EC6.13

IVVQ strategy Assurance Cases for ML-based systems

Requirements Specification to support Assurance Cases development in CAPELLA

contact@confiance-ai.fr | www.confiance.ai

Specific information to be provided in ANR reports only

X	Progress report		Final report
From:	AAAA/MM/JJ	To:	AAAA/MM/JJ
Brief description of the project (reminder)			
Contractual description of the project:			~~nom du fichier~~
Reference budget:			~~nom du fichier~~

RECOMMANDATIONS

Strikethrough text is used to illustrate and explain the headings. It should be removed and replaced with non-strikethrough text. Similarly, this recommendations block should be removed, along with the statement "Specific information to be provided in ANR reports only", prior to dissemination of the document.

Document reference: XXX

Contributors

	Name	Organisation	Role
Responsible for the deliverable	E. JENN	IRT Saint Exupéry	Action sheet leader
Scientific responsible	E. JENN	IRT Saint Exupéry	Action sheet leader
Co-authors	Eric JENN Vincent MUSSOT Ramon CONEJO-LAGUNA Florent CHENEVIER Bernard BOTELLA Morayo ADEJOUMA Anthony FERNANDES-PIRES	IRT Saint Exupery IRT Saint Exupery IRT Saint Exupery THALES AVS CEA CEA ONERA	Research engineers

Document control

Revision	Date	Commentary	Author
1.0a	2023/04/12	Creation	E. JENN
1.0	2023/04/14	Version for delivery	All co-authors

A. Introduction

This document collects the requirements for the integration of Assurance Cases in the Confiance.ai engineering modelling environment based on Capella. This specification is “inspired” by the ARTEFACTS tool developed in Confiance.ai batch #2.

In order to focus the document on the main requirements, features provided “by default” by the Capella environment (e.g., save / restore) are not reproduced here.

Nota: In the rest of this document, the implementation of requirements is referred to as “the tool”.

The role of the tool is depicted on Figure 1.

The tool provides the capability to:

- Select an engineering item I from the workflow model
- Select a property P about engineering item I
- Build the argumentation model defining the various *strategies* to demonstrate that P holds (the *goal* or *claim)*. The argumentation is a tree in which the initial goal is decomposed iteratively into sub-goals. The decomposition (or refinement) is carried out iteratively down to the point where sub-goals can be directly associated with *solutions*.
- A *solution* refers to some engineering item that demonstrates that the goal has been reached. A solution may use some of the technological bricks (e.g., a method or a tool) produced by the project.
- This engineering item is produced by some activity of the workflow. In some cases, this activity already exists in the workflow; in other cases, new activity needs to be created to provide the evidences supporting the argumentation.

In addition, the tool aims at providing metrics to support the selection of an argumentation strategy. Metrics can refer to cost, maturity, and uncertainty.

Enventually, the tool will be integrated in the overall “Confidence environment” developed in the project, in particular with the “companion” tool.

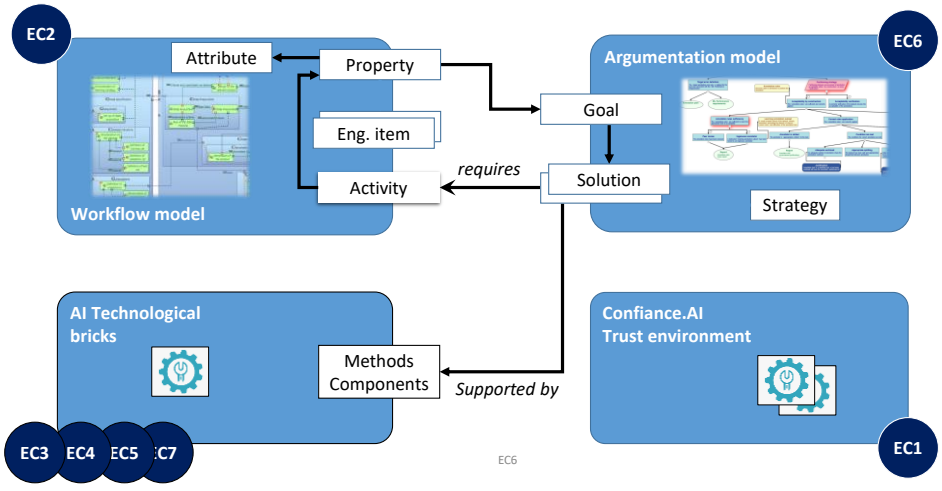


Figure 1. Relationships between workflow, the argumentation, the technological bricks developed in Confiance.ai and the overall environment.

B. Specification

B.1 GSN concepts

This section collects requirements concerning the GSN's concepts captured in the tool.

REQ-00001: Capture of GSN V3 concepts

The tool shall capture all concepts of GSN standard V3 defined in [1], sections 1.2 to 1.5. Concepts with priority level 1 shall be addressed first.

Concept	Prio	Comments[1]
Core element		
Goal	1	Presents a claim forming part of the argument.
Strategy	1	Describes the inference that exists between a goal and its supporting goal(s).
Solution	1	Presents a reference to an evidence item.
Context	1	Presents a contextual artefact.
Assumption	1	Presents an intentionally unsubstantiated statement.
Justification	1	Presents a statement of rationale.
Undeveloped element	1	Indicates that a line of argument has not been developed.
SupportedBy (rel)	1	Supports relationships between elements to be documented. (also extended to support the modular extensions)
InContextOf (rel)	1	Declares a contextual relationship. (also extended to support the modular extensions)
Pattern extension		
multiplicity	1	*Expresses a generalised n-ary relationships between GSN elements* *(Associated with the SupportedBy relation)*
optionality	1	*Expresses optional and alternative relationships between GSN elements (Associated with the SupportedBy relation)*
choice	1	*Denotes possible alternative I, satisfying a relationship* *(Associated with the SupportedBy relation)* *(nb) Choices will capture User Strategies*
Uninstanciated element	2	*Denotes that the attached element remains to be instantiated.* *This also includes the instantiation table.*
Undeveloped and uninstanciated element	2	Denotes that the attached element requires both further development and instantiation.
Instanciation data reference	2	Indicates that the GSN argument below the attached element is to be instantiated as a template argument.
Modular extension		
Away goal	1	Repeats a claim presented in another argument module
Away solution	1	Repeats a reference to evidence items presented in another argument module.
Away context	1	Repeats a reference to context presented in another argument module.
Away assumption	1	Repeats an assumption presented in another argument module and is typically used only in Contract Modules.
Module reference	2	Presents a reference to a module containing an argument.
Contract reference	2	Presents a reference to a contract module
Public decorator	2	Indicates that the element is publicly visible in one or more interfaces of the module and can be referenced as an away element.
To be supported by contract	2	Denotes that support for the claim presented by the attached goal is intended to be provided from an argument in another module linked by an as-yet-undisclosed contract.
Module	2	*To be discussed. Such structuring elements may be replaced by navigation features*

Contract	2	Represents a special type of module that defines the relationship between argument module interfaces and shows how one module supports the argument in another.
Module interface connector	2	Optionally be added to aid clarity of the specific interface (specified by the {interface identifier}) used by the inter-module relationship
ModuleSupportedBy (rel)	2	Represents one or more support relationship(s) between the elements within the modules
ModuleInContextOf (rel)	2	Represents one or more context relationship(s) between the elements within the modules
Composite (rel)	2	Used where both a supported and a context relationship exists between modules.
Confidence argument extension		
ACP decorator	2	
Dialectic extension		
Defeated element	2	Indicates that the element is defeated
Challenges (rel)	2	Allows a Challenge to any GSN entity to be documented.
Defeated relationship	2	Indicates that the relationship is defeated

■

Note 1: Concepts shall be covered from §1.2 onwards (most critical is §1.2).
Note 2: a metamodel for version 2 of the GSN standard, defined as an extension of the SACM metamodel [2] is given in GSN_metamodelV2-2-1210.pdf (scsc.uk). Differences between GSN V2 and GSN V3 are listed in GSN Standard (Version 2 to Version 3 main changes (scsc.uk).

REQ-00002: Graphical representation

The tool shall provide a graphical representation of the GSN concepts. The graphical representation shall comply with the GSN 3.0 standard.

■

B.2 Other concepts

This section collects requirements about additional concepts besides those defined in GSN.

B.2.1 Glossaries

In GSN, definitions are introduced using Contexts. In order to limit the cluttering of the diagram[1], we propose to add the concept of Glossary. An element enclosed in square brackets in the text of any node (e.g., a Goal) is considered to be defined in a glossary (in a "glossary entry"). Since the same term may have multiple definitions, multiple glossaries shall be supported with appropriate scoping rules.

REQ-00003: Glossary

The tool shall provide the capability to define terms in a **glossary**. A Glossary associates a set of terms (or an expression) with their definitions. A glossary may be attached to one or multiple Context nodes or to the project itself.

■

[1] Even though a filtering feature is also available.

REQ-00004: Glossary – Adding entries

The tool shall support the definition of a new glossary entry in two different ways:

- The User enters a new term in the glossary by using the square brackets notation (e.g., "`[this term] is defined in a glossary…`" in a node description).

  In that case, the tool shall present a list of glossaries in which the term is already defined. This list shall be organized according to the increasing distance from the current node to the context node to which the glossary is associated. If the term is not yet defined, the tool shall propose to create a new glossary in any of the existing glossaries (the farthest being presented first).
- The User create a new glossary entry in some existing glossary.

■

REQ-00005: Glossary – Removing entries

The tool shall support the deletion of a specific glossary entry or of a complete glossary.

■

REQ-00006: Glossary – Verification

The tool shall provide the capability to check that all terms of the model enclosed in square brackets are actually defined in a glossary. The tool shall also verify that a term only appears once in glossaries located at the same level(*). Verification shall be done starting from a Goal and transitively through all child nodes following the GSN relations down to the leaves of the argumentation tree.

The error (absence of definition of multiple definitions) shall be signalled to the user (e.g., by writing the term in red, e.g., [this is not defined]).

Verification shall be done automatically when the model is saved or after any modification of a glossary. An option shall be provided to enable / disable this function and select the checking mode.

(*) Two glossaries are "at the same level" if they are attached to sibling nodes.

■

REQ-00007: Glossary – Navigation

The tool shall provide the capability to navigate to the glossary entry of a term between square brackets. The applicable entry is the one defined in the glossary that is the "upstream closest" to the node in which the term appears considering the graph of nodes associated with glossaries.

■

REQ-00008: Glossary – InfoTips

The tool shall provide the capability to display the definition of a glossary term using an "infotip" when hovering over it.

■

B.2.2 Review support

This section collects requirements concerning the review / checking of the Assurance Cases.

REQ-00009: Reviewing status

The tool shall provide the capability to associate a review status (none, ready for review, being reviewed, reviewed) to any node of the Assurance Case.

■

B.3 Link between the engineering workflow and the assurance cases

This section gives the list of requirements related to the relationship between Workflow and Assurance Cases elements. The principle is that (i) any Engineering Item of the Workflow may be associated with a list of properties that must hold for this item to be correct and valid, and (ii) demonstration that a property holds is captured by the argumentation tree rooted at the Goal associated with the property.

These requirements applies to the version of Capella already extended to support Confiance.AI workflow modelling activities.

B.3.1 Properties

REQ-00010: Properties – Pre- and post-conditions

The tool shall allow capturing the functional properties about an engineering activity.

A functional property is either a pre- or a post-condition of the activity.

A property is expressed as a textual statement involving one or several engineering items in the set of inputs and outputs of the activity.

■

Nota 1: In the general case, the semantics of pre- and post- conditions is the following: "if all pre-conditions p1, p2, are satisfied then, if the activity is correctly executed, then all post-conditions are satisfied". In our context, the semantic is slightly different: "if some pre-condition is not satisfied, then some post-conditions are not satisfied".

Nota 2: Pre- and Post- conditions do not need to be discriminated explicitly. A pre-condition only involves input engineering items whereas a post-condition may involve input and output engineering items.

REQ-00011: Properties –Reference to engineering items

The tool shall allow referencing engineering items in the textual statement of properties.

Engineering items shall be identified using a specific delimiter (e.g. "<>") or color highlighting.

For instance, for the activity that splits a dataset into a training, validation and test dataset:

- Post-1: "The union of the <training dataset>, <validation dataset> and <test dataset> is equal to the <dataset>."
- Post-2: "The intersection between the <training dataset> and <validation dataset> is empty."
- Post-3: "The intersection between the <training dataset> and <test dataset> is empty."
- etc.

Or, more simply:

- Post-1: "The <training dataset>, <validation dataset>, <test dataset> form a partition of the <dataset>."

■

REQ-00012: Properties – Verification

The tool shall provide the capability to verify that all items in angle brackets ("<>") are actual engineering items of the model. In case of error (the item does not exist), it shall be signalled to the user (e.g., by writing the term in red, e.g., [this is not defined]).

■

REQ-00013: Properties – Detection of absence of properties

The tool shall allow signalling situations where an engineering item is not involved in any pre or post condition.

■

REQ-00014: Properties – Activity properties

The tool shall allow capturing activity properties, i.e., property about the way an activity is implemented. For instance:

"The activity uses a formal verification method."

■

REQ-00015: Properties – List of properties

The tool shall allow displaying the list of all properties in which an engineering item is involved.

■

Remark – Relationship between solutions and properties

Solutions are leaves of the argumentation tree. A solution does not express a claim but references an evidence that supports its parent goal.

The evidence may be either some "obvious evidence", i.e. a fact that is commonly admitted but not necessarily "proved" (e.g., "any integer number greater than 3 is the sum of two prime numbers") or an evidence materialized by an engineering item (e.g., a report).

In the latter case, the engineering item may be produced by a dedicated activity (e.g., a test activity) that need to be added to the workflow or by some activity already in the workflow.

In both cases, the solution will usually be associated to one engineering item themselves associated with properties.

For instance, a "software test report" is an engineering item produced by a "test definition and execution activity". The post-conditions for this activity are:

```
Post 1 - All tests passed OK

Post 2 - The tests cover 100% of the code.
```

Remark: Properties – Immutability of engineering items

By default, we adopt a functional approach in which an activity takes a set of input engineering items to generate new engineering items. In this approach, an engineering item is immutable.

For instance, an "annotated dataset" is not some "input dataset" to which "annotations" are added: it is a new engineering item built out of the "input dataset" by copying it and adding an annotation. If needed, the designer will create specific post-conditions to express the relationship existing between the “input dataset” and the “annotated dataset”.

An engineering item being invariant, if it satisfies some property (expressed by some post-conditions of the activity) it will always satisfy it. In particular, if an activity uses some engineering item I input, there is no need to express the fact that I is not modified after the execution of the activity.

REQ-00016: Properties – Relation between pre- and post-conditions

The tool shall allow capturing the relation between pre- and post-conditions using simple logical expressions. For instance:

```
(not Pre-1 or not Pre-2) implies not Post-1
```

where `Pre-1`, `Pre-2` and `Post-1` are pre- and post-conditions for some activity.

This relation will be associated to the activity consuming the inputs and producing the outputs.

In the absence of an explicit relation, we consider that no post-condition is satisfied if at least one pre-condition is not satisfied, i.e.,

```
not P1 or not P2 or ... or not Pn => not P'1 and not P'2 and ... and not P'm.
```

where `Pi` are pre-conditions and P’ are post-conditions.

Rationale: this element of the model allows more accurate impact analysis to be performed. For instance, it can be used to compare the criticality of various properties and determine the argumentation strategy.

■

REQ-00017: Properties – Relation between Properties and Goals

The tool shall allow associating a Property with one GSN Goal.

The semantics of the relation is the following: “the Goal states that the Property holds.”

Navigation shall be possible between Goals and Properties in both directions.

■

Remark: If a property can be supported by multiple arguments, then this shall be captured using appropriate GSN constructs.

REQ-00018: Properties – Risk

The tool shall provide the capability to associate a risk level (i.e, a couple (Probability,Severity)) to a Property. The probability denotes the likelihood for the property to be violated and the severity captures the effect of a property violation at "system level". (The definition of a "system" in this context is user dependent.)

- Possible values for Severity: any integer value in [0,10]
- Possible values for Probability: any float value in [0,1]

■

Rationale: The risk level may be used by the V&V engineer to choose the most appropriate argumentation strategy considering the argumentation and V&V efforts, the uncertainties, and the risk level.

B.3.2 Solutions

REQ-00019: Solutions – Relation with Engineering Items

The tool shall provide the capability to associate zero or one Engineering Item to a Solution. Navigation shall be possible between Solutions and Engineering items in both directions.

■

REQ-00020: Solutions – Verification

The tool shall signal when a solution is associated with no engineering item.

■

B.3.3 User-Strategies capabilities

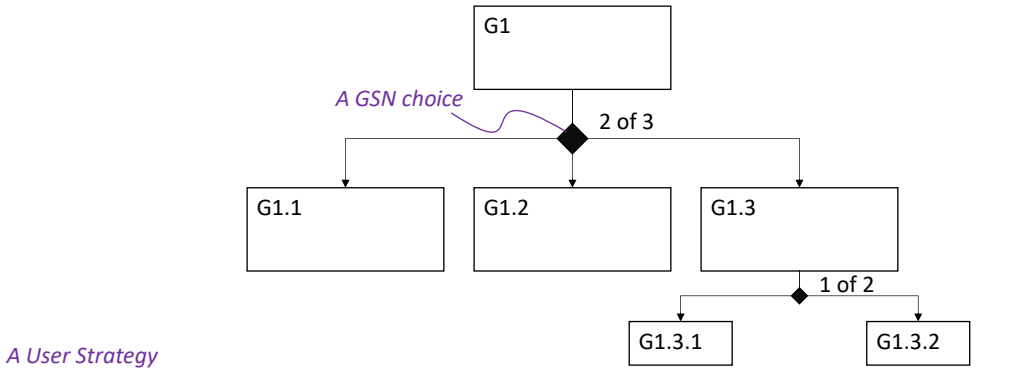
This section collects the requirements related to the management of User-strategies. A User-strategy is a selection of argumentation branches when multiple argumentation choices are possible. A *complete* argumentation is an argumentation with no open choices.

A "**user**-strategy" is different from GSN's "strategy" that is used to justify the decomposition of goals into sub-goals.

REQ-00021: User-strategies – Concept

A user-strategy is expressed by a table showing the choices done by the user with respect to the GSN 3.0 choice decorators ("diamond") of a given Assurance Case.

The table gives, for a given AC, for all goals of this AC associated with sub-goals via a choice relation, the selected sub-goal(s). This is illustrated below:



A User Strategy

Parent node	Choice	Rationale
G1	G1.2, G1.3	
G1.3	G1.3.2	

A User-Strategy is an instance of such table.

■

REQ-00022: User-strategies – Multiple User Strategies

The tool shall support the definition of multiple User-strategies. A User-strategy is identified by a unique name.

■

REQ-00023: User-strategies – Instantiation

The tool shall support the instantiation of a user-strategy, i.e., the application of the choices expressed by the user-strategy. The resulting model shall only contain the modelling elements pertinent with respect to the user-strategy. For instance, activities only providing Engineering Items associated with Solutions **not selected** by the user-strategy must not appear in the resulting model.

■

REQ-00024: User-strategies – Visualization

The tool shall allow visualizing the effect of applying/selecting a specific user-strategy in the argumentation tree: when a user-strategy$S$ is selected, only the elements depending on the user-strategy shall be displayed. The elements excluded by the user-strategy shall be either not displayed at all (which allows simplifying the diagrams if auto-layout is implemented), or simply displayed in a dimmed color. This behaviour shall be user-selectable.

Note that the application of a user-strategy may impact the workflow. Indeed, the workflow contains all activities that may be required by any possible user-strategy. When a user-strategy is selected, some activities may not be necessary and should be hidden. Transitively, the engineering item generated by those activities are not produced, so the properties attached to them to not need to be demonstrated …

■

B.4 Other capabilities

REQ-00025: Documentation - hypertext

The tool shall provide means to add documentation hypertext (including images, hyperlinks, etc.) to any model element.

■

REQ-00026: Documentation generation

The tool shall provide means to generate Word documents from a model using templates (cf. M2Doc).

■

REQ-00027: Coverage status

The tool shall provide a visual indication showing the “completeness” status of an AC (or part of it). For a given node, the indicator shall show the percentage of the leave goals that are associated with solution nodes.

For instance, if all leaves of the subtrees of node X are solution nodes, then the coverage is 100% and the indicator shall be green. If no leave are solution nodes, then the coverage is 0% and the indicator shall be red. The color scale goes from green (100% of the leaves are solution nodes) to red (0% of the leaves are solution nodes).

■

B.5 Edition, Presentation and Navigation capabilities

This section collects requirements about presentation, edition, and searching. We consider that the model will benefit from the “standard’” features of the Capella environment, so focus is placed on features specific (or supposedly specific) to our needs.

B.5.1 Presentation

REQ-00028: AC presentation – Automatic layout

The tool shall allow presenting / editing the AC graphically. The tool shall provide the capability to automatically layout the tree.

■

REQ-00029: Presentation – Tree folding

The tool shall support folding / unfolding parts of the argumentation tree. A visual cue shall be provided to show that a subtree is folded.

■

REQ-00030: Presentation – Textual presentation and edition

The tool shall support the presentation and edition of the argumentation model as text. The editor shall come with automatic completion (when a term in a glossary is entered “[]” or when an engineering item is refered to “<>”).

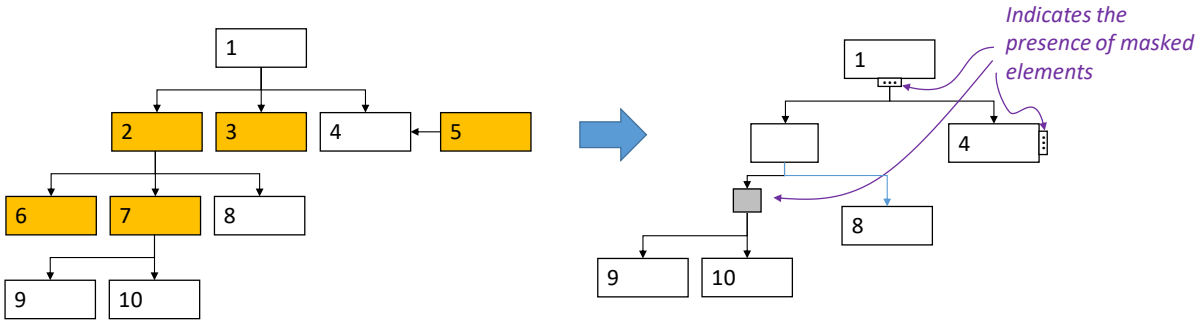
The textual representation shall reflect the model and reciprocally.

■

REQ-00031: Presentation – Masking

The tool shall support masking any element of an assurance case.

- If all elements to be masked are leaves of the tree, a visual clue shall be added to their parents node to indicate that their child nodes are actually masked
- If the elements are not leaves of the tree, they are replaced by a grey box, materializing the hidden nodes. The elements not selected for masking that are linked to masked elements are attached to the grey box representing the masked element.



■

REQ-00032: Presentation – Automatic creation of views

The tool shall allow the automatic generation of diagrams (model views) out of an existing argumentation tree and a pagination constraint (e,g. page size). Generated diagrams shall be such that (i) they cover the complete AC and (ii) there is a "small" overlap between diagrams (e.g., the root goal of a diagram is a leaf goal of another diagram).

Rationale: argumentation tree tends to be too large to be displayed on a unique A4 page. Providing the capability to split a large tree in a unique view (e.g., the view used to *create* the argumentation) in multiple smaller view is necessary and technically feasible since we are dealing with trees.

■

B.5.2 Search

REQ-00033: Search – Search nodes

The tool shall allow the user to enter a keyword or phrase to search for a specific node in the assurance case argumentation tree. The search box should display a list of nodes that match the keyword or phrase entered by the user. In addition, it should highlight the searched keyword or phrase in the search results list. It shall also highlight the nodes for which a match is found.

REQ-00034: Search – Filter search

The tool shall allow the user to filter search results by different categories such as node type or status. It shall also provide an option to search for nodes within a specific branch or level of the argumentation tree.

B.5.3 Edition

REQ-00035: Edition – Autocompletion

The tool shall provide autocompletion (ctrl-space) to reference Engineering Items, Properties, Goals, Glossary entries in textual descriptions.

■

REQ-00036: AC edition – Identifiers

The tool shall automatically assign an identifier to an AC modeling element (excluding relations), according to the following rules:

Node type	Identifier structure
Definition	“D<number>”
Goal	“G<number>”
Context	“C<number>”
Strategy	“S<number>”
Solution	“SO<number>”
Assumption	“A<number>”
Justification	“J<number>”

Where <number> is a 6 digits integer value, e.g., “123456”, “001234”.

All identifiers shall be unique in a given model.

For a given AC, identifiers shall not change or be reused except upon explicit request of the user (see next req).

■

REQ-00037: AC edition – Identifier renumbering

The tool shall provide the capability to renumber all nodes of an AC automatically. Renumbering shall be done on the sub-tree designated by its root node. Numbering shall be done in a breadth-first manner. The starting value shall be given by the user. Ids already allocated to nodes that have been deleted can be reused.

■

B.6 Impact analysis

B.6.1 Attributes

This section collects requirements related to the attributes that can be associated with GSN elements. Those attributes are used to perform evaluations.

REQ-00038: Impact analysis

The tool shall provide the capability to determine the activities to be redone when an Engineering Item is updated.

This means:

- Identifying the Properties that need to be reverified (i.e., any property involving E or an engineering items E' that may be modified directly or indirectly by an update of E)
- Identifying the Goals that need to be re-demonstrated (i.e., all goals related to the properties that need to be reverified)
- Identifying the Solutions that need to be re-generated (i.e., all Solutions related to Goals that need to be re-demonstrated)
- Identifying Activities that need to be redone (all activities depending directly of indirectly of E, or to the production of a Solution that need to be regerenrated.)

■

B.6.2 Evaluation capabilities

This section collects the requirements related to the evaluation of an Assurance Case. As of version 1.0, the following evaluation metrics are considered:

- Metrics related to the argumentation:
  - Uncertainty (*units to be defined*)
  - Cost (integer with no dimension)
- Metrics related to the design of the argumentation:
  - Level of completion (%)
  - Level of maturity (low, medium, high)

Those evaluation metrics are computed using data associated with the modelling elements. Those data are either provided by the user in the model or loaded from a file.

The level of completion is different from the coverage status indicator for it refers to any user-defined metric whereas the coverage indicator only refers to the presence of Solutions.

In the same manner, the level of maturity is a user-defined metric with no specified semantics.

Note: another solution could be to allow user to add any attribute and provide a few combination laws to compute attribute values on the complete tree. *To be discussed with the developers.*

REQ-00039: Evaluation – Metrics

The tool shall provide the capability to compute the following metrics for an assurance case

- Uncertainty
- Cost
- Maturity

These metrics are associated with Goal and Solution nodes.

These metrics are computed as follows:

- **Uncertainty** - *The case of "uncertainty" is very specific and will be specified in a future version of this document.*
- **Cost:** the cost associated with a node is calculated from the cost of the child nodes. The calculation depends on the relation between the parent node and child nodes. E.g., if the relation is a "and" (i.e., all sub-goals must be reached for the parent goal to be reached), then the cost of the parent goal is the sum of the costs of the child goals; if the relation is a "n out of m" (i.e., at least n sub-goals must be reached out of m sub-goals), then the cost of the parent goal is the sum of the costs of the sub-goals in the less costly combination.

  "cost" is a floating point value whose actual meaning (time, €, etc) is left to the user.
- **Maturity:** same as "cost" but maturity does not sum up: when several sub-goals must be reached for the parent goal to be reached, the maturity of the parent goal is the "min" of the maturity of the associated sub-goals.

■

Note: The same metrics can be applied during the design and exploitation of the Assurance Case. During design, the cost may correspond to the "expected" cost of providing a solution whereas during exploitation, the cost may be the actual cost of providing the solution.

REQ-00040: Evaluation – Attributes setting

For all attributes involved in the computation of a metric, the attribute may be either computed automatically by propagating the values using the combination operators or set explicitly by the user.

Rationale: during the early phases of AC elaboration, some metrics may be at first set by the user (as a "budget") and later be computed when the model is refined.

■

REQ-00041: Uncertainty – Capture

The tool shall provide a means to set the belief and disbelief attributes of a node according to the method described in [3].

This requirement has to be elaborated. It shall not be considered in the first development batch.

■

REQ-00042: Uncertainty – Computation

The uncertainty of a goal is evaluated according to the method described in [4].

(The algorithm is currently being designed. It will be provided in a future version of the document.)

■

B.7 Generation of Verification and Validation plans

REQ-00043: V&V plan – Plan Generation

The tool shall allow generating a Verification plan (Word format) for a selected Goal and User-strategy. The V&V plan shall contain:

- Pertinent glossary elements
- Solutions (incl. their description)

- V&V Activities providing solutions (incl. their description)

(Templates will be provided in a future version of the specification.)

■

B.8 Import and export capabilities

REQ-00044: Import-export capability

The tool shall allow exporting / importing ACs in textual format.

The concrete textual syntax is defined in Annex 1.

Rationale: entering / editing AC using a textual notation is often more convenient than using a graphical notation.

■

C. References

[1] 'GSN Community Standard V3', The Assurance Case Working Group (ACWG), SCSC-141C, May 2021. [Online]. Available: https://www.goalstructuringnotation.info/

[2] 'Structured Assurance Case Metamodel (SACM)', Object Management Group (OMG), Standard ptc/21-02-02, 2021.

[3] Y. Idmessaoud, J. Guiochet, and D. Dubois, 'Questionnaire for estimating uncertainties in assurance cases', LAAS/CNRS, Rapport LAAS n° 22142. 2022.

[4] Y. Idmessaoud, D. Dubois, and J. Guiochet, 'Quantifying Confidence of Safety Cases with Belief Functions', in *Belief Functions: Theory and Applications*, T. Denœux, E. Lefèvre, Z. Liu, and F. Pichon, Eds., in Lecture Notes in Computer Science, vol. 12915. Cham: Springer International Publishing, 2021, pp. 269–278. doi: 10.1007/978-3-030-88601-1_27.

Appendix 1 – Example of Textual syntax for Assurance Cases

The BNF given hereafter is provided as an example. It refers to a simpler version of ACs.

```
<AssuranceCase> ::= <Identifier> <EOL> <SuppElement>* <Node>*
<SuppElement>   ::= <Definition> | <Context> | <Assumption>
<Definition>    ::= "[D]" <Identifier>  <RichLine> <EOL>
<Context>       ::= "[C]" [<Identifier> ":"] ( <RichLine> | <Reference>)
<Assumption>    ::= "[A]" [<Identifier> ":"] ( <RichLine> | <Reference>)

<Node>          ::= <Indent> <CoreElement>

<CoreElement>   ::= <Goal> | <Strategy> | <Evidence>
<Goal>          ::= "[G]" [<Digit>] [<Identifier> ":"] (<RichLine> | <Reference>)
<Strategy>      ::= "[S]" [<Digit>] [<Identifier> ":"] (<RichLine> | <Reference>)
<Solution>      ::= "[SO]" <Identifier>

<Indent>        ::= <Tab>*
<RichLine>      ::= { <Sentence> | <Identifier> | <Formula>}* <EOL>
<Reference>     ::= "=" <Identifier>
<Identifier>    ::= "<" <Sentence> ">"
<Digit> =       ::= [1-9][0-9]*
<Sentence>      ::= [<String> | "\s"]*
<String>        ::= [a-zA-Z0-9_]*
<Tab>           ::= [\s]
<EOL>           ::= CRLF
<Formula>       ::= "$" {mathematical_operators_regex | <Digit> }* "$"
```

Example of AC using this grammar:

```
<Dummy assurance case>

[D] <another definition>: another description, which may also reference <a context> or anything else
[C] <a context>: the description of supporting
elements and references
```

```
[C] <another context>: a different context, or additional one, with unique titles

[A] <an assumption>: when we assume something for the rest of the argumentation

/* Multiline comment

supported */

[G] the description of a goal or claim

    [G] the description of the subgoal of parent goal, with hierarchy represented by indentation

        [S] the description of the argument used to decompose the parent goal into sub elements

            [G] <first strategy subgoal>: the first subgoal of the strategy

                [SO] <solution for the goal>

                    [C] The <solution for the goal> is a evidence item that serves as a proof  to complete the upstream
subgoal and must contain de following characteristics...

            [G] <second strategy subgoal>: another subgoal

                [SO] <another solution>

                [C] The <another solution> is a proof that must contain the following details...

            [G] <third subgoal>: and another

                [G] <yet another subgoal>: even more subgoal hierarchy

                    [SO] <unsurprising solution>

    [G]1 One of the sub elements is sufficient to satisfy current goal, supported by <an assumption>

        [G]2 <another disjunctive goal>: at least 2 sub elements satisfied are required

            [S] <an optional strategy>: a decomposition referencing <a context>

                [G] <a subgoal X>: one element of the strategy

                    [SO] <solution>

                      [C} The <solution> is based on <a definition> and referencing an <evidence item X>

                            [D]<mandatory title>: the definition of $test=8$ something, which can reference <another
definition>

                [SO]  <evidence item>

                    [C] This context adds information about the <evidence item>such as it must contain specific
evidential information.

            [G] <a subgoal A>: description, referencing <another definition>

                [SO] <a copy solution> = <a direct solution>

                [SO] = <another solution>

                [G] <a ref to an away goal> = <name of an away-goal>

                [SO] <second evidence item for A>

                    [C]description of <second evidence item for A> based on a <specific method>

            [G] <a subgoal B>: and one of these 3 sub elements can be ignored, or not fulfilled

                [SO] <solution item for B>

                 [C] description of <solution item for B> also based on <a definition>

        [G] <another possible goal>: the other choice to satisfy parent goal, based on the formula 724 >= p_x, p^{i=0}

            [SO] <the item solution>

                [C] adescription of <the item solution>
```

Titre : Requirements Specification to support Assurance Cases development in CAPELLA

Mots clefs : Machine Learning, Vérification et Validation, Assurance Cases

Ce document spécifie les capacités à intégrer dans l'outil CAPELLA pour permettre la mise en œuvre de l'approche d'argumentation définie dans le projet EC6.

Title : Requirements Specification to support Assurance Cases development in CAPELLA

Keywords : Machine learning, Verification and validation, Assurance Cases

This document specifies the capabilities to be implemented in CAPELLA in order to support the argumentation approach defined in EC6.