Table of Contents

1 Scope

1.1 General

This specification defines a metamodel for representing structured assurance cases. An Assurance Case is a set of auditable claims, arguments, and evidence created to support the claim that a defined system/service will satisfy the particular requirements. An Assurance Case is a document that facilitates information exchange between various system stakeholder such as suppliers and acquirers, and between the operator and regulator, where the knowledge related to the safety and security of the system is communicated in a clear and defendable way. Each assurance case should communicate the scope of the system, the operational context, the claims, the safety and/or security arguments, along with the corresponding evidence.

Systems Assurance is the process of building clear, comprehensive, and defensible arguments regarding the safety and security properties of systems. The vital element of Systems Assurance is that it makes clear and well-defined claims about the safety and security of systems. Certain claims are supported through reasoning. Reasoning is expressed by explicit annotated links between claims, where one or more claims (called sub-claims) when combined provide inferential support to a larger claim. Certain associations (recorded as assertions) between claims and subclaims can require supporting arguments of their own (e.g., justification of an asserted inference). Claims are propositions which are expressed by statements in some natural language. The degree of precision in formulation of the claims may contribute to the comprehensiveness of an assurance case. The context is important to communicate the scope of the claim, and to clarify the language used by the claim by providing necessary definition and explanations. Context involves assumptions made about the system and its environment. Explicit statement of the assumptions contributes to the comprehensiveness of the argument. Argumentation flow between claims is structured to facilitate communication of the entire assurance case.

1.2 Structured Arguments

Part of this specification defines a metamodel for representing structured arguments. A convincing argument that a system meets its assurance requirements is at the heart of an assurance case, which also may contain extensive references to evidence. The Argumentation Metamodel facilitates projects by allowing them to effectively and succinctly communicate in a structured way how their systems and services are meeting their assurance requirements. The scope of the Argumentation Metamodel is therefore to allow the interchange of structured arguments between diverse tools by different vendors. Each Argumentation Metamodel instance represents the argument that is being asserted by the stakeholder that is offering the argument for consideration.

This specification is designed to stand alone, or may be used in combination with the SACM Artefact Metamodel. The Artefact Metamodel is designed to represent aspects of evidence and properties about evidence in further detail. In the Argumentation Metamodel we have simplified support to model the relation of evidence to a structured argument. Standardization will ensure that end users are investing not just in individual tools but also rather in a coordinated strategy.

The metamodel for argumentation provides a common structure and interchange format that facilitates the exchange of system assurance arguments contained within individual tool models. The metamodel represents the core concepts for structured argumentation that underlie a number of existing argumentation notations.

1.3 Evidence

Part of this specification provides a metamodel for communicating the way in which evidence artefacts are collected by various participants using techniques, resources and activities. This allows users to build a repository of evidence that communicates its provenance and how it was gathered. This Artefact Metamodel identifies the main elements that determine the evidence collection process: artefacts, participants, resources, activities and techniques. Artefacts may be exchanged as packages or combined into composites.

The SACM Artefact Metamodel defines a catalog of elements for constructing and interchanging packages of evidence that communicate how the evidence was collected.

In conjunction with the Argumentation Metamodel, certain claims may be expressed to be supported by evidence that is within the Artefact Metamodel, to permit the authors of the assurance claims to offer evidentiary support for their positions. Evidence is usually collected by applying systematic methods and procedures and is often collected by automated tools.

Evidence is information or objective artefacts, based on established fact or expert judgment, which is presented to show that the claim to which it relates is valid (i.e., true). Various and diverse things may be produced as evidence, such as documents, expert testimony, test results, measurement results, records related to process, product, and people, etc.

1.4 History, Motivation, and Rationale

The original Structured Assurance Case Metamodel version 1.0 was the composite of two efforts within the OMG's Systems Assurance Task Force. One effort, the Structured Assurance Evidence Metamodel (SAEM) was created through the OMG Request For Proposal (RFP) approach and the other, the Argumentation Metamodel (ARG) was created through the OMG Request For Comment (RFC) approach. Both were completed in the mid-2010 timeframe and then put into the same Finalization Task Force (FTF) due to the interconnectedness of their topics and concepts. The first version of SACM was eventually produced in the spring of 2012 consisting of a top-level container object joining SAEM and ARG without significantly altering the two original metamodels.

A Revision Task Force (RTF) was convened to drive further integration of the two original parts of SACM into one Metamodel and that effort formulated a set of goals to shape and guide the integration. Basically the stated goals were:

- Improve support for ISO/IEC 15026-2. In order to facilitate the use of structured assurance cases for producing and reviewing ISO/IEC 15026-2 conformant assurance cases, the structured assurance case metamodel needs to more fully support the constructs and entities in ISO/IEC 15026-2.

- Improve support for "Goal Structuring Notation." In order to facilitate the use of structured assurance cases by the existing community of practitioners across the world that are currently using Goal Structuring Notation (GSN) and the specific capabilities in GSN for working with assurance cases, the structured assurance case metamodel needs to more fully support the constructs and entities in GSN.

- Harmonization of Parts. In order to facilitate acceptance and successful use of SACM, the argumentation and evidence container metamodels need to be more consistently aligned and integrated. Areas of focus include elimination of overlap, making useful facilities now available on one side generalized to be useful on both sides, achieving uniform terminology and consistency, and using common concepts.

- Add initial support for Patterns/Templates. In order to make the use of assurance cases more practical and efficient for users including those that do not have in-depth experience within the assurance case domain (e.g., acquisition officials, systems integrators, auditors, regulators, and tool vendors), the structured assurance case metamodel needs to support the concept of assurance case patterns and templates. Patterns will provide support to enable reuse and the effective composition of assurance cases along with the underlying argumentation supporting goals. Templates will provide support for defining and describing constraining conventions that a community may require for assurance cases within a particular domain due to regulatory requirements or accepted practices in that domain/industry/community.

- Improve the modularity and simplicity of SACM

- Provide for future concepts such as structured expressions and other formalisms

The SACM 1.1 was subsequently worked to attempt to meet these goals and a draft metamodel was created during the summer OMG 2013 Berlin meeting. However the magnitude of the changes necessary to actually integrate the two original metamodels into one cohesive approach and achieve some of the other goals turned out to be too big of a change for a point release. The final SACM 1.1, released in July 2015, was scaled back to address some of the issues and it cleaned up some terminology and logical issues but it did not substantially alter the underlying metamodel.

During this same timeframe other efforts in the OMG (the Dependability Assurance Framework for Safety-Sensitive Consumer Devices (DAF)) and in The Open Group (the Dependability Assurance Framework (O-DA), as well as the work of the Food and Drug Administration (FDA) in the U.S. started making use of the assurance case concept and articulated implicit requirements/needs for tools that would work with assurance case models and their exchange.

Additionally, the Open Platform for EvolutioNary Certification of Safety-critical Systems (OPENCOSS) effort in Europe was exploring different uses of assurance cases, including the creation of a Common Certification Language, and the OMG's Architecture Driven Modernization Task Force crafted a Structured Pattern Metamodel Standard (SPMS) that provided a

method for describing patterns within models. Together these new needs and the new openly available capabilities represented in OPENCOSS and SPMS offer a way forward.

This version 2.0 of SACM has been created as a major version release since pursuing another point release revision of SACM would appear to be incompatible with achieving the integration and harmonization that is critical to obtain wide-spread adoption and implementation within the tooling market and allow that market to deliver on some of the potential capabilities they could provide to address the emerging and evolving need for assurance case tools, such as:

- Improving the Understandability of an Assurance Case to a 3rd Party
- Improving Rigor of Assurance Cases through Modeling
- Allowing for Reexamination of Assumptions, Argument Structuring, and the Appropriateness of Evidence
- Allowing for Reuse of Sub-Claim/Evidence Constructs That "Work"
- Authoring/Sharing Libraries of Sub-Claims/Supporting Evidence
- Providing for Assurance Case Analytics/Validation
- Providing for Exchange of Assurance Cases (Import/Export)
- Providing for Enforcing Community of Interest Norms of Practice

The resulting metamodel in this version of SACM come from the ideas in the 2013 Berlin metamodel, along with the approaches utilized for modeling artefact- and process-related concepts in OPENCOSS Common Certification Language and the pattern metamodel and concepts from the SPMS.

2 Conformance

2.1 Introduction

The Structured Assurance Case Metamodel (SACM) specification defines the following three compliance points:

- Argumentation
- Artefact Model
- Assurance Case

2.2 Argumentation compliance point

Software that conforms to the SACM specification at the Argumentation compliance point shall be able to import and export XMI documents that conform with the SACM XML Schema produced by applying XMI rules to the normative MOF metamodel defined in the Argumentation subpackage of the SACM specification, including the common elements defined in the Common and Predefined diagrams of the SACM. The top object of the Argumentation package as a unit of interchange shall be the Argumentation::ArgumentPackage element of the SACM.

Conformance to the Argumentation compliance point does not entail support for the Evidence subpackage of SACM, or the terminology sub package of the SACM. The 'ArtefactElementCitation' class shall not be used.

This compliance point facilitates interchange of the structured argumentation documents produced by existing tools supporting existing structured argument notations such as the Goal Structuring Notation (GSN) and the Claims-Arguments-Evidence (CAE) notation which provide their own mapping onto SACM argumentation aspects. Further details of these mappings are given in Annex A.

2.3 Artefact model compliance point

Software that conforms to the specification at the Artefact Model compliance point shall be able to import and export XMI documents that conform with the SACM XML Schema produced by applying XMI rules to the normative MOF metamodel defined in this Artefact subpackage of the SACM specification, including the common elements defined in the Common and

Predefined diagrams of the SACM. The top object of the Evidence package as a unit of interchange shall be the ArtefactModel::ArtefactPackage element of the SACM.

Conformance to the Artefact model compliance point does not entail support for the Argumentation subpackage of SACM, or the terminology diagram of the SACM. This compliance point facilitates interchange of the packages of evidence. In particular, this compliance point facilitates development of evidence repositories in support of software assurance and regulatory compliance.

2.4 Assurance Case compliance point

Software that conforms to the specification at the Assurance Case compliance point shall be able to import and export XMI documents that conform with the SACM XML Schema produced by applying XMI rules to the normative MOF metamodel defined in this entire specification. The top object of the Assurance Case package as a unit of interchange shall be the SACM::AssuranceCasePackage element.

The Conformance clause identifies which clauses of the specification are mandatory (or conditionally mandatory) and which are optional in order for an implementation to claim conformance to the specification.

3 References

3.1 Normative References

The following normative documents contain provisions which, through reference in this text, constitute provisions of this specification. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply.

- ISO/IEC 15026-1:2013 Systems and software engineering - Systems and software assurance - Part 1: Concepts and vocabulary, 2013. <http://www.iso.org/iso/catalogue_detail.htm?csnumber=62526>
- ISO/IEC 15026-2: 2011 Systems and software engineering - Systems and software assurance - Part 2: Assurance case, 2011. <http://www.iso.org/iso/catalogue_detail.htm?csnumber=52926>
- OMG UML 2.5 Infrastructure Specification formal/15-03-01. <http://www.omg.org/spec/UML/>
- OMG Meta-Object Facility (MOF) version 2.5 formal/2015-06-05. <http://www.omg.org/spec/MOF/>
- OMG MOF XML Metadata Interchange (XMI) Specification, version 2.5.1, formal/2015-06-07 <http://www.omg.org/spec/XMI/>

3.2 Non-normative References

The following non-normative documents contain provisions which, through reference in this text, provide informative context for material in this specification.

- Goal Structuring Notation (GSN) Community Standard, Nov 2011. <http://www.goalstructuringnotation.info/documents/GSN_Standard.pdf>
- Open Platform for EvolutioNary Certification of Safety-critical Systems (OPENCOSS) WP4: Common Certification Language, 2012-2015. <http://www.opencoss-project.eu/node/7>
- Open Platform for EvolutioNary Certification of Safety-critical Systems (OPENCOSS) WP6: Evolutionary Evidential Chain, 2012-2015. <http://www.opencoss-project.eu/node/7>.
- Evidence management for compliance of critical systems with safety standards: A survey on the state of practice, Information and Software Technology 60: 1-15, Elsevier (North-Holland) (2015). <http://www.sciencedirect.com/science/article/pii/S0950584914002560>
- OMG Structured Pattern Metamodel Standard (SPMS), beta2, ptc/14-09-31 <http://www.omg.org/spec/SPMS/>

- Open Group Dependability Assurance Framework (O-DA), Jul 2013. <https://www2.opengroup.org/ogsys/catalog/C13F>
- OMG Dependability Assurance Framework for Safety-Sensitive Consumer Devices (DAF), beta1, May 2015. <http://www.omg.org/spec/DAF/>
- Infusion Pumps Total Product Life Cycle Guidance for Industry and FDA Staff, Dec 2014. <http://www.fda.gov/medicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/ucm206153.htm>

4 Terms and Definitions

For the purposes of this specification, the following terms and definitions apply.

Argument

A body of information presented with the intention to establish one or more claims through the presentation of related supporting claims, evidence, and contextual information.

Assurance Case

A collection of auditable claims, arguments, and evidence created to support the contention that a defined system/service will satisfy its assurance requirements.

Claim

A proposition being asserted by the author or utterer that is a true or false statement.

Evidence

Objective artefacts being offered in support of one or more claims.

Evidence Repository

A software service providing access to, and information about, a collection of evidence items, such as records, documents, and other exhibits together with related information that facilitates management of evidence, the interpretation of evidence, and understanding the evidentiary support provided to claims.

Structured argument

A particular kind of argument where the relationships between the asserted claims, and from the evidence to the claims are explicitly represented.

5 Symbols

There are no symbols defined in this specification.

6 Additional Information

6.1 Changes to Adopted OMG Specifications [optional]

This specification completely replaces the SACM 1.1 specification.

6.2 Acknowledgements

The following companies submitted this specification:

- MITRE Corporation
- Adelard LLP
- KDM Analytics
- Lockheed Martin
- Benchmark Consulting
-

The following companies supported this specification:

- University of York
- Universidad Carlos III de Madrid
- Carnegie Mellon University
-

6.3 How to Proceed

The rest of this document contains the technical content of this specification.

Clause 7. Specification overview - Provides design rationale for the SACM Argumentation Metamodel specification.

Part 1 of the specification defines the normative common elements. This part includes three clauses. Material in this part of the specification is related to all compliance points.

Clause 8. SACM Base classes define the common base classes of the Structured Assurance Case Metamodel.

Clause 9. SACM Packages define the common packages of the Structured Assurance Case Metamodel.

Clause 10. SACM Terminology defines the common terminology classes of the Structured Assurance Case Metamodel.

Part 2 of the specification defines the SACM Argumentation metamodel. The Argumentation Metamodel defines the catalog of elements for constructing and interchanging structured statements describing argumentations. Material in this part of the specification is related to the Assurance Case and Argumentation compliance points, and is not required for the Evidence Container compliance point. This part includes a single clause. The non-normative Annex B contains some examples of the SACM XML interchange format for Argumentation, and describes how SACM Argumentation is related to existing graphical notations for describing structured arguments, such as the Goal Structuring Notation (GSN) and the Claims-ArgumentsEvidence (CAE) notation.

Clause 11. The SACM Argumentation Metamodel - Provides the details of the Argumentation Metamodel specification.

Part 3 of the specification defines the SACM Artefact Metamodel. The Artefact Metamodel defines the catalog of elements for constructing and interchanging precise statements involved in evidence-related efforts. This part includes a single clause. Material in this part of the specification is related to the Assurance Case and the Evidence Container compliance points, and it is not required for the Argumentation compliance point.

Clause 12 defines the key elements of the Artefact Metamodel.

arguments comprise argument elements (primarily claims) that are being asserted by the author of the argument, together with relationships that are asserted to hold between those elements.

7.5 Precise statements related to evidence

In the simplest form, evidence consists of a collection of documents, records or artefacts that provide evidentiary support to a set of claims.

Artefacts may be structured together into composite artefacts or collections. For higher degrees of assurance it is pertinent to know how these artifacts have been created and managed over their lifecycle, and what techniques and resources were used in their generation – i.e., the provenance of the artefact.

The Artefact Metamodel defines the vocabulary for constructing and interchanging precise statements describing evidence-related efforts, including

- Describing artefacts and their properties and associated events
- Collection and management of evidence by participants, using resources, techniques, and activities, by describing the relationships between them
- Structuring of artefacts – e.g., as composite artefacts or collections

An extensible approach is presented whereby users of an Artefact Model may specify the relationships that hold between the artefact assets. If necessary a terminology package may be used to reuse common relationships.

Describing artefacts – artefacts have properties and associated events. An artefact event can be used to communicate, for example, the review date or release date for the artefact.

Collection and Management of Evidence – can be described by means of an extensible set of relationships between participants, activities, resources and the associated evidence artefacts.

Structuring of artefacts – Artefacts may be part of a larger composite by means of artefact to artefact relationships, or within a common artefact package.

Part I - Common Elements

The first part of the specification defines the common elements of the Structured Assurance Case Metamodel, including the Base Classes, the Structured Assurance Case Terminology Classes, and the Structured Assurance Case Packages. Subsequent parts define the Argumentation Metamodel and the Artefact Metamodel.

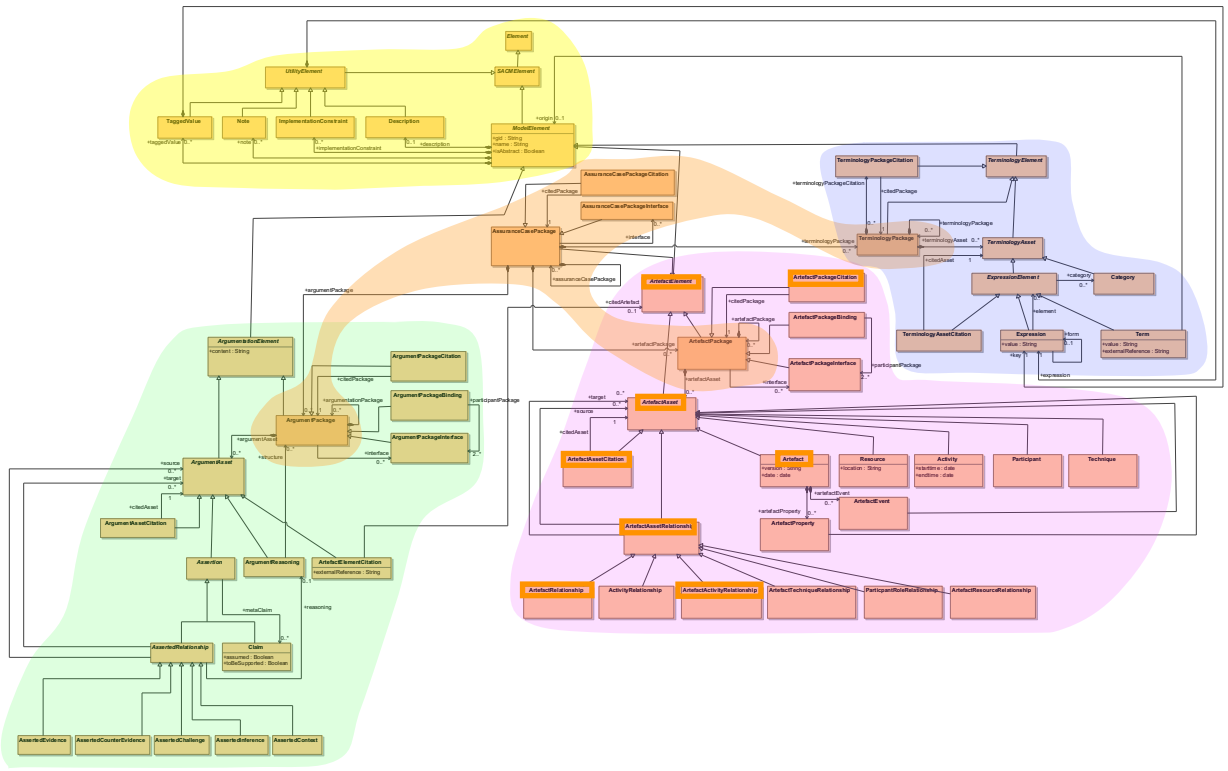


Figure 7.2 - Overall SACM Class Diagram

Yellow denotes items covered in Clause 8, Structured Assurance Case Base Classes.

Orange denotes items covered in Clause 9, Structured Assurance Case Packages.

Blue denotes items covered in Clause 10, Structured Assurance Case Terminology Classes.

Green denotes items covered in Clause 11, Argumentation Metamodel.

Purple denotes items covered in Clause 12, Artefact Metamodel.

8.7 Note

This class specifies a generic note that may be associated with a ModelElement. For example a note may include a number of explanatory comments.

Superclass

UtilityElement

Semantics

Notes are used to specify additional (typically optional) generic, unstructured, untyped information about a ModelElement. An example of this kind of information could be a comment about a ModelElement.

8.8 TaggedValue

This class represents a simple key/value pair that can be attached to any element in SACM. This is a simple extension mechanism to allow users to add attributes to each element beyond those already specified in SACM.

Superclass

UtilityElement

Attributes

key: Expression – the key of the tagged value

Semantics

TaggedValues can be used to specify attributes, and their corresponding values, for ModelElements.

Constraints

TaggedValues should not be used to document attributes that already form part of SACM (e.g., ArtefactProperty).

9 Structured Assurance Case Packages

9.1 General

This chapter presents the normative specification for the SACM Packages Metamodel. It begins with an overview of the metamodel structure followed by a description of each element.

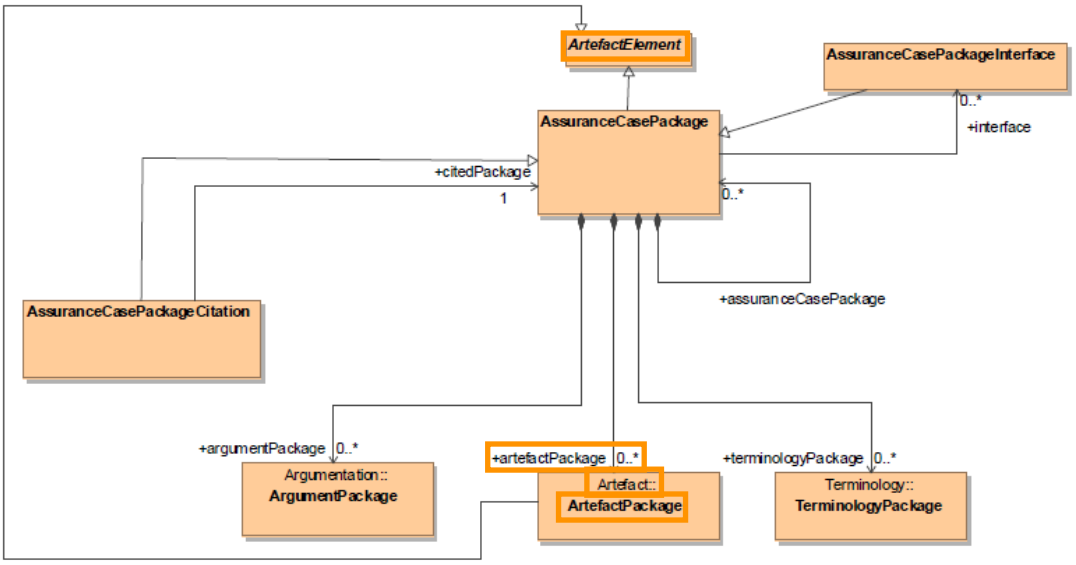


Figure 9.1 - Structured Assurance Case Packages Class Diagram

In SACM, the parent container element is AssuranceCasePackage. AssurancesCasePackages can be thought of assurance case 'modules'. Packages can contain other packages, including citations to other packages not contained within the same package hierarchy. Packages optionally can have a separately declared interface (AssuranceCasePackageInterface) (analogous to a public header file) that declares selected packages contained by a package.

Assurance cases (AssuranceCasePackages) consist of arguments (containined in ArgumentPackages), evidence descriptions (contained in ArtefactPackages) and Terminology definitions (contained in TerminologyPackages).

9.2 ArtefactElement (abstract)

ArtefactElement is an abstract class that serves as a parent class for Artefacts and AssuranceCasePackage elements.

Superclass

ModelElement

Semantics

ArtefactElement correspond to the base class for specifying all the identifiable units of data modelled and managed in a structured assurance case effort.

9.3 AssuranceCasePackage

AssuranceCasePackage is an exchangeable element that may contain a mixture of artefacts, argumentation and terminology. When users exchange content, it is expected they use this as the top level container. It is a recursive container, and may contain one or more sub-packages.

This follows the existing practice of considering an assurance case when fully completed to comprise both argumentation and evidence, although each may be exchanged individually.

AssuranceCasePackage is a sub-class of ArtefactElement. Semantically an AssuranceCasePackage can be considered as an artefact of evidence (e.g. from the perspective of another AssuranceCasePackage).

Superclass

ArtefactElement

Associations

assuranceCasePackageCitation: AssuranceCasePackageCitation [0..*] – a collection of optional citations to other AssuranceCasePackages

assuranceCasePackage: AssuranceCasePackage [0..*] – a number of optional sub-packages

interface: AssuranceCasePackageInterface [0..*] – a number of optional assurance case package interfaces that the current package may implement

artefactPackage: ArtefactPackage [0..*] – a number of optional artefact sub-packages

terminologyPackage: TerminologyPackage [0..*] – a number of optional terminology sub-packages

Semantics

AssuranceCasePackage is the root class for creating structured assurance cases.

9.4 AssuranceCasePackageInterface

AssuranceCasePackageInterface is a kind of AssuranceCasePackage that defines an interface that may be exchanged between users. An AssuranceCasePackage may declare one or more ArtefactPackageInterfaces.

Superclass

AssuranceCasePackage

Semantics

AssuranceCasePackageInterface enables the declaration of the elements of an AssuranceCasePackage that might be referred to (cited) in another AssuranceCasePackage, thus the elements can be used for assurance in the scope of the latter AssuranceCasePackage.

Constraints

AssuranceCasePackageInterface are only allowed to contain the following: ArgumentPackageInterfaces, ArtefactPackageInterfaces, and TerminologyPackages.

9.5 AssuranceCasePackageCitation

AssuranceCasePackageCitation is used to cite another AssuranceCasePackage. The citation can be used where an assurance case author wishes to refer to an AssuranceCasePackage outside of the current AssuranceCasePackage hierarchy.

Superclass

ArtefactElement

Associations

citedPackage: AssuranceCasePackage – the existing AssuranceCasePackage being referenced.

Constraints

The citedPackage referred to by a AssuranceCasePackageCitation must be outside of the containment hierarchy containing the citation.

9.6 ArgumentPackage

ArgumentPackage is a container for the structured argument aspect of the assurance case. It contains the structure of assertions which comprise the structured argument.

Superclass

ArgumentationElement

Associations

argumentPackageCitation: ArgumentPackageCitation [0..*] – an optional set of citations to other ArgumentPackages

argumentPackage: ArgumentPackage [0..*] – an optional set of sub ArgumentPackages, allowing for recursive

containment argumentAsset: ArgumentAsset [0..*] an optional set of ArgumentAssets

Semantics

ArgumentPackage is the base class for specifying the results of the argumentation efforts for a structured assurance case (i.e., an AssuranceCase).

9.7 TerminologyPackage

TerminologyPackage is a container element for terminology that may be exchanged. Terminology can define terms, expressions or categories, used elsewhere in the assurance case.

Superclass

TerminologyElement

Associations

terminologyPackageCitation: TerminologyPackageCitation [0..*] – an optional set of citations to other TerminologyPackage elements

terminologyAsset: TerminologyAsset [0..*] – an optional set of terminology assets (expressions, terms and categories)

terminologyPackage: TerminologyPackage [0..*] – an optional set of contained TerminologyPackage elements, allowing for recursive containment.

Semantics

TerminologyPackage is the base class for specifying all the terminology needs and constraints (via TerminologyAssets) for a structured assurance case (i.e., an AssuranceCase).

9.8 ArtefactPackage

ArtefactPackage is a container element for the assets that are used as evidence or cited in support of a structured argument. These assets form the evidential basis for the assurance case.

Superclass

ArtefactElement

Associations

artefactPackageCitation: ArtefactPackageCitation [0..*] – an optional set of citations to other ArtefactPackage elements

artefactAsset:

ArtefactAsset [0..*] – an optional set of ArtefactAsset elements, such as citations, artefacts, resources, activities, etc.

artefactPackage: ArtefactPackage [0..*] - an optional set of contained ArtefactPackage elements, allowing for recursive containment.

Semantics

ArtefactPackage is the base class for specifying and structuring the ArtefactAssets of a structured assurance case (i.e., an AssuranceCase).

10 Structured Assurance Case Terminology Classes

10.1 General

This chapter presents the normative specification for the SACM Terminology Metamodel. It begins with an overview of the metamodel structure followed by a description of each element.

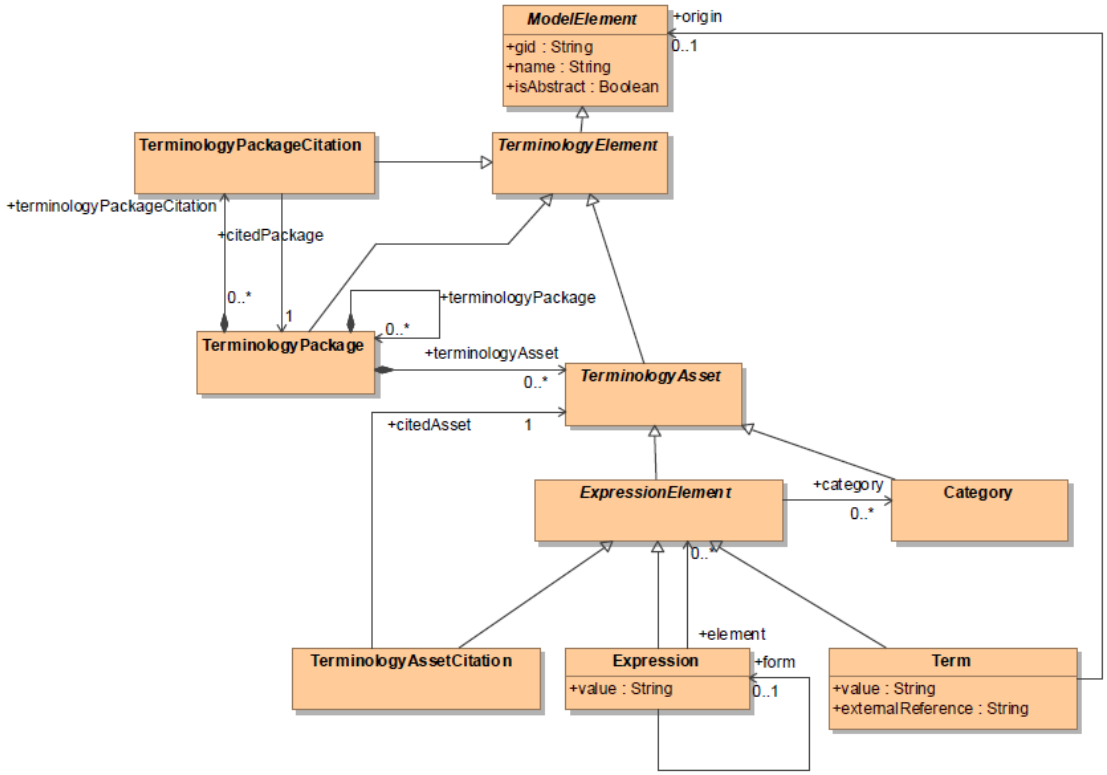


Figure 10.1 - Terminology Class Diagram

This portion of the SACM metamodel describes and defines the concepts of term, expression and an external interface to terminology information from others. This area of the Structured Assurance Case Metamodel also provides the starting foundation for formalism in the assembly of terms into expressions without mandating the formalism for those that do not need it.

10.2 TerminologyElement (abstract)

TerminologyElement is an abstract class that serves as a parent class for all SACM terminology assets (TerminologyAsset) and the packaging of these assets (TerminologyPackage and TerminologyPackageCitation).

Superclass

ModelElement

Semantics

TerminologyElement is the base class for specifying the terminology aspects of an assurance case (AssuranceCasePackage).

10.3 TerminologyPackage

The TerminologyPackage Class is the container class for SACM terminology assets.

Superclass

TerminologyElement

Associations

TerminologyAsset:TerminologyAsset[0..*]

The TerminologyAssets contained in a given instance of a TerminologyPackage.
terminologyPackage:TerminologyPackage[0..*]

The nested terminologyPackage contained in a given instance of a TerminologyPackage
terminologyPackageCitation:TerminologyPackageCitation[0..*]

The nested terminologyPackageCitation contained in a given instance of a TerminologyPackage

Semantics

TerminologyPackages contain the TerminologyAssets that can be used within the naming and description of SACM arguments and artefacts. TerminologyPackage elements can be nested, and can contain citations (references) to other TerminologyPackages.

10.4 TerminologyPackageCitation

The TerminlogyPackageCitation is a citation (reference) to another TerminologyPackage.

Superclass

TerminologyElement

Associations

citedPackage: TerminologyPackage[0..1]

The TerminologyPackage being cited by the TerminologyPackageCitation.

Semantics

TerminologyPackageCitations make it possible to cite other TerminologyPackages.For example, within a TerminologyPackage it can be useful to refer to another TerminologyPackage (to reference terminology) that is not contained with the same TerminologyPackage and is defined elsewhere.

Constraints

The citedPackage referred to by a TerminologyPackageCitation must be outside of the containment hierarchy containing the citation.

10.5 TerminologyAsset (abstract)

The TerminologyAsset Class is the abstract class for the different types of terminology elements represented in SACM.

Superclass

TerminologyElement

Semantics

TerminologyAssets represent all of the elements required to model and categorize expressions in SACM (expressions and terminology categories).

10.6 Category

The Category class describes categories of ExpressionElements (Terms and Expressions) and can be used to group these elements within TerminologyPackages.

Superclass

TerminologyAsset

Semantics

Terms and ExpressionElements can be said to belong to Categories. Categories can group Terms, Expressions, or a mixture of both. For example, a Category could be used to describe the terminology associated with a specific assurance standard, project, or system.

10.7 ExpressionElement (abstract)

The ExpressionElement class is the abstract class for the elements in SACM that are necessary for modeling expressions.

Superclass

TerminologyAsset

Associations

category: Category [0..*] – optionally associates the ExpressionElement with one or more terminology categories.

Semantics

ExpressionElements are used to model (potentially structured) expressions in SACM. All ModelElements contain a Description whose value is provided by means of an Expression.

10.8 Expression

The Expression class is used to model both abstract and concrete phrases in SACM. Abstract Expressions are denoted by the inherited isAbstract attribute being set true. A concrete expression (denoted by isAbstract being false) is one that has a literal string value and references only concrete ExpressionElements.

Superclass

ArtefactElement

Attributes

value: String – An attribute recording the value of the expression

Associations

element: ExpressionElement [0..*] – an optional reference to other ExpressionElements forming part of the StructuredExpression.

Semantics

Expressions are used to model phrases and sentences. These are defined using the value attribute. The value attribute can be a simple literal string. Alternatively, the expression can also be defined (using the value string) as a production rule involving other ExpressionElements. In this case, the value string must use a suitable (string) form for denoting the position of involved ExpressionElements (e.g. "$<ExpressionElement.name>$") within the production rule, and expressing production rule operators (e.g. Extended Backus-Naur Form operators).

Constraints

Where an Expression has associated ExpressionElements these should be referenced by name within the Expression.value.

Where an Expression.value references ExpressionElements by name, these ExpressionElements should be associated (using the element association) with Expression.

11 SACM Argumentation Metamodel

11.1 General

This chapter presents the normative specification for the SACM Argumentation Metamodel. It begins with an overview of the metamodel structure followed by a description of each element.

11.2 Argumentation Class Diagram

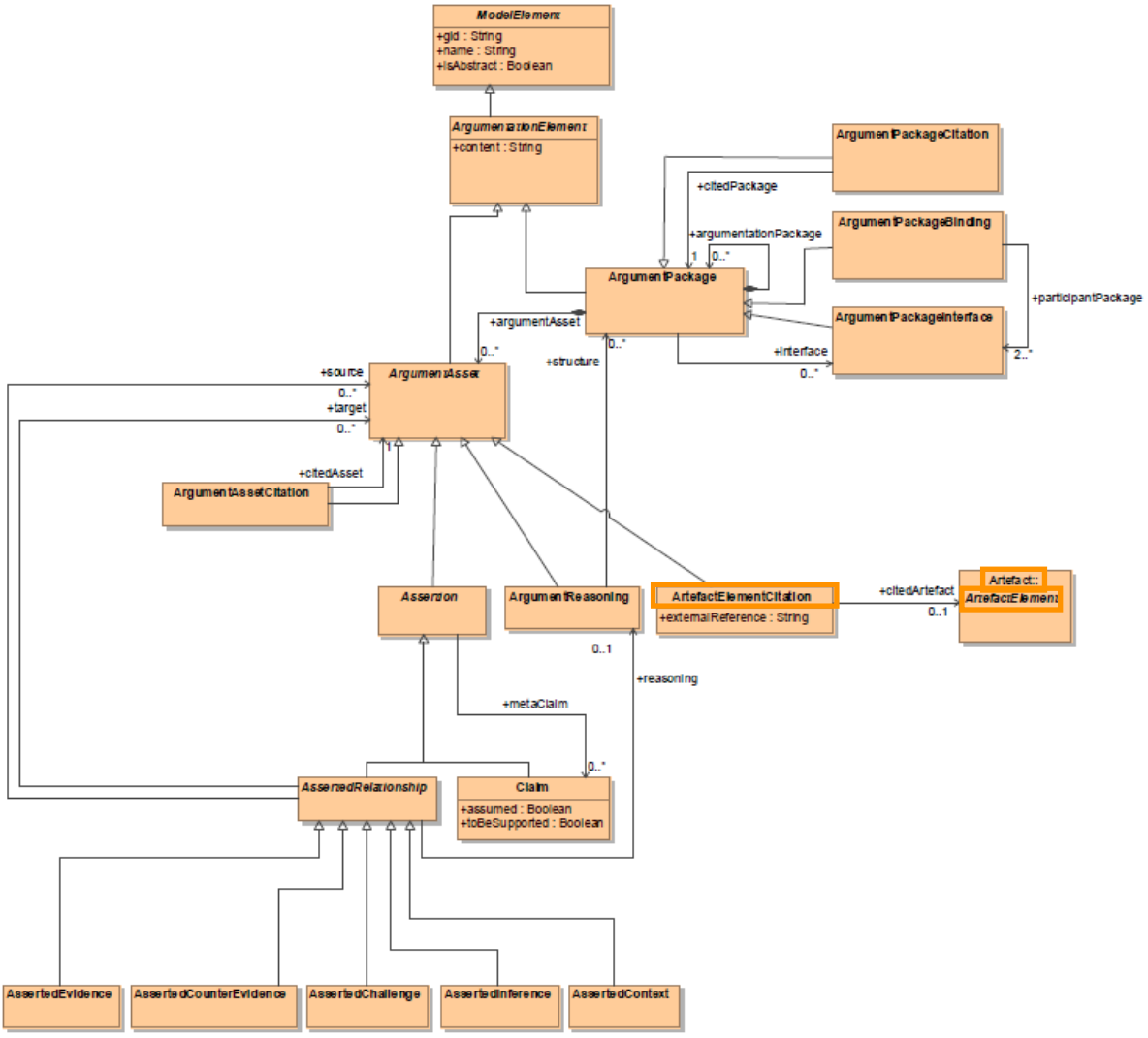


Figure 11.1 - Argumentation Class Diagram

This portion of the SACM model describes and defines the concepts required to model structured arguments. represented in SACM through explicitly representing the Claims and citation of artefacts (e.g., as evidence) (ArtefactElementCitation), and the 'links' between these elements – e.g., how one or more Claims are asserted to Claim, or how one or more artefacts are asserted as providing evidence for a Claim (AssertedEvidence). In core elements, in SACM it is possible to provide additional description of the ArgumentReasoning associated with and evidential relationships, represent counter-arguments (through AssertedChallenge), counter-evidence (through AssertedCounterEvidence), and represent how artefacts provide the context in which arguments should be AssertedContext.)

For example, within an ArgumentPackage it can be useful to refer to another ArgumentPackage that is not contained within the same ArgumentPackage.

Constraints

ArgumentPackageCitations have no contents other than the association to the citedPackage.

The citedPackage referred to by an ArgumentPackageCitation must be outside of the containment hierarchy containing the citation.

11.2.4 ArgumentPackageBinding Class

The ArgumentPackageBinding is a sub type of ArgumentPackage used to record the mapping (agreement) between two or more ArgumentPackages.

Superclass

ArgumentPackage

Associations

participantPackage:ArgumentPackageInterface[2..*]

The ArgumentPackages being mapped together by the ArgumentPackageBinding.

Semantics

ArgumentPackageBindings can be used to map resolved dependencies between the Claims of two or more ArgumentPackages.

For example, one ArgumentPackage may contain a claim that is toBeSupported (i.e. currently has no supporting argument). An ArgumentPackageBinding can be used to record the mapping (by means of containing a structured argument linking ArgumentAssetCitations to the claims in question) between this claim and a supporting claim in another ArgumentPackage.

An ArgumentPackageInterface is a sub type of ArgumentPackage that can be used to create an explicit interface to an existing ArgumentPackage.

Constraints

The ‘root’ ArgumentAssets contained by an ArgumentPackageBinding (i.e. the ArgumentAssets only associated as target of an AssertedRelationship) and ‘leaf’ ArgumentAssets (i.e. the ArgumentAssets only associated as source of an AssertedRelationship) must be ArgumentAssetCitations to Claims or ArtefactElementCitations contained within the ArgumentPackages associated by the participantPackage association.

11.2.5 ArgumentPackageInterface Class

Superclass

ArgumentPackage

Semantics

ArgumentPackageInterfaces can be used to declare (by means of containing ArgumentAssetCitations) the ArgumentAssets contained in an ArgumentPackage that form part of the explicit, declared, interface of the ArgumentPackage.

For example, whilst an ArgumentPackage may contain many Claims, it may be desirable to create an ArgumentPackageInterface that cites only a subset of those claims that are intended to be mapped / used (e.g. by means of an ArgumentPackageBinding) by other ArgumentPackages. There may be more than one ArgumentPackageInterface for a given ArgumentPackage that reveal different aspects of the ArgumentPackage for different audiences.

Constraints

ArgumentPackageInterfaces are only allowed to contain ArgumentAssetCitations to ArgumentAssets within the ArgumentPackage with which this ArgumentPackageInterface is associated (by the interface association).

11.2.6 ArgumentAsset Class (abstract)

The ArgumentAsset Class is the abstract class for the elements of any structured argument represented in SACM.

Superclass

ArgumentationElement

Semantics

ArgumentAssets represent the constituent building blocks of any structured argument contained in an ArgumentPackage.

For example, ArgumentAssets can represent the Claims made within a structured argument contained in an ArgumentPackage.

11.2.7 Assertion Class (abstract)

Assertions are used to record the propositions of Argumentation (including both the Claims about the subject of the argument and the structure of the Argumentation being asserted). Propositions can be true or false, but cannot be true and false simultaneously.

Associations

metaClaim:Claim[0..*]

references Claims concerning (i.e., about) the Assertion (e.g., regarding the confidence in the Assertion)

Semantics

Structured arguments are declared by stating claims, citing evidence and contextual information, and asserting how these elements relate to each other.

11.2.8 ArtefactElementCitation Class

The ArtefactElementCitation Class enables the citation of an artefact that relates to the structured argument.

Superclass

ArgumentAsset

Attributes

externalReference: String An attribute recording a URL to external evidence.

Associations

citedArtefact:ArtefactElement[0..1]

The ArtefactElements cited by the current ArtefactElementCitation object.

Semantics

It is necessary to be able to cite artefacts that provide supporting evidence, context, or additional description for the core reasoning of the recorded argument. ArtefactElementCitations allow there to be an objectified citation of this information within the structured argument, thereby allowing the relationship between this artefact and the argument to also be explicitly declared.

The externalReference attribute can be used when wishing to cite an Artefact not being modeled by an SACM ArtefactElement.

11.2.11 ArgumentReasoning Class

ArgumentReasoning can be used to provide additional description or explanation of the asserted inference or challenge that connects one or more Claims (premises) to another Claim (conclusion). ArgumentReasoning elements are therefore related to AssertedInferences and AssertedChallenges. It is also possible that ArgumentReasoning elements can refer to other structured Arguments as a means of documenting the detail of the argument that establishes the asserted inferences.

Superclass

ReasoningElement

Associations

structure:ArgumentPackage[0..1]

Optional reference to another the ArgumentPackage that provides the detailed structure of the argument being described by the ArgumentReasoning.

Semantics

The AssertedRelationship that relates one or more Claims (premises) to another Claim (conclusion), or evidence cited by an ArtefactElementCitation to a Claim, may not always be obvious. In such cases ArgumentReasoning can be used to provide further description of the reasoning involved.

11.2.12 AssertedRelationship Class (abstract)

The AssertedRelationship Class is the abstract association class that enables the ArgumentAssets of any structured argument to be linked together. The linking together of ArgumentAssets allows a user to declare the relationship that they assert to hold between these elements.

Superclass

Assertion

Associations

source:ArgumentAsset[0..*]

Reference to the ArgumentAsset(s) that are the source (start-point) of the relationship.

target:ArgumentAsset[0..*]

Reference to the ArgumentAsset(s) that are the target (end-point) of the relationship.

reasoning:ArgumentReasoning[0..*]

Reference to the ArgumentReasoning being described by the ArgumentReasoning.

Semantics

In SACM, the structure of an argument is declared through the linking together of primitive ArgumentAssets. For example, a sufficient inference can be asserted to exist between two claims (“Claim A implies Claim B”) or sufficient evidence can be asserted to exist to support a claim (“Claim A is evidenced by Evidence B”). An inference asserted between two claims (A – the source – and B – the target) denotes that the truth of Claim A is said to infer the truth of Claim B.

11.2.13 AssertedInference Class

The AssertedInference association class records the inference that a user declares to exist between one or more Assertion (premises) and another Assertion (conclusion). It is important to note that such a declaration is itself an assertion on behalf of the user.

Superclass

AssertedRelationship

Semantics

The core structure of an argument is declared through the inferences that are asserted to exist between Assertions (e.g., Claims). For example, an AssertedInference can be said to exist between two claims ("Claim A implies Claim B"). An AssertedInference between two claims (A – the source – and B – the target) denotes that the truth of Claim A is said to infer the truth of Claim B.

Constraints

The source of AssertedInference relationships must be Claims, or ArgumentElementCitations that cite a Claim.

The target of AssertedInference relationships must be Assertions, or ArgumentElementCitations that cite an Assertion.

11.2.14 AssertedEvidence Class

The AssertedEvidence association class records the declaration that one or more artefacts of Evidence (cited by ArtefactElementCitations) provide information that helps establish the truth of a Claim. It is important to note that such a declaration is itself an assertion on behalf of the user. The artefact (cited by an ArtefactElementCitation) may provide evidence for more than one Claim.

Superclass

AssertedRelationship

Semantics

Where evidence (cited by ArtefactElementCitation) exists that helps to establish the truth of a Claim in the argument, this relationship between the Claim and the evidence can be asserted by an AssertedEvidence association. An AssertedEvidence association between an artefact cited by an ArtefactElementCitation and a Claim (A – the source evidence cited – and B – the target claim) denotes that the evidence cited by A is said to help establish the truth of Claim B.

Constraints

The source of AssertedEvidence relationships must be ArtefactElementCitation.

The target of AssertedEvidence relationships must be Assertions, or ArgumentElementCitations that cite an Assertion.

11.2.15 AssertedChallenge Class

The AssertedChallenge association class records the challenge (i.e. counter-argument) that a user declares to exist between one or more Claims and another Claim. It is important to note that such a declaration is itself an assertion on behalf of the user.

Superclass

AssertedRelationship

Semantics

An AssertedChallenge by Claim A (source) to Claim B (target) denotes that the truth of Claim A challenges the truth of Claim B (i.e., Claim A leads towards the conclusion that Claim B is false).

Constraints

The source of AssertedChallenge relationships must be Claims, or ArgumentElementCitations that cite a Claim.

The target of AssertedChallenge relationships must be Assertions, or ArgumentElementCitations that cite an Assertion.

11.2.16 AssertedCounterEvidence Class

AssertedCounterEvidence can be used to associate evidence (cited by ArtefactElementCitations) to a Claim, where this

evidence is being asserted to infer that the Claim is false. It is important to note that such a declaration is itself an assertion on behalf of the user.

Superclass

AssertedRelationship

Semantics

An AssertedCounterEvidence association between some evidence cited by an InformationNode and a Claim (A – the source evidence cited – and B – the target claim) denotes that the evidence cited by A is counter-evidence to the truth of Claim B (i.e., Evidence A suggests the conclusion that Claim B is false).

Constraints

The source of AssertedCounterEvidence relationships must be ArtefactElementCitation.

The target of AssertedCounterEvidence relationships must be Assertions, or ArgumentElementCitations that cite an Assertion.

11.2.17 AssertedContext Class

The AssertedContext association class can be used to declare that the artefact cited by an ArtefactElementCitation(s) provides the context for the interpretation and scoping of a Claim or ArgumentReasoning element. In addition, the AssertedContext association class can be used to declare a Claim asserted as necessary context (i.e. a precondition) for another Assertion or ArgumentReasoning.

Superclass

AssertedRelationship

Semantics

Contextual information often needs to be cited in order to make clear the interpretation and scope of a Claim or ArgumentReasoning description. For example, a Claim can be said to be valid only in a defined context ("Claim A is asserted to be true only in a context as defined by the information cited by Artefact B" or conversely "InformationItem B is the asserted context for Claim A"). A declaration (AssertedContext) of context (ArtefactElementCitation B) for a ReasoningElement A records that B is asserted to be contextual information required for the interpretation and scoping of A (i.e., B defines the context where the reasoning presented by A is asserted as true).

Contextual Claims often need to be cited as preconditions for a Claim or ArgumentReasoning. For example, a Claim may be asserted only in the context of another claim ("Claim A is asserted to be true only in a context where Claim B is true". Similarly, a description of ArgumentReasoning A may only be considered true in a context where Claim B is true".

Constraints

The source of AssertedContext relationships must be ArtefactElementCitations or Claims.

The target of AssertedContext relationships must be Assertions, ArgumentElementCitations that cite an Assertion, "ArgumentReasoning" elements or ArgumentElementCitations that cite ArgumentReasoning elements.

Part III - Artefact Metamodel

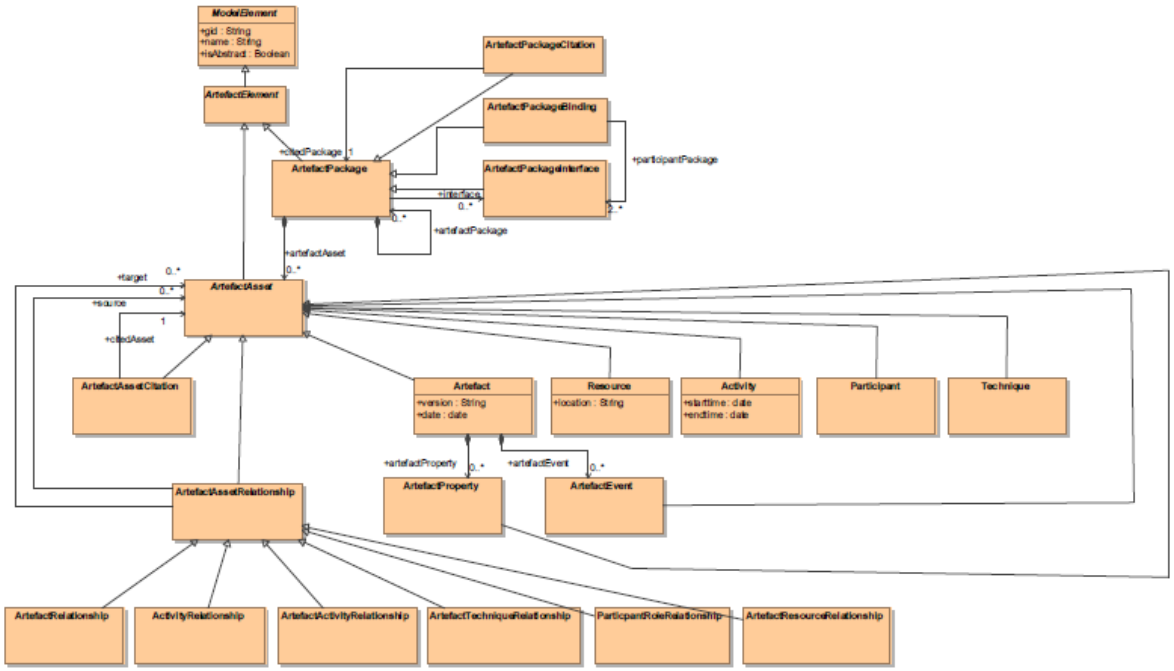
This part of the specification defines the Artefact Metamodel.

12 Artefact Classes

12.1 General

This chapter presents the normative specification for the SACM Artefact Metamodel. It begins with an overview of the metamodel structure followed by a description of each element.

Figure 12.1 - Artefact Class Diagram



Artefacts correspond to the main evidentiary elements of an assurance case. By means of assertions (AssertedEvidence and AssertedCounterEvidence), artefacts are used for supporting claims and arguments.

In general, artefacts are managed when the corresponding objects are available. For example, a test case is linked to the requirement that validates once the test case has already been created. However, artefact management might also require the specification of patterns (or templates) in order to allow a user, for instance, to indicate that a given artefact must be created but it has not yet. A common scenario of this situation corresponds to the process during which a supplier and a certifier have to agree upon the artefacts that the supplier will have to provide as assurance evidence for a system. As a result of this process, artefact patterns could be specified, and such patterns would need to be made concrete during the lifecycle of the system. Artefact patterns are specified by mean of the attribute 'isAbstract' (ModelElement). For example, a supplier and a certifier might agree upon the need for maintaining a hazard log during a system's lifecycle. Such a hazard log would initially be modeled as an Artefact that is abstract. Once created, the value of this attribute of the hazard log would be 'false'. The specification of artefact patterns also facilitates their reuse, as the corresponding artefacts might have to be created in the scope of more than one assurance case effort. Using again hazard logs as an example, their structure might be the same for several systems, thus all the corresponding hazard logs might be based on a same abstract Artefact.

When made concrete, an Artefact can relate to many different types of information necessary for developing confidence in the Artefact and thus for assurance purposes. Such information can be regarded as meta-data or provenance information about an Artefact, provides information about its management, and is specified with the rest of specializations of ArtefactAsset (different to ArtefactAssetCitation). Using a design specification as an example, properties (ArtefactProperty) could be specified regarding its quality (completeness, consistency...), and it would have a

lifecycle with events such as its creation and modifications. The specification could be created by using UML (Technique) in an Activity named 'Specify system design', stored in a Resource corresponding to a diagram created with some modeling tool, and later used as input for another Activity called 'Verify system design'. A given person (Participant) playing the role of system designer could be the owner of the design specification, which would also relate to other artefacts: the requirements specification that satisfies, the architecture that implements, its verification report, etc. Further relationships might be specified between other artefact assets, such precedence between activities ('Specify system design' precedes 'Verify system design') and the participants in an Activity.

12.2 ArtefactPackageCitation

ArtefactPackageCitation is used to cite another ArtefactPackage. The citation can be used where an assurance case author wishes to refer to an existing ArtefactPackage.

Superclass

ArtefactPackage

Associations

citedPackage: ArtefactPackage [1] – the ArtefactPackage cited by the ArtefactPackageCitation

Semantics

ArtefactPackageCitations enable the reference, in a given ArtefactPackage, to another ArtefactPackage.

Constraints

ArtefactPackageCitations have no contents other than the association to the citedPackage.

12.3 ArtefactPackageBinding

The ArtefactPackageBinding is a sub type of ArtefactPackage used to record ArtefactAssetRelationships between the ArtefactAssets of two or more ArtefactPackages.

Superclass

ArtefactPackage

Associations

participantPackage:ArtefactPackageInterface[2..*]

The ArtefactPackages containing the ArtefactAssets being related together by the ArtefactPackageBinding.

Semantics

ArtefactPackageBindings can be used to map dependencies between the cited ArtefactAssets of two or more ArtefactPackages. For example, a binding could be used to record a 'derivedFrom' ArtefactAssetRelationship between the ArtefactAsset of one package to the ArtefactAsset of another.

Contraints

ArtefactPackageBindings must only contain ArtefactAssetRelationships with source and target ArtefactAssetCitations citing ArtefactsAssets contained within the ArtefactPackageInterfaces associated by participantPackage.

12.4 ArtefactPackageInterface

ArtefactPackageInterface is a kind of ArtefactPackage that defines an interface that may be exchanged between users. A typical use case might be for a component supplier to provide its customers with ArtefactPackageInterfaces that contain the relevant supplier's ArtefactElements for the customers' ArtefactPackages. An ArtfefactPackage may also declare

that it implements or conforms to a particular ArtefactPackageInterface.

Superclass

ArtefactPackage

Associations

artefactPackageCitation: ArtefactPackageCitation [0..*] – an optional set of citations to other ArtefactPackage elements

artefactAsset: ArtefactAsset [0..*] – an optional set of ArtefactAsset elements, such as citations, artefacts, resources, activities, etc.

artefactPackage: ArtefactPackage [0..*] - an optional set of contained ArtefactPackage elements, allowing for recursive containment.

Semantics

ArtefactPackageInterface enables the declaration of the elements of an ArtefactPackage that might be referred to (cited) in another ArtefactPackage, thus the elements can be used for assurance in the scope of the latter ArtefactPackage.

Constraints

ArtefactPackageInterfaces are only allowed to contain ArtefactAssetCitations to ArtefactAssets within the ArtefactPackage with which this ArtefactPackageInterface is associated (by the interface association).

12.5 ArtefactAsset class (abstract)

The ArtefactAsset class represents the artefact-specific pieces of information of an assurance case, in contrast to the argument-specific pieces of information.

Superclass

ArtefactElement

Semantics

Information about artefacts is essential for any assurance case. The artefacts correspond, for instance, to the evidence provided in support of the arguments and claims of an assurance case. It is also important to have access to related pieces of information such as the provenance of an artefact, its lifecycle, and its properties. All this information might have to be consulted for developing confidence in the validity of an assurance case.

12.5.1 ArtefactAssetCitation class

The ArtefactAssetCitation class allows an ArtefactPackage to refer to the components of another ArtefactPackage.

Superclass

ArtefactAsset

Associations

citedAsset:ArtefactAsset[1]

The ArtefactAsset that the ArtefactAssetCitation cites

Constraints

The citedAsset of an ArtefactAssetCitation must be part of an ArtefactPackageInterface.

The citedAsset of an ArtefactAssetCitation must be part of a different ArtefactPackage.

The citedAsset of an ArtefactAssetCitation cannot be an ArtefactAssetCitation.

The citedAsset of an ArtefactAssetCitation cannot be an ArtefactAssetRelationship.

Semantics

ArtefactAssets belong to single ArtefactPackages. Nonetheless, the ArtefactAssets can be referred to in other ArtefactPackages in order to, for instance, specify that a relationship exists between ArtefactAssets of different ArtefactPackages. For example, an ArtefactPackage might be specified for all the V&V results of an assurance case, and another for the requirements specifications. The fist ArtefactPackage might refer to the second for further specifying that a given V&V result corresponds to the validation of a given requirement.

12.5.2 Artefact class

The Artefact class represents the distinguishable units of data used in an assurance case.

Superclass

ArtefactAsset

Attributes

version: String
The version of the Artefact

date: Date
The date on which the artefact was created.

Associations

artefactProperty::ArtefactProperty[0..*]
The ArtefactProperties of the Artefact

artefactEvent::ArtefactEvent[0..*]
The set of ArtefactEvents that represent the lifecycle of the Artefact

Semantics

Artefacts correspond to the main evidentiary support for the arguments and claims of an assurance case: an Artefact can play the role of evidence of a Claim (AssertedEvidence), or of counterevidence (AssertedCountedEvidence). An Artefact can take several forms, such as a diagram, a plan, a report, or a specification, both in electronic (e.g., a pdf file) or physical (e.g., a paper document) formats. Typical examples of Artefacts include system lifecycle plans, dependability (e.g., safety) analysis results, system specifications, and V&V results.

12.5.3 ArtefactProperty class

The ArtefactProperty class enables the specification of the characteristics of an Artefact.

Semantics

An Artefact can have different, specific characteristics independent of the argumentation structure in which the Artefact is used. Some can be objective (e.g., the result of a test case execution, as passed or not passed) and others can be based on a person's judgement (e.g., regarding a quality aspect of a report).

12.5.4 ArtefactEvent class

The ArtefactEvent class enables the specification of the events in the lifecycle of an Artefact.

Attributes

date: Date
The date on which the ArtefactEvent occurred.

Semantics

Artefacts change during their lifecycle, and different types of happenings can occur at different moments: creation, modification, revocation... ArtefactEvents serve to maintain a history log of an Artefact, and can be consulted to know how an Artefact has evolved and to develop confidence in its adequate management.

12.5.5 Resource class

The Resource class corresponds to the tangible objects representing an Artefact.

Superclass

ArtefactAsset

Attributes

location: String
The path or URL specifying the location of the Resource.

Semantics

Artefacts are located and accessible somewhere, usually in the form of some electronic file for an assurance case. Such information is specified by means of Resources.

12.5.6 Activity class

The Activity class represents units of work related to the management of ArtefactAssets.

Superclass

ArtefactAsset

Attributes

startTime: Date
Time when the Activity started.

endTime: Date
Time when the Activity ended.

Semantics

The Artefacts used in an assurance case are the result of and managed via the execution of processes, which consist of Activities: specification of requirements, design of the system, integration of system components, etc. ArtefactActivityRelationships can be used to specify the relationship between Activities and Artefacts. Activities can, for instance, be described as using a given Artefact as input or producing an Artefact as output. Activities can be related to one another using ActivityRelationships (e.g., 'preceding'). The purpose of an activity can be specified in its description.

12.5.7 Technique class

The Technique class represents techniques associated with Artefacts (e.g., associated with the creation, inspection, review or analysis of an Artefact).

Superclass

ArtefactAsset

Semantics

Artefacts are created, or managed from a more general perspective, via some method whose use results in specific

characteristics for the Artefacts. For example, the use of UML (as a Technique) for designing a system results in a design specification with a set of UML diagrams that could represent static and dynamic internal aspects of the system.

12.5.8 Participant class

The Participant class enables the specification of the parties involved in the management of ArtefactAssets.

Superclass

ArtefactAsset

Semantics

Different parties can participate in an assurance case effort, such as specific people, organizations, and tools.

12.5.9 ArtefactAssetRelationship class

The ArtefactAssetRelationship class enables the ArtefactAssets of an AssuranceCase to be linked together. The linking together of ArtefactAssets allows a user to specify that a relationship exists between the assets.

Superclass

ArtefactAsset

Associations

source:ArtefactAsset[0..*]
The source of the ArtefactRelationship

target:ArtefactAsset[0..*]
The target of the ArtefactRelationship

Constraints

The source or target of an ArtefactAssetRelationship cannot be another ArtefactAssetRelationship.

Semantics

An ArtefactAsset can be related to other ArtefactAssets. This kind of information is specified by means of ArtefactAssetRelationships, which can also have a specific type depending on the ArtefactAssets being linked together.

12.5.10 ArtefactRelationship class

The ArtefactRelationship class enables two Artefacts to be linked together.

Superclass

ArtefactAssetRelationship

Constraints

The source and target of an ArtefactRelationhsip must be Artefacts, or ArtefactAssetCitations citing an Artefact.

Semantics

The Artefacts managed during a system's lifecycle do not exist in isolation, but relationships typically exist between them: the test cases that validate some requirement, the design standard followed in a design specification, etc. These relationships are specified by means of ArtefactRelationships.

12.5.11 ActivityRelationship class

The ActivityRelationship class enables two Activities to be related together.

Superclass

ArtefactAssetRelationship

Constraints

The source and target of an ActivityRelationship must be Activities or ArtefactAssetCitations citing an Activity.

Semantics

ActivityRelationships aim to support the specification of how Activities, and citations to them, relate each other: an Activity that precedes another, an Activity decomposed into others, etc.

12.5.12 ArtefactActivityRelationship class

The ArtefactActivityRelationships class enables an Artefact and an Activity to be linked together.

Superclass

ArtefactAssetRelationship

Constraints

The source of an ArtefactActivityRelationship must be an Artefact, or an ArtefactAssetCitation citing an Artefact.

The target of an ArtefactActivityRelationship must be an Activity, or an ArtefactAssetCitation citing an Activity.

Semantics

Artefacts are managed in the scope of Activities, which usually use the Artefact as input and output. Such information is specified by means of ArtefactActivityRelationships.

12.5.13 ArtefactTechniqueRelationship class

The ArtefactTechniqueRelationship class enables an Artefact and a Technique to be linked together.

Superclass

ArtefactAssetRelationship

Constraints

The source of an ArtefactActivityRelationship must be an Artefact, or an ArtefactAssetCitation citing an Artefact.

The target of an ArtefactActivityRelationship must be a Technique, or an ArtefactAssetCitation citing a Technique.

Semantics

Artefacts result from the application of Techniques, such as the application of UML for a design specification. ArtefactTechniqueRelationships are used to specify such a kind of information.

12.5.14 ParticipantRoleRelationship class

The ParticipantRoleRelationships class enables a Participant to be linked to other ArtefactAssets.

Superclass

ArtefactAssetRelationship

Constraints

The source of an ParticipantRoleRelationship must be a Participant or an ArtefactAssetCitation citing a Participant.

Semantics

The information about the roles and functions that a Participant plays with regard to other ArtefactAssets is specified by means of ParticipantRoleRelationships. Examples of roles and functions include the owner of an Artefact, the executor of an Activity, and possible relationships between Participants (e.g., supervisor).

12.5.15 ArtefactResourceRelationship class

The ArtefactResourceRelationship class enables an Artefact and a Resource to be linked together.

Superclass

ArtefactAssetRelationship

Constraints

The source of an ArtefactActivityRelationship must be an Artefact, or an ArtefactAssetCitation citing an Artefact.

The target of an ArtefactActivityRelationship must be a Resource, or an ArtefactAssetCitation citing a Resource.

Semantics

The specific Resources where an Artefact is located are specified by means of ArtefactResourceRelationships.