ECSEL Research and Innovation actions (RIA)

AMASS

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

Integrated AMASS platform (c)
D2.8

Work Package:	WP2 Reference Architecture and Integration
Dissemination level:	PU = Public
Status:	Final
Date:	14th December 2018
Responsible partner:	Morayo Adedjouma/ Bernard Botella (CEA)
Contact information:	{morayo.adedjouma, bernard.botella } AT cea.fr
Document reference:	AMASS_D2.8_WP2_CEA_V1.0

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'energie Atomique et aux Energies Alternatives (CEA)
A. Debiasi, L. Cristoforetti	Fondazione Bruno Kessler (FBK)
Isaac Moreno	Thales Alenia Spain (TAS)
S. Puri	Intecs (INT)
Isaac Moreno	Thales Alenia Space – Spain (TAS)
Marc Sango	Alliance pour les technologies de l'informatique (A4T)
Luis Alonso	The Reuse Company (TRC)
Helmut Martin, Bernhard Winkler, Robert Bramberger	Virtual Vehicle (VIF)
Jan Mauersberger, Ines Hoffmann	Medini Technologies AG (KMT)
Angel López, Garazi Juez, Alejandra Ruiz	Tecnalia Research & Innovation (TEC)

Reviewers

Reviewers Names	Organisation
Ran Bi, Daniel Wright (Peer-reviewers)	Rapita Systems (RPT)
Detlef Scholle, Staffan Skogbe (Peer-reviewers)	ALTEN SE (ALT)
Cristina Martinez (Quality review)	Tecnalia Research & Innovation (TEC)
Barbara Gallina (TC-review)	Maelardalens Hoegskola (MDH)

TABLE OF CONTENTS

Executive Summary **7**

1. Introduction **8**
1.1 Scope 8
1.2 Purpose of the deliverable 9
1.3 Relations to other deliverables 9
1.4 Structure of this document 10

2. AMASS Platform Architecture **11**
2.1 Conceptual and Implementation Architecture 11
2.2 AMASS Platform Tools 12
2.3 Installation and User Manuals 13
2.4 AMASS Prototype P1 Validation Results 13
2.5 AMASS Prototype P2 Testing and Validation Methodology 13
3.1 Functionalities 16
3.2 Test Cases 16
3.3 Test Results 25

4. Testing and Validation for WP4-related Blocks **27**
4.1 Functionalities 27
4.2 Test Cases 27
4.3 Test Results 31

5. Testing and Validation for WP5-related Blocks **33**
5.1 Functionalities 33
5.2 Test Cases 34
5.3 Test Results 42

6. Testing and Validation for WP6-related Blocks **45**
6.1 Functionalities 45
6.2 Test Cases 45
6.3 Test Results 51

7. AMASS Prototype P2 Validation Summary **52**

8. Conclusion **54**

Abbreviations and Definitions **55**

References **57**

List of Figures

Figure 1. AMASS Reference (High-Level) Architecture (Prototype P2) 11
Figure 2. Functional Decomposition of the AMASS Platform 12
Figure 3. Validation status of the AMASS Platform P1 13
Figure 4. AMASS Prototype P2 validation and testing process 14
Figure 5. AMASS platform features validation synthesis per WP 52
Figure 6. AMASS platform features validation 53

List of Tables

Table 1. System Component Specification and Architecture driven assurance functionalities....16
Table 2. Test Case WP3_FBK_TC_001 for WP3_APL_001....16
Table 3. Test Case WP3_FBK_TC_002 for WP3_APL_002....17
Table 4. Test Case WP3_FBK_TC_003 for WP3_APL_003....18
Table 5. Test Case WP3_FBK_TC_004 for WP3_APL_005....18
Table 6. Test Case WP3_A4T_TC_001 for WP3_VVA_004....19
Table 7. Test Case WP3_A4T_TC_002 for WP3_VVA_005....19
Table 8. Test Case WP3_A4T_TC_003 for WP3_CAC_001....19
Table 9. Test Case WP3_A4T_TC_004 for WP3_CAC_005....20
Table 10. Test Case WP3_A4T_TC_005 for WP3_CAC_008....20
Table 11. Test Case WP3_A4T_TC_006 for WP3_CAC_008....20
Table 12. Test Case WP3_A4T_TC_007 for WP3_CAC_011....21
Table 13. Test Case WP3_A4T_TC_008 for WP3_VVA_010....21
Table 14. Test Case WP3_A4T_TC_009 for WP3_VVA_002....22
Table 15. Test Case WP3_CEA_TC_001 for WP3_SAM_003 and WP3_VVA_012....22
Table 16. Test Case WP3_CEA_TC_002 for WP3_SAM_004....23
Table 17. Test Case WP3_CEA_TC_003 for WP3_VVA_003....23
Table 18. Test Case WP3_CEA_TC_004 for WP3_VVA_007....23
Table 19. Test Case WP3_CEA_TC_005 for WP3_SAM_002....24
Table 20. Test Case WP3_CEA_TC_006 for WP3_VVA_011....24
Table 21. Test Case WP3_CEA_TC_007 for WP3_VVA_006....25
Table 22. Test results for functionalities implemented in WP3....25
Table 23. Assurance case Specification and multi-concern assurance functionalities....27
Table 24. Test Case WP4_CEA_TC_001 for WP4_ACS_001....27
Table 25. Test Case WP4_CEA_TC_002 for WP4_ACS_005....28
Table 26. Test Case WP4_CEA_TC_003 for WP4_ACS_004....28
Table 27. Test Case WP4_CEA_TC_004 for WP4_ACS_008....28
Table 28. Test Case WP4_CEA_TC_005 for WP4_ACS_010....29
Table 29. Test Case WP4_CEA_TC_006 for WP4_SDCA_002....29
Table 30. Test Case WP4_CEA_TC_007 for WP4_SDCA_001 and WP4_SDCA_003....29
Table 31. Test Case WP4_CEA_TC_008 for WP4_ACS_011 and WP4_ACS_013....30
Table 32. Test Case WP4_CEA_TC_009 for WP4_CAC_010....30
Table 33. Test Case WP4_CEA_TC_010 for WP4_CMA_002....30
Table 34. Test Case WP4_CEA_TC_011 for WP4_ACS_006....31
Table 35. Test results for functionalities implemented in WP4....31
Table 36. Evidence Management and seamless interoperability functionalities....33
Table 37. Test Case WP5_TRC_TC_001 for WP5_AM_001....34
Table 38. Test Case WP5_TRC_TC_002 for WP5_AM_002....34
Table 39. Test Case WP5_TRC_TC_003 for WP5_AM_003....35
Table 40. Test Case WP5_TRC_TC_004 for WP5_AM_004....36
Table 41. Test Case WP5_TRC_TC_005 for WP5_AM_005....36

Table 42. Test Case WP5_TRC_TC_006 for WP5_DM_001 37
Table 43. Test Case WP5_TRC_TC_007 for WP5_DM_002 37
Table 44. Test Case WP5_TRC_TC_008 for WP5_DM_005 37
Table 45. Test Case WP5_TRC_TC_009 for WP5_DM_006 37
Table 46. Test Case WP5_TRC_TC_010 for WP5_DM_007 38
Table 47. Test Case WP5_TRC_TC_011 for WP5_TI_001 38
Table 48. Test Case WP5_TRC_TC_012 for WP5_TI_002 38
Table 49. Test Case WP5_TAS_TC_001 for WP5_CW 38
Table 50. Test Case WP5_TAS_TC_002 for WP5_TQ_001 to WP5_TQ_005 39
Table 51. Test Case WP5_TAS_TC_003 for WP5_EM 40
Table 52. Test Case WP5_TAS_TC_004 for WP5_CW 41
Table 53. Test Case WP5_CEA_TC_001 for WP5_TI 41
Table 54. Test Case WP5_CEA_TC_002 for WP5_DM_003 and WP5_DM_004 42
Table 55. Test results for functionalities implemented in WP5 42
Table 56. Compliance management and Cross and intra domain reuse functionalities 45
Table 57. Test Case WP6_VIF_TC_001 for WP6_PPA_001 45
Table 58. Test Case WP6_VIF_TC_002 for WP6_PPA_004 46
Table 59. Test Case WP6_VIF_TC_003 for WP6_PPA_005 46
Table 60. Test Case WP6_CEA_TC_001 for WP6_CM_001 47
Table 61. Test Case WP6_KMT_TC_002 for WP6_RA 47
Table 62. Test Case WP6_KMT_TC_003 for WP6_RA_002 to WP6_RA_004 48
Table 63. Test Case WP6_KMT_TC_004 for WP6_RA_001 to WP6_RA_005 49
Table 64. Test Case WP6_KMT_TC_005 for WP6_PPA_002 49
Table 65. Test Case WP6_CEA_TC_006 for WP6_PPA_003 49
Table 66. Test Case WP6_KMT_TC_008 for WP6_CM and WP6_SEM_001 50
Table 67. Test results for functionalities implemented in WP6 51
Table 68. Results of the test cases for functionalities implemented in Prototype P2 52

Executive Summary

The AMASS Open Tool Platform is one of the results 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 (D2.8) is the last one produced in Task 2.4 AMASS Platform Validation. It concerns the validation of the AMASS Platform Prototype P2.

The functionalities of the AMASS Platform are described in the AMASS deliverable D2.4 (AMASS Reference Architecture) [6]. The Prototype Core has been built upon three pre-existing toolsets from the OPENCOSS project [1] , the CHESS project (Polarsys Platform) [12] and the SafeCer project [2] (which was built on top of the Eclipse Process Framework project [27]). Prototype P2 extends Prototype P1 with functionalities and tools addressing the AMASS STOs: Architecture-Driven Assurance (STO1), Multi-Concern Assurance (STO2), Seamless Interoperability (STO3), and Cross/intra-Domain Reuse (STO4).

Prototype P2 has been released as an Eclipse bundle. Two manuals have been provided with the platform: a Developers Guide [5] written for AMASS Platform developers, and a User Manual [4] written for AMASS Platform users.

This deliverable:

- Explains the architecture of the overall AMASS Platform and its building blocks.
- Presents the validation activities that have been conducted on Prototype P2. The validation has been based on an analysis of the requirements and corresponding functionalities planned for the Prototype P2 and defined in D2.1 [7], and the usage scenarios defined in D2.4 [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 the prototypes Core and P1 have been revised as well as the functionalities that were postponed for P2.
- Summarizes the validation results.

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. Its aim is to lower certification costs in face of rapidly changing product features and market needs. This has been 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 results are:

a) The **AMASS Reference Tool Architecture**, which extends the OPENCOSS [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 corresponds 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 is 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 open-source release of the AMASS building blocks.

c) The **Open AMASS Community**, which manages the project outcomes for maintenance, evolution and industrialization. The Open Community is 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 (which will enrich AMASS feature). As Eclipse Foundation is part of the AMASS consortium, the PolarSys/Eclipse community [3] has been chosen as the best 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 (Core Prototype), 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 were developed, integrated with previous prototype and benchmarked at TRL 4.
3. Finally, at the **third prototyping** iteration (Prototype P2), all AMASS building blocks have been integrated in a comprehensive toolset operating at TRL 5 (technology validated in a relevant environment).

[1] https://open-services.net

1.2 Purpose of the deliverable

This deliverable is the third one resulting from Task 2.4 “AMASS Platform Validation”. The purpose of this deliverable is to serve as a complement to Prototype P2. 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 D2.4 [6].

This document briefly explains the AMASS platform architecture and its component 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 P2, in order to check the global functionality of the platform according to the requirements defined in D2.1 [7]. For the validation activities, the usage scenarios defined in D2.4 [6] have been collected in order to refine these items into test cases that are compatible with the current developments of the AMASS Platform. Additional test cases have also been defined for those functionalities of Core Prototype and Prototype P1 that were implemented during the previous iterations but whose test results were not found satisfactory. The manual execution of the test cases lets us provide direct feedback regarding implementation status and potential further enhancements for the functionalities.

The testing results, together with the validation team feedback, will allow WP1 and T1.4 “Case Study Implementation and Benchmarking” 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 other deliverables

D2.8 is related to the following other AMASS deliverables:

- D2.1 [7] (Business cases and high-level requirements), which 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.
- D2.4 [6] (AMASS Reference Architecture (c)), which describes the overall architecture of the AMASS platform including needs from the case studies that must be covered by the platform.
- The deliverables related to the development of a tooling framework to support the AMASS Prototype P2 platform. These deliverables describe the tools whose testing is reported in the current document:
    - D3.6 [8] Prototype for Architecture-Driven Assurance (c)
    - D4.6 [9] Prototype for multi-concern assurance (c)
    - D5.6 [10] Prototype for seamless interoperability (c)
    - D6.6 [11] Prototype for cross/intra-domain reuse (c)
- The methodological guides:
    - D3.8 [13] Methodological Guide for Architecture-Driven Assurance (b)
    - D4.8 [14] Methodological Guide for Multiconcern Assurance (b)
    - D5.8 [15] Methodological Guide for Seamless Interoperability (b)
    - D6.8 [16] Methodological Guide for Cross-Intra Domain Reuse (b)
- D2.7 [28] (Integrated AMASS Platform (b)), which is the previous deliverable about the validation of the AMASS platform. This deliverable extends and complete the latter.
- The AMASS Prototype User Manual [4], which describes how to use the AMASS platform. The targeted audience of this manual is the AMASS Platform users.

- The AMASS Prototype Developers Guide [5], which describes how to set up the AMASS development environment and the tools integrated in the AMASS platform. The targeted audience of this guide is the AMASS Platform developers. It was written by the AMASS developers at the same time as the AMASS platform was developed and has been validated by them.

The D2.4 [6] deliverable and the AMASS User Manual [4] have been the main reference documents from which new test cases have been derived. Test cases have been produced to validate features described in those deliverables. Documentation related to external tools that communicate with the platform have also been used.

1.4 Structure of this document

This deliverable is structured as follows:

- Section 2 provides a brief presentation of the AMASS platform and the tooling architecture and the technologies used to implement them and describes the testing and validation procedure.
- Sections 3 to 6 describe the current status of the Prototype P2 functionalities, the test cases that have been defined to evaluate those functionalities, and the results from the execution of those test cases.
- Section 7 provides a synthesis of the validation results for Prototype P2.
- Finally, in Section 8, to sum up, some conclusions about validation on Prototype P2 have been included.

2. AMASS Platform Architecture

2.1 Conceptual and Implementation Architecture

A general top-level architecture of the AMASS platform was designed in the D2.4 deliverable [6]. As part of the overall platform, the AMASS Core Prototype was produced by merging existing technologies from OPENCOSS [1] and SafeCer [2], and other related projects such as CHESS [12]. The AMASS Prototype P1 includes building blocks composed of tools extending the functionalities provided by the building blocks of the Core Prototype in order to address the following concerns: architecture-driven assurance, multi-concern assurance, seamless interoperability and cross/intra-domain reuse. In the AMASS Prototype P2, the functionalities developed for Prototype P1 have been improved and integrated in a comprehensive toolset operating at TRL5.

Figure 1 illustrates the AMASS Reference Tool Architecture (ARTA), where the basic building blocks constituting the Core Prototype are surrounded by a red dashed line, and the building blocks implemented in Prototypes P1 and P2 are depicted in green boxes.

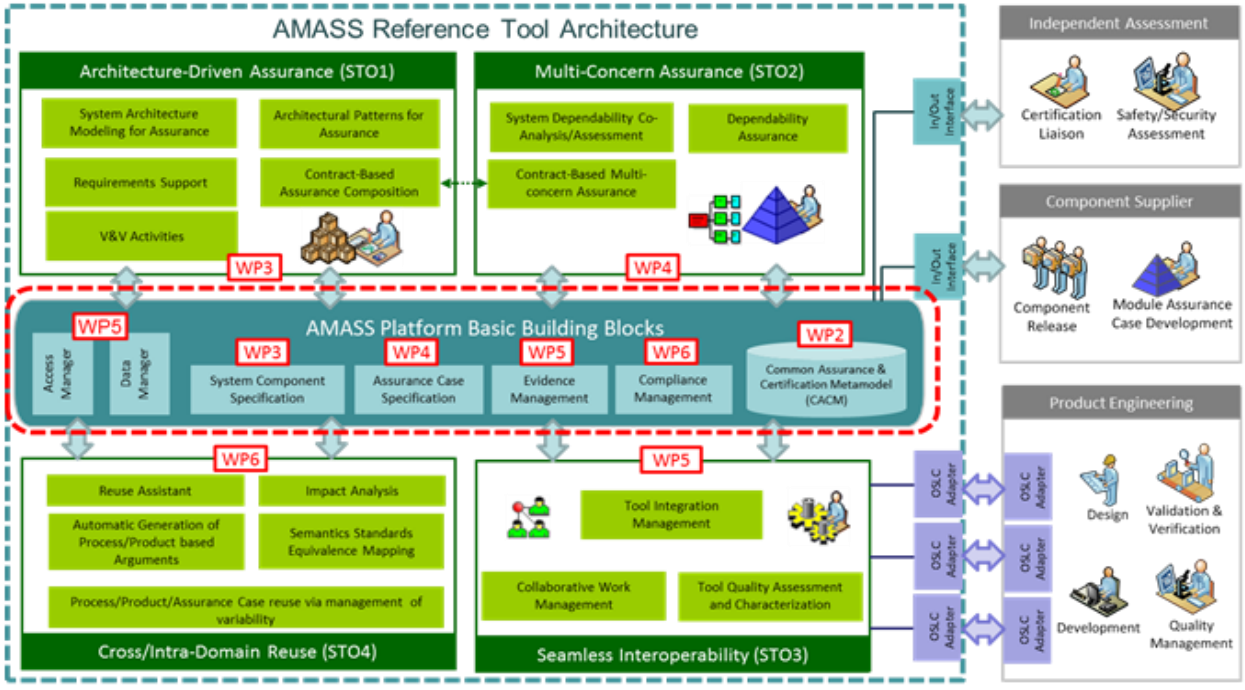

Figure 1. AMASS Reference (High-Level) Architecture (Prototype P2)

The AMASS platform comprises a set of tools which provide the functionalities described in the AMASS deliverable D2.4 [6] (AMASS Reference Architecture (c)). Figure 2 presents an overall picture of the layered structure of the AMASS functional architecture. This architecture was implemented in tasks T3.3, T4.3, T5.3 and T6.3.

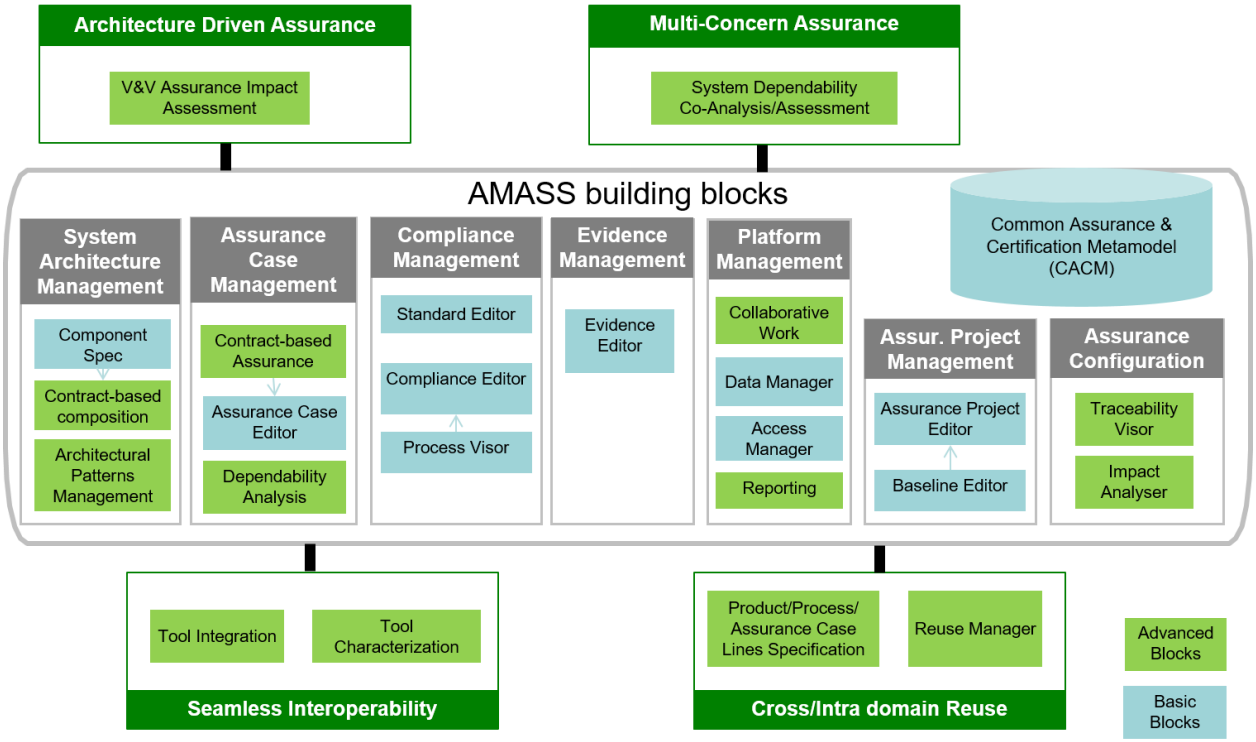

Figure 2. Functional Decomposition of the AMASS Platform

2.2 AMASS Platform Tools

The AMASS Prototype P2 has been built upon the following baseline technologies and toolsets:

1. **OpenCert** [1] – AMASS Core edition, which supports Assurance case specification, Dependability Assurance Modelling, and Contract–based multi-concern assurance.
2. **Papyrus** [23] and **CHESS** [12] – AMASS Core edition, which supports Contract modelling, Contract-Based Multi-concern Assurance, and Contract-based trade-off analysis in parameterized architectures with OCRA, NuXmv, XSAP.
3. **EPF-Composer** [27], which supports Assurance process modelling and tailoring to the individual project (resulting process model is used by WEFACT [19]).
4. **Concerto-FLA** [30], which supports Failure Logic Analysis (FLA) for safety and security-related failure modes.
5. **Capra tool** [24], which supports traceability management.
6. **BVR tool** [18], which supports orthogonal variability management and which, integrated with EPF-Composer supports variability management at process level (enabling process co-assessment as well as cross-concern, cross-domain, and intra-domain reuse of process information), integrated with CHESS supports variability management at product level and, integrated with OpenCert supports variability management at argumentation level.
7. **Open Service for Lifecycle Collaboration** (OSLC) technology and **Knowledge Manager** (KM) toolset [21][22], which support interoperability features.

In addition, the AMASS platform supports or provides an interface to the following external tools:

1. **WEFACT** [9] [14], which supports the assurance process workflow.
2. **FMVEA** [9] [14], which supports model-based system-dependability co-analysis and –assessment.

3. **MORETO** [9] [14], which supports security analysis and manual or standards-based automated generation of security requirements.
4. **Medini Analyze** [31], which supports the assurance process workflow and allows safety and security analyses.
5. **SAVONA** [35], which supports contract modelling.
6. **Sabotage** [34], which supports fault injection simulation.
7. **SafetyArchitect** [32] and **CyberArchitect** [33], which support dependability co-analysis.
8. **V&V Manager**[8] [13] for verification and validation features.
9. **Verification Studio** [36] for management of V&V evidence.

2.3 Installation and User Manuals

The steps necessary to install the AMASS Prototype P2 are described in the AMASS User Manual [4]. A packaged distribution of AMASS Prototype P2 has been released as an Eclipse bundle [29].

The AMASS User Manual also describes how to use the AMASS platform tool. It provides detailed information about how to install the tool, how it works, and how to use it. The manual also includes shorts descriptions for external tools used by the platform, the manuals for some of these tools are available in the tool providers' websites.

2.4 AMASS Prototype P1 Validation Results

Figure 3 summarizes the results from the validation phase for the AMASS Prototype P1 release [28]:

- 27% of functionalities met their specifications.
- 16% of functionalities partially met their specifications.
- 14% of functionalities failed to meet their specifications.
- 43% of functionalities were not tested because information needed to run tests for them was not available.

The validation results obtained in the validation of the AMASS Platform P1 have been considered during this third validation iteration, particularly those from functionalities that were not successfully validated.

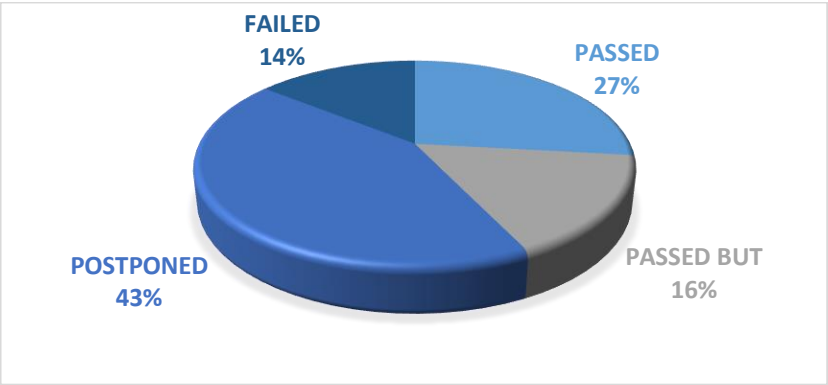

Figure 3. Validation status of the AMASS Platform P1

2.5 AMASS Prototype P2 Testing and Validation Methodology

Figure 4 presents the overall validation and testing process that has been followed for the AMASS Prototype P2.

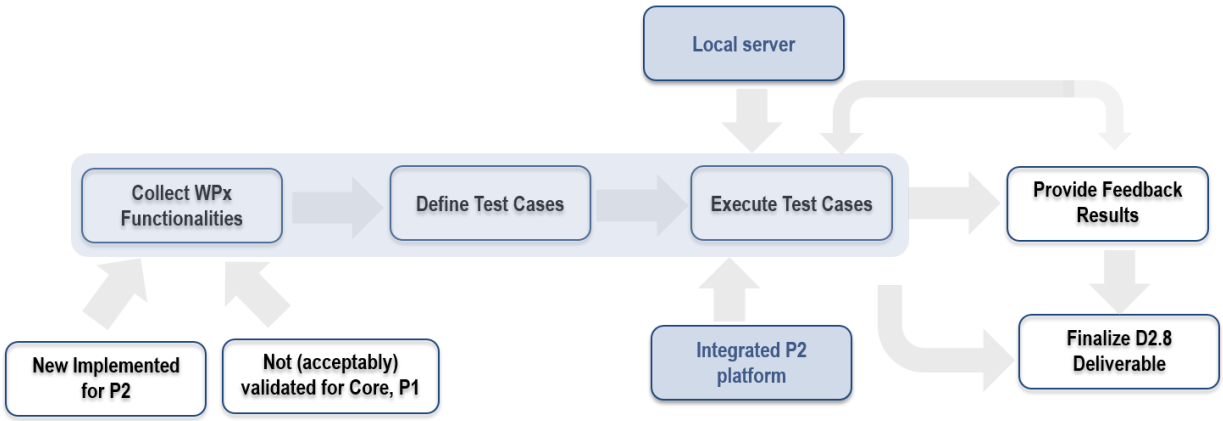

Figure 4. AMASS Prototype P2 validation and testing process

The methodology aims to validate that the AMASS Prototype P2 platform satisfies its requirements and to check the system behaviour against user needs and case studies (see D2.1 [7] and D2.4 [6] deliverables). It also checks that functionalities specified for the Core Prototype and Prototype P1 for which validation results were not found satisfactory during their last validation phase now meet their specifications.

The test cases listed in this document are mainly based on the scenarios defined in the use cases specified in deliverable D2.4 [6]. 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 [7] requirements of the AMASS Prototype P2 (and some of the Core Prototype and Prototype P1 as well) to ensure their theoretical coverage.

We have also used the AMASS User Manual [4] and the methodological guides D3.8 [13] , D4.8 [14], D5.8 [15], and D6.8 [16], provided for WP3 to WP6 as reference documents to enhance the input(s), steps, and expected result(s) for some test cases.

As during the validation of the previous AMASS Platform prototypes, each test case specification comprises:

- A *Test Case ID*, which uniquely identifies the test case.
- A *Scope*, which provides the context and summarizes the purpose of the test case.
- A *Feature ID*, which refers to the AMASS related requirements that must be validated.
- *Related use cases*, which refer to related use case scenarios.
- *Input*, which specifies the necessary input data needed to execute the test case.
- *Steps*, which specify the execution steps to follow in order to run the test case.
- *Expected results*, which specify the behaviour or results expected from the execution of the test case.
- *MoSCoW Priority*[2], as defined for the AMASS requirements in deliverable D2.1 [7].

Dedicated partners have installed the platform, manually executed test cases and checked its implementation. These partners have indicated the material used to run the test cases including machine configuration, validation data, etc. We report the status of test case execution as:

- **Passed**: feature that works as required.
- **Passed but**: feature that works but could be enhanced.
- **Failed**: feature that does not work.

[2] Must have, Should have, Could have, and Won't have but would like

- **Not executed**: feature that was implemented but not tested.

For each “Passed but” or “Failed” status, a rationale has been provided describing why that status was given. Also, some tickets for those “Passed but” or “Failed” tests have been generated within the AMASS Issue-Tracker system to report the problems to the Implementation Team.

The overall validation results are summarized in Section 7.

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 the deliverable D2.1 [7]. Table 1 lists the functionalities planned for Prototype P2 and those from Prototype P1 which needed to be revised for Prototype P2.

Table 1. System Component Specification and Architecture driven assurance functionalities

ID	Feature	Prototype
WP3_VVA_004	Trace requirements validation checks	P1
WP3_VVA_005	Verify (model checking) state machines	P1
WP3_CAC_001	Validate composition of components by validating their contracts	P1
WP3_CAC_005	General management of contract-component assignments	P1
WP3_CAC_008	Contract-based validation and verification	P1
WP3_CAC_011	Overview of contract-based validation for behavioural models	P1
WP3_VVA_010	Model-based safety analysis	P1
WP3_VVA_002	Trace model-to-model transformation	P1
WP3_APL_001	Drag and drop an architectural pattern	P2
WP3_APL_002	Edit an architectural pattern	P2
WP3_APL_003	Use of architectural patterns at different levels	P2
WP3_APL_005	Generation of argumentation fragments from architectural patterns/decisions	P2
WP3_SAM_002	Impact assessment if the component changes	P2
WP3_SAM_003	Compare different architectures according to different concerns which have been specified before	P2
WP3_SAM_004	Integration with external modelling tools	P2
WP3_VVA_003	Validate requirements checking consistency, redundancy, … on formal properties	P2
WP3_VVA_006	Automatic provision of HARA/TARA-artefacts	P2
WP3_VVA_007	Generation of reports about system description/ verification results	P2
WP3_VVA_011	Simulation-based Fault Injection	P2
WP3_VVA_012	Design Space Exploration	P2

3.2 Test Cases

This section includes a table describing the test case for each feature listed in Table 1. The listed test cases are based on the use case scenarios for each feature defined in deliverable D2.4 [6], where these are available.

Table 2. Test Case WP3_FBK_TC_001 for WP3_APL_001

ID	WP3_FBK_TC_001
Scope	Drag and drop an architectural pattern

Feature ID	WP3_APL_001
Related use cases	"Instantiate an architectural pattern"
Input	An existing CHESS model with at least one component created.
Steps	1. Open the CHESS model where the pattern has to be applied. 2. Open the "Model Explorer" view, right click the CHESS root model entity and select: 2.1. Import Registered Package to load the CHESS library, or 2.2. Import Package from User Model to load the available user-defined patterns Papyrus project (requires the availability of a pattern model in the current workspace). 3. In the "Model Explorer" view, select a system component where the component instances playing the pattern roles will be identified/created; right click and select "CHESS -> DesignPatterns -> Select and Apply a Design Pattern". A dedicated wizard is shown. 4. Select the pattern to instantiate from the Available Patterns lists (this list is initialized with the patterns library/projects that have been imported in step 2 and click "Apply". 5. Bind the information available in the pattern into the current system model.
Expected results	The architectural pattern is instantiated in the context of the selected (composite) component.
Priority	Should

Table 3. Test Case WP3_FBK_TC_002 for WP3_APL_002

ID	**WP3_FBK_TC_002**
Scope	Edit an architectural pattern
Feature ID	WP3_APL_002
Related use cases	"Edit an architectural pattern"
Input	A Papyrus project with the *CHESS Design Pattern* registered profile applied.
Steps	1. Create a new UML Package (from the "Model Explorer" view) to contain all the elements of the new pattern. 2. Create a Class Diagram (from the "Model Explorer" view). 3. From the Model Explorer or from the Class Diagram Palette, create UML Class elements to be used as types for the roles of the pattern. 4. Create a UML Collaboration (from the "Model Explorer" view) and stereotype it as <<Pattern>>. 5. Set the properties of the pattern, if available, by using the Pattern stereotype properties (from the tab "Profile" of the Eclipse Properties View). 6. Create (from the Model Explorer) a Composite Structure Diagram (CSD) for the UML Collaboration. 7. Create Property elements inside the CSD (from the CSD Palette/Model Explorer) and set their type to the appropriate UML Class created at 2. 8. Set each Property element as CollaborationRole (from the CSD Palette); so, create a CollaborationRole by using the CSD Palette and specify for it a Property created in the previous step. Repeat this step for each defined Property. 9. Add Ports and Connectors to the CSD elements (from the CSD Palette/Model Explorer).
Expected results	The architectural pattern is created, ready to be instantiated in a given CHESS model.

Priority	Should

Table 4. Test Case WP3_FBK_TC_003 for WP3_APL_003

ID	**WP3_FBK_TC_003**
Scope	Use of architectural patterns at different levels
Feature ID	WP3_APL_003
Related use cases	"Edit an architectural pattern"
Input	An existing CHESS model with at least one component created
Steps	1. Open the CHESS model where the pattern must be applied. 2. Open the "Model Explorer" view, right click the CHESS root model entity and select: 2.1. Import Registered Package to load the CHESS library, or 2.2. Import Package from User Model to load the available user-defined patterns Papyrus project (requires the availability of a pattern model in the current workspace). 3. In the Model Explorer select a system component where the component instances playing the pattern roles will be identified/created; right click and select "CHESS -> DesignPatterns -> Select and Apply a Design Pattern". A dedicated wizard is shown. 4. Select the pattern to instantiate from the Available Patterns lists (this list is initialized with the patterns library/projects that have been imported in step 2 and click "Apply". 5. Bind the information available in the pattern into the current system model.
Expected results	The architectural pattern is instantiated.
Priority	Should

Table 5. Test Case WP3_FBK_TC_004 for WP3_APL_005

ID	**WP3_FBK_TC_004**
Scope	Generation of argumentation fragments from architectural patterns/decisions
Feature ID	WP3_APL_005
Related use cases	Missing use case in D2.4
Input	An existing CHESS model with an architectural pattern instantiated, and an established connection to a CDO server where assurance cases are stored.
Steps	1. Open the CHESS model where the architectural pattern is instantiated. 2. Select the block System and associate it with the CHGaResourcePlatform stereotype. 3. In the Model Explorer, go to AnalysisView package and create a class diagram in the "DependabilityAnalysisView" 4. Add the ContractRefinementAnalysisContext element to the diagram. 5. In the profile of the created element select the block System as system platform. 6. Run the argument generation from the menu "CHESS → Argumentation → Generate Argument-Fragments" (OpenCert)
Expected results	Argumentation fragments from architectural patterns/decisions in the destination assurance project.
Priority	Should

Table 6. Test Case WP3_A4T_TC_001 for WP3_VVA_004

ID	WP3_A4T_TC_001
Scope	Trace requirements validation checks
Feature ID	WP3_VVA_004
Related use cases	"Trace requirement validation"
Input	The requirement to be traced is available in the system model. An analysis context has been used to store the information needed to perform the V&V analysis related to the requirement to be traced.
Steps	1. Select the requirements and the analysis context to be traced. 2. Create a traceability link between the selected entities.
Expected results	Traceability links between the selected entities.
Priority	Should

Table 7. Test Case WP3_A4T_TC_002 for WP3_VVA_005

ID	WP3_A4T_TC_002
Scope	Verify (model checking) state machines
Feature 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	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	The check results are displayed in the "V&V Results" view.
Priority	Must

Table 8. Test Case WP3_A4T_TC_003 for WP3_CAC_001

ID	WP3_A4T_TC_003
Scope	Validate composition of components by validating their contracts
Feature 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	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 "Validation property trace" view.
Expected results	The status of the check in the "Validation property trace" view
Priority	Should

Table 9. Test Case WP3_A4T_TC_004 for WP3_CAC_005

ID	**WP3_A4T_TC_004**
Scope	General management of contract-component assignments.
Feature 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	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 to check the status of the components currently defined in the system architecture, together with its associated contracts.
Expected results	A view in terms of components and their associated contracts in the platform. Contracts assigned for each component in the "System Architectures" column of the "Hierarchical Model View".
Priority	Should

Table 10. Test Case WP3_A4T_TC_005 for WP3_CAC_008

ID	**WP3_A4T_TC_005**
Scope	Contract-based validation and verification
Feature 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	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 Results" view.
Priority	Must

Table 11. Test Case WP3_A4T_TC_006 for WP3_CAC_008

ID	**WP3_A4T_TC_006**
Scope	Contract-based validation and verification

Feature 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	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	The system shall allow the user to generate and view fault trees.
Priority	Must

Table 12. Test Case WP3_A4T_TC_007 for WP3_CAC_011

ID	**WP3_A4T_TC_007**
Scope	Overview of contract-based validation for behavioural models
Feature 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	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	A status of the check is shown in the "V&V Results" view.
Priority	Could

Table 13. Test Case WP3_A4T_TC_008 for WP3_VVA_010

ID	**WP3_A4T_TC_008**
Scope	Model-based safety analysis.
Feature 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	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. 3. 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

Table 14. Test Case WP3_A4T_TC_009 for WP3_VVA_002

ID	WP3_A4T_TC_009
Scope	Trace model-to-model transformation
Feature ID	WP3_VVA_002
Related use cases	"Trace system model with assurance assets"
Input	A CHESS project with fault tree generated (as resulting from WP3_A4T_TC_008)
Steps	1. Open the CHESS project. 2. Select the Analysis Context CHESS model entity that has been selected to perform the fault tree generation (the Analysis Context, in particular its Platform property, stores the information about the entities of the CHESS model that have been considered for the fault tree generation). 3. Right click and choose "Capra → Add to Trace Source". 4. In the Project Explorer, select both the .fei and .smv files located in the NuSMV3-XSAP/Files folder under the current the CHESS project (these SMV model files have been generated by the model transformation automatically executed during the fault tree generation). 5. Right click and choose "Capra → Add to Trace Targets". 6. Go to the "Create Tracelinks" view and click the "Create Trace Link" button located on the right side of the view toolbar.
Expected results	A trace is created and stored in the traceability model. To check the trace, select the .fei (or .smv) file and go to the Traceability View; the view shows the selected file and the traced Analysis Context CHESS model element.
Priority	Should

Table 15. Test Case WP3_CEA_TC_001 for WP3_SAM_003 and WP3_VVA_012

ID	WP3_CEA_TC_001
Scope	Design space exploration. Compare different architectures according to different concerns which have been specified before.
Feature ID	WP3_SAM_003, WP3_VVA_12
Related use cases	"Compare parameterized architecture"
Input	A CHESS project with BDD and IBD diagram
Steps	1. Open the CHESS project. 2. Parametrize the model by creating a static Flowport as parameter (in the Model Explorer or BDD editor) and set its static attribute to "true" in the Properties view. 3. Assign the parameter to an owner by creating a delegationConstraint (from the Model Explorer or IBD editor) and set a value or expression. 4. Instantiate the parametrized architecture by clicking on the menu "Instantiate the parameterized architecture" from the root component of the model. 5. In the wizard that appears, select the parameters and assign them values, and then import the instantiated architecture into the project by choosing a destination package on the right side of the wizard page. 6. Click ok. 7. Select the menu "CHESS → Trade off analysis" from the root component of the parametrized model. 8. In the wizard that appears, select at least two configurations on which to run the analysis, and then the time model on the architecture.
Expected results	Results of the trade-off analysis appears in a tab "Trade-off" with graphical

	interpretation for each architecture and for each aspect, qualitative and quantitative results.
Priority	Could

Table 16. Test Case WP3_CEA_TC_002 for WP3_SAM_004

ID	**WP3_CEA_TC_002**
Scope	Integration with external modelling tools
Feature ID	WP3_SAM_004
Related use cases	"Import model"
Input	A CHESS project, an oss file
Steps	1. Open the CHESS project. 2. In the "Model Explorer" view, right click on a package and select the menu "CHESS → Basic Operations → Import <<SystemView>> components from .oss file " 3. From the wizard that appears, select an oss file and click ok.
Expected results	Components are generated, along with their structural description (ports, contracts, refinements, connections, etc.) and imported in the selected package.
Priority	Could

Table 17. Test Case WP3_CEA_TC_003 for WP3_VVA_003

ID	**WP3_CEA_TC_003**
Scope	Validate requirements checking consistency, redundancy, etc. on formal properties.
Feature ID	WP3_VVA_003
Related use cases	"Verify contract refinement"
Input	A CHESS project with existing contracts
Steps	1. Open the CHESS project. 2. From the root component of the model, select the menu "CHESS → Functional verifications → Check contract refinement on selected component". 3. Choose the timing model of the architecture.
Expected results	Results of the contract refinement appears in a tab "Contract refinement trace".
Priority	Must

Table 18. Test Case WP3_CEA_TC_004 for WP3_VVA_007

ID	**WP3_CEA_TC_004**
Scope	Generation of reports about system description/verification results ….
Feature ID	WP3_VVA_007
Related use cases	"Generate documentation from the system model"
Input	A CHESS project
Steps	1. Open the CHESS project. 2. From any component in the "Model Explorer" view, select the menu "CHESS → Safety case → Document generation → Generate documentation from selected component".
Expected results	A generated pdf report that includes the model elements description, the diagrams and the analyses results.
Priority	Must

Table 19. Test Case WP3_CEA_TC_005 for WP3_SAM_002

ID	WP3_CEA_TC_005
Scope	Impact assessment if the component changes
Feature ID	WP3_SAM_002
Related use cases	Impact assessment
Input	Existing traceability links between a CHESS system component and assurance case or evidence entities
Steps	1. Open the CHESS project. 2. Open the Traceability View (if not already available in the Eclipse workbench, select it from the "Window → Show View → Others → CHESS" Eclipse main menu). 3. Select the CHESS component for which an impact assessment has to be performed, right click it and select "Capra → Add to trace sources". 4. Select a Claim in an AssuranceCase diagram, right click it and select "Capra → Add to trace targets". 5. Go to the "Create Tracelinks" view and select the "Create trace link" button.
Expected results	The assurance case and/or evidence entities that could be impacted by a modification on the selected component are listed in the Traceability View.
Priority	Could

Table 20. Test Case WP3_CEA_TC_006 for WP3_VVA_011

ID	WP3_CEA_TC_006
Scope	Simulation-based Fault Injection
Feature ID	WP3_VVA_011
Related use cases	"Simulation-based fault injection"
Input	The system model is extended with faulty information
Steps	1. The user annotates the system model with contracts and faulty information in CHESS. Fault Models can be expressed in CHESS or be later filled in as expressed in Step 2 by means of the Sabotage editor. 2. The user configures the rest of the needed information to create the fault list: where to inject the fault target, fault model (if not already specified in the previous step, this includes fault models such as stuck-at0, delay, too-low or too high among others), when to trigger the fault (fault injection time), and for how long to insert the fault in the system (fault duration). 3. The user runs the fault injection simulations. 4. Simulation Results are analysed, and proper measures are taken. If the inserted faults are not detected or the measures are not sufficient, the user redesigns the safety concept/safety mechanisms.
Expected results	The expected results can differ depending on the phase of the product development the technique is applied: • HARA tables partially automated: During the hazard identification process, fault injection in combination with monitoring techniques allows to automate part of the current manual process of completing the hazard identification process (Hazard Analysis and Risk Assessment in the automotive ISO 26262 standard, in industrial IEC 61508 "Hazard and Risk Analysis", "Functional Hazard Analysis (FHA) in aerospace ARP4761. Fault Injection simulation aims at deriving potential effects or hazards based on the Fault Injection simulation results. • An early validated safety concept and verification of safety mechanisms.

	This would be one of the main objectives of applying fault injection simulations in AMASS.
Priority	Should

Table 21. Test Case WP3_CEA_TC_007 for WP3_VVA_006

ID	**WP3_CEA_TC_007**
Scope	The system shall provide the capability for generating HARA (Hazard Analysis Risk Assessment) and TARA (Threat Assessment & Remediation Analysis).
Feature ID	WP3_VVA_006
Related use cases	"Define/Perform Safety/Security Analysis"
Input	A Medini Analyze project
Steps	1. Select the HARA option in the Medini tool. 2. Fill manually the showed table. 3. Save the results.
Expected results	HARA/TARA-tables
Priority	Must

3.3 Test Results

Table 22 presents the results of executing the test cases listed in section 0, including the achieved status and the rationale for this. The instructions for installing the used testing environment are described in the AMASS Developers Guide [5]. The functionalities provided for System component specification and Architecture-driven assurance in Prototype P2 are described in the AMASS User Manual [4].

Table 22. Test results for functionalities implemented in WP3

Test Case ID	Execution Results	Status	Rationale
WP3_FBK_TC_001	The architectural pattern is instantiated	Passed	
WP3_FBK_TC_002	A new architectural pattern is created	Passed	
WP3_FBK_TC_003	The architectural pattern is instantiated. This test case is covered by WP3_FBK_TC_001 for WP3_APL_001	Passed	
WP3_FBK_TC_004	The argument-fragments are generated.	Passed	
WP3_A4T_TC_001	Traceability links between the selected entities. It is validated with a CHESS model with discrete time (SSR_no_fi)	Passed	
WP3_A4T_TC_002	The check results in the "V&V Results" view. It is validated with a CHESS model with discrete time (Battery_Multistate)	Passed	
WP3_A4T_TC_003	The status of the check in the "Validation property trace". It is validated with a CHESS model with discrete time (SSR_no_fi) by using the OSLC service.	Passed	
WP3_A4T_TC_004	Contracts assigned for each component in the "System Architectures" Column of "Hierarchical Model View". It is validated with a CHESS model with discrete time (SSR_no_fi) by using the OSLC service.	Passed	
WP3_A4T_TC_005	A status of the check is shown in the "V&V	Passed	

Test Case ID	Execution Results	Status	Rationale
	result" view. It is validated with a CHESS model with discrete time (SSR_no_fi) by using the OSLC service.		
WP3_A4T_TC_006	The system shall allow the user to generate and view fault trees. It is validated with a CHESS model with discrete time (SSR_no_fi) by using the OSLC service.	Passed	
WP3_A4T_TC_007	A status of the check is shown in the "V&V Results" view. It is validated with a CHESS model with discrete time (SSR_no_fi) by using the OSLC service.	Passed	
WP3_A4T_TC_008	The system shall allow the user to generate and view fault trees. It is validated with a CHESS model with discrete time (SSR_no_fi) by using the OSLC service.	Passed	
WP3_A4T_TC_009	A trace is created and stored in the traceability model. It is validated with a CHESS model with discrete time (SSR_no_fi) and the result of WP3_A4T_TC_008.	Passed	
WP3_CEA_TC_001	Results from trade off analysis appears in a tab "trade-off".	Passed	
WP3_CEA_TC_002	A package containing the component model extracted from the oss file.	Passed	
WP3_CEA_TC_003	Contract refinement results appears in a tab.	Passed	
WP3_CEA_TC_004	A generated report with system model diagrams screenshots, description and analyses results.	Passed	
WP3_CEA_TC_005	Traceability view shows the entities that are related to each other between system model and the argumentation model elements.	Passed but	It would be useful to have some pop-up to confirm that the link has been created.
WP3_CEA_TC_006	The resulting simulation traces (i.e. the results coming from the fault injection simulations) can be graphically seen in Matlab/Simulink. In addition, those results are saved in a MLDATX/CSV file.	Passed	
WP3_CEA_TC_007	HARA and TARA tables are generated	Passed	

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 lists such functionalities planned for Prototype P2, as well as functionalities of Core Prototype and Prototype P1 which needed to be revised.

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

ID	Feature	Prototype
WP4_ACS_001	Assurance case edition	Core
WP4_ACS_005	Provide a structured language to the text inside the claims	Core
WP4_ACS_004	Provide guidelines for argumentation patterns	P1
WP4_ACS_008	Traceability of the dependability case	P1
WP4_ACS_010	Composition of the overall argument	P1
WP4_SDCA_002	System dependability co-verification and co-validation	P1
WP4_SDCA_003	The system shall allow combinations of safety and security analysis	P1
WP4_ACS_011	Assurance case status report	P2
WP4_ACS_013	Provide quantitative confidence metrics about an assurance case in a report	P2
WP4_CAC_010	Contract-based trade-off analysis	P2
WP4_CMA_002	Component contracts must support multiple concerns	P2
WP4_SDCA_001	System dependability co-architecturing and co-design	P2
WP4_ACS_006	Provide guidelines for argumentation	P2

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 functionalities specified in Table 23. The listed test cases are based on the use case scenarios for each feature defined in deliverable D2.4 [6], where these are available.

Table 24. Test Case WP4_CEA_TC_001 for WP4_ACS_001

ID	WP4_CEA_TC_001
Scope	Edit an assurance case in a scalable way.
Feature ID	WP4_ACS_001
Related use cases	“Define and navigate an assurance case structure”
Input	A reference framework
Steps	1. Create an assurance project 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 new diagram view 10. Drag and drop an 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_CEA_TC_002 for WP4_ACS_005

ID	**WP4_CEA_TC_002**
Scope	Provide a structured language to the text inside the claims
Feature ID	WP4_ACS_005
Related use cases	None
Input	An argumentation model
Steps	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 26. Test Case WP4_CEA_TC_003 for WP4_ACS_004

ID	**WP4_CEA_TC_003**
Scope	Provide guidelines for argumentation patterns
Feature ID	WP4_ACS_004
Related use cases	None
Input	An argumentation model pattern. Methodological guidance in D4.8 [14], Section 3.2. “Dependability Assurance Case Modelling”
Steps	Read the methodological guidance to check that provides details instruction to use and instantiate argument patterns (concerning safety and security) presented in the actual assurance case.
Expected results	Guidelines for argumentation patterns
Priority	Should

Table 27. Test Case WP4_CEA_TC_004 for WP4_ACS_008

ID	**WP4_CEA_TC_004**
Scope	Traceability of the dependability case
Feature ID	WP4_ACS_008
Related use cases	“Assign to component specification”
Input	There is a system architecture definition and an assurance case project.
Steps	1. Open the “Model explorer” and the “Assurance case structure” views. 2. Select an entity in the CHESS model to be linked. 3. Open the “OpenCert” tab in the Properties view. This tab shows the relationships that can be created. 4. Pin the Properties view. 5. Drag the assurance related entity (Claim, Artefact, Agreement, Argumentation Element) from the Project Explorer view to the Property

	area of the “OpenCert” tab.
Expected results	The system architecture and the assurance case specifications are correlated.
Priority	Should

Table 28. Test Case WP4_CEA_TC_005 for WP4_ACS_010

ID	**WP4_CEA_TC_005**
Scope	Provide the capability of generating a compositional assurance case.
Feature ID	WP4_ACS_010
Related use cases	“Define and navigate an assurance case structure”
Input	None
Steps	For every argument module: 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 manually claims (away goals) to the external claims (public goals) that support them (in other argument modules).
Expected results	A compositional assurance case is defined.
Priority	Must

Table 29. Test Case WP4_CEA_TC_006 for WP4_SDCA_002

ID	**WP4_CEA_TC_006**
Scope	System dependability co-verification and co-validation
Feature 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 V&V Tools to process 6. Execute a process step 7. Collect the Tools outputs in Co-V&V engine
Expected results	Outputs or Co-V&V tools, such as FMVEA tables or FT&AT.
Priority	Must

Table 30. Test Case WP4_CEA_TC_007 for WP4_SDCA_001 and WP4_SDCA_003

ID	**WP4_CEA_TC_007**
Scope	The system shall allow combinations of safety and security analysis. System dependability co-architecturing and co-design.
Feature ID	WP4_SDCA_001, WP4_SDCA_003
Related use cases	“Define/Perform Safety/Security Analysis”
Input	System/component definition fully specified at the planned analysis level.
Steps	1. Identify assets to be protected. 2. Select appropriate method and tool. 3. Decide on which level to analyse the [Sub-]System/Component. 4. Identify for each item all conceivable failure and threat modes with all possible causes and vulnerabilities and assess the possibility to detect the

	failures/attacks. 5. 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 31. Test Case WP4_CEA_TC_008 for WP4_ACS_011 and WP4_ACS_013

ID	**WP4_CEA_TC_008**
Scope	Assurance case status report. Provide quantitative confidence metrics about an assurance case in a report.
Feature ID	WP4_ACS_011, WP4_ACS_013
Related use cases	"Monitor status of argumentation"
Input	Argument diagram
Steps	For every Argument diagram: 1. Claims with no supported evidence or decomposed are identified. 2. The overall assurance case completion is presented.
Expected results	An assurance case being maintained.
Priority	Could

Table 32. Test Case WP4_CEA_TC_009 for WP4_CAC_010

ID	**WP4_CEA_TC_009**
Scope	Contract-based trade-off analysis
Feature ID	WP4_CAC_010
Related use cases	"Compare parameterized architecture"
Input	A parametrized architecture with available instantiation
Steps	1. Select a root component of the parametrized architecture. 2. Launch the trade-off analysis menu. 3. Choose a type of contract analysis. 4. Choose a set of instantiated architecture (at least two). 5. Use existing command "trade off analysis" in CHESS (see steps in the AMASS Platform User Manual [4]).
Expected results	Results of trade off analysis per concern appears in a new window.
Priority	Could

Table 33. Test Case WP4_CEA_TC_010 for WP4_CMA_002

ID	**WP4_CEA_TC_010**
Scope	Component contracts must support multiple concerns
Feature ID	WP4_CMA_002
Related use cases	"Structure properties into contracts", "Assign a contract to the component"
Input	The actor has proven knowledge about modelling languages for contracts. The component is available in the model.
Steps	1. Select the component model and open it. 2. Creates a new component contract. 3. Select the option of editing contract properties. 4. Use the Properties view to bind Formal Properties for the *concern 1* from the component as contract's assumption and guarantee. 5. Create a new component contract. 6. Select the option of editing contract properties. 7. Uses the Properties view to bind Formal Properties for the *concern 2* from

	the component as contract's assumption and guarantee.
Expected results	Contracts conform to *concern 1* and *concern 2* requirements.
Priority	Must

Table 34. Test Case WP4_CEA_TC_011 for WP4_ACS_006

ID	**WP4_CEA_TC_011**
Scope	Provide guidelines for argumentation.
Feature ID	WP4_ACS_006
Related use cases	None
Input	An argumentation model. Methodological guidance in D4.8 [14], Section 3.2. "Dependability Assurance Case Modelling"
Steps	Read the methodological guidance to check that provides guidelines about the assurance case edition based on the system/component development phase status.
Expected results	Guidelines for argumentation.
Priority	Could

4.3 Test Results

Table 35 presents, for each test case defined for the implemented Assurance Case Specification and Multi-concern assurance functionalities, the results of the execution, the status, and a rationale if the execution "Failed". The functionalities listed are described in the AMASS User Manual [4].

Table 35. Test results for functionalities implemented in WP4

Test Case ID	Execution Results	Status	Rationale
WP4_CEA_TC_001	An assurance case is modified with associated argumentation model	Passed	
WP4_CEA_TC_002	Cannot use the vocabulary in the argumentation model	Failed	The command to use the elements in the vocabulary does not work (Ctrl + space)
WP4_CEA_TC_003	Detailed guidelines provided in the methodological guidance.	Passed	*Improvement:* Include the guidelines information in the tools
WP4_CEA_TC_004	The system architecture and the assurance case specifications are not correlated.	Failed	Cannot drag and drop element from the assurance case to the OpenCert tab of the system model element.
WP4_CEA_TC_005	A compositional assurance case is defined	Passed but	The tool could be improved to reducing the manual steps. The user manual does not make clear how to do it in the tool.
WP4_CEA_TC_006	No results obtained	Failed	We were not able to configure the workflow

Test Case ID	Execution Results	Status	Rationale
			engine for testing
WP4_CEA_TC_007	This test case is covered by WP3_CEA_TC_007 for WP3_VVA_006 when fault tree is generated by invoking ConcertoFLA (for safety and security co-analysis)	Passed	
WP4_CEA_TC_008	Report contains the metrics	Passed	
WP4_CEA_TC_009	Results of trade off analysis per concern appears in a new window	Passed	
WP4_CEA_TC_0010	Defined security and safety related contracts	Passed	
WP4_CEA_TC_0011	Detailed guidelines provided in the methodological guidance	Passed	*Improvement:* Include the guidelines information in the tools

5. Testing and Validation for WP5-related Blocks

5.1 Functionalities

The functionalities concerning Evidence Management and Seamless interoperability blocks are defined in deliverable D2.1 [7]. Table 36 lists such functionalities planned for Prototype P2, as well as functionalities of Core Prototype and Prototype P1 which needed to be revised.

Table 36. Evidence Management and seamless interoperability functionalities

ID	Feature	Prototype
WP5_CW_003	Collaborative management of compliance with standards and of process assurance	Core
WP5_CW_004	Collaborative re-certification needs & consequences analysis	Core
WP5_CW_007	Collaborative assurance evidence management	Core
WP5_CW_008	Collaborative product reuse needs & consequences analysis	Core
WP5_CW_009	Collaborative assurance case specification	Core
WP5_CW_010	Collaborative compliance needs specification	Core
WP5_CW_011	Collaborative assurance assessment	Core
WP5_CW_012	Collaborative compliance assessment	Core
WP5_TQ_001	Tool qualification information needs	Core
WP5_TQ_002	Tool quality evidence management	Core
WP5_TQ_003	Tool quality information import	Core
WP5_TQ_004	Tool quality needs indication	Core
WP5_TQ_005	Tool quality requirements fulfilment	Core
WP5_EM_007	Derivation of evidence characterization model	Core
WP5_EM_016	Evidence report generation	Core
WP5_AM_003	User action log	Core
WP5_DM_001	Multi-platform availability	Core
WP5_DM_002	Simultaneous data access	Core
WP5_DM_005	System artefact information storage	Core
WP5_DM_006	Standard formats storage	Core
WP5_DM_007	Data versioning	Core
WP5_TI_001	Automatic data collection	Core
WP5_TI_002	Automatic data export	Core
WP5_EM_008	Visualization of chains of evidence	P1
WP5_EM_015	Resource part selection	P1
WP5_TI_003	Tool chain deployment support	P1
WP5_TI_005	System specification tools interoperability	P1
WP5_TI_006	V&V tools interoperability	P1
WP5_TI_014	Client-server support	P1
WP5_TI_017	Standards-based interoperability	P1
WP5_TI_018	Extended standard-based interoperability	P1
WP5_EM_009	Suggestion of evidence traces	P2
WP5_EM_012	Evidence trace verification	P2

WP5_AM_001	User authentication	P2
WP5_AM_002	User access	P2
WP5_AM_004	User profiles	P2
WP5_AM_005	Access rights groups	P2
WP5_DM_003	Consistent data access	P2
WP5_DM_004	Real-time data access feedback	P2
WP5_TI_004	System analysis tools interoperability	P2
WP5_TI_010	Interoperability throughout CPS lifecycle	P2
WP5_TI_011	Non-proprietary data exchange	P2
WP5_TI_012	Data entry effort	P2
WP5_TI_013	Continuous data management	P2
WP5_TI_015	Service offer and discovery	P2
WP5_TI_016	Performance monitoring	P2
WP5_CW_001	Collaborative system analysis	P2
WP5_CW_002	Collaborative system specification	P2
WP5_CW_005	Collaborative system V&V	P2
WP5_CW_006	Collaborative model-based systems engineering	P2
WP5_CW_013	Metrics & measurements reports	P2

5.2 Test Cases

This section defines the test cases to validate the implementation of the Evidence Management and Seamless interoperability functionalities specified in Table 36. The listed test cases are based on the use case scenarios for each feature defined in deliverable D2.4 [6], where these are available.

Table 37. Test Case WP5_TRC_TC_001 for WP5_AM_001

ID	**WP5_TRC_TC_001**
Scope	User authentication
Feature ID	WP5_AM_001
Related use cases	None
Input	AMASS Platform
Steps	1. Start the AMASS Platform 2. Wait for the Credentials form to appear 3. Provide a suitable username and password 4. Wait for the AMASS application to load 5. Check in the “Project explorer” window that the public Assurance projects are visible.
Expected results	Access to the public Assurance project resources
Priority	Must

Table 38. Test Case WP5_TRC_TC_002 for WP5_AM_002

ID	**WP5_TRC_TC_002**
Scope	User access
Feature ID	WP5_AM_002
Related use cases	None
Input	AMASS Platform

Steps	1. Start the AMASS Platform 2. Wait for the Credentials form to appear 3. Log in as Administrator 4. Go to the Manage Security under the connected CDO session 5. Manage another user and: a. Forbid the access to one public project b. Restrict to read-only the access to a resource inside other public project 6. Quit the AMASS Platform 7. Start the AMASS Platform 8. Log in as the modified user in step 5 9. Check that this user cannot see the project whose access rights have been modified in step 5. a 10. Check that this user cannot modify the resource whose access rights have been modified in step 5. b
Expected results	The modified user cannot see the project whose access rights have been modified in step 5. a The modified user cannot modify the resource whose access rights have been modified in step 5. b
Priority	Should

Table 39. Test Case WP5_TRC_TC_003 for WP5_AM_003

ID	**WP5_TRC_TC_003**
Scope	User action log
Feature ID	WP5_AM_003
Related use cases	None
Input	AMASS Platform
Steps	1. Start the AMASS platform 2. Wait for the Credentials form to appear 3. Log in as a non-administrator user 4. Perform some modification of any resource 5. Quit the AMASS platform 6. Start the AMASS platform 7. Wait for the Credentials form to appear 8. Log in as Administrator 9. Go to the CDO Repository view, right-click and select “History…” 10. Check that there is a new entry in the History log tracking the change performed by the user used in step 3 11. Go to the CDO session, right-click and select “History …” 12. Select the date of the previous date and click on accept 13. Locate the resource modified in step 4 14. Check that the modifications performed in step 4 are not present in the History view.
Expected results	The History View in step 12 shows the state of the platform before being modified.
Priority	Must

Table 40. Test Case WP5_TRC_TC_004 for WP5_AM_004

ID	WP5_TRC_TC_004
Scope	User profiles
Feature ID	WP5_AM_004
Related use cases	None
Input	AMASS platform
Steps	1. Start the AMASS Platform 2. Wait for the Credentials form to appear 3. Log in as Administrator 4. Go to the Manage Security under the connected CDO session 5. Create two different profiles with different access rights over the existing resources 6. Assign a non-administrator user to both created profiles 7. Quit the AMASS platform 8. Start the AMASS platform 9. Wait for the Credentials form to appear 10. Log in as the modified user 11. Check that the permissions given in step 5 are hold for this user.
Expected results	The non-administrator user has access exactly to the rights defined in the profiles.
Priority	Should

Table 41. Test Case WP5_TRC_TC_005 for WP5_AM_005

ID	WP5_TRC_TC_005
Scope	Access rights groups
Feature ID	WP5_AM_005
Related use cases	None
Input	AMASS Platform
Steps	1. Start the AMASS platform 2. Wait for the Credentials form to appear 3. Log in as Administrator 4. Go to the Manage Security under the connected CDO session 5. Create two different profiles with different access rights over the existing resources 6. Assign a non-administrator user for each created profile 7. Quit the AMASS platform 8. Start the AMASS platform 9. Wait for the Credentials form to appear 10. Log in as the first modified user 11. Check that the permissions given in step 5 are hold for this user 12. Quit the AMASS Platform 13. Start the AMASS Platform 14. Wait for the Credentials form to appear 15. Log in as the second modified user 16. Check that the permissions given in step 5 are hold for this user.
Expected results	Each user has access exactly to the rights defined in the profiles.
Priority	Must

Table 42. Test Case WP5_TRC_TC_006 for WP5_DM_001

ID	WP5_TRC_CEA_TC_006
Scope	Multi-platform availability
Feature ID	WP5_DM_001
Related use cases	None
Input	AMASS Platform
Steps	1. Start the AMASS platform from the bundle installation 2. Go to the website http://amass.tecnalia.com:8080
Expected results	From both applications (web and desktop) the repository is available
Priority	Should

Table 43. Test Case WP5_TRC_TC_007 for WP5_DM_002

ID	WP5_TRC_TC_007
Scope	Simultaneous data access
Feature ID	WP5_DM_002
Related use cases	None
Input	AMASS Platform
Steps	1. Start the AMASS platform from the bundle installation 2. Open a model 3. Start another instance of the AMASS Platform from the bundle installation 4. Open the same model as done in step 2
Expected results	Both instances can see and modify the model opened in steps 2 and 4 respectively
Priority	Must

Table 44. Test Case WP5_TRC_TC_008 for WP5_DM_005

ID	WP5_TRC_TC_008
Scope	System artefact information storage
Feature ID	WP5_DM_005
Related use cases	None
Input	AMASS Platform
Steps	1. Start the AMASS tool platform from the bundle installation. 2. In the “OSLC-KM” menu option, choose “Import Evidence from File” for the different engineering workproducts.
Expected results	In the Import wizard there are several more than 10 types of files to be imported as evidences.
Priority	Must

Table 45. Test Case WP5_TRC_TC_009 for WP5_DM_006

ID	WP5_TRC_TC_009
Scope	Standard formats storage
Feature ID	WP5_DM_006
Related use cases	None
Input	AMASS platform
Steps	1. Start the AMASS tool platform from the bundle installation. 2. In the “OSLC-KM” menu option, choose “Import Evidence from File” for the different engineering standards.
Expected results	In the Import wizard there are several standards to be imported.

Priority	Must

Table 46. Test Case WP5_TRC_TC_010 for WP5_DM_007

ID	**WP5_TRC_TC_010**
Scope	Data versioning
Feature ID	WP5_DM_007
Related use cases	None
Input	AMASS Platform
Steps	1. Start the AMASS tool platform from the bundle installation 2. Open any model 3. Perform any modification 4. Store the model 5. Open the History view of the model
Expected results	There are traces of the actions performed in the selected model.
Priority	Must

Table 47. Test Case WP5_TRC_TC_011 for WP5_TI_001

ID	**WP5_TRC_TC_011**
Scope	Automatic data collection
Feature ID	WP5_TI_001
Related use cases	None
Input	AMASS Platform
Steps	1. Start the AMASS tool platform from the bundle installation 2. In the “OSLC-KM” menu option, choose “Import Evidence from File” 3. Choose a papyrus model to be imported 4. Import it
Expected results	The Papyrus model is imported as a new evidence in the selected place in the repository.
Priority	Must

Table 48. Test Case WP5_TRC_TC_012 for WP5_TI_002

ID	**WP5_TRC_TC_012**
Scope	Automatic data export
Feature ID	WP5_TI_002
Related use cases	None
Input	VERIFICATION Studio tool
Steps	1. Open the VERIFICATION Studio tool 2. Connect to a Papyrus model via OSLC-KM 3. Log in 4. Assess correctness quality the papyrus requirements in VERIFICATION Studio.
Expected results	VERIFICATION Studio shall assess the correctness quality of the requirements of the Papyrus file.
Priority	Must

Table 49. Test Case WP5_TAS_TC_001 for WP5_CW

ID	**WP5_TAS_TC_001**
Scope	Collaborative Assurance
Feature ID	WP5_CW_003

	WP5_CW_004 WP5_CW_007 WP5_CW_008 WP5_CW_009 WP5_CW_010 WP5_CW_011 WP5_CW_012
Related use cases	"Concurrent Assurance Information Edition"
Input	Existing Assurance Project and Compliance Map.
Steps	1. An AMASS user accesses some data: a. Assurance Project b. Project Baseline c. Compliance Maps 2. Another AMASS user accesses the same data. 3. The first AMASS user changes some data: a. Create and Assurance Project and Baseline b. Create or update a Project Baseline c. Edit a Project Baseline d. Edit Compliance Maps e. Cross-Standard reuse f. Cross-Project reuse 4. The second AMASS user is notified of the data change.
Expected results	1. Collaboration for management of compliance with standards and process assurance is supported (WP5_CW_003). 2. Collaboration for re-certification needs & consequences analysis is supported (WP5_CW_004). 3. Collaboration for assurance evidence management is supported (WP5_CW_007). 4. Collaboration for product reuse needs & consequences analysis is supported (WP5_CW_008). 5. Collaboration for assurance case specification is supported (WP5_CW_009). 6. Collaboration for compliance needs specification is supported (WP5_CW_010). 7. Collaboration for assurance assessment is supported (WP5_CW_011). 8. Collaboration for compliance assessment is supported (WP5_CW_012). # If a (possible) conflict arises from concurrent data access, the AMASS users will be notified about it.
Priority	Should

Table 50. Test Case WP5_TAS_TC_002 for WP5_TQ_001 to WP5_TQ_005

ID	**WP5_TAS_TC_002**
Scope	Tool Qualification
Feature ID	WP5_TQ_001 WP5_TQ_002 WP5_TQ_003 WP5_TQ_004 WP5_TQ_005
Related use cases	None
Input	Existing CHESS model with requirements.

Steps	1. Launch VERIFICATION Studio. 2. Create a new OSLC-KM connection in RQA to reference the CHESS project. 3. In the OSLC-KM Connection window select SysML Type, Papyrus SysML SubType and the SysML File. 4. In order to assess the quality of the model, it is necessary to select a template that stores the set of metrics (e.g. TRC Metric Configuration). 5. When the configuration is created it is possible to connect to the project and the artefacts of the model are imported to the RQA tool. 6. CCC metrics can be specified in the "Project configuration" menu. 7. Assess the CCC quality for the whole specification. 8. The quality of the different artefacts of the model is shown in the RQA tool. 9. This quality measure can be exported as evidence to an AMASS repository. For this it is necessary to include the location of the server (e.g. amass.tecnalia.com:2036), select an assurance project and set an evidence name. 10. Launch the AMASS Platform. 11. Open the OpenCert perspective and browse in the Repository Explorer to check that the evidences have been uploaded to the selected assurance project.
Expected results	1. Needs regarding qualification for the engineering tools used in a CPS' lifecycle is specified (WP5_TQ_001). 2. Managed evidences of tool quality in CCC reports and in the AMASS repository (WP5_TQ_002). 3. Tool quality information (e.g. CCC reports) exported to the AMASS repository (WP5_TQ_003). 4. Degree to which tool quality requirements and needs have been fulfilled (WP5_TQ_004 & WP5_TQ_005).
Priority	Should

Table 51. Test Case WP5_TAS_TC_003 for WP5_EM

ID	**WP5_TAS_TC_003**
Scope	Evidence Management
Feature ID	WP5_EM_007 WP5_EM_016 WP5_EM_008 WP5_EM_015 WP5_EM_009 WP5_EM_012
Related use cases	"Link Artefact with External Tool", "Characterise Artefact", "Specify Traceability between Assurance Assets".
Input	Existing Assurance Project and Artefact.
Steps	1. The Assurance Manager creates an Artefact for an Artefact Definition. 2. The Assurance Manager specifies the information of the Artefact. 3. The Assurance Manager selects an Assurance Asset. 4. The Assurance Manager adds a Trace Link to the Asset. 5. The Assurance Manager indicates the Assurance Asset or Assets that corresponds to the target of the Trace Link. 6. The Assurance Manager selects an Artefact. 7. The Assurance Manager adds a Resource to the Artefact.

	8. The Assurance Manager specifies the information about an External Tool in the Resource.
Expected results	1. An evidence characterisation model is derived from the baseline of an assurance project (WP5_EM_007). 2. The chains of evidence to which an evidence artefact belongs are displayed (WP5_EM_008). 3. When specifying relationships for an evidence artefact, evidence artefacts to which the first evidence artefact might relate are suggested (WP5_EM_009). 4. Quality of the relationships between evidence artefacts is analysed (WP5_EM_012). 5. When indicating the location of the resource that an evidence artefact represents in the system, it is allowed to select a part of the resource (WP5_EM_015). 6. Reports, checklists, and evidence for certification purposes are automatically generated (WP5_EM_007).
Priority	Should

Table 52. Test Case WP5_TAS_TC_004 for WP5_CW

ID	**WP5_TAS_TC_004**
Scope	Collaborative System Analysis
Feature ID	WP5_CW_001 WP5_CW_002 WP5_CW_005 WP5_CW_006 WP5_CW_013
Related use cases	“Concurrent Assurance Information Edition”
Input	Existing CHESS model in CDO.
Steps	1. User1 opens/loads a CHESS model in the CDO repository. 2. User2 accesses to the same model. 3. User1 can modify some objects in the model (e.g. BDD diagrams) and User2 can also view the changes. 4. User2 can analyse some objects in the model and User1 can also view the changes. 5. User1/User2 can see the number of CDO connections using the CDO Collaboration view.
Expected results	1. Collaboration for system modelling and model-based systems engineering will be supported (WP5_CW_002 & WP5_CW_006). 2. AMASS platform will support the collaboration for system analysis and V&V (WP5_CW_001 & WP5_CW_005). 3. AMASS Platform will manage metrics and measurements about collaborative work (WP5_CW_013)
Priority	Should

Table 53. Test Case WP5_CEA_TC_001 for WP5_TI

ID	**WP5_CEA_TC_001**
Scope	Interoperability features
Feature ID	WP5_TI_003, WP5_TI_04, WP5_TI_005, WP5_TI_006, WP5_TI_010, WP5_TI_011, WP5_TI_012, WP5_TI_013, WP5_TI_014, WP5_TI_015, WP5_TI_016, WP5_TI_017, WP5_TI_018

Related use cases	“Characterise Toolchain”, “Specify Tool Connection Information”
Input	Tool chain information is available
Steps	1. The Assurance Manager selects the tools that will be part of the toolchain. 2. The Assurance Manager specifies the interactions between the tools. 3. The Assurance Manager specifies the necessary information to enable the toolchain. 4. The Assurance Manager is informed about the success of toolchain connection. 5. The Assurance Manager creates a new tool connection. 6. The Assurance Manager specifies the required information to the tool connection. 7. The Assurance Manager is provided information about the success of tool connection.
Expected results	Tool chain information is available in the platform
Priority	Should

Table 54. Test Case WP5_CEA_TC_002 for WP5_DM_003 and WP5_DM_004

ID	**WP5_CEA_TC_002**
Scope	Data access
Feature ID	WP5_DM_003 WP5_DM_004
Related use cases	“Concurrent Assurance Information Edition”
Input	Data consistency
Steps	1. An AMASS user accesses some data 2. Another AMASS user accesses the same data 3. Both users are notified of the concurrent access 4. The first AMASS user changes some data 5. The second AMASS user is notified of the data change
Expected results	Data remains consistent
Priority	Should

5.3 Test Results

Table 55 presents, for each test case defined for the implemented Evidence Management and Seamless interoperability functionalities, the results of the execution, the status and a rationale when the execution status is “Failed” or “Passed but”. The installation instructions for the tools used for the validation are provided in the AMASS Developers Guide [5]. The AMASS User Manual [4] was used to understand how the selected functionalities were working.

Table 55. Test results for functionalities implemented in WP5

Test Case ID	Execution Results	Status	Rationale
WP5_TRC_TC_001	The credentials are required when the platform is started.	Passed	
WP5_TRC_TC_002	Work as expected.	Passed	
WP5_TRC_TC_003	Work as expected.	Passed	
WP5_TRC_TC_004	Work as expected.	Passed	
WP5_TRC_TC_005	Work as expected.	Passed	
WP5_TRC_TC_006	From both applications, an	Passed	

Test Case ID	Execution Results	Status	Rationale
	instance of AMASS can work with the CDO repository		
WP5_TRC_TC_007	Works as expected	Passed	
WP5_TRC_TC_008	Done as expected. There are more than 10 types of files to be imported.	Passed	
WP5_TRC_TC_009	Works as expected. One found several standards, such as SysML, XMI, FMI/FMU, etc.	Passed	
WP5_TRC_TC_010	Works as expected. Changes are tracked each time a model is saved.	Passed	
WP5_TRC_TC_011	Works as expected. The Papyrus model is imported as a new evidence.	Passed	
WP5_TRC_TC_012	VERIFICATION Studio can assess the correctness quality of the requirements of the Papyrus file	Passed	
WP5_TAS_TC_001	VERIFICATION Studio can load the requirements in the Papyrus file and perform quality assessment on it.	Passed but	Incomplete setup data to perform all the steps.
WP5_TAS_TC_002	Tool Qualification is present in CCC reports and exported to the AMASS platform	Passed but	*Note*: Nothing is said in the User Manual concerning how to edit or create a CCC metric, so default metrics have been used.
WP5_TAS_TC_003	Evidence management features works mostly as expected	Passed but	Artefact creation and edition raises several errors that makes it difficult to perform the steps
WP5_TAS_TC_004	Most of the features work as expected, except those related to the comments in the “Rationale” column	Passed but	Validation commands are supported only for local CHESS models. Loading OSLC Service Provider Catalogue has failed. *Note*: Nothing is said in the User Manual concerning: • CDO Administration • CDO Collaboration • CDO Time Machine • CDO Watch List
WP5_CEA_TC_001	Tool connection established with formal verification tool	Passed	

Test Case ID	Execution Results	Status	Rationale
	(OCRA, NuXmv) and with external tools through OSLC		
WP5_CEA_TC_002	Concurrent access to data from different users	Passed	

6. Testing and Validation for WP6-related Blocks

6.1 Functionalities

Table 56 is an excerpt of some functionalities defined in the D2.1 deliverable [7] for Compliance management and Cross and intra-domain reuse for Prototype P2, in addition to the functionalities of Core Prototype and Prototype P1 to be revised.

Table 56. Compliance management and Cross and intra domain reuse functionalities

ID	Feature	Prototype
WP6_CM_001	Modelling of standards	P1
WP6_CM_006	Compliance Status to Externals	P1
WP6_CM_010	Compliance map generation from argument evidences	P1
WP6_RA_001	Intra-Domain, Intra standard, Reuse Assistance	P1
WP6_RA_002	Intra-Domain, Cross standards, Reuse Assistance	P1
WP6_RA_003	Intra-Domain, Cross versions, Reuse Assistance	P1
WP6_RA_004	Cross-Domain Reuse Assistance	P1
WP6_RA_005	Intra-Domain, Intra standard, Different Stakeholders, Reuse/Integration Assistance	P1
WP6_PPA_001	The AMASS tools must support variability management at process level	P1
WP6_PPA_002	Semi-automatic generation of product arguments	P1
WP6_PPA_003	Semi-automatic generation of process arguments	P1
WP6_PPA_004	The AMASS tools must support management of variability at the component level	P1
WP6_PPA_005	The AMASS tools must support variability management at the assurance case level	P2
WP6_CM_003	Correlating processes to the requirements	P2
WP6_CM_004	Triggering compliance Checking	P2
WP6_CM_007	Useful Feedback Upon Violations	P2
WP6_CM_009	Process Compliance (formal) management	P2
WP6_RA_006	Reusable off the shelf components	P2
WP6_SEM_001	Semantics-based mapping of standards	P2

6.2 Test Cases

This section defines the test cases used to validate the implementation of the Compliance management and Cross and intra-domain reuse functionalities. The listed test cases are based on the use case scenarios for each feature defined in deliverable D2.4 [6], where these are available.

Table 57. Test Case WP6_VIF_TC_001 for WP6_PPA_001

ID	WP6_TC_VIF_001
Scope	Variability management at process level
Feature ID	WP6_PPA_001
Related use cases	“Manage process variability”

Input	EPF-C process library (base model)
Steps	1. Import the Base Model into a project via “Seamless Integration between EPF Composer and BVR Tool”. A folder “error_free_models” is generated. 2. Open “model.xmi” stored in the folder “delivery processes” with “Sample reflective Ecore model editor” 3. Manage variability via the Variability, Resolution, and Realization editors. 3.1. In the Realization Editor, Placements, Replacements, “Fragment Substitutions” and “Bindings” are defined. 3.2. The Resolution editor is used to define correct values for each “choice”. 3.3. The Resolution has to be validated 3.4. Execute the model in the Resolution editor (select “model.xmi” in the navigator) 3.5. The generated “model.xmi” is available in the folder of the original “Delivery process” (not in the folder “error_free_models”). 4. Open the exported “Delivery process” in EPF-C and check the tailored “Work Breakdown Structure”.
Expected results	EPF-C process library which has been tailored according to variability parameters.
Priority	Must

Table 58. Test Case WP6_VIF_TC_002 for WP6_PPA_004

ID	**WP6_TC_VIF_002**
Scope	Variability management at the component level
Feature ID	WP6_PPA_004
Related use cases	“Manage product variability”
Input	CHESSS Component model
Steps	1. Import the test model into work space “Existing Projects into Work Space” 2. Open Resolution and Realization editor 3. Open uml model with “UML Model Editor” 4. Manage variability via the Variability, Resolution, and Realization editors. 4.1. In the Realization Editor, Placements, Replacements, “Fragment Substitutions” and “Bindings” are defined. 4.2. The Resolution editor is used to define correct values for each “choice”. 4.3. The Resolution has to be validated 4.4. Execute the model in the Resolution editor (select uml model (*.uml) in the navigator) 4.5. The generated “.uml” is available in the folder and the original model will be renamed to “*_old.uml” 5. Open the uml model (UML Model Editor) where the defined changes have been realized. 6. For the graphical view use “Papyrus editor core” to open the “*.di” file
Expected results	CHESS Component model which has been tailored according to variability parameters.
Priority	Shall

Table 59. Test Case WP6_VIF_TC_003 for WP6_PPA_005

ID	**WP6_TC_VIF_003**
Scope	Variability management at the assurance case level

Feature ID	WP6_PPA_005
Related use cases	"Manage assurance case variability"
Input	OpenCert argumentation diagram
Steps	1. Import the test model into work space "Existing Projects into Work Space" 2. Open Resolution and Realization editor 3. Open *.arg model with "ARG model editor" and for a better overview the *.arg_diagram" with "ARG diagram editing" 4. Manage variability via the Variability, Resolution, and Realization editors. 4.1. In the Realization Editor, Placements, Replacements, "Fragment Substitutions" and "Bindings" are defined. 4.2. The Resolution editor is used to define correct values for each "choice". 4.3. The Resolution has to be validated 5. Execute the model in the Resolution editor (select "*.arg" file in the navigator) 5.1. Select a name for the new file 6. The generated "*.arg" is available in the folder and the original stays untouched 7. Open the new file (ARG model editor) and check the realized changes. 8. To create the graphical view, use the command "Initialize arg_diagram diagram file" 8.1. Check the "outline view" and select all elements under "case transient" and drop them into the "*.arg_diagram" 8.2. Use the command "Arrange all" to create the GSN structure.
Expected results	OpenCert argumentation diagram which has been tailored according to variability parameters.
Priority	Shall

Table 60. Test Case WP6_CEA_TC_001 for WP6_CM_001

ID	**WP6_CEA_TC_001**
Scope	Retrieve, digitalize and store a set of norms, recommendations, standards, or quality models.
Feature ID	WP6_CM_001
Related use cases	"Capture information from standards"
Input	Standard information
Steps	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 61. Test Case WP6_KMT_TC_002 for WP6_RA

ID	**WP6_KMT_TC_002**
Scope	Intra-Domain, Cross standards, Cross versions, Reuse Assistance
Feature ID	WP6_RA_002 WP6_RA_003 WP6_RA_004

	WP6_RA_006
Related use cases	"Assist for Cross-Standards Assurance Assets Reuse"
Input	• Two standards A and B • Equivalence maps model between the standard B with standard A. • A source assurance project which includes assurance assets (evidence, process, argumentation, compliance models) based in a standard model A. • A newly created assurance project based in the standard model B.
Steps	1. The actor opens the newly created assurance project (target project), starts the reuse assistant (cross-standard interface), and selects the reusable assurance project (source project). 2. Evidence models from the target project must be created automatically. The model elements will follow the same structure and naming as the standards (baseline in this case) model. 3. Select the Equivalence Map model and the Equivalence Map Group between the standard B and the Standard A. 4. Once a baseline model element of the target project is selected, the actor can discover reuse opportunities by using equivalence maps. 5. Once selected all the desired source evidence model elements to be reused, click the "Reuse" button and those evidence elements are copied from the source to the target project. 6. The actor opens the newly created assurance project (target project), starts the reuse assistant (Reuse View interface), and selects the reusable assurance project (source project). 7. All the source project assurance models are shown in selectable lists, including evidence, process, argumentation and baseline (compliance information) models and can be individually selected. 8. Double clicking over some of the listed model of a type different to evidence to select model subparts to be copied. 9. Index all the evidence model contents for Elastic Search. 10. Index the source assurance project using OSLC-KM tool. 11. Double click over any evidence model of the source assurance project. 12. Right click over the "Artefact Model Element" and search reusable assets in the selected source evidence model according a text using both searching methods. (The text should be any word appearing in model content). 13. Once selected all the desired source assurance models and elements to be reused, click the "Reuse" button. Just those elements are copied from the source to the target project.
Expected results	AMASS models updated according to the reuse scope, including evidence models, argumentation models, process models and compliance information
Priority	Must

Table 62. Test Case WP6_KMT_TC_003 for WP6_RA_002 to WP6_RA_004

ID	**WP6_KMT_TC_003**
Scope	Intra-Domain, Cross standards, Cross versions, Reuse Assistance.
Feature 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	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 63. Test Case WP6_KMT_TC_004 for WP6_RA_001 to WP6_RA_005

ID	**WP6_KMT_TC_004**
Scope	Intra/Cross-Domain, Intra/Cross standard, Cross versions, Different Stakeholders, Reuse/Integration Assistance
Feature 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	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 64. Test Case WP6_KMT_TC_005 for WP6_PPA_002

ID	**WP6_KMT_TC_005**
Scope	Semi-automatic generation of product arguments.
Feature ID	WP6_PPA_002
Related use cases	"Semi-automatic generation of product arguments"
Input	Strong and weak component contracts shall be already defined and associated with claims, context statements and evidence artefacts. The weak contracts shall be either selected for usage in the given context, or all weak contract assumptions shall be validated. The contract refinement analysis shall be already performed, either for the selected contracts, or for all the weak contracts.
Steps	1. Select the "Generate argumentation fragments" feature. 2. Select either new or existing assurance project as the destination for the argument-fragments. 3. The platform validates the system model and extracts the information needed for the argument-fragment generation for each component. 4. The platform 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 65. Test Case WP6_CEA_TC_006 for WP6_PPA_003

ID	**WP6_CEA_TC_006**
Scope	Semi-automatic generation of process arguments
Feature ID	WP6_PPA_003
Related use cases	"Automatic generation of process arguments"
Input	A process model

Steps	1. Select the “Generate argumentation fragments” feature. 2. Select 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 66. Test Case WP6_KMT_TC_008 for WP6_CM and WP6_SEM_001

ID	**WP6_KMT_TC_007**
Scope	Compliance map
Feature ID	WP6_CM_006, WP6_CM_003, WP6_CM_004, WP6_CM_007, WP6_CM_009, WP6_CM_010, WP6_SEM_001
Related use cases	None
Input	Artefact Model (.evidence), Process Model (.process), Argumentation Model (.arg)
Steps	1. Create Compliance Maps by clicking on the “Mapping Set” button on the Properties form of the baseline configuration using the tree view editor. 2. In the *left zone*, load the type of elements for which we want to make the compliance maps. 3. In the *middle zone,* create different filters for activities, artefacts, requirements, roles and techniques: a. Select a Mapping Model and a Map Group b. Select the Filter Map element c. Select the target model d. Select or create the Compliance Map 4. Select an element from the source model and check or uncheck elements from the target model. 5. Monitor the compliance status by clicking on the “Mapping Table” button on the properties view of the Base Framework element of one Baseline using the tree view editor. 6. Filter the mapping by criticality level, applicability level, map model, a map group of the selected map model, a type of element that could be mapped and the mapping type. 7. Double click in any element of the table to access to the Compliance Map tailored feature, to create or modify the compliance map information of the double-clicked element. 8. Double click in one element of the target list to access to detailed information of the selected target element
Expected results	1. Compliance map created 2. Compliance map checking
Priority	Must

6.3 Test Results

Table 67 presents, for each test case defined for WP6-related functionalities, the results of the execution, the status, and a rationale when the execution was not fully satisfying the expected results.

The installation instructions for the validation environment and the description of the selected functionalities can be found in the AMASS Developers Guide [5] and the AMASS User Manual [4], respectively. As testing data, we used validation data from the AMASS SVN, EPF-C libraries and an OpenCert argumentation diagram.

Table 67. Test results for functionalities implemented in WP6

Test Case ID	Execution Results	Status	Rationale
WP6_VIF_TC_001	The process modified according to the variability parameters has been created.	Passed	
WP6_VIF_TC_002	The "*.uml" file modified according to the variability parameters has been created.	Passed	
WP6_VIF_TC_003	The "*.arg " file modified according to the variability parameters has been created.	Passed	
WP6_CEA_TC_001	A standard model is created	Passed	
WP6_KMT_TC_002	Artefacts could be index, are found based on selection and can be reused	Passed	
WP6_KMT_TC_003	Equivalent model elements were selected (and can be reused in next step)	Passed	
WP6_KMT_TC_004	Copies of assets have been created	Passed	
WP6_KMT_TC_005	An argument fragment was created	Passed	This test passed once with a simple example (later facing test problems so could not repeat)
WP6_CEA_TC_006	Arguments model is generated from process model	Passed	
WP6_KMT_TC_007	Compliance map was created	Passed	

7. AMASS Prototype P2 Validation Summary

Table 68 presents the validation status of the planned functionalities at the time of release of the AMASS Prototype P2. In total, 103 functionalities have been taken into consideration for the validation of Prototype P2 and 59 test cases have been defined. 50 test cases successfully "Passed", 6 test cases resulted with the status "Passed but" and 3 test cases "Failed" to provide the expected results. For those test cases with the status "Failed" or "Passed but", some tickets have been created in the AMASS issue tracker to report the issues to the partners responsible for implementing the respective functionalities.

Table 68. Results of the test cases for functionalities implemented in Prototype P2

Test Results Status	WP3 related functionalities	WP4 related functionalities	WP5 related functionalities	WP6 related functionalities	AMASS platform
Passed	19	7	14	10	50
Passed but	1	1	4	0	6
Failed	0	3	0	0	3
Total	**20**	**11**	**18**	**10**	**59**

Figure 5 summarizes the overall validation results at this stage per WP. The deliverable D2.9 "AMASS platform validation" will document the final validation of AMASS platform – this deliverable will review functionalities that have not been validated yet, statuses of tickets in the issue tracker and how they have been taken into account.

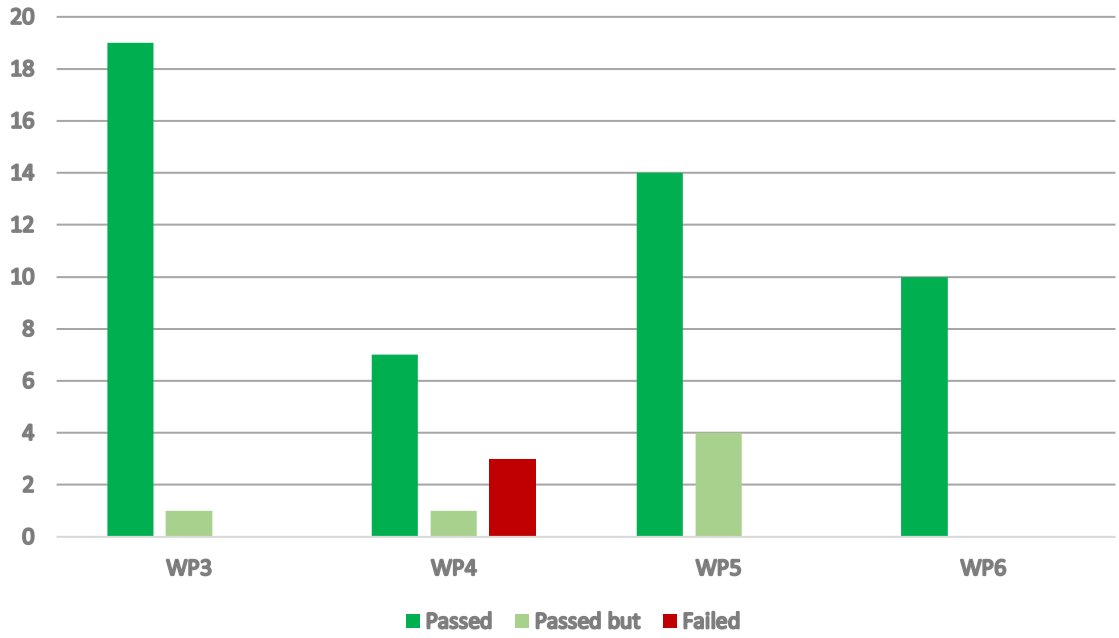

Figure 5. AMASS platform features validation synthesis per WP

Figure 6 shows the overall results from the validation phase for the AMASS Prototype P2 release:

- 85% of functionalities have met their specifications.
- 10% of functionalities have partially met their specifications.
- 5% of functionalities have failed to meet their specifications.
- 0 % of functionalities were not tested.

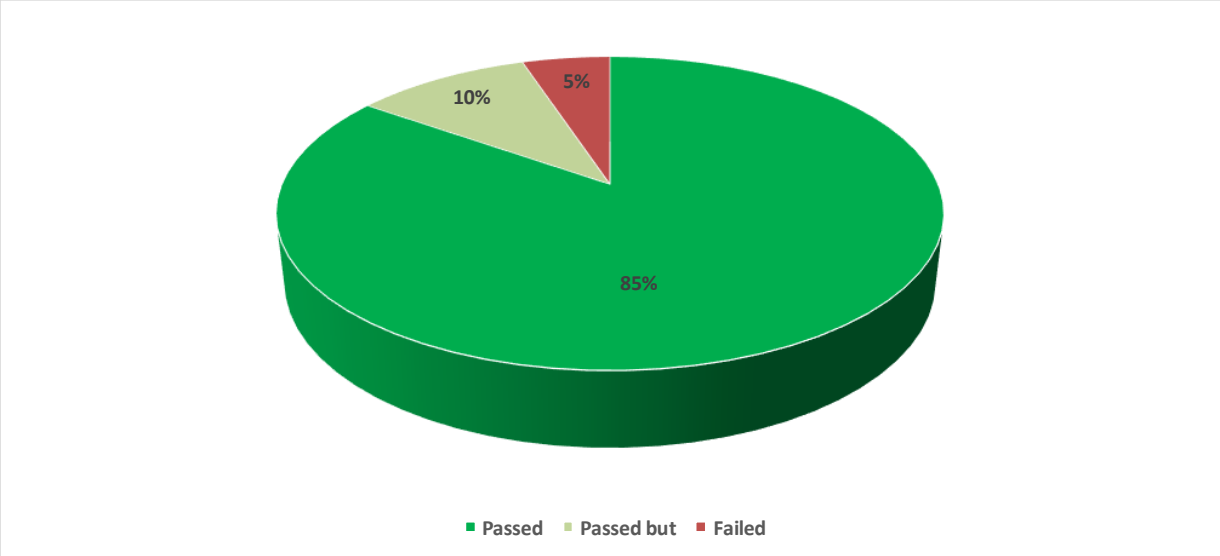

Figure 6. AMASS platform features validation

8. Conclusion

This report documents the current results of testing and validation of the AMASS Prototype P2. The validation has been based on the analysis of requirements and corresponding functionalities planned for the AMASS platform. These have been refined into test cases compatible with the current version of the AMASS platform. The previous validation results of Core Prototype and Prototype P1 have also been revised.

Three main activities were performed during the Prototype P2 validation phase:

1. Testing and validation of pending (not implemented) and postponed (not tested) functionalities with respect to previous developments. This was undertaken prior to validating new developments.
2. Validating the new implemented tool features.
3. Validating that the AMASS Platform is integrated in a comprehensive toolset.

All the test cases defined for the validation have been executed. Also, the obtained results are better than those achieved during the validation of Prototype P1, since only 5% of the test cases have failed.

All issues have been reported to the implementation team and mitigation actions have been internally planned to handle them before they impact in the use cases development.

Deliverable D2.9 “AMASS platform validation” will document the final validation of the AMASS platform – this deliverable will provide a global overview of the different validation campaigns with regards to coverage of the user requirements and business cases. The deliverable will also include a TRL assessment of key components of the platform to demonstrate that it operates at TRL5.

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
CCC	Correctness, Consistency and Completeness
CDO	Connected Data Objects
CHESS	Composition with Guarantees for High-integrity Embedded Software Components Assembly
CPS	Cyber Physical Systems
CPU	Central Processing Unit
CSD	Composite Structure Diagram
EPF	Eclipse Process Framework
FMVEA	Failure Modes, Vulnerabilities and Effect Analysis
FT&AT	Fault Tree & Attack Tree
FTA	Fault tree analysis
FMI	Functional Mock-up Interface
FMU	Functional Mock-up Unit
GB	Gigabyte
HARA	Hazard Analysis and Risk Assessment
HMI	Human Machine Interface
IMA	Integrated Modular Avionics
IBD	Internal Block Diagram
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
OPENCOSS	Open Platform for EvolutioNary Certification Of Safety-critical Systems
OSLC	Open Services for Lifecycle Collaboration
RAM	Random-access memory
RQA	Requirement Quality Analyser
STO	Scientific and Technical Objective
UML	Unified Modelling Language
URL	Uniform Resource Locator
TARA	Threat Analysis and Risk Assessment
TRL	Technology Readiness Level
V&V	Verification & Validation
WBS	Work Break Down Structure
WP	Workpackage
XMI	XML Metadata Interchange

XML	eXtensible Markup Language
XSAP	Symbolic model checking tool for safety assessment of synchronous finite-state and infinite-state systems

References

[1] OPENCOSS project, 2015. http://www.opencoss-project.eu

[2] SafeCer Project, 2015. (Certification of Software-intensive Systems with Reusable Components) http://cordis.europa.eu/project/rcn/103721_en.html and http://cordis.europa.eu/project/rcn/105610_en.html

[3] PolarSys. https://www.polarsys.org

[4] AMASS Platform User Manual[3]. D2.5 AMASS User guidance and methodological framework, November 2018.

[5] AMASS Platform Developers Guide[4]. D2.5 AMASS User guidance and methodological framework, November 2018.

[6] AMASS D2.4 AMASS reference architecture (c), June 2018.

[7] AMASS D2.1 Business cases and high-level requirements, February 2017.

[8] AMASS D3.6 - Prototype for architecture-driven assurance (c), August 2018.

[9] AMASS D4.6 - Prototype for multiconcern assurance (c), August 2018.

[10] AMASS D5.6 - Prototype for seamless interoperability (c), September 2018.

[11] AMASS D6.6 - Prototype for cross/intra-domain reuse (c), October 2018.

[12] CHESS project http://www.chess-project.org/

[13] AMASS D3.8 Methodological guide for architecture-driven assurance (b), October 2018.

[14] AMASS D4.8 Methodological guide for cross/intra-domain reuse (b), October 2018.

[15] AMASS D5.8 Methodological guide for seamless interoperability (b), October 2018.

[16] AMASS D6.8 Methodological guide for cross/intra-domain reuse (b), November 2018.

[17] AMASS D4.3 Design of the AMASS tools and methods for multiconcern assurance (b), April 2018.

[18] OMG - Semantics of Business Vocabulary and Rules™ (SBVR™) version 1.3, 2015 http://www.omg.org/spec/SBVR/1.3

[19] WEFACT 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.

[27] Eclipse Process Framework Project https://eclipse.org/epf/

[28] AMASS D2.7 Integrated AMASS platform (b), January 2018

[3] The AMASS Platform User Manual has been included as an Annex in *D2.5 AMASS User guidance and methodological framework*.

[4] The AMASS Platform Developers Guide has been included as an Annex in *D2.5 AMASS User guidance and methodological framework*.

[29] AMASS Platform bundle, 2018. https://www.polarsys.org/opencert/
[30] CONCERTO Deliverable D3.3 November 2015 Design and implementation of analysis methods for non-functional properties – Final version
[31] Medini Analyzer, https://www.ansys.com/fr-fr/products/systems/ansys-medini-analyze
[32] Safety Architect, https://www.all4tec.net/documentation-safety-architect
[33] Cyber Architect, https://www.all4tec.net/documentation-cyber-architect
[34] Sabotage, https://www.cyberssbytecnalia.com/node/271
[35] SAVONA, https://www.assystem-germany.com/en/products/savona/
[36] Verfication Studio, https://www.reusecompany.com/verification-studio