

ECSEL Research and Innovation actions (RIA)



AMASS

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

Prototype for multiconcern assurance (a) D4.4

Work Package:	WP4 Multi-Concern Assurance
Dissemination level:	PU = Public
Status:	Final
Date:	31 January 2017
Responsible partner:	Thomas Gruber (AIT)
Contact information:	Thomas.gruber@ait.ac.at
Document reference:	AMASS_D4.4_WP4_AIT_V1.0

PROPRIETARY RIGHTS STATEMENT

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

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

Contributors

Names	Organisation
Christoph Schmittner, Thomas Gruber, Zhendong Ma and Petr Böhm	AIT Austrian Institute of Technology GmbH
Alejandra Ruiz, Ángel López, Garazi Juez and Huáscar Espinoza	Tecnalia Research & Innovation
Barbara Gallina	Mälardalens Hoegskola

Reviewers

Names	Organisation
Irfan Sljivo (Peer reviewer)	Mälardalens Hoegskola
Bernhard Winkler (Peer reviewer)	Virtual Vehicle
Cristina Martínez	Tecnalia Research & Innovation

TABLE OF CONTENTS

Executive Summary.....	6
1. Introduction	7
2. Implemented Functionality	9
2.1 Scope	9
2.2 Implemented Requirements.....	10
2.2.1 Assurance case edition.....	10
2.2.2 Composition of the overall argument.....	12
2.2.3 Connection from the supporting evidences to evidence information	13
2.2.4 Edition of argument patterns	14
2.2.5 Drag and Drop argumentation patterns	14
2.2.6 Assurance case structure navigation	15
2.2.7 Provide a structured language to the text inside the claims.....	15
2.2.8 Capability to edit an assurance case in collaboration with other people from the project.....	16
2.3 Installation and User Manuals	17
2.4 Outlook and Next Steps	18
3. Implementation Description	20
3.1 Implemented Modules for the Assurance Case Specification Basic Building Block	20
3.2 Source Code Description.....	21
Abbreviations and Definitions	23
References.....	24

List of Figures

Figure 1. AMASS Building blocks	7
Figure 2. Functional decomposition for the AMASS platform	9
Figure 3. Argumentation Editor displaying an argumentation in GSN.....	11
Figure 4. Modelling elements of the Goal Structuring Notation (GSN)	11
Figure 5. Modelling elements for GSN extension for compositional argumentation	12
Figure 6. View of the preferences for the argumentation editor	12
Figure 7. Module Explorer tab highlighted in the Templates View	13
Figure 8. Example of a solution node connected with a specific piece of evidence	13
Figure 9. Modelling elements for GSN patterns	14
Figure 10. Example of software contribution safety argument pattern [19]	14
Figure 11. Templates view let the user select the library of patterns and the modules stored	15
Figure 12. Example of a claim with mark-ups	16
Figure 13. Example of content assistant using a vocabulary in the ISO 26262 context	16
Figure 14. Open Database-based Argumentation Diagram	17
Figure 15. Tool modules for Assurance Case Specification.....	20
Figure 16. Assurance Case Specification plugins.....	22

List of Tables

Table 1. Requirements implemented in the first prototype of the AMASS platform.....	10
Table 2. Functionality to be implemented in future iterations of the AMASS platform	18

Executive Summary

This deliverable, D4.4 Prototype for multi-concern assurance (a), is the output of the task T4.3 *Implementation for Multi-Concern Assurance*. Based on the results from task T2.2 *AMASS Reference Tool Architecture and Integration*, task T4.3 develops a tooling framework to develop prototype tooling for multi-concern assurance. Particular attention is paid to support the architectural approach to assurance being developed in WP3 and to support the requirements for tooling developed in WP4. This task is being carried out iteratively, in close connection with the conceptual tasks (T4.2 *Conceptual Approach for Multi-Concern Assurance* as well as those in the other WPs, namely T3.2, T5.2 and T6.2), with validation results from the implementation being used to guide further refinement of the conceptual approach. The implementation is closely guided by the requirements of the case studies, which are used to validate the prototype.

The first prototype iteration releases the basic building blocks (Prototype Core) as a consolidation/integration of previous projects OPENCOS [1] and SafeCer [2].

The developed tools in the first prototype support the following two functional areas:

- Argumentation Editor
- Argument Patterns Editor

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

Other important parts of D4.4 deliverable are:

- Installable AMASS Platform tools for the first prototype
- User Manuals and installation instruction [16]
- Source code description [15]

1. Introduction

The AMASS approach focuses on the development and consolidation of an open and holistic assurance and certification framework for CPS, which constitutes the evolution of the OPENCOS [1] and SafeCer [2] approaches towards an architecture-driven, multi-concern assurance, reuse-oriented, and seamlessly interoperable tool platform.

The expected tangible AMASS results are:

- The **AMASS Reference Tool Architecture**, which will extend the OPENCOS and SafeCer conceptual, modelling and methodological frameworks for architecture-driven and multi-concern assurance, as well as for further cross-domain and intra-domain reuse capabilities and seamless interoperability mechanisms (based on OSLC specifications [12]).
- The **AMASS Open Tool Platform**, which will correspond to a collaborative tool environment supporting CPS assurance and certification. This platform represents a concrete implementation of the AMASS Reference Tool Architecture, with a capability for evolution and adaptation, which will be released as an open technological solution by the AMASS project. AMASS openness is based on both standard OSLC APIs with external tools (e.g. engineering tools including V&V tools) and on open-source release of the AMASS building blocks.
- The **Open AMASS Community**, which will manage the project outcomes, for maintenance, evolution and industrialization. The Open Community will be supported by a governance board, and by rules, policies, and quality models. This includes support for AMASS base tools (tool infrastructure for database and access management, among others) and extension tools (enriching AMASS functionality). As Eclipse Foundation is part of the AMASS consortium, the Polarsys/Eclipse community (www.polarsys.org) is a strong candidate to host AMASS Open Tool Platform.

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

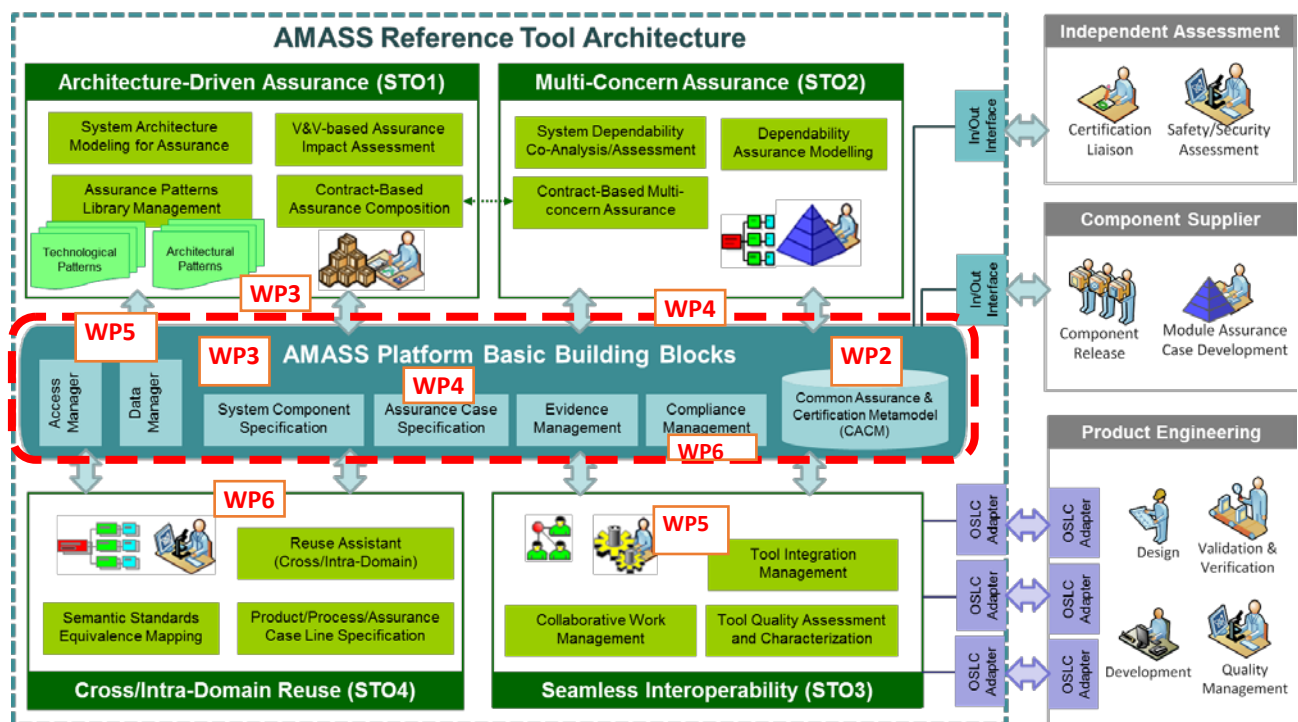


Figure 1. AMASS Building blocks

Since AMASS targets high-risk objectives, the AMASS Consortium decided to follow an incremental approach by developing rapid and early prototypes. The benefits of following a prototyping approach are:

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

AMASS has planned three prototype iterations:

1. During the **first prototyping** iteration (Prototype Core), the AMASS Platform Basic Building Blocks (see Figure 1), will be aligned, merged and consolidated at TRL4¹.
2. During the **second prototyping** iteration (Prototype P1), the AMASS-specific Building Blocks will be developed and benchmarked at TRL4; this comprises the blue basic building blocks as well as the green building blocks in Figure 1. Regarding multiconcern assurance, in this second prototype, the specific building blocks will provide functionalities regarding system dependability co-analysis/assessment, dependability assurance modelling or contract-based multiconcern assurance.
3. Finally, at the **third prototyping** iteration (Prototype P2), all AMASS building blocks will be integrated in a comprehensive toolset operating at TRL5. Functionalities specific for multi-concern assurance developed for the second prototype will improve and integrate with functionalities from other technical work packages.

Each of these iterations has the following three prototyping dimensions:

- **Conceptual/research development:** development of solutions from a conceptual perspective.
- **Tool development:** development of tools implementing conceptual solutions.
- **Case study development:** development of industrial case studies using the tool-supported solutions. The application of the building blocks in case studies for this first prototype will be described in D1.1 [18].

As part of the Prototype Core, WP4 is responsible for consolidating the previous works on single-concern assurance as well as multi-concern assurance in order to design and implement the basic building block called “**Assurance Case Specification**” (Figure 1). This part of the AMASS platform manages argumentation information in a modular fashion. It includes mechanisms to support structured arguments and argument patterns management.

This deliverable reports the **tool development** results of the “Assurance Case Specification” basic building block. It presents in detail the design of the functionality implemented in the AMASS platform tools, their software architecture, the technology used, and source code references. The design is based on the requirements for multiconcern assurance already mentioned in D4.1, focusing on the core and basic requirements.

Other important parts of D4.4 deliverable are:

- Installable AMASS Platform tools for the first prototype
- User Manuals and installation Instruction
- Source code description

¹ In the context of AMASS, the EU H2020 definition of TRL is used, see http://ec.europa.eu/research/participants/data/ref/h2020/other/wp/2016_2017/annexes/h2020-wp1617-annex-g-trl_en.pdf

2. Implemented Functionality

2.1 Scope

In this first prototype, the purpose for the WP4-related basic building block is the provision of modelling tools for creating and editing the assurance case specification. In order to support compositional and efficient assurance case specification, the block includes modular argument structures and assurance patterns as well as supplementary functions such as preliminary pattern instantiation, vocabulary support and contract definition. The major scope is highlighted with a red circle in Figure 2 showing the general functional overview of the AMASS platform (from deliverable D2.2 [14]).

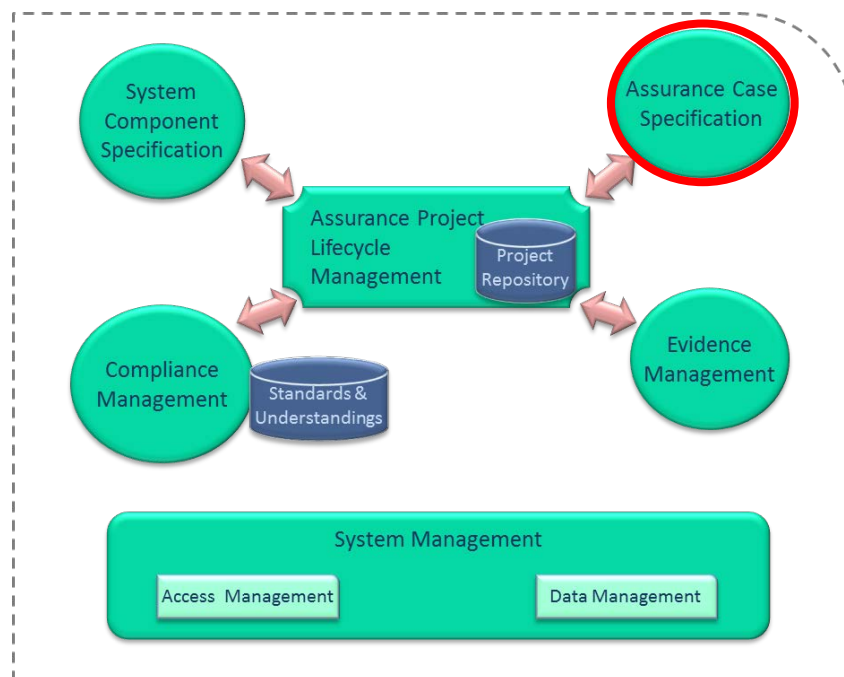


Figure 2. Functional decomposition for the AMASS platform

As stated above, the “Assurance Case Specification” part of the AMASS platform is in charge of the argumentation along the assurance project. The following functions have been realized:

1. Editing an argument structure in GSN in a scalable way.
2. Provide capabilities for creating and editing modular arguments (support for GSN extension for modular argumentation).
3. Re-using predefined argument structure parts (GSN argument modules) from a library.
4. Storing argument structure parts (GSN argument patterns) in a library for re-use.
5. Basic support for instantiating argument structure parts from a library for the individual project.
6. Basic automatic generation of process based argument structures based on the decisions made about standard compliance.
7. Filling the textual parts of the argument structure with predefined elements from a context-specific vocabulary (context refers normally to a domain-specific standard), and
8. Editing these vocabularies with a vocabulary editor.

More specifically, the user will be able to benefit from reuse by using previously approved modules for argumentation or by instantiating argumentation patterns that reflect a set of best practices. Moreover, they are supported by additional functions depending on the environment of the project and the compliance management (i.e. the industry standard or other standards the project has to deal with). The following section details both the satisfied requirements and the deployed components to show the implementation scope of the WP4-related part of the first prototype.

2.2 Implemented Requirements

From the requirements point of view this phase focuses on a set of AMASS requirements as defined in deliverable D2.1 [13]. The following requirements have been implemented in the first iteration of the AMASS platform. The column "Related requirement" refers to the list items in the deliverable D2.1.

Table 1. Requirements implemented in the first prototype of the AMASS platform

Function name	Description
Assurance case edition	The system shall be able to edit an assurance case in a scalable way.
Composition of the overall argument	The system shall provide the capability of generating a compositional assurance case argument.
Connection from the supporting evidences to evidence information	The system shall be able to navigate from an evidence supporting a claim to the information about the evidence such as the evidence characterization and the actual artefact.
Edition of argument patterns	The system shall be able to edit and store argumentation patterns for later use.
Drag and drop argumentation patterns	The system shall be able to instantiate in the actual assurance case an argument pattern (concerning safety and security) selected from the list of stored patterns.
Provide a structure language to the text inside the claims	The system shall be able to provide support for language formalization inside argument claims.
Capability to edit an assurance case in collaboration with other people from the project	The system shall be able to let different users edit an assurance case in a collaborative manner.

Each requirement together with the implementation done so far that implements the requirement is shortly outlined in the following sections.

2.2.1 Assurance case edition

Assurance Case specification is supported by a graphical argumentation editor (see Requirement 4.2 in D2.1). For the first prototype, a graphical argumentation editor has been evolved from the OPENCOS project. The evolved editor implements the SACM V1.1 [4] argumentation meta-model and the GSN [5] graphical notation. The main change with respect to previous development in OPENCOS is the use of GSN naming in the user interface while keeping the SACM model internally and, thus, fulfilling requirement "Argumentation import/export" from D2.1 implicitly. The code has also been updated to the latest eclipse version "Neon", facilitating maintenance and access to the latest improvements. As can be seen in Figure 3, the editor contains several parts, highlighted with red/gray rectangles:

- The **Project Explorer** shows the contents of the argumentation models in the context of AMASS assurance projects
- The **Diagram Editor** permits the graphical modelling of GSN goal structures.
- The **Palette** is a toolbox with the modelling elements and the connections between them to add to the diagram.
- The **Properties** allows editing the properties of the selected element in the model.

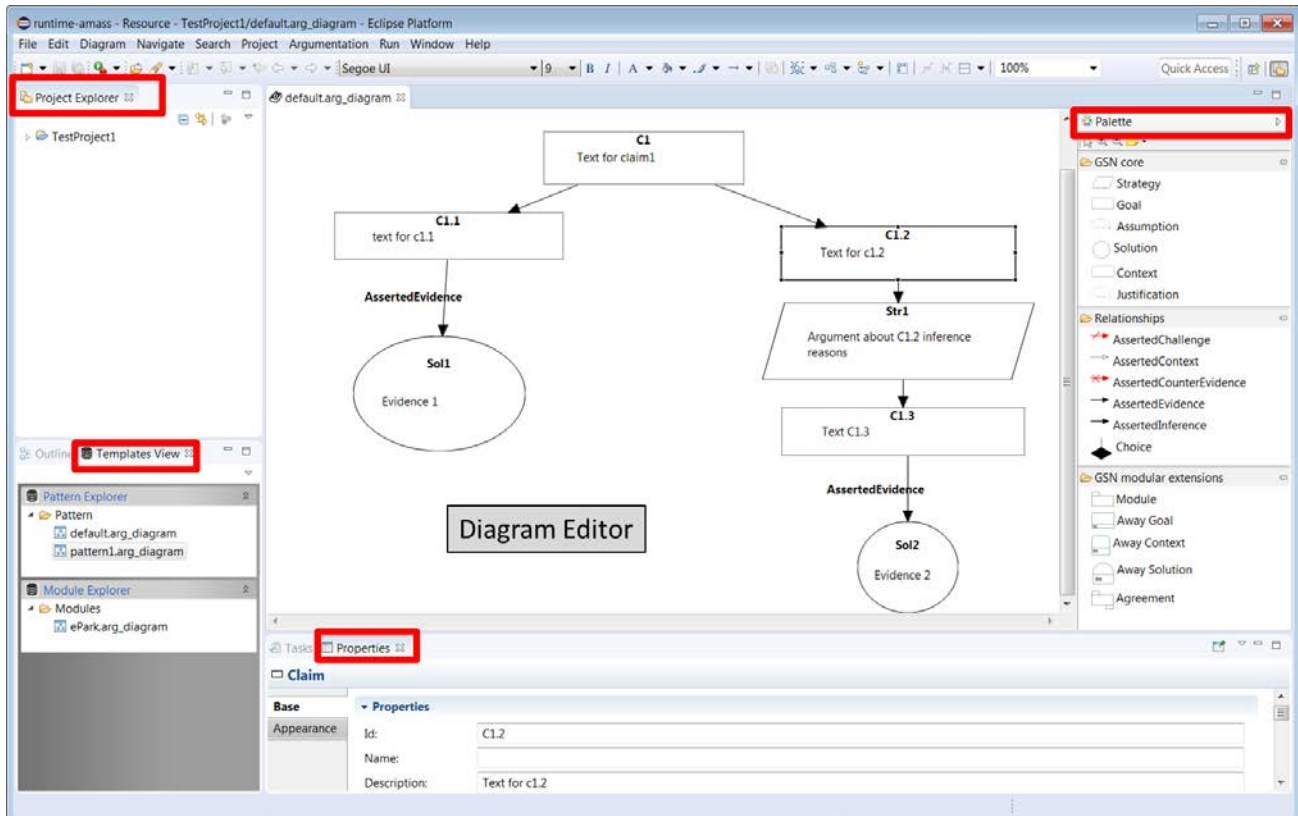


Figure 3. Argumentation Editor displaying an argumentation in GSN

Assurance cases and assurance case patterns are stored as SACM elements model together with their associated GSN graphical diagrams. The elements of GSN are shown in Figure 4 below.

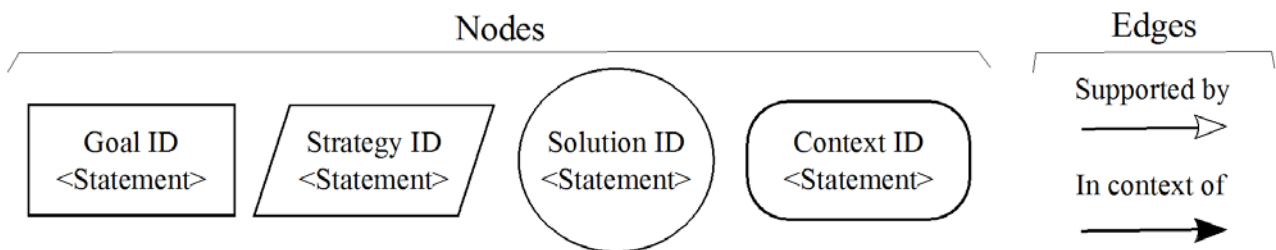


Figure 4. Modelling elements of the Goal Structuring Notation (GSN)

Assurance cases can be stored either in individual model files in the Eclipse workspace or in a shared database using CDO technology.

2.2.2 Composition of the overall argument

Arguments can be defined for a specific system/component and environment, but also these systems can be reused. In a similar way, arguments can be encapsulated can be specified in a series of modularised interconnected arguments. These capsules can be seen as ready to be used arguments, these are focused on a specific topic and do not need to be instantiated for a specific environment just referenced when they are reused. GSN describes an appendix including graphical notation associated with these concepts and SACM v2.0 will also include in more detail these topics.

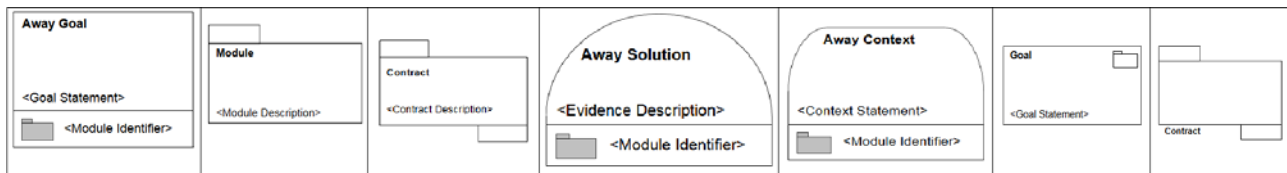


Figure 5. Modelling elements for GSN extension for compositional argumentation

The argumentation editor defines a class called **argumentation module** which lets users store arguments in an encapsulated and modular way. Previously, specified argumentation which is ready for reuse can be instantiated as an argumentation module on the actual model and, by doing so, all the encapsulated argumentation will be reused in the new project. This fulfills requirement “Composition of the overall argument” in D2.1.

When creating the arguments which shall be encapsulated in an argumentation module, the user shall store the associated assurance case in files and in a directory predefined in the preferences (see Figure 6).

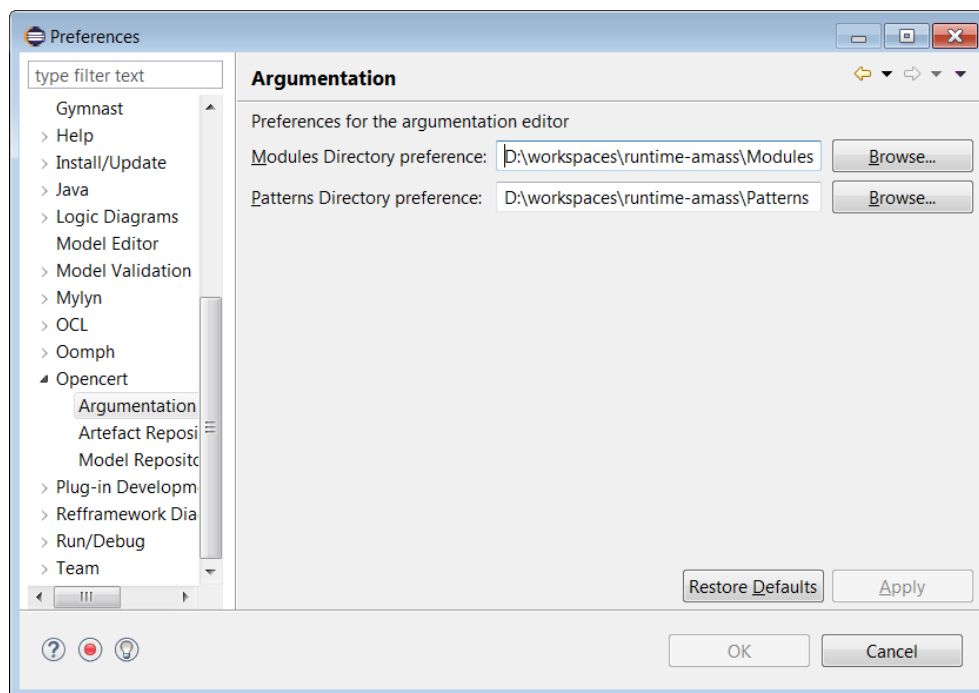


Figure 6. View of the preferences for the argumentation editor

All argument modules are stored in a directory and the user can have access to the contents and reference the arguments in the actual assurance case from the “Module Explorer” tab included in the “Templates view” (see Figure 7).

These modules do not need the phase of instantiation because the included information is ready to be used and no adaptation or modification is required.

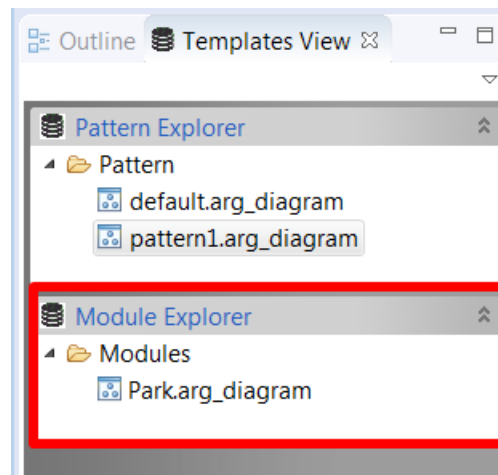


Figure 7. Module Explorer tab highlighted in the Templates View

The composition of the assurance case from modules supports collaborative assurance case edition as requested by requirement “Capability to edit an assurance case in collaboration with other people from the project”.

2.2.3 Connection from the supporting evidences to evidence information

When editing an assurance case, the assertions made in the arguments shall be supported by evidences. In the specification, this is done by using the *Solution node* and connecting it with the goal that supports. The user should provide more information in the properties of the Solution and is able to link it with an artefact which is already specified in the evidences model. The evidences editor is part of the basic building block described in D5.4 Prototype for seamless interoperability (a). This functionally links both editors and explicitly shows the connection of the argumentation metamodel and the evidences metamodel described in D2.2.

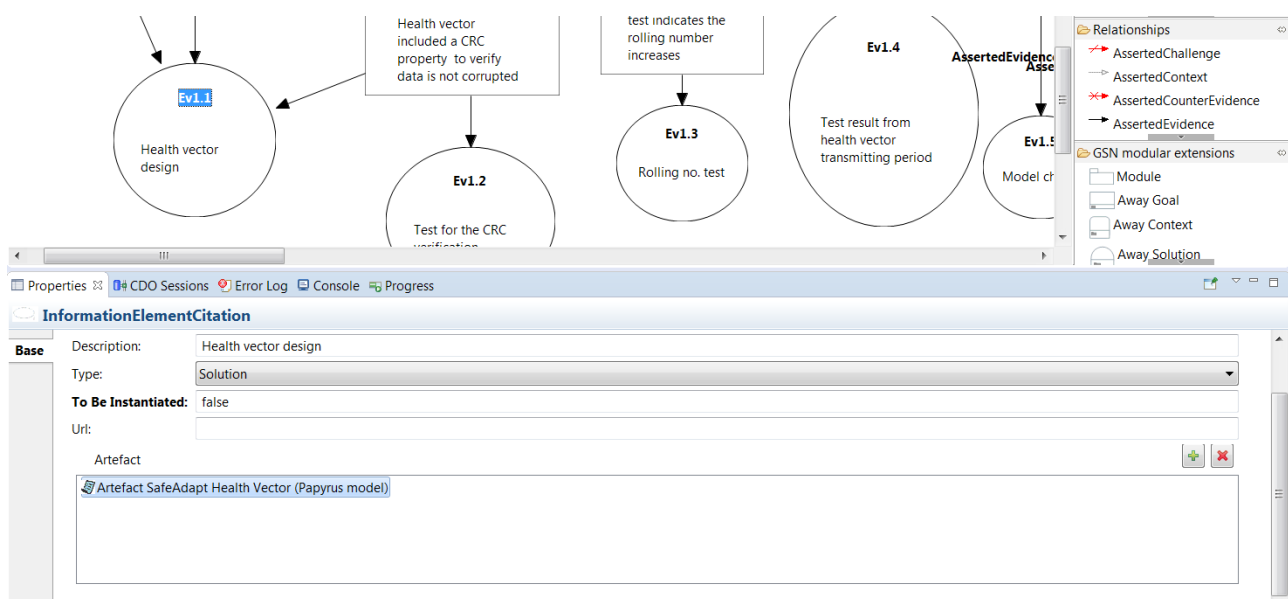


Figure 8. Example of a solution node connected with a specific piece of evidence

2.2.4 Edition of argument patterns

According to the GSN standard, argument patterns are generic arguments that can be useful for reusable reasoning, akin to software development patterns. GSN creates an extension for these abstractions, and so the assurance editor also supports this extension.

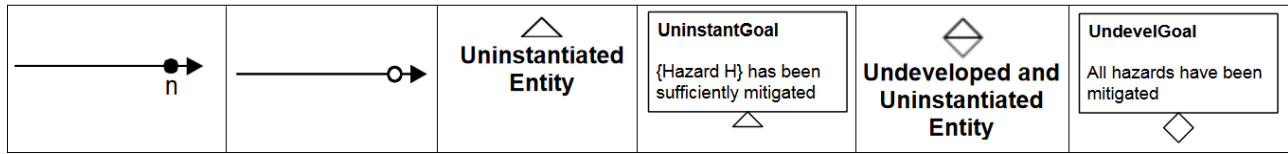


Figure 9. Modelling elements for GSN patterns

The editor supports the creation and storage of argumentation patterns. In the tool, an argument pattern is stored in a special directory that is defined in the preferences (see Figure 6).

2.2.5 Drag and Drop argumentation patterns

When users are specifying a new assurance case they can take advantage of the use of patterns. Argument patterns can be checked while using the “Pattern Explorer” (see non-highlighted tab in Figure 7 which let users select stored argument patterns and, just by using the drag and drop function, patterns are copied in the target argumentation (see Requirement “Drag and drop argumentation patterns” in D2.1). Figure 10 shows an example for an argument pattern during its creation.

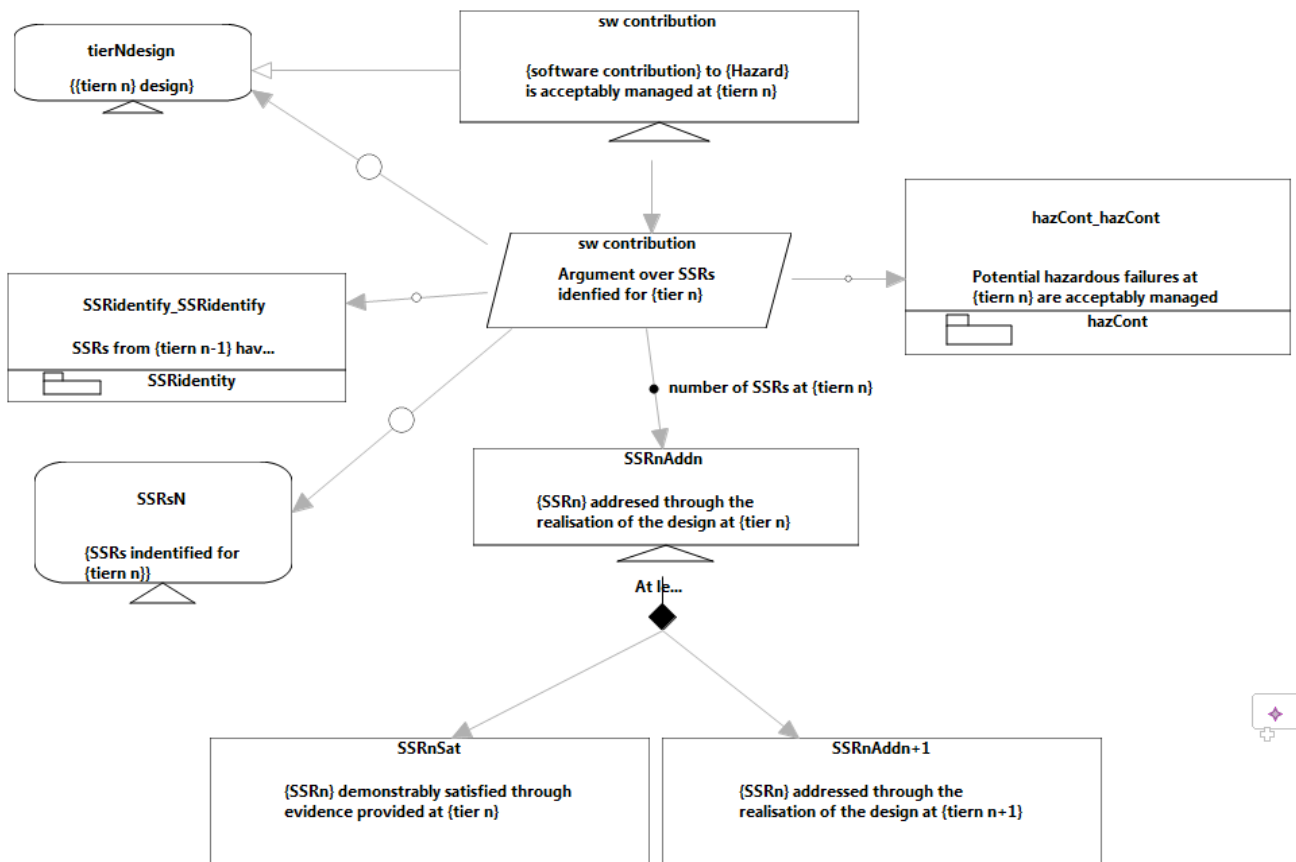


Figure 10. Example of software contribution safety argument pattern [19]

The user should instantiate and adapt the argument pattern to the actual assurance case by updating the information contained in brackets and making decisions in the edges when there is more than one option specified.

2.2.6 Assurance case structure navigation

Patterns and Modules are displayed in the Pattern Explorer and the Module Explorer, which are contained in the Templates View, as can be seen in Figure 11. Modules allow Assurance case navigation as requested by requirement "Assurance case structure navigation" in D2.1 and with the argument Modules, the requirements is partially covered.

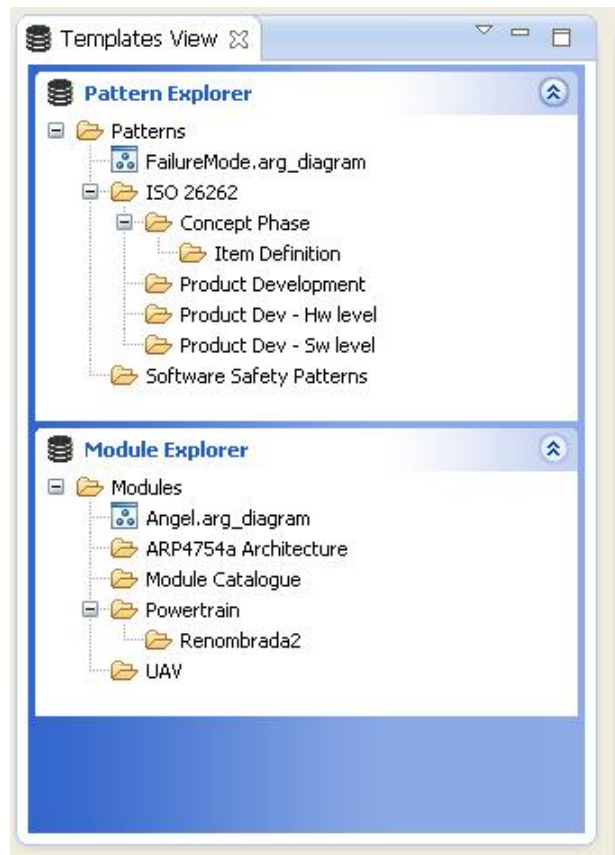


Figure 11. Templates view let the user select the library of patterns and the modules stored

2.2.7 Provide a structured language to the text inside the claims

2.2.7.1 Markups

In the OPENCROSS project, there has been work in order to define functional safety vocabularies. The same concept is reproducible to include other concerns, such as security, maintainability, performance or other quality attributes like, for instance, those comprised by dependability; this implements requirement "Provide a structure language to the text inside the claims" in D2.1. Based on the initial support for vocabularies in OPENCROSS and on means to support a more structured argumentation, a set of additional supporting features has been implemented atop the graphical argumentation editor. The language used to express claims has been enriched by supporting dedicated markups. For example to express references to other elements in the argumentation structure by means of *id:* markup or to express typed pattern variables by using *var:* markups. References to external files or hyperlinks are supported as well. Markups can be used while editing an argument text and they are rendered (similar to hypertext markups and in line with the GSN standard) in case the argument element is shown on a GSN diagram. Using the provided set of markups a first step towards semi-formal or even formal argumentation is supported.

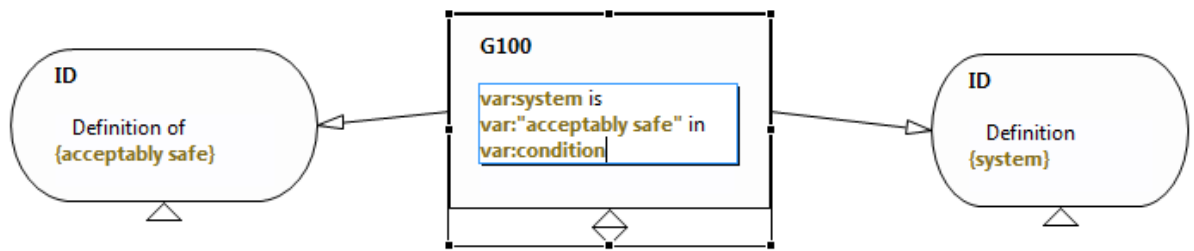


Figure 12. Example of a claim with mark-ups

2.2.7.2 Content Assistant

Beside the markups, the user is provided with a content assistant that – depending on the context, on the already entered text and on the vocabularies that are available in the current project – proposes a set of texts or markups that consistently fit the current context. For that purpose a simple vocabulary editor is available to create and organize vocabulary terms. Available vocabularies (vocabulary files in the project, potentially the SBVR [3]) are dynamically loaded and the terms are used by the content assistant (see Figure 13). Content assistance is also available for all markups, e.g. for the *id:* markup the IDs of referable elements are provided.

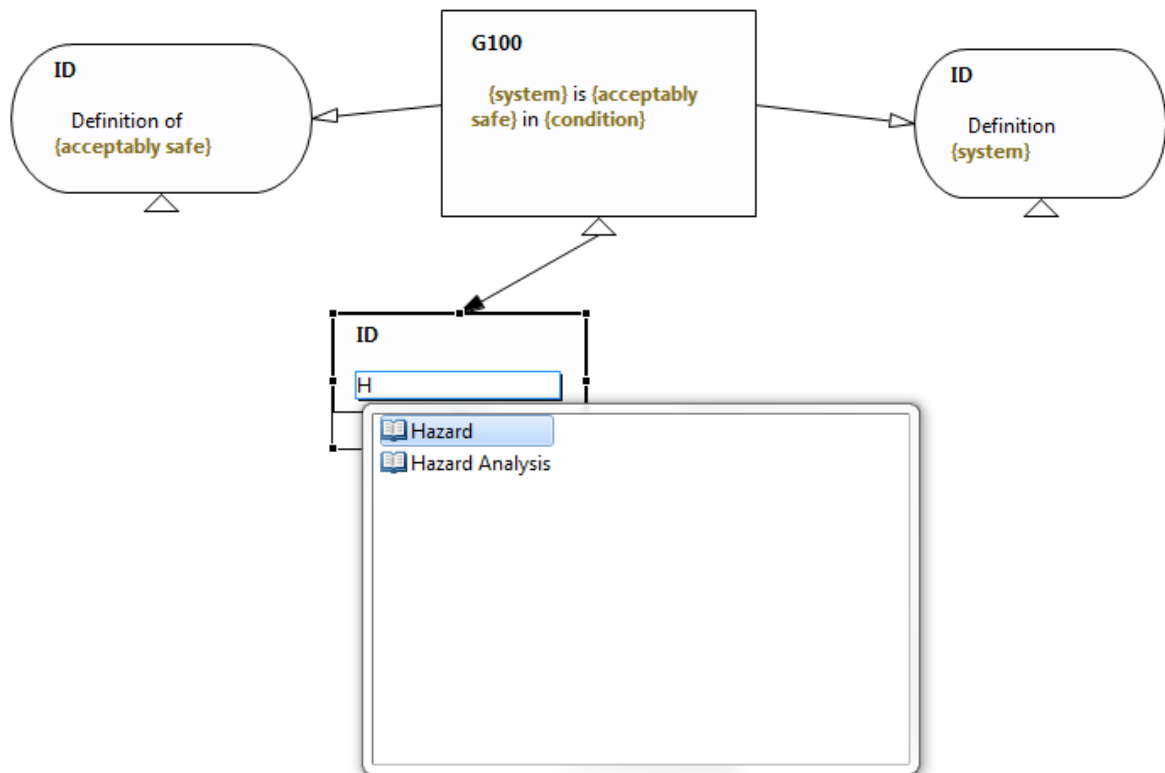


Figure 13. Example of content assistant using a vocabulary in the ISO 26262 context

2.2.8 Capability to edit an assurance case in collaboration with other people from the project

The assurance case specification edition takes advantage of the CDO technology. This technology supports real-time collaboration on models which are stored in a database. The graphical editors using CDO are generated similar to the ones that used file-based technology, and can also be customized.

CDO technology includes a security model that allows to specify the access rights to the objects of repositories, which is especially important for the access management and data management basic building blocks. These blocks are not part of the scope of this document.

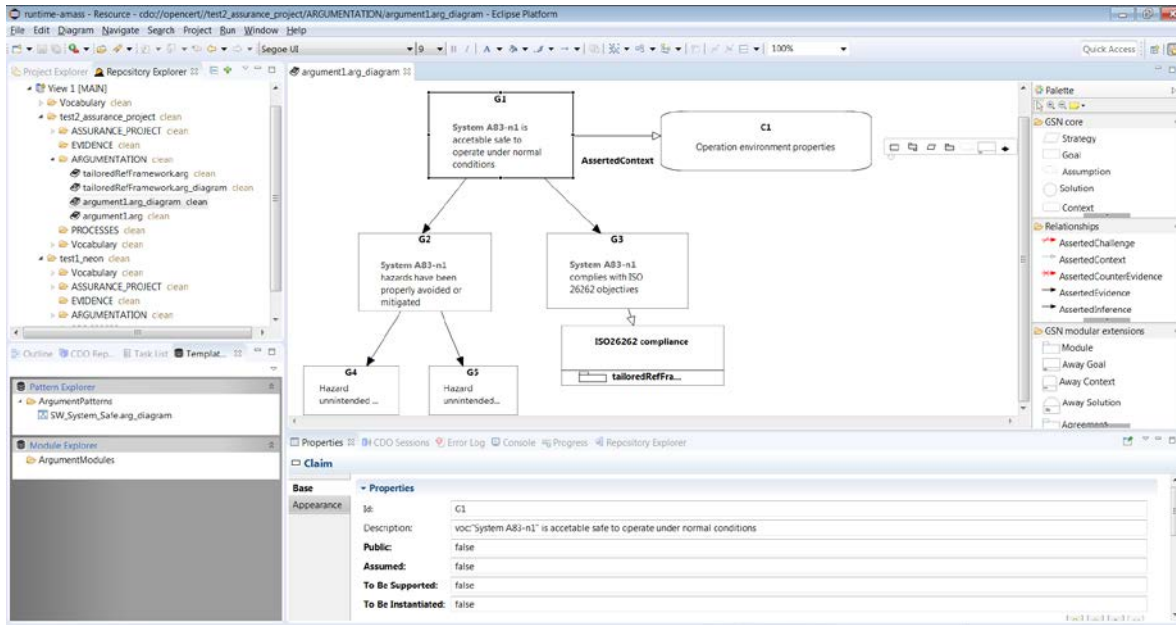


Figure 14. Open Database-based Argumentation Diagram

2.3 Installation and User Manuals

The steps necessary to install the first prototype are exhaustively described in the AMASS User Manual (currently under elaboration for all the AMASS building blocks) [16] and will not be repeated here. That document contains all required steps and document references to set up the tools. A pre-packaged distribution will be supplied in the second iteration of the AMASS platform.

In summary, that document is a user manual of the first AMASS tool prototype implementation. The users can find there the installation instructions, the tool environment description, and the functionalities for not just the assurance case specification but also for the other basic building blocks.

2.4 Outlook and Next Steps

For the second iteration of the AMASS platform an extension of the prototype for multi-concern assurance with workflow capabilities is planned by integrating the tool WEFACT [17]. This Eclipse RCP application, which is currently being developed based on a prior, DOORS requirements based edition, allows workflow execution, supports continuous impact management in the event of changing requirements, models or implementations and triggers the necessary re-assurance activities. A first presentation of the functionality is envisaged for the first Annual review; then the WEFACT core modules shall be integrated with the ARTA.

According to the requirements defined in D2.1 the following functionality shall be implemented in future iterations.

Table 2. Functionality to be implemented in future iterations of the AMASS platform

Function name	Description
Argumentation architecture	The system shall be able to edit an argument architecture associated with a system and/or component.
Semi-automatic generation of product arguments	The system shall reduce efforts of manual creation of product-based assurance case arguments. This could be done by enabling semi-automatic generation of product-based arguments-fragments.
Assurance case status report	The system shall provide the capability for querying the assurance case in order to detect: 1) undeveloped goals; 2) fallacies.
Assurance case structure navigation	The system shall let the user browse the assurance case structure. Note: in case GSN-like modelling elements are used, this requirement may be translated as follows: The system shall let the user navigate from top-level assurance case overview to the nested assurance case fragments that are encapsulated within modules.
Provide guidelines for argumentation patterns	The system shall be able to provide guidelines to use and instantiate argument patterns (concerning safety and security) presented in the actual assurance case.
Compliance map generation from argument evidences	The system shall be able to detect when a claim about a requirement from a standard (compliance claim) is supported by an evidence and generate the compliance indicator in a transparent way.
Formal validation of assumptions and context when arguments modules are connected	The system shall be able to indicate the validation of assumptions contained in argument modules every time the modules are connected and/or modified.
Provide quantitative confidence metrics about an assurance case in a report	The system could produce a status report indicating a quantitative confidence metric for the assurance case.

Provide guidelines for argumentation	The system shall be able to provide guidelines about the assurance case edition based on the system/component development phase status.
The AMASS tools must support specification of variability at the argumentation level	<p>The system shall provide the capability for modelling a multi-concern and multi-context assurance case.</p> <p>Note: variability modelling could be a solution. If GSN-like modelling elements are considered, the diamond for representing alternatives as well as the octagon for extrinsic variability could be considered.</p>
Argumentation import/export	The system shall be able to import/export argumentations to SACM.

3. Implementation Description

3.1 Implemented Modules for the Assurance Case Specification Basic Building Block

The Assurance Case Specification building block is an Eclipse-Based Argumentation Editor based on the SACM data structure standard. It contains plugins for editing argumentation models and plugins for management of argument patterns and module libraries. (Please note that the term "module" used for argumentation modules differs from the "implemented modules" described in this chapter.)

The purpose of the first tool module is to provide services for modular argumentation. The second tool module offers services to store and instantiate argument patterns.

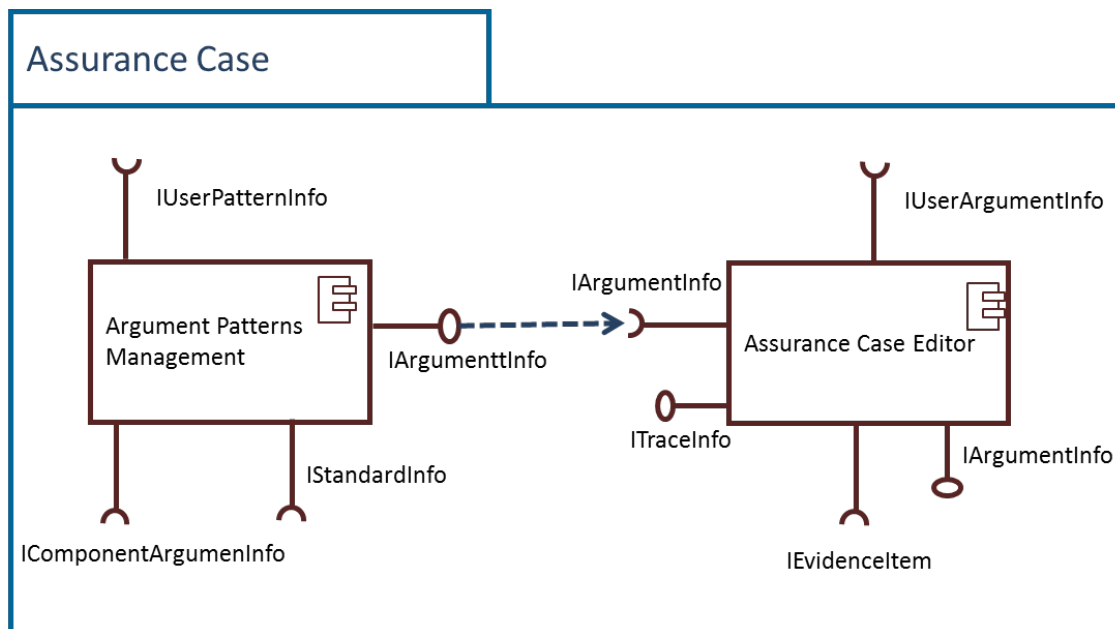


Figure 15. Tool modules for Assurance Case Specification

In this first iteration of the AMASS platform, main work has been done to consolidate previous developments from the OPENCROSS project, mainly in the Assurance Case Editor module. The developments on the Argument Patterns Management module have focused on basic functionality and we have plans to include enhancements in future iterations.

The technologies used to develop the Assurance Case Specification building block are:

- To generate Editors: GMF [6], EMF [7], Eugenia [10]
- For model transformations: Epsilon (ETL) [8]
- For storage: CDO [11] (Patterns & Modules are stored in Files)
- For vocabulary: Xtext [9]

3.2 Source Code Description

The source code of the first AMASS prototype can be found in the source code SVN repository [15]. The code for the assurance case modules first prototype will be stored together with the other basic building blocks in the repository under “tag” to distinguish the state of the code at the time of the integrated release.

Once all the plugins are installed, these are the necessary ones for the Assurance Case Specification:

- **GSN.figures**
This plugin provides utilities to draw model elements according to the Goal Structuring Notation (GSN) standard.
- **org.opencert.sam.arg**
In this plugin, the argumentation metamodel is defined and stored, and the Java implementation classes for this model are generated.
- **org.opencert.sam.arg.diagram**
This plugin is the diagram editor itself. It manages diagrams and includes a canvas to draw on, a palette with creation tools and default selecting and zooming capabilities, a property view and an outline view.
- **org.opencert.sam.arg.edit**
The edit plugin includes adapters that provide a structured view and perform command-based edition of the model objects.
- **org.opencert.sam.arg.editor**
This plugin provides the user interface to view instances of the model using several common viewers and to add, remove, cut, copy and paste model objects, or to modify the objects in a standard property sheet.
- **org.opencert.sam.arg.ui**
This is an additional plugin. It offers several utilities such as drawing model elements not included in the GSN standard, accessing to patterns and modules files.
- **org.opencert.sam.arg.preferences**
This plugin manages the default preferences required by the Argumentation diagram editor. The parameters which can be defined are the Modules Directory (with all argumentation modules stored from previous argumentation phases) and the Patterns Directory (that contains all argumentation patterns templates).
- **org.opencert.sam.vocabulary**
Contains the vocabulary meta model, which is part of the previous results from OPENCOS CCL (Common Certification Language).
- **org.opencert.sam.vocabulary.edit**
The edit plugin includes adapters that provide a structured view and perform command-based edition of the model objects. It contains the CCL vocabulary meta model respective the related EMF based tree editor and GMF based graphical editor to create and edit vocabulary models.
- **org.opencert.sam.vocabulary.editor**
This plugin provides the user graphical interface to view instances of the model using an EMF based tree editor and GMF based graphical editor to create and edit vocabulary models.

In addition, the following plugins are necessary to manage assurance project and to handle the corresponding evidences:

- **org.opencert.apm.assuranceassets**

In this plugin, the assurance assets metamodel is defined and stored, and the Java implementation classes for this model are generated.

- `org.opencert.apm.assuranceassets.edit`
The edit plugin includes adapters that provide a structured view and perform command-based edition of the assurance assets model objects.
- `org.opencert.evm.evidspes`
In this plugin, the evidence metamodel is defined and stored, and the Java implementation classes for this model are generated.
- `org.opencert.evm.evidspec.edit`
The edit plugin includes adapters that provide a structured view and perform command-based edition of the model objects.
- `org.opencert.infra.properties`
This plugin contains the definition of the Property metamodel, and the Java implementation classes for this model.
- `org.opencert.infra.properties.edit`
In relation with the edit plugin for evidence, this plugin contains a provider to display the model in a user interface.

Figure 16 lists all the plugins described above.














 `org.eclipse.opencert.sam.arg`
 `org.eclipse.opencert.sam.arg.diagram`
 `org.eclipse.opencert.sam.arg.diagram.dawn`
 `org.eclipse.opencert.sam.arg.edit`
 `org.eclipse.opencert.sam.arg.editor`
 `org.eclipse.opencert.sam.arg.editor.dawn`
 `org.eclipse.opencert.sam.arg.ui`
 `org.eclipse.opencert.sam.preferences`
 `org.eclipse.opencert.vocabulary`
 `org.eclipse.opencert.vocabulary.edit`
 `org.eclipse.opencert.vocabulary.editor`
 `org.eclipse.opencert.vocabulary.generatefrommodel`
 `org.eclipse.papyrus.diagram.common`

Figure 16. Assurance Case Specification plugins

Abbreviations and Definitions

Definitions are common to the whole AMASS project and are given in the AMASS glossary (deliverable D2.2 [14]). Only the definition of RCP is missing there:

<i>Term</i>	<i>Explanation</i>
RCP	Rich Client Platform - an Eclipse add-on framework allowing the development of Eclipse applications

Following abbreviations are used in this document:

<i>Abbreviation</i>	<i>Explanation</i>
API	Application Programming Interface
ARTA	AMASS Reference Tool Architecture
CACM	Common Assurance and Certification Metamodel
CCL	Common Certification Language
CDO	Connected Data Object
CPS	Cyber-Physical Systems
DOORS	Dynamic Object-Oriented Requirements System
ECSEL	Electronic Components and Systems for European Leadership
EMF	Eclipse Modelling Framework
ETL	Epsilon Transformation Language
GMF	Graphical Modeling Framework
GSN	Goal Structured Notation
ISO	International Organization for Standardization
OMG	Object Management Group
OSLC	Open Services for Lifecycle Collaboration
RCP	Rich Client Platform
SACM	Structured Assurance Case Metamodel
SBVR	Semantics of Business Vocabulary and Rules
SVN	Subversion
TRL	Technology Readiness Level
V&V	Verification and Validation
WEFACT	Workflow Engine for Analysis, Certification and Test
WP	Work Package

References

- [1] The OPENCROSS project <http://www.opencross-project.eu/>
- [2] The SafeCer project <http://www.safecer.eu/>
- [3] OMG - Semantics of Business Vocabulary and Rules™ (SBVR™) version 1.3, 2015 <http://www.omg.org/spec/SBVR/1.3>
- [4] OMG - SACM - Object Management Group version 1.1, 2015 <http://www.omg.org/spec/SACM/1.1>
- [5] Origin Consulting GSN Community Standard Version 1 (2011)
- [6] Graphical Modeling Project (GMP) <http://www.eclipse.org/modeling/gmp/>
- [7] Eclipse Modeling Framework (EMF) <https://www.eclipse.org/modeling/emf/>
- [8] Epsilon Transformation Language <http://www.eclipse.org/epsilon/doc/etl/>
- [9] Xtext <http://www.eclipse.org/Xtext/>
- [10] Eugenia <http://www.eclipse.org/epsilon/doc/eugenia/>
- [11] CDO <http://www.eclipse.org/cdo/>
- [12] OSLC <http://open-services.net/specifications/>
- [13] AMASS D2.1 Business cases and high-level requirements Version 0.3 (2016/17)
- [14] AMASS D2.2 AMASS Reference Architecture (a) Version 1.0 (30 November 2016)
- [15] AMASS source code https://services.medini.eu/svn/AMASS_source/²
- [16] AMASS Platform – Prototype Core User Manual https://services.medini.eu/svn/AMASS_collab/WP-transversal/ImplementationTeam/PrototypeCore/AMASS_Prototype1_UserManual.docx Version 0.1 (2017)³
- [17] WEFACT <http://www.ait.ac.at/en/research-fields/verification-validation/methods-and-tools/wefact/>
- [18] AMASS [D1.1 Case studies description and business impact](#) Version 1.0 (30 November 2016)
- [19] Richard Hawkins, Software Contribution Safety Argument Pattern (2009) <http://www.goalstructuringnotation.info/archives/234>

² The AMASS SVN code repository is open to AMASS partners with the same credentials as the SVN document repository. In case that people outside the project need access, please contact the AMASS Project Manager (huascar.espinoza@tecnalia.com)

³ The current User Manual is a draft document; the final version of the manual will be integrated in D2.5 AMASS User guidance and methodological framework (m31).