed in the related use case specification in D2.3 [6]	A4T
WP3_TC_011	A fault tree is generated	Passed_ But	The fault tree is not generated for all the projects. There seems to be an issue with the export functionality.	FBK

4. Testing and Validation for WP4-related Blocks

4.1 Functionalities

The functionalities concerning the Assurance Case Specification and Multi-concern assurance blocks are defined in the deliverable D2.1 [7]. Table 23 is an excerpt of these functionalities planned for prototype P1 plus the ones postponed in prototype Core, their implementation status and the partner responsible for their implementation. Among the thirteen collected functionalities, two functionalities have not been implemented and are postponed for the next version of the AMASS platform.

Table 23. Assurance case Specification and multi-concern assurance functionalities

ID	Functionality	Status	Responsible
WP4_ACS_001	Edit an assurance case in a scalable way	Implemented	TEC
WP4_ACS_002	Argumentation architecture: Edit a modular structure (argument architecture) associated with a system and/or component	Implemented	TEC
WP4_ACS_003	Drag and drop argumentation patterns	Implemented	TEC
WP4_ACS_004	Semi-automatic generation of process arguments	Pending	MDH
WP4_ACS_005	Provide support for language formalization inside argument claims	Implemented	TEC
WP4_ACS_010	Provide the capability of generating a compositional assurance case argument	Implemented	TEC
WP4_DAM_001	Capability to model relationships between concerns	Implemented	TEC
WP4_DAM_002	Capability to capture conflicts occurring during system development and the trade-off process	Implemented	TEC
WP4_ACS_007	Argumentation import/export	Implemented	AIT (TEC)
WP4_ACS_006	Provide guidelines for argumentation	Pending	AIT
WP4_SDCA_002	System dependability co-verification and co-validation	Implemented	AIT, ANSYS
WP4_SDCA_003	The system shall allow combinations of safety and security analysis	Implemented	AIT
WP4_CMA_003	Contract based multi-concern assurance	Implemented	INT

4.2 Test Cases

This section presents the set of test cases defined to validate the implementation of the Assurance Case Specification and Multi-concern assurance blocks of Prototype P1. The test cases are based on the use case scenarios defined in the D2.3 deliverable [6] for the concerned functionalities when existing.

Table 24. Test Case WP4_TC_001 for WP4_ACS_001 functionality

ID	WP4_TC_001
Scope	Edit an assurance case in a scalable way.
Functionality ID	WP4_ACS_001
Related use cases	"Define and navigate an assurance case structure"
Input	A reference framework
Steps	1. Create an assurance project



	<ol style="list-style-type: none"> 2. Create a baseline from a big reference framework (ISO 26262) 3. Choose to create automatically the argumentation diagram 4. Browse the argument diagram elements 5. Create new elements, links 6. Update the elements 7. Delete some elements 8. Save the argumentation diagram 9. Create a diagram view 10. Drag and drop element from Outline menu to Diagram editor 11. Hide an element on the diagram 12. Delete an element on the diagram 13. Create a new diagram from the argumentation model
Expected results	Modified assurance case
Priority	Must

Table 25. Test Case WP4_TC_002 for WP4_ACS_010 functionality

ID	WP4_TC_002
Scope	Provide the capability of generating a compositional assurance case argument.
Functionality ID	WP4_ACS_010
Related use cases	"Define and navigate an assurance case structure"
Input	None
Steps	<p>For every argument module:</p> <ol style="list-style-type: none"> 1. Specify manually the claims set 2. Provide stated and valid assumptions applied to the claims 3. Specify contextual information to define or constraint the scope over which the arguments are assumed to be valid 4. Map claims (away goals) to the external claims (public goals) that support to (in other argument modules)
Expected results	A compositionally defined assurance case
Priority	Must

Table 26. Test Case WP4_TC_003 for WP4_ACS_005 functionality

ID	WP4_TC_003
Scope	Provide support for language formalization inside argument claims.
Functionality ID	WP4_ACS_005
Related use cases	None
Input	An argumentation model
Steps	<ol style="list-style-type: none"> 1. Create a vocabulary diagram 2. Add categories and terms 3. Open an argumentation diagram 4. Edit the description of the elements using the defined terms 5. Save the vocabulary on an xml file
Expected results	A vocabulary and an argumentation using it inside claims.
Priority	Must

Table 27. Test Case WP4_TC_004 for WP4_ACS_002 functionality

ID	WP4_TC_004
Scope	Argumentation architecture: edit a modular structure (argument architecture) associated with a system and/or component.



Functionality ID	WP4_ACS_002
Related use cases	"Define and navigate an assurance case structure"
Input	None
Steps	In an argumentation diagram the user will: 1. Define the appropriate granularity by using argument modules to encapsulate arguments. 2. Inside each argument module, include appropriate arguments taking into account: hazard mitigation, requirements, and integration. 3. Drag and drop argument module into the desired diagram of assurance case.
Expected results	The modular assurance structure for a given project has been detailed.
Priority	Must

Table 28. Test Case WP4_TC_005 for WP4_ACS_003 functionality

ID	WP4_TC_005
Scope	Argumentation architecture: edit a modular structure (argument architecture) associated with a system and/or component.
Functionality ID	WP4_ACS_003
Related use cases	"Reuse an argument pattern"
Input	Assurance argumentation is under edition.
Steps	1. Library of patterns is available to be used in a specific assurance case model. 2. Drag and drop argument pattern into the desired diagram of assurance case. 4. Pattern parameters must be defined by the user.
Expected results	Changes are registered
Priority	Must

Table 29. Test Case WP4_TC_006 for WP4_ACS_002 functionality

ID	WP4_TC_006
Scope	Argumentation architecture: edit a modular structure (argument architecture) associated with a system and/or component.
Functionality ID	WP4_ACS_002
Related use cases	"Reuse an argument pattern"
Input	A system architecture definition.
Steps	1. Select the component specification from the system component's architecture specification to be connected. 2. Select one of the architecture element (block, contract, ...). 3. Open the assurance case structure view. 4. Select one of the argument modules (claim, evidence, agreement) 5. Connect the argument module with the system architecture by drag and drop in the "OpenCert" tab property of the architecture element.
Expected results	System architecture and the assurance case specifications are correlated.
Priority	Must

Table 30. Test Case WP4_TC_007 for WP4_DAM_001 and WP4_DAM_002 functionality

ID	WP4_TC_007
Scope	Capability to model relationships between concerns.
Functionality ID	WP4_DAM_001, WP4_DAM_002
Related use cases	"Specify impact of claims"
Input	The current argumentation module has been created. The pieces of evidence addressed by the current project have been established.



Steps	For each of the claims referring to a specific concern 1. Analyse the impact of another claim 2. Specify the possible impact relationships described in D4.2 [17]: a) Dependency relationship, b) Conflicting relationship, or c) Supporting relationship
Expected results	The current Argumentation Module is under edition.
Priority	Must

Table 31. Test Case WP4_TC_008 for WP4_DAM_001 functionality

ID	WP4_TC_008
Scope	Capability to model relationships between concerns
Functionality ID	WP4_DAM_001
Related use cases	"Tag multi-concern to contracts"
Input	The contracts are already defined.
Steps	1. The user selects an existing contract. 2. The user associates selected contract to a property/concern.
Expected results	The contract has data associated referring the concern.
Priority	Must

Table 32. Test Case WP4_TC_009 for WP4_ACS_007 functionality

ID	WP4_TC_009
Scope	Argumentation import/export.
Functionality ID	WP4_ACS_007
Related use cases	None
Input	Argumentation exported and imported
Steps	1. The user edits an argumentation diagram. 2. The user exports this argumentation model in a file. 3. The user reuses this model to create a new argumentation diagram.
Expected results	Argumentation exported and imported.
Priority	Must

Table 33. Test Case WP4_TC_010 for WP4_SDCA_002 functionality

ID	WP4_TC_010
Scope	System dependability co-verification and co-validation.
Functionality ID	WP4_SDCA_002
Related use cases	None
Input	Dependability workflow engine (WEFACT) and Co-V&V tools
Steps	1. Define the requirements to be verified and validated 2. Create a Co-V&V Process (e.g., Verification of safety and Security concepts) 3. Link Requirement to Process 4. Create Co-V&V Tools 5. Link Tools to process 6. Execute a process step 7. Collect the Tools outputs in Co-V&V engine
Expected results	Outputs of Co-V&V tools, such as FMVEA tables or FT&AT.
Priority	Must

**Table 34.** Test Case WP4_TC_011 for WP4_SDCA_003 functionality

ID	WP4_TC_011
Scope	The system shall allow combinations of safety and security analysis.
Functionality ID	WP4_SDCA_003
Related use cases	"Define/Perform Safety/Security Analysis"
Input	System/component definition fully specified at the planned analysis level.
Steps	<ol style="list-style-type: none">1. Identify assets to be protected2. Select appropriate method and tool3. Decide on which level to analyse the [Sub-]System/Component4. Identify for each item all conceivable failure and threat modes with all possible causes and vulnerabilities and assess possibility to detect the failures/attacks5. Identify mitigation measures already in place
Expected results	Safety and Security artefacts (FMVEA or FT&AT) generated by the tool for Safety/Security Analysis.
Priority	Must

Table 35. Test Case WP4_TC_012 for WP4_CMA_003 functionality

ID	WP4_TC_012
Scope	The system must provide features that support contract based assurance with respect to multiple concerns; i.e. it must be possible to specify relations between safety contracts, security contracts and other-concerns-related contracts in order to take care of the influence of system modifications for mitigating the risks associated with one quality attribute on the contract belonging to another quality attribute.
Functionality ID	WP4_CMA_003
Related use cases	"Tag multi-concerns to contracts"
Input	CHESS project with the formal property already defined.
Steps	<ol style="list-style-type: none">1. Select the formal property (in the "Model Explorer" view) or the corresponding graphical representation (in the Diagram editor).2. In the "Property" view – Profile Tab, FormalProperty – concern, select the concern (unspecified/safety/security/performance).
Expected results	The concern is specified for the formal property
Priority	Must

4.3 Test Results

Table 36 presents, for each test case defined for the implemented Assurance Case Specification and Multi-concern assurance functionalities, the results of the execution, the status, a rationale when the execution failed and finally the AMASS project partner who is responsible for the validation of the test case. The functionalities are detailed in the AMASS User manual [4].

The test cases have been performed with the following machine configurations: Windows 10 Enterprise (64 bits) operating system, Intel(R) Core(TM) i7-6700HQ and i7-5600U processors, CPU @ 2.60 GHz, 16 GB of RAM. The instructions to install the used testing environment are described in the AMASS Developer Guide [5].

Between the twelve test cases that have been defined, eight test cases were successfully PASSED, three test cases were executed with the status PASSED_BUT, and one test case FAILED.

**Table 36.** Test results for the WP4 implemented functionalities

Test Case ID	Execution Results	Status	Rationale	Responsibility
WP4_TC_001	An argumentation model	Passed_But	Step 11: "Delete from diagram" menu option is always greyed.	CEA
WP4_TC_002	A compositionally defined assurance case	Passed		CEA
WP4_TC_003	A vocabulary and an argumentation using it inside claims	Failed	We created a vocabulary, both on a file or in the remote repository. However, we did not succeed in associating the vocabulary to the assurance case. When we use CTRL-SPACE during claim editing, nothing happens.	CEA
WP4_TC_004	A modular assurance structure	Passed		CEA
WP4_TC_005	An assurance argumentation	Passed		CEA
WP4_TC_006	A system architecture definition	Passed		CEA
WP4_TC_007	An argumentation module	Passed		CEA
WP4_TC_008	Contracts with associated concern	Passed		CEA
WP4_TC_09	Argumentation exported and imported	Passed		CEA
WP4_TC_010	Outputs or Co-V&V tools, such as FMVEA tables or FT&AT	Passed_But	WEFACT tool is installed perfectly thanks to WEFACT Handbook. The test steps are executed but a Co-V&V tool is not available for the full test.	A4T
WP4_TC_011	Safety and Security artefacts (FMVEA or FT&AT) generated by the tool for Safety/Security Analysis	Passed_But	WEFACT tool is installed and linked to Safety and Security tools for separated Safety and Security analysis, but the combined safety and security analysis is not ready for the full test.	A4T
WP4_TC_012	The concern is specified for the formal property	Passed		FBK

5. Testing and Validation for WP5-related Blocks

5.1 Functionalities

The functionalities concerning Evidence Management and Seamless interoperability blocks are defined in the D2.1 deliverable [7]. Table 37 is an excerpt of these functionalities, their implementation status and the AMASS project partner who is responsible for the validation of the test case. Three functionalities among the ten identified for this Prototype iteration have not been implemented and postponed for the next version of the AMASS platform.

Table 37. Evidence Management and seamless interoperability functionalities

ID	Functionality	Status	Responsibility
WP5_EM_006	Evidence information export	Implemented	TEC, UC3, TRC
WP5_EM_008	Visualization of chains of evidence	Pending	AMT, INT
WP5_EM_009	Suggestion of evidence traces	Pending	UC3, TRC
WP5_EM_012	Evidence trace verification	Pending	UC3, TRC
WP5_EM_015	Resource part selection	Implemented	AMT
WP5_TI_018	Extended standard-based interoperability	Implemented	UC3, TRC, FBK, HON
WP5_TI_017	Standards-based interoperability	Implemented	UC3, TRC, FBK, HON
WP5_TI_003	Tool chain deployment support	Implemented	UC3, TRC, FBK, HON
WP5_TI_005	System specification tools interoperability	Implemented	UC3, TRC, FBK
WP5_TI_006	V&V tools interoperability	Implemented	UC3, TRC, FBK, HON, UOM

5.2 Test Cases

Table 38 in this section defines the test cases to validate the implementation of the Evidence Management and Seamless interoperability basic building blocks of the Prototype Core. The test cases are based on the use case scenarios defined in the D2.3 deliverable [6] for the concerned functionalities when existing.

Table 38. Test Case WP5_TC_001 for WP5_EM_006 functionality

ID	WP5_TC_001
Scope	Evidence information export
Functionality ID	WP5_EM_006
Related use cases	"Link Artefact with External Tool"
Input	Assurance Project. An Artefact has been created.
Steps	1. Select an Artefact 2. Add a Resource to the Artefact 3. Specify the information of the Resource and the location of an External Tool in the Resource.
Expected results	Exported data to the external tool.
Priority	Must

**Table 39.** Test Case WP5_TC_002 for WP5_TI_006 functionality

ID	WP5_TC_002
Scope	V&V tools interoperability
Functionality ID	WP5_TI_006
Related use cases	"Specify Tool Connection Information"
Input	System / component model with formal properties and contracts
Steps	<ol style="list-style-type: none">1. Setup external V&V tools2. Check contract refinement using external V&V tool3. Check contract implementation using external V&V tool4. Verify V&V tool results
Expected results	External V&V tools produce adequate (expected) results
Priority	Should

Table 40. Test Case WP5_TC_003 for WP5_TI_003 functionality

ID	WP5_TC_003
Scope	Toolchain deployment
Functionality ID	WP5_TI_003
Related use cases	"Characterise Toolchain"
Input	External tools
Steps	<ol style="list-style-type: none">1. Configure at least two tools that will be part of a toolchain2. Configure at least one interaction between those tools3. Connect both tools4. Verify the connection and validate the expected result
Expected results	Tool chains can be "characterized" aka configured.
Priority	Can

Table 41. Test Case WP5_TC_004 for WP5_EM_015 functionality

ID	WP5_TC_004
Scope	Resource part selection
Functionality ID	WP5_EM_015
Related use cases	"Link Artefact with External Tool"
Input	An Artefact has been created
Steps	<ol style="list-style-type: none">1. The Assurance Manager selects an Artefact2. The Assurance Manager adds a Resource to the Artefact3. The Assurance Manager specifies the information about an External Tool in the Resource4. The Assurance Manager selects a part of the Resource5. The AMASS Platform retrieves data from the external tool6. The AMASS Platform exports data to the external tool
Expected results	The link with the external tool is stored in the AMASS Platform.
Priority	Should

Table 42. Test Case WP5_TC_005 for WP5_TI_017 and WP5_TI_018 functionalities

ID	WP5_TC_005
Scope	Extended standard-based interoperability
Functionality ID	WP5_TI_017, WP5_TI_018
Related use cases	"Characterise Toolchain"
Input	Tool information is available in the AMASS Tool Platform.
Steps	<ol style="list-style-type: none">1. The Assurance Manager selects the tools that will be part of the toolchain.



	<ol style="list-style-type: none"> The Assurance Manager specifies the interactions between the tools. The Assurance Manager specifies the necessary information to enable the toolchain. The Assurance Manager is informed about the success of toolchain connection. The Assurance Manager will be notified of the specified interactions between tools that are not supported.
Expected results	The toolchain information is available in the AMASS Tool Platform.
Priority	Must

Table 43. Test Case WP5_TC_006 for WP5_TI_005 functionality

ID	WP5_TC_006
Scope	System specification tools interoperability
Functionality ID	WP5_TI_005
Related use cases	"Specify Tool Connection Information"
Input	None
Steps	<ol style="list-style-type: none"> The Assurance Manager creates a new tool connection. The Assurance Manager specifies the required information to the tool connection. The Assurance Manager is provided information about the success of the tool connection. The required tool connection information can vary among tools, and it can include user, password, and a URL. <p>Alternatives can be offered for connection with a tool, e.g. ad-hoc connection or OSLC-based.</p>
Expected results	The tool connection information is available in the AMASS Tool Platform.
Priority	Should

5.3 Test Results

Table 44 presents, for each test case defined for the implemented Evidence Management and Seamless interoperability functionalities, the results of the execution, the status, a rationale when the execution failed and the AMASS project partner who is responsible for the validation of the test case. The installation instructions for the tools used for the validation are provided in the AMASS Developer Guide [5]. The AMASS user manual [4] was used to understand how the selected functionalities were working.

The test case WP5_TC_001 has been performed with the following machine configuration: Windows 10 Enterprise (64 bits) operating system, Intel(R) Core(TM) i7-6700HQ processor, CPU @ 2.60 GHz, 16 GB of RAM. The test cases WP5_TC_002 and WP5_TC_003 have been performed on Windows 7 Professional (64 bit) with Intel Xeon @ 2.8 GHZ CPU(W3530) with 12GB of RAM.

Between the six test cases that have been defined, one test case was successfully PASSED and five test cases FAILED.

Table 44. Test results for the WP5 implemented functionalities

Test Case ID	Execution Results	Status	Rationale	Responsibility
WP5_TC_001	Exported data to the external tool	Passed		A4T
WP5_TC_002		Failed	Incomplete test procedure. V&V tool setup partially passed, however, setup of system model with contracts and	AMT



			properties was not possible with prototype P1.	
WP5_TC_003		Failed	Missing tool features. This test failed due to missing tool functionality. External tools can be used in various places but there is neither a feature to “characterize” (model) them nor any mean to model toolchains.	AMT
WP5_TC_004		Failed	Nothing is said in the manual concerning how to specify a part of a resource. The implementation progress notes that Capra could provide support for it but no detail is given.	CEA
WP5_TC_005		Failed	The only use case associated to this requirement is “Characterize toolchain”. Nothing is said in the manual concerning the definition of toolchains.	CEA
WP5_TC_006		Failed	The requirement “WP5_TI_005” is too vague. It should better characterize the tools addressed. Neither the implementation progress nor the D5.5 [10], explain how this requirement is solved. We have considered that it is related to the use of OSLC-KM to access Papyrus models. We have tried to test it (import Papyrus model as evidence) but the corresponding web-services do not respond.	CEA

6. Testing and Validation for WP6-related Blocks

6.1 Functionalities

Table 45 is an excerpt of some functionalities defined in the D2.1 deliverable [7] for Compliance management and Cross and intra-domain reuse. It presents their implementation status and the AMASS project partner who is responsible for their validation. Among the planned functionalities for prototype P1, thirteen functionalities have been implemented and two functionalities are still pending at the time of the validation.

Table 45. Compliance Management and cross and intra-domain reuse functionalities

ID	Functionality	Status	Responsibility
WP6_CM_002	Tailoring of Standards models to specific projects	Implemented	MDH + TEC
WP6_CM_005	Compliance Monitoring	Implemented	MDH + TEC
WP6_CM_008	Process Compliance (informal) management	Implemented	TEC
WP6_CM_006	Compliance status to externals.	Pending	MDH + TEC
WP6_CM_001	Retrieving, digitalizing and storing of a set of industrial standards (including the parts, objectives, practices, goals/requirements, criticality levels from the standards).	Implemented	TEC
WP6_RA_001	Intra-Domain, Intra standard, Reuse Assistance	Implemented	TEC
WP6_RA_002	Intra-Domain, Cross standards, Reuse Assistance	Implemented	TEC
WP6_RA_003	Intra-Domain, Cross versions, Reuse Assistance	Implemented	TEC
WP6_RA_004	Cross-Domain Reuse Assistance	Implemented	TEC
WP6_RA_005	Intra-Domain, Intra standard, Different Stakeholders, Reuse/Integration Assistance	Pending	TEC
WP6_RA_006	Reuse of pre-developed components and their accompanying artefact.	Implemented	TRC
WP6_PPA_001	The AMASS tools must support variability management at process level	Implemented	MDH
WP6_PPA_002	Semi-automatic generation of product arguments	Implemented	MDH
WP6_PPA_003	Semi-automatic generation of process arguments	Implemented	MDH
WP6_PPA_004	The AMASS tools must support management of variability at the component level	Implemented	MDH

6.2 Test Cases

Table 46 and Table 47 in this section define the test cases to validate the implementation of the Compliance management and Cross and intra-domain reuse functionalities. The test cases have been defined based on the use case scenarios defined in the D2.3 deliverable [6] for the concerned functionalities when existing.

Table 46. Test Case WP6_TC_001 for WP6_CM_001 functionality

ID	WP6_TC_001
Scope	Retrieve, digitalize and store a set of norms, recommendations, standards, or quality models.



Functionality ID	WP6_CM_001
Related use cases	"Capture information from standards"
Input	Standard information
Steps	<ol style="list-style-type: none"> 1. Create a new standard model. 2. Specify the characteristics that define the standard in the properties view. 3. Structure/Categorize the standard by parts, objectives, activities, practices, goals and requirements. 4. Describe the parts, objectives, activities, practices, goals and requirements contained in the standard in the properties view.
Expected results	Standard model
Priority	Must

Table 47. Test Case WP6_TC_002 for WP6_RA_006 functionality

ID	WP6_TC_002
Scope	Reuse of pre-developed components and their accompanying artefact.
Functionality ID	WP6_RA_006
Related use cases	None
Input	Pre-developed components and their accompanying artefact.
Steps	None
Expected results	Import of pre-developed components and their accompanying artefact.
Priority	Must

Table 48. Test Case WP6_TC_003 for WP6_RA_001, WP6_RA_005 functionalities

ID	WP6_TC_003
Scope	Intra-Domain, Intra standard, Reuse Assistance
Functionality ID	WP6_RA_001, WP6_RA_005
Related use cases	"Assist for Cross-System Assurance Assets Reuse"
Input	<ul style="list-style-type: none"> • A source assurance project which includes assurance assets (evidence, process, argumentation, compliance models) . • A target assurance project.
Steps	<ol style="list-style-type: none"> 1. The actor opens the newly created assurance project (target project), starts the reuse assistant, and selects the reusable assurance project (source project). 2. A number of assurance models from the source project are available for navigation, including evidence, process, argumentation and baseline (compliance information) models and can be individually selected. 3. Once selected a specific model in the source project, the actor can call the impact analysis functionality to select model elements and visualise the dependent (impacted) model elements. 4. Additional selection criteria can be applied such as e.g. subset of assurance assets associated to a given criticality level. 5. Once selected all the desired model elements to be reused from the various models, the actor can store the subset before executing the reuse operation.
Expected results	AMASS models updated according to the reuse scope, including evidence models, argumentation models, process models and compliance information.
Priority	Must

**Table 49.** Test Case WP6_TC_004 for WP6_CM_002 functionality

ID	WP6_TC_004
Scope	Create, modify and drop assurance information.
Functionality ID	WP6_CM_002
Related use cases	"Manage Assurance Project"
Input	Library and configuration models exported from EPF.
Steps	1. Import process related information from EPF.
Expected results	Process and Artefact (Evidence) models.
Priority	Must

Table 50. Test Case WP6_TC_005 for WP6_CM_002, WP6_CM_008 functionalities

ID	WP6_TC_005
Scope	Create, modify and drop assurance information.
Functionality ID	WP6_CM_002, WP6_CM_008
Related use cases	"Manage Assurance Project"
Input	A model containing information of the standard available in the platform
Steps	1. Create a new assurance project 2. Specify the baseline in association with a standard which will be followed in the project 3. Specify the compliance maps/links through the project lifecycle.
Expected results	Assurance Project
Priority	Must

Table 51. Test Case WP6_TC_006 for WP6_CM_005 functionality

ID	WP6_TC_006
Scope	Information about the assurance activities.
Functionality ID	WP6_CM_005
Related use cases	"Monitor Assurance Project Status"
Input	Assurance project in the platform.
Steps	1. Select an assurance project 2. Define a filter to find specific compliance information
Expected results	Compliance information
Priority	Must

Table 52. Test Case WP6_TC_007 for WP6_RA_002, WP6_RA_003, WP6_RA_004 functionality

ID	WP6_TC_007
Scope	Intra-Domain, Intra/Cross standards, Cross versions, Reuse Assistance
Functionality ID	WP6_RA_002, WP6_RA_003, WP6_RA_004
Related use cases	"Assist for Cross-Standards Assurance Assets Reuse"
Input	<ul style="list-style-type: none">• A source assurance project which includes assurance assets (evidence, process, argumentation, compliance models)• It exists an equivalence map model between the source and target standards• A target assurance project based in a standard model with equivalence maps with the standards model in which is based the source assurance project.
Steps	1. The actor opens the newly created assurance project (target project), starts the reuse assistant, and selects the reusable assurance project (source project).



	<p>2. Assurance models from the target project can already exist or may be created automatically. In the latter case, the model elements will follow the same structure and naming as the standards (baseline in this case) model.</p> <p>3. A number of assurance models from the target project are available for navigation, including evidence, process, argumentation and baseline (compliance information) models and can be individually selected.</p> <p>4. Once a source model element is selected, the actor can discover reuse opportunities by using equivalence maps (extended use case).</p> <p>5. Once selected all the desired model elements to be reused from the various models, the actor can store the subset before executing the reuse operation. Storing the subset of assets permits to check the actions that need to be performed before executing them. This is important for example if it is necessary to have an internal approval or if the user wants to check the impact of the operation before executing it.</p>
Expected results	AMASS models updated according to the reuse scope, including evidence models, argumentation models, process models and compliance information
Priority	Must

Table 53. Test Case WP6_TC_008 for WP6_RA_002, WP6_RA_003, WP6_RA_004 functionalities

ID	WP6_TC_008
Scope	Intra-Domain, Intra/Cross standards, Cross versions, Reuse Assistance.
Functionality ID	WP6_RA_002, WP6_RA_003, WP6_RA_004
Related use cases	"Discover Reuse Opportunities by using Standards Equivalences"
Input	An equivalence map model between the source and target standards.
Steps	<ol style="list-style-type: none"> 1. Select target model elements. 2. Visualise the equivalent model elements in the source assurance projects. 3. Look at the reuse post-conditions identified in the equivalence map model. 4. Decide if the reusable element will be selected for reuse.
Expected results	Identification of model elements with associated equivalence standard model elements.
Priority	Must

Table 54. Test Case WP6_TC_009 for WP6_RA_001, WP6_RA_002, WP6_RA_003, WP6_RA_004, WP6_RA_005 functionalities

ID	WP6_TC_009
Scope	Intra/Cross-Domain, Intra/Cross standard, Cross versions, Different Stakeholders, Reuse/Integration Assistance
Functionality ID	WP6_RA_001, WP6_RA_002, WP6_RA_003, WP6_RA_004, WP6_RA_005
Related use cases	"Reuse Selected Assurance Assets"
Input	A subset of assurance assets has been selected.
Steps	<ol style="list-style-type: none"> 1. Visualise the subset of selected assurance assets 2. Perform reuse operation 3. Visualise results of the reuse operation
Expected results	Copy operation in the AMASS repository.
Priority	Must

Table 55. Test Case WP6_TC_010 for WP6_PPA_001 functionality

ID	WP6_TC_010
Scope	AMASS tools must support variability management at process level.
Functionality ID	WP6_PPA_001



Related use cases	"Manage process variability"
Input	A Process library (Base Model)
Steps	<ol style="list-style-type: none"> 1. The user imports the Base Model into a project. 2. The user manages variability via the Variability, Resolution, an Realization editors. 3. The user generates/exports the new process model, obtained as tailoring of the Base Model.
Expected results	A new Base Model.
Priority	Must

Table 56. Test Case WP6_TC_011 for WP6_PPA_002 functionality

ID	WP6_TC_011
Scope	Semi-automatic generation of product arguments.
Functionality ID	WP6_PPA_002
Related use cases	"Semi-automatic generation of product arguments"
Input	<p>Strong and weak component contracts shall be already defined and associated with claims, context statements and evidence artefacts.</p> <p>The weak contracts shall be either selected for usage in the given context, or all weak contract assumptions shall be validated.</p> <p>The contract refinement analysis shall be already performed, either for the selected contracts, or for all the weak contracts.</p>
Steps	<ol style="list-style-type: none"> 1. The user selects the "Generate argumentation fragments" functionality. 2. The user selects either new or existing assurance project as the destination for the argument-fragments. 3. The ARTA validates the system model and extract the information needed for the argument-fragment generation for each component. 4. The ARTA generates the corresponding argument-fragments, and notifies the user of their location.
Expected results	An argument model with the argument fragments included.
Priority	Should

Table 57. Test Case WP6_TC_012 for WP6_PPA_003 functionality

ID	WP6_TC_012
Scope	Semi-automatic generation of process arguments
Functionality ID	WP6_PPA_003
Related use cases	"Automatic generation of process arguments"
Input	A process model
Steps	<ol style="list-style-type: none"> 1. The user selects the "Generate argumentation fragments" functionality. 2. The user selects either new or existing assurance project as the destination for the argument-fragments. 3. The information needed for the argument-fragment generation is extracted from the process model. 4. The corresponding argument-fragments are generated; the location is notified to the user.
Expected results	An argument model with the argument fragments included.
Priority	Should

Table 58. Test Case WP6_TC_013 for WP6_PPA_004 functionality

ID	WP6_TC_013
-----------	-------------------



Scope	The AMASS tools must support management of variability at the component level.
Functionality ID	WP6_PPA_004
Related use cases	"Manage product variability"
Input	A component warehouse (Base Model)
Steps	<ol style="list-style-type: none"> 1. The user manages variability via the variability, resolution, realization editors. 2. The user generates/export the new component model, obtained as tailoring of the Base Model.
Expected results	A new component model.
Priority	Shall

6.3 Test Results

Table 59 presents, for each test case defined for WP6 related functionalities, the results of the execution, the status, a rationale when the execution was not fully satisfying the expected results, and the AMASS project partner who is responsible for the validation of the test case.

The installation instructions for the validation environment and the description of the selected functionalities are respectively found in the AMASS Developer Guide [5] and the AMASS User Manual [4]. As testing data, we use a database backup containing examples of assurance project and evidence model. We also used some files exported from EPF tool for the process model⁵.

The test cases have been performed with the following machine configuration: Windows 7 Enterprise (64 bits) operating system, Intel Core i7-5600U processor, CPU @ 2.60 GHz, 16 GB of RAM.

Between the thirteen test cases that have been defined, three test cases were successfully PASSED, one test case was executed with the status PASSED_BUT, and one test case FAILED.

Table 59. Test results for the WP6 implemented functionalities

Test Case ID	Execution Results	Status	Rationale	Responsibility
WP6_TC_001	A standard's model with its characteristics	Passed_ But	The argumentation model is not generated from the baseline, but together with the baseline from a new assurance project creation. When a different baseline is created, we are not able to generate the corresponding argumentation model.	CEA, AMT
WP6_TC_002	Import of pre-developed components and their accompanying artefact	Failed	No description of an usage scenario defined in D2.3 deliverable [6][6].	VIF
WP6_TC_003		Postponed	Consistent requirement information, (owner, allocated prototype, implementation status)	

⁵ The EPF files used are located in the SVN repository:
https://services-medini.kpit.com/svn/AMASS_collab/WP-transversal/ImplementationTeam/PrototypeCore/Vaditation_Data/EPF/Exported XML



			were not available at the time of the validation.	
WP6_TC_004	Process and artefact models imported from EPF.	Passed		CEA
WP6_TC_005	An Assurance Project with compliance links done. Summary can be checked through the mapping table.	Passed		CEA
WP6_TC_006	The Compliance information related to a specific element type (activity, requirement, etc.).	Passed		CEA
WP6_TC_007		Postponed	Consistent requirement information, (owner, allocated prototype, implementation status) were not available at the time of the validation	
WP6_TC_008		Postponed	Consistent requirement information, (owner, allocated prototype, implementation status) were not available at the time of the validation	
WP6_TC_009		Postponed	Consistent requirement information, (owner, allocated prototype, implementation status) were not available at the time of the validation	
WP6_TC_010		Postponed	Consistent requirement information, (owner, allocated prototype, implementation status) were not available at the time of the validation	
WP6_TC_011		Postponed	Consistent requirement information, (owner, allocated prototype, implementation status) were not available at the time of the validation	
WP6_TC_012		Postponed	Consistent requirement information, (owner, allocated prototype, implementation status) were not available at the time of the validation	



WP6_TC_013		Postponed	Consistent requirement information, (owner, allocated prototype, implementation status) were not available at the time of the validation	
------------	--	-----------	--	--

7. Prototype P1 Validation Synthesis

7.1 Analysis of Test Results

Table 60 summarizes the implementation status of the planned functionalities at the time of the release of the prototype P1. In total, 50 functionalities have been planned for the prototype P1: 11 functionalities have been postponed, while 39 functionalities have successfully been implemented.

Table 60. Prototype P1 Implementation Status

Functionalities	WP3 related System Component	WP4 related Assurance Case Specification	WP5 related Evidence Management	WP6 related Compliance Management	AMASS prototype P1
Implemented	8	11	7	13	39
Pending	4	2	3	2	11
Total	12	13	10	15	50

We have defined 50 test cases to test and validate the (39) implemented functionalities: Nineteen test cases have been successfully PASSED. Fifteen test cases result with the status PASSED_BUT, seven test cases FAILED to provide the expected results. Finally, eight test cases have been postponed for being tested during next validation.

For each test case with the status FAILED or PASSED_BUT, we have created a ticket corresponding to each problem identified in the software or user guide in the AMASS wiki to report them to the implementation responsible. It results in 13 opened tickets. In priority, we must address the problems described in these tickets with respect to new developments and implement the pending functionalities before tackling the prototype P2 functionalities implementation. The future D2.8 deliverable about the final validation of AMASS Platform will review these tickets statuses and how they have been taken into account.

Table 61. Results of the test cases for prototype P1 implemented functionalities

Test Results Status	WP3 related functionalities	WP4 related functionalities	WP5 related functionalities	WP6 related functionalities	AMASS prototype P1
Passed	7	8	1	3	19
Passed but	11	3	0	1	15
Failed	0	1	5	1	7
Postponed	1	0	0	8	9
Total	19	12	6	13	50

7.2 Recommendations

Although the recommendations elicited during the prototype Core validation have been followed, there is still some room for improvements. Some required functionalities have not been tested because of some issues that the validation team faced:

- Some usage scenarios are defined vaguely and are found difficult to interpret.
- There is missing information about the related use case(s) or required guideline(s) for some requirements.
- Some requirements are cross- tools and or cross-WP, sometimes leading to duplication among them.



- The set of inputs needed for performing some use cases are incompletely defined.
- The status of some requirements was not up to date at the time of the validation.

To avoid such issues for prototype P2 validation, we recall some recommendations already elicited during last validation:

- The requirements, use case scenarios, guidelines and user manual must be updated to create a better alignment between these documents. Ideally, we need a homogeneous (in the different WP3-6) description allowing to follow the links between the requirements that were planned, those that have been implemented, how prototype functionalities solve these requirements, which are the relevant use cases, and how these use cases are realized in terms of actions described in the user manual.
- To enhance further the validation results, the test cases definition by the validation team must be carried out in closer collaboration with the implementation team prior to their execution, to early identify any comprehension discrepancies of the implemented functionalities. We propose defining a test cases review and validation phase before their execution.

Furthermore, the requirements information, (i.e. owner, prototype number on which it should be made available, implementation status, modification date, etc.), must be updated, and that in a coherent manner within all the WPs. For example, it was missing the information whether an already implemented and satisfactory tested requirement for prototype Core has been enhanced for some reason for prototype P1, so that the validation team can confirm that the test status of the concerned requirement is still valid.



8. Conclusion

This report reflects the results of testing and validation on the AMASS prototype P1. The validation has been based on an analysis of the planned requirements and corresponding functionalities planned for the AMASS platform. These items have been refined into test cases that are compatible with the current developments of the AMASS platform. The previous validation results of prototype Core have been revised as well as the functionalities that were postponed for P1. Some issues were detected and reported as well as recommendations given for improvement.

Three main topics will be tackled in the prototype P2 validation phase:

1. Perform testing and validation of pending (not implemented) and postponed (not tested) functionalities with respect to old developments prior to tackling the new developments.
2. Validate the new implemented tool parts.
3. Validate that the AMASS platform is integrated in a comprehensive toolset operating at TRL 5.



Abbreviations and Definitions

AMASS	Architecture-driven, Multi-concern and Seamless Assurance and Certification of Cyber-Physical Systems
API	Application Programming Interface
ARTA	AMASS Reference Tool Architecture
AUTOSAR	AUTomotive Open System ARchitecture
BVR	Base Variability Resolution
CDO	Connected Data Objects
CPS	Cyber Physical Systems
CPU	Central Processing Unit
EPF	Eclipse Process Framework
FMVEA	Failure Modes, Vulnerabilities and Effect Analysis
FT&AT	Fault Tree & Attack Tree
FTA	Fault tree analysis
GB	Gigabyte
HMI	Human Machine Interface
IMA	Integrated Modular Avionics
ISO	International Organization for Standardization
KM	Knowledge Management
NuSMV	New Symbolic Model Verifier (a symbolic model checker tool for finite state systems)
OCRA	Othello Contracts Refinement Analysis
OPENCOS	Open Platform for Evolutionary Certification Of Safety-critical Systems
OSLC	Open Services for Lifecycle Collaboration
RAM	Random-access memory
STO	Scientific and Technical Objective
URL	Uniform Resource Locator
TRL	Technology Readiness Level
V&V	Verification & Validation
WBS	Work Break Down Structure
WP	Workpackage
XML	eXtensible Markup Language



References

- [1] OPENCROSS project. 2015. <http://www.opencross-project.eu>
- [2] SafeCer Project. 2015. <http://safecer.eu>
- [3] PolarSys. <https://www.polarsys.org>
- [4] AMASS project: Prototype P1_ User Manual⁶. 2017.
https://services.medini.eu/svn/AMASS_collab/WP-transversal/ImplementationTeam/PrototypeP1/AMASS_PrototypeP1_UserManual.docx
- [5] AMASS Developer Guide⁷. 2017
https://services.medini.eu/svn/AMASS_collab/WP-transversal/ImplementationTeam/PrototypeP1/AMASS_PrototypeP1_DeveloperGuide.doc
- [6] AMASS D2.3 AMASS reference architecture (b). 30 September 2017.
- [7] AMASS [D2.1 Business cases and high-level requirements](#). 28 February 2017.
- [8] AMASS [D3.5 - Prototype for architecture-driven assurance \(a\)](#). 23 December 2016.
- [9] AMASS [D4.5 - Prototype for multiconcern assurance \(a\)](#). 31 January 2017.
- [10] AMASS [D5.5 - Prototype for seamless interoperability \(a\)](#). 31 March 2017.
- [11] AMASS [D6.5 - Prototype for cross/intra-domain reuse \(a\)](#). 31 March 2017.
- [12] CHESS project. <http://www.chess-project.org/>
- [13] AMASS D3.7 Methodological guide for architecture-driven assurance (a). 30 November 2017.
- [14] AMASS D4.7 Methodological guide for cross/intra-domain reuse (a). 30 December 2017.
- [15] AMASS D5.7 Methodological guide for seamless interoperability (a). 30 December 2017.
- [16] AMASS D6.7 Methodological guide for cross/intra-domain reuse (a). 31 January 2018.
- [17] AMASS D4.2 Design of the AMASS tools and methods for multiconcern assurance (a), 30 June 2017.
- [18] OMG - Semantics of Business Vocabulary and Rules™ (SBVR™) version 1.3, 2015
<http://www.omg.org/spec/SBVR/1.3>
- [19] WEFAC <http://www.ait.ac.at/en/research-fields/verification-validation/methods-and-tools/wefact/>
- [20] Eclipse Process Framework Project. <https://eclipse.org/epf/>
- [21] OSLC, <http://open-services.net/specifications/>
- [22] OSLC-KM for Knowledge Management <http://trc-research.github.io/spec/km/>, Llorens, J., Morato, J., Genova, G., Fuentes, M., Quintana, V., & Díaz, I. (2004). RHSP: An information representation model based on relationship. *Studies in fuzziness and soft computing*, 159, 221-253.
- [23] Papyrus Eclipse project: <https://eclipse.org/papyrus/>
- [24] Capra project, <https://projects.eclipse.org/proposals/capra>
- [25] AUTomotive Open System ARchitecture <http://www.autosar.org>
- [26] Gaska, T., Watkin, C., & Chen, Y. (2015). Integrated modular avionics-past, present, and future. *IEEE Aerospace and Electronic Systems Magazine*, 30(9), 12-23.

⁶ The current User Manual is a draft document; the final version of the manual will be integrated in D2.5 - AMASS User guidance and methodological framework (m31).

⁷ The current Developer Guide is a draft document; the final version of the manual will be integrated in D2.5 - AMASS User guidance and methodological framework (m31).



Appendix A: Validation status of the AMASS Prototype P1

Table 62 summarizes the validation status of the prototype P1 implementation. Each functionality of the prototype P1 is traced to the test cases that evaluated them, if existing. The colour code indicates the status of associated test cases to the functionalities:

- Green indicates all test cases have the status PASSED, so the functionality is correctly implemented.
- Red indicates that the test cases FAILED, hence the functionality was not correctly implemented.
- Orange indicates that the test cases PASSED BUT we identified some needed improvements to fully meet the expected results for the functionality.
- Yellow indicates that the validation have been postponed for the next prototype P2 validation, either because the requirements were not implemented yet, or some information were missing to perform the validation.

Table 62. Prototype P1 Functionalities Status

Functionality		Test Cases Status
System component specification and architecture driven assurance		
WP3_VVA_004	Trace requirements validation checks	Yellow
WP3_SC_007	Fault injection (includes faulty behaviour of a component)	Green
WP6_PPA_004	The AMASS tools must support specification of variability at the component level	Yellow
WP3_CAC_001	Validate composition of components by validating their contracts	Orange
WP3_CAC_005	General management of contract-component assignments	Orange
WP3_CAC_006	Refinement-based overview	Green
WP3_CAC_007	Overview of check refinements results	Green
WP3_CAC_008	Contract-based validation and verification	Orange
WP3_CAC_009	Improvement of Contract definition process	Green
WP3_CAC_011	Overview of contract-based validation for behavioural models	Orange
WP3_VVA_005	Verify (model checking) state machines	Yellow
WP3_VVA_010	Model-based safety analysis	Orange
WP3_VVA_002	Trace model-to-model transformation	Yellow
Assurance Case Specification and multi-concern assurance		
WP4_ACS_001	Edit an assurance case in a scalable way	Orange
WP4_ACS_002	Argumentation architecture: Edit a modular structure (argument architecture) associated with a system and/or component	Green
WP4_ACS_003	Drag and drop argumentation patterns	Green
WP4_ACS_004	Semi-automatic generation of process arguments	Yellow
WP4_ACS_005	Provide support for language formalization inside argument claims	Red
WP4_ACS_010	Provide the capability of generating a compositional assurance case argument	Green
WP4_DAM_001	Capability to model relationships between concerns	Green
WP4_DAM_002	Capability to capture conflicts occurring during system development and the trade-off process	Green
WP4_ACS_007	Argumentation import/export	Green
WP4_ACS_006	Provide guidelines for argumentation	Yellow
WP4_SDCA_002	System dependability co-verification and co-validation	Orange
WP4_SDCA_003	The system shall allow combinations of safety and security analysis	Orange
WP4_CMA_003	Contract based multi-concern assurance	Green
Evidence Management and seamless interoperability		
WP5_EM_006	Evidence information export	Green
WP5_EM_008	Visualization of chains of evidence	Yellow
WP5_EM_009	Suggestion of evidence traces	Yellow
WP5_EM_012	Evidence trace verification	Yellow
WP5_EM_015	Resource part selection	Red



WP5_TI_018	Extended standard-based interoperability	
WP5_TI_017	Standards-based interoperability	
WP5_TI_003	Tool chain deployment support	
WP5_TI_005	System specification tools interoperability	
WP5_TI_006	V&V tools interoperability	
Compliance management and cross/intra-domain reuse		
WP6_CM_006	Compliance status to externals	
WP6_CM_001	Retrieving, digitalizing and storing of a set of industrial standards (including the parts, objectives, practices, goals/requirements, criticality levels from the standards)	
WP6_RA_006	Reuse of pre-developed components and their accompanying artefact	
WP6_CM_004	Triggering compliance Checking	
WP6_CM_009	Process Compliance (formal) management	
WP6_SEM_001	Semantics-based mapping of standards	
WP6_PPA_005	The AMASS tools must support variability management at the assurance case level	
WP6_CM_002	Tailoring of Standards models to specific projects	
WP6_CM_005	Compliance Monitoring	
WP6_CM_008	Process Compliance (informal) management	
WP6_RA_001	Intra-Domain, Intra standard, Reuse Assistance	
WP6_RA_002	Intra-Domain, Cross standards, Reuse Assistance	
WP6_RA_003	Intra-Domain, Cross versions, Reuse Assistance	
WP6_RA_004	Cross-Domain Reuse Assistance	
WP6_RA_005	Intra-Domain, Intra standard, Different Stakeholders, Reuse/Integration Assistance	
WP6_RA_006	Reuse of pre-developed components and their accompanying artefact.	
WP6_PPA_001	The AMASS tools must support variability management at process level	
WP6_PPA_002	Semi-automatic generation of product arguments	
WP6_PPA_003	Semi-automatic generation of process arguments	
WP6_PPA_004	The AMASS tools must support management of variability at the component level	