

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 defensible 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.

assurance claim (a claim possibly may be a requirement of the system)

assessors

assurance

specific desired assurance

, constrained, or formal

Classically, systems are desired to be assured as secure or safe, but a system may also need to be assured of other critical properties.

1.2 Structured Arguments and Assessments

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.

The Argument Metamodel assurance case developer and / or Terminology parts of the

This specification is designed to stand alone, or may be used in combination with the SACM Artifact Metamodel. The Artifact 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 Argument Metamodel 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

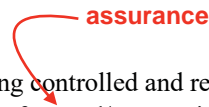
Part of this specification provides a metamodel for communicating the way in which evidence artifacts are collected by various participants using techniques, resources and activities. This allows users to build a repository of evidence that communicates its provenance and how it was gathered. This Artifact Metamodel identifies the main elements that determine the evidence collection process: artifacts, participants, resources, activities and techniques.

Artifacts may be exchanged as packages or combined into composites.

The SACM Artifact 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 Artifact 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 artifacts, 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 Controlled Vocabulary



Part of this specification provides a metamodel for defining controlled and reusable vocabulary used in the argumentation of an Assurance Case. In the argument of ~~safety and/or security of~~ a system, a set of vocabulary is often referred to repeatedly. Thus, it is important to ensure that the usage of vocabulary refer to the terms with the same semantics. In addition, for model based system assurance, such vocabularies can relate to external heterogeneous models that define the semantics for the terms. Therefore, controlled vocabulary ensures the consistency of the semantics of the terms used in the argumentation, and provides a mean to define the terms used in the argumentation through external models (e.g. standard models).

The SACM Terminology Metamodel defines a number of elements for constructing and interchanging packages of terminologies, to ensure the consistency of semantics.

1.5 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.

In SACM 2.4, updates, restructuring, and additions to the SACM 2.3 model have been made as noted issues. With these changes SACM 2.4:

- is a "standard complete" MOF model
- is usable by a wider set of demographics (including System Engineers)
- is usable by people using their favorite tools (e.g. Profiles)
- is embeddable in other modeling languages
- is able to reference other models (and elements within those models)
- is based on UML and KerML (the basis of SysML2) concepts
- provides tighter semantics in the metamodels to replace unneeded OCL statements

the existing notations in the domain such as GSN and CAE are also considered in the design process. In SACM 2.2, minor clarifications, expanded explanations for enhanced readability and consistency have been incorporated along with a new annex to provide additional details about the concepts of package interfaces and bindings and their use within the standard.

In SACM 2.3, a new normative annex and corresponding compliance point to provide a UML profile allowing UML-based tools to work with SACM concepts and the allowing for categories to contain categories as a mechanism for organizing concepts in the Terminology class.

- provides strength weights for defining the strength of the relationships between argument elements
- provides various Assurance Case Diagram types
- widens the applicability of various elements in the model (e.g. Properties)
- is more specific about packaging semantics
- now supports all concepts from GSN, CAE, Assurance 2.0
- supports Architecture Views (see GSN)
- supports other types of logics (e.g. Inductive, Deductive, Abductive, other)
- supports (multiple conflicting) Assessments of an Assurance Case
- supports Joins (compare with GSN)

2 Conformance

2.1 Introduction

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

- Argumentation Model
- Artifact Model
- Assurance Case Model
- Terminology Model
- SACM UML Profile

2.2 Argumentation Model compliance point

Software that conforms to the SACM specification at the Argumentation 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 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 Model compliance point does not entail support for the Evidence subpackage of SACM, or the terminology sub package of the SACM.

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 Artifact Model compliance point

Software that conforms to the specification at the Artifact 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 Artifact 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 Artifact::ArtifactPackage element of the SACM.

Conformance to the Artifact 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 Model compliance point

This compliance point is mandatory. Software that conforms to the specification at the Assurance Case Model compliance point shall be able to import and export XMI documents that conform with the SACM XML Schema

8.2 Element [Abstract Class]

An Element is a constituent of a model. As such, it has the capability of owning other Elements.

Association Ends

/ownedElement : Element [0..*] {union,readOnly}

The Elements owned by this Element.

~~**/owner : Element [0..1] {union,readOnly}**~~

~~*The Element that owns this Element.*~~

Constraints

not_own_self

An element may not directly or indirectly own itself.

inv: not self->closure(e | e->ownedElement)->includes(self)

8.3 NamedElement [Abstract Class]

A NamedElement is an Element in a model that may have a name.

Attributes

name : String [0..1]

The name of the NamedElement.

/qualifiedName : String [0..1] {readOnly}

A name that allows the NamedElement to be identified within a hierarchy of nested Namespaces. It is constructed from the names of the containing Namespaces starting at the root of the hierarchy and ending with the name of the NamedElement itself. (Separator of names is "::".)

visibility : VisibilityKind [0..1]

Determines whether and how the NamedElement is visible outside its owning Namespace.

~~Association Ends~~

~~**/namespace : Namespace [0..1] {union}**~~

~~*Specifies the Namespace that owns the NamedElement.*~~

Constraints

visibility_needs_ownership

If a NamedElement is owned by something other than a Namespace, it does not have a visibility.

inv: (namespace = null and owner <> null) implies visibility = null

8.4 Namespace [Abstract Class]

A Namespace is an Element in a model that owns and/or imports a set of NamedElements that can be identified by name.

Association Ends

{union,readOnly,subsets ownedElement}
/ownedMember : NamedElement [0..*]{union, subsets Element::ownedElement}

A collection of NamedElements owned by the Namespace.

8.5 Package [Class]

A package can have one or more profile applications to indicate which profiles have been applied. Because a profile is a package, it is possible to apply a profile not only to packages, but also to profiles.

attributes

{subsets ownedElement}
packageableElement : PackageableElement [0..*]{subsets Namespace::ownedMember}

Specifies the packageable elements that are owned by this Package.

8.6 PackageableElement [Abstract Class]

A PackageableElement is a NamedElement that may be owned directly by a Package.

Attributes

visibility : VisibilityKind [0..1] = public {redefines visibility}

A PackageableElement must have a visibility specified if it is owned by a Namespace. The default visibility is public.

VisibilityKind [Enumeration]

VisibilityKind is an enumeration type that defines literals to determine the visibility of Elements in a model.

Literals

public

A Named Element with public visibility is visible to all elements that can access the contents of the Namespace that owns it.

private

A NamedElement with private visibility is only visible inside the Namespace that owns it.

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

~~9.8.11~~ 8 **Note** SACMComment

This class specifies a generic note that may be associated with ~~a ModelElement~~ ^{an Element}. For example a note may include a number of explanatory comments.

Superclass BaseElement
~~SACMCoreElement~~
~~UtilityElement~~

Semantics
~~SACMComments~~
~~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.

Attributes body:ExpressionLangString[0..1]{redefines elementName}
~~body:ExpressionLangString[0..1]~~
 AssociationEnds
 annotatedElement:Element[0..*] – Elements to which the SACMComment applies

~~9.8.12~~ 9 **TaggedValue** NamedValue

This class represents ~~a simple key/value pair~~ ^{attributes} 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 BaseElement, Foundation::Namespace
~~BaseElement~~
~~UtilityElement~~
 Attributes

Associations
~~value :String [0..*] {ordered, nonunique}~~
~~key:MultiLangString[1] (composition) – the key of the TaggedValue.~~
 Associations
 ownedNamedValue [0..*] {subsets ownedMember} - NamedValues owned by this NamedValue

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

9.10 Group

Group can be used to associate a number of SACMElements to a common group (e.g., representing a common type or purpose, or being of interest to a particular stakeholder).

Supertype SACMElement, SACMModel
~~SACMElement~~
 BaseElement
 ownedSACMElement [0..*] {subsets ownedMember, subsets member} - SACMElements that are owned by this Group

Association End
~~element: SACMElement [0..*] {ordered} – a collection of SACMElements that comprise a group~~
~~member:SACMElement [0..*] {redefines element, ordered} - a collection of SACMElements that compromise a group~~

~~9.11~~ **9.10 IRI**

IRI is a international resource identifier is a String and is formatted according to RFC 3987.

Supertype String
~~String~~
 Foundation::String

Supertype ModelElement
 ModelElement

Attributes
 isPattern : Boolean [0..1] = false - if true elements in SACMModel are part of a pattern.
 isModel : Boolean [0..1] = false - if true elements in SACMModel are part of a model.

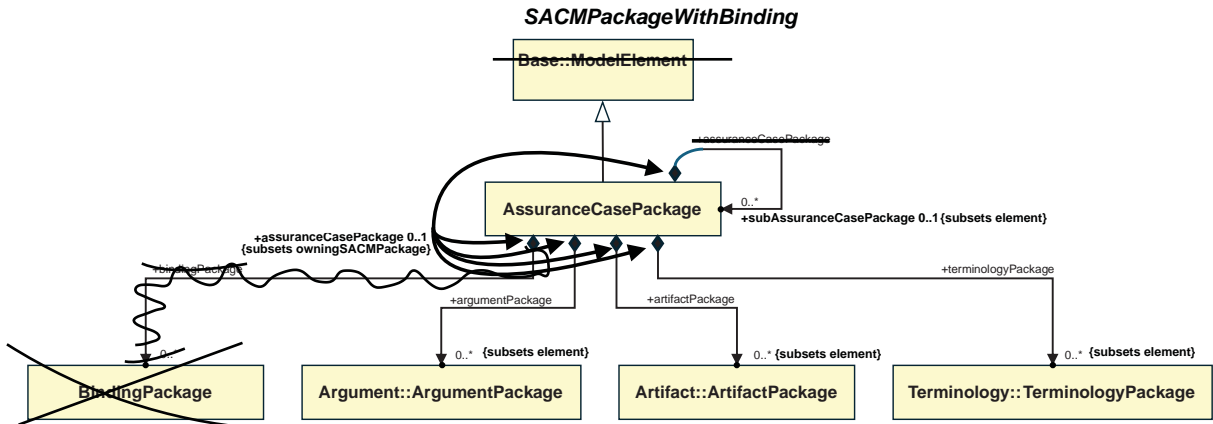
Association
 +subModel:SACMModel [0..*] - subModel is a recursive containment of elements in the model.
 +elements:SACMElement [0..*] - elements are items in the model. **{ordered}**

11
~~10~~ 9
 11
~~10~~ 9.1

Structured Assurance Case Packages

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.



Packages, which are specializations of SACMModel, can set isModel=true to indicate that what is contain in the Package should be considered a completed model. If isPattern=true, then what is in the package should be considered a pattern.

Figure 9.1 - Structured Assurance Case Packages Class Diagram

In SACM, the parent container element is AssuranceCasePackage. AssuranceCasePackages 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 (contained in ArgumentPackages), evidence descriptions (contained in ArtifactPackages) and Terminology definitions (contained in TerminologyPackages).

11 ~~10~~ 9.2 AssuranceCasePackage

AssuranceCasePackage is an exchangeable element that may contain a mixture of artifacts, 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 Base::ArtifactElement. Semantically an AssuranceCasePackage can be considered as an artifact of evidence (e.g., from the perspective of another AssuranceCasePackage).~~

Superclass

~~Base::ModelElement SACMPackageWithBinding
 Base::ArtifactElement~~

Associations

~~subAssuranceCasePackage {subsets element}
 assuranceCasePackage: AssuranceCasePackage [0..*] (composition) – a collection of optional sub-packages
 interface: AssuranceCasePackageInterface [0..*] – a number of optional assurance case package interfaces that the current package may implement~~

12
~~11~~ 10

Structured Assurance Case Terminology Classes

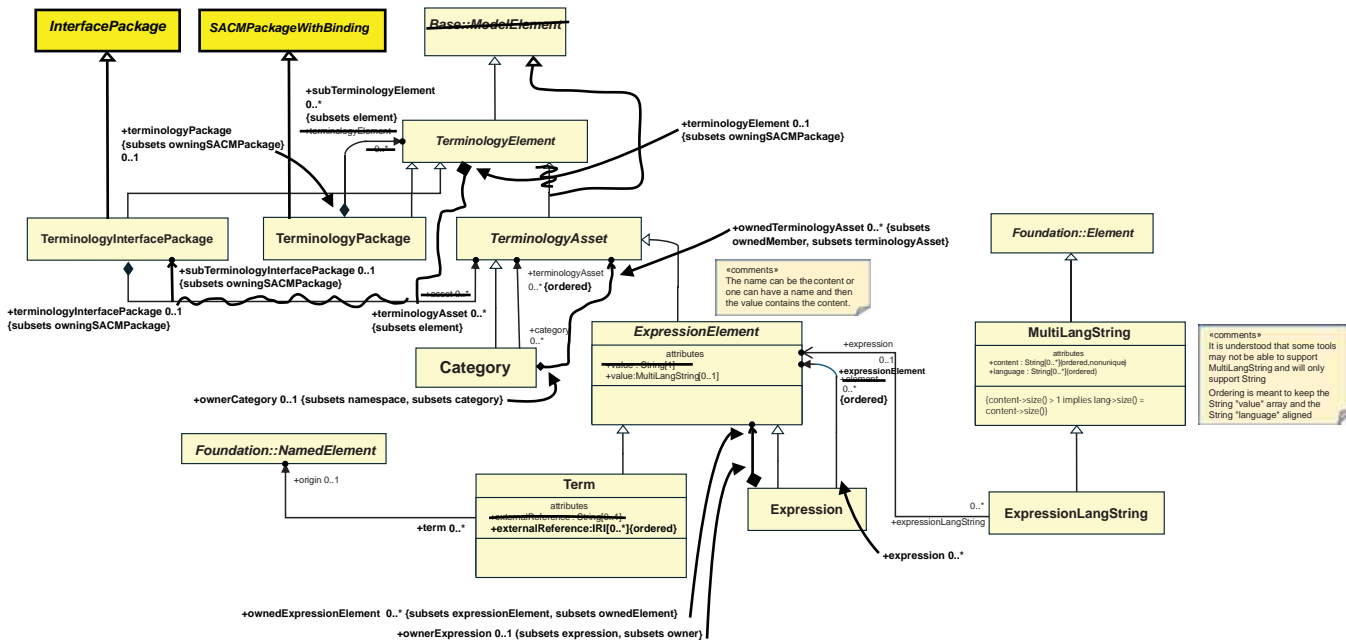
12
~~11~~ 10.1

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.~~

TerminologyElement is an abstract class that serves as a parent class for all packaging in Terminology (TerminologyPackage and TerminologyInterfacePackage).

TerminologyConcept



12
~~11~~ 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.

12
~~11~~ 10.2

TerminologyElement (abstract)

~~TerminologyElement is an abstract class that serves as a parent class for all SACM terminology assets (TerminologyAsset) and the grouping of TerminologyElements (TerminologyGroup). TerminologyElement extends Base::ArtifactElement, this implies that all elements in the Terminology package are artifacts.~~

~~Superclass~~
~~Base::ModelElement~~ TerminologyConcept

TerminologyElement is an abstract class that serves as a parent class for all packaging in Terminology (TerminologyPackage and TerminologyInterfacePackage).

~~Base::ArtifactElement~~
~~Semantics~~ Association
terminologyAsset [0..*] {subsets element} - TerminologyAssets that are contained in this TerminologyElement

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

10.3 TerminologyGroup

TerminologyGroup can be used to associate a number of TerminologyElements to a common group (e.g., representing a common type or purpose, or being of interest to a particular stakeholder).

Superclass

TerminologyElement

Associations

terminologyElement[0..*] – an optional collection of TerminologyElements that are organised within the TerminologyGroup.

Semantics

TerminologyGroup can be used to associate a number of TerminologyElements to a common group (e.g. representing a common type or purpose, or being of interest to a particular stakeholder). The name and the description of the TerminologyGroup should provide the semantic for understanding the TerminologyGroup. TerminologyGroups serve no structural purpose in the formation of the argument network, nor are they meant as a structural packaging mechanism (this should be done using TerminologyPackages).

10.4 TerminologyPackage

The TerminologyPackage is the container element for SACM terminology assets.

Superclass

TerminologyElement

Associations

~~TerminologyElement:TerminologyElement[0..*] (composition) – TerminologyElements contained in the TerminologyPackage, it can be either TerminologyPackage (and its sub-types) or TerminologyAssets (or its sub-types).~~

Semantics

~~← subTerminologyElement:TerminologyElement[0..*] {subsets element} - subTerminologyElement contained in the TerminologyPackage~~

TerminologyPackage contains the TerminologyElements that can be used within the naming and description of SACM arguments and artifacts. TerminologyPackages can be nested.

10.5 TerminologyPackageInterfacePackage

TerminologyPackageInterface is a kind of TerminologyPackage that defines an interface that may be exchanged between users. An TerminologyPackage may declare one or more TerminologyPackageInterfaces.

Superclass

TerminologyElement

Associations

~~implements:TerminologyPackage[1] – the TerminologyPackage that the TerminologyPackageInterface declares.~~

Semantics

~~← subTerminologyInterfacePackage:TerminologyInterfacePackage[0..*] {subsets element}~~

TerminologyPackageInterface enables the declaration of the elements of an TerminologyPackage that might be referred to (cited) in another TerminologyPackage, thus the elements can be used for assurance in the scope of the latter AssuranceCasePackage. A TerminologyPackageInterface resides inside the TerminologyPackage to which it refers. It refers to TerminologyElements using isCitation=True that reside within the same TerminologyPackage as itself.

11.2 ArgumentGroup

ArgumentGroup can be used to associate a number of ArgumentElements to a common group (e.g., representing a common type or purpose, or being of interest to a particular stakeholder).

Superclass
ArgumentAsset
~~ArgumentationElement~~

Associations

argumentAsset:ArgumentAsset[0..*]{ordered}
~~argumentationElement:ArgumentationElement[0..*]~~ as optional collection of ~~ArgumentationElements~~ ArgumentAssets organised within the ArgumentGroup.

Semantics

ArgumentGroup can be used to associate a number of ArgumentElements to a common group (e.g., representing a common type or purpose, or being of interest to a particular stakeholder). The name and the description of the ArgumentGroup should provide the semantic for understanding the ArgumentGroup. ~~ArgumentGroups serve no structural purpose in the formation of the argument network, nor are they meant as a structural packaging mechanism (this should be done using ArgumentPackages).~~

13
12

11.3 ArgumentationElement (abstract)

~~An ArgumentationElement is the top level element of the hierarchy for argumentation elements. ArgumentationElement extends Base::ArtifactElement. Subsequently, all argument elements are considered artifacts.~~

Superclass
Base::ModelElement
Base::ArtifactElement

ArgumentConcept

ArgumentElement is an abstract class that serves as a parent class for all packaging in Argument (ArgumentPackage and ArgumentInterfacePackage).

Semantics

Association
argumentAsset [0..*] (subsets element) - ArgumentAssets contained in this ArgumentElement

~~The ArgumentationElement is a common class for all elements within a structured argument.~~

13
12

11.4 ArgumentPackage Class

ArgumentPackage is the containing element for a structured argument represented using the SACM ~~Argumentation~~ Metamodel.

Superclass

ArgumentationElement

Associations

argumentationElement:ArgumentationElement[0..*] (composition) – a collection of ArgumentationElements forming a structured argument

subArgumentElement [0..*] (subsets element) - ArgumentElements contained in this ArgumentPackage

Semantics

ArgumentPackages contain structured arguments. These arguments are composed of ArgumentAssets. ArgumentPackages elements can also be nested.

Constraints

If an ArgumentPackage has nested ArgumentPackages, then it is only allowed to contain ArgumentPackages.

12
11

11.5 ArgumentPackageBinding

ArgumentBindingPackage

ArgumentBindingPackages

ArgumentBindingPackage

ArgumentElement within the ArgumentPackage can be bound together by means of ArgumentPackageBinding. An ArgumentPackage can provide ArgumentPackageInterfaces, which export ArgumentationElements to be used by other ArgumentPackages. ArgumentPackageInterfaces contain citations to ArgumentationElements (e.g. an ArgumentPackageInterface may contain an 'asCited' Claim, with its 'citedElement' pointing to the Claim inside the

Superclass

ArgumentAsset

~~Associations~~

~~referencedArtifactElement:Base::ArtifactElement[0..*] - reference to a collection of ArtifactElements.~~

Semantics

It is necessary to be able to cite artifacts that provide supporting evidence, context, or additional description within an argument structure. ArtifactReferences allow there to be an objectified citation of this information within the structured argument, thereby allowing the relationship between this artifact and the argument to also be explicitly declared.

13 9 12 11.10 Assertion (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.

Superclass

ArgumentAsset

Attributes

AssertionDeclarationKind[0..1]

assertionDeclaration:AssertionDeclaration[1] = asserted – the declaration indicating the state of the Assertion.

~~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.

13 10 12 11.11 Claim

Claims are used to record the propositions of any structured argument contained in an ArgumentPackage. Propositions are instances of statements that could be true or false, but cannot be true and false simultaneously.

Superclass

Assertion

Attributes

value:MultiLangString[0..1]

Semantics

The core of any argument is a series of claims (premises) that are asserted to provide sufficient reasoning to support a (higher-level) claim (a conclusion). The name can be the content or one can have a name and then the value contains the content.

The core of any argument is a series of claims (premises) that are asserted to provide sufficient reasoning to support a (higher-level) claim (a conclusion).

A Claim that is intentionally declared without any supporting evidence or argumentation can be declared as being assumed (i.e., assertionDeclared = assumed). It is an assumption. However, it should be noted that a Claim that is not 'assumed' (i.e., assertionDeclared = asserted) is not being declared as false. However, there is the expectation of the provision of a supporting argument structure (e.g., it may represent part of an incomplete structure).

A Claim that is intentionally declared as requiring further evidence or argumentation can be denoted by setting +assertionDeclaration to "needsSupport".

A Claim that is being declared as axiomatically true can be denoted by setting +assertionDeclaration to "axiomatic".

A Claim that is defeated by counter evidence or counter argument can be denoted by setting +assertionDeclaration to "defeated".

A Claim which cites another claim and supported by the cited claim can be denoted by setting +assertionDeclaration to "asCited".

The name can be the content or one can have a name and then the value contains the content.

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.

13 14 12 11.15 ~~15~~ AssertedEvidence

AssertedEvidence association records the declaration that one or more artifacts of Evidence (cited by ArtifactReference) 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 artifact (cited by an ArtifactReference) may provide evidence for more than one Claim.

Superclass

AssertedRelationship

Semantics

Associations

+assertion:Assertion [1] - the assertion is that the evidence for this Assertion is enough to make the AssertedEvidence undefeated.
+evidence:ArtifactReference [1] - the assertion is that the evidence for this Assertion is enough to make the AssertedEvidence undefeated.

Where evidence (cited by ArtifactReference) 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 artifact cited by an ArtifactReference 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 ArtifactReference.

OCL:

self.source->forall(s|s.oclIsTypeOf(ArtifactReference))

AssertedContext can be used to declare that the Artifact or ArtifactReference provides the context for the interpretation and scoping of an ArgumentAsset (ArtifactReference, Artifact, Claim, ArgumentReasoning, or any AssertedRelationship specialization element).

13 15 12 11.16 ~~15~~ AssertedContext

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

Superclass

AssertedRelationship

Semantics

Associations

+context:ArtifactReference [1] - the context provides further clarification or constraint of the contexttable
+contexttable:ArgumentAsset [1] - the context provides further clarification or constraint of the contexttable

~~Contextual information often needs to be cited in order to make clear the interpretation and scope of an Assertion and supporting argumentation. 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 ArtifactReference B or conversely ArtifactReference B is the asserted context for Claim A”).~~

~~Contextual Claims often need to be cited as preconditions for an Assertion. 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”).~~

13 16 12 11.17 ~~17~~ AssertedArtifactSupport

~~AssertedArtifactSupport records the assertion that~~

Superclass

AssertedRelationship

Semantics

~~The truth of the assertions associated with an artifact are supported by the assertions that are associated with one or more other artifacts. Note: this can be an ambiguous relationship if the nature of these Assertions is unclear. In such cases, it would be clearer to declare explicit AssertedInferences between Claims drawn out from the ArtifactReference.~~

Constraints

~~The source and target of AssertedArtifactSupport must be of type ArtifactReference.~~

Contextual information often needs to be cited in order to make clear the interpretation and scope of an ArgumentAsset. 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 ArtifactReference B or conversely ArtifactReference B is the asserted context for Claim A”).

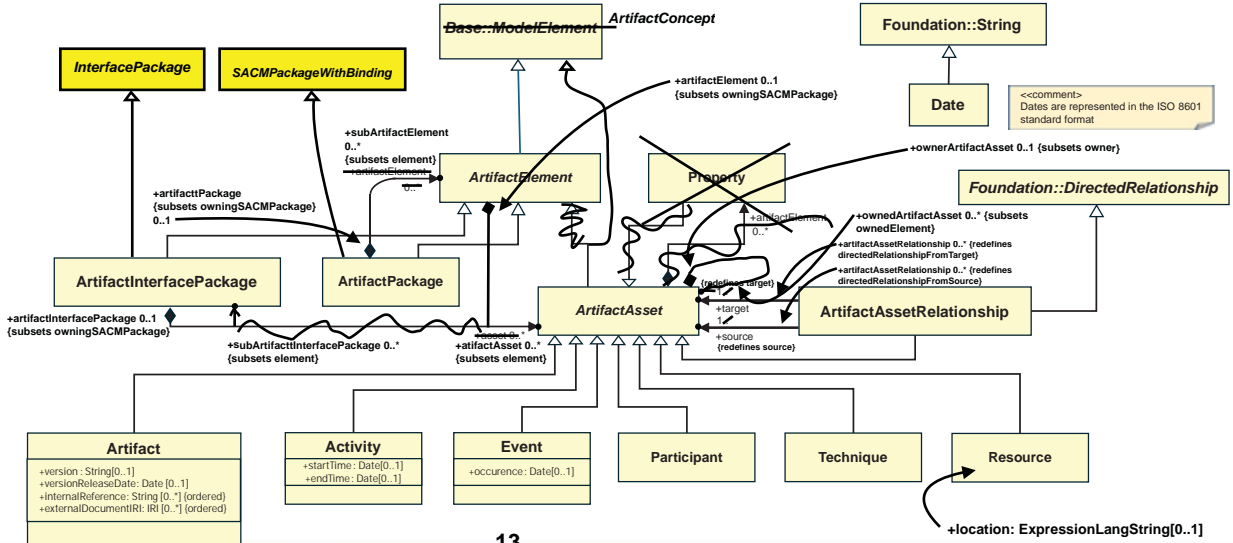
13 17 12 11.18 ~~18~~ AssertedArtifactContext

~~AssertedArtifactContext records the assertion that one or more artifacts provide context for another artifact.~~

Artifact Classes

General

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



13
Figure 13.1 - Artifact Package Diagram

'isSACMAbstract'

Artifacts correspond to the main evidentiary elements of an assurance case. By means of assertions (AssertedEvidence with isCounter = true/false), artifacts can be referenced (using ArtifactReferences) as supporting claims and arguments.

In general, artifacts 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, artifact management might also require the specification of patterns (or templates) in order to allow a user, for instance, to indicate that a given artifact 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 artifacts that the supplier will have to provide as assurance evidence for a system. As a result of this process, artifact patterns could be specified, and such patterns would need to be made concrete during the lifecycle of the system. Artifact patterns are specified by means of the attribute **'isAbstract'** (SACMElement). 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 Artifact that is abstract. Once created, the value of this attribute of the hazard log would be 'false'. The specification of artifact patterns also facilitates their reuse, as the corresponding artifacts 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 Artifact.

When made concrete, an Artifact can relate to many different types of information necessary for developing confidence in the Artifact and thus for assurance purposes. Such information can be regarded as meta-data or provenance information about an Artifact, provides information about its management, and is specified with the rest of specializations of ArtifactAsset. Using a design specification as an example, properties (Property) 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 artifacts: the requirements specification that satisfies, the architecture that implements, its verification report, etc. Associations between Artifacts and Activities /Events/Participants/ Resources/Techniques can be recorded by means ArtifactAssetRelationships.

14 ~~13~~ 12.2 ArtifactPackage

ArtifactPackage is the containing element for artifacts involved in a structured assurance case.

Superclass

ArtifactConcept

~~Base::ArtifactElement~~

Associations

subArtifactElement [0..*] (subsets element)

~~artifactElement:Base::ArtifactElement[0..*] (composition)~~ – a collection of ArtifactElements forming an artifact package in a structured assurance case.

~~Semantics~~

~~ArtifactPackages contain ArtifactElements that represent the artifact forming part of a structured assurance case.~~

~~ArtifactPackages can also be nested.~~

~~12.3~~ ArtifactGroup

ArtifactGroup can be used to associate a number of ArtifactElements to a common group (e.g., representing a common type or purpose, or being of interest to a particular stakeholder).

Superclass

Base::ArtifactElement

Associations

artifactElement:ArtifactElement[0..*] – an optional collection of ArtifactElements organised within the ArtifactGroup.

Semantics

ArtifactGroup can be used to associate a number of ArtifactElements to a common group (e.g., representing a common type or purpose, or being of interest to a particular stakeholder). The name and the description of the ArtifactGroup should provide the semantic for understanding the ArtifactGroup. ArtifactGroups serve no structural purpose in the formation of the argument network, nor are they meant as a structural packaging mechanism (this should be done using ArtifactPackage).

13 ~~12~~ 4.2 ArtifactPackageBinding

The ~~ArtifactPackageBinding~~ is a sub type of ArtifactPackage used to record ArtifactAssetRelationships between the ArtifactAssets of two or more ArtifactPackages.

Superclass

ArtifactElement

~~ArtifactPackage~~

Associations

participantPackage:ArtifactPackage[2..*] - the ArtifactPackages containing the ArtifactAssets being related together by the ~~ArtifactPackageBinding~~.

Semantics

~~ArtifactPackageBindings~~ can be used to map dependencies between the cited ArtifactAssets of two or more ArtifactPackages. For example, a binding could be used to record a 'derivedFrom' ArtifactAssetRelationship between the ArtifactAsset of one package to the ArtifactAsset of another. ~~An ArtifactPackageBinding resides within an~~

ArtifactBindingPackage

Associations

source:ArtifactAsset[1,*] - the source of the ArtifactAssetRelationship

target:ArtifactAsset[1,*] - the target of the ArtifactAssetRelationship

Semantics

An ArtifactAsset can be related to other ArtifactAssets. This kind of information is specified by means of ArtifactAssetRelationships name and description of the ArtifactAssetRelationship can be used to describe the semantics of the ArtifactAssetRelationship.

14
~~13~~ 12.14³ Date

Date is a type, whose primitive type is String and represented in the ISO 8601 standard format

14.4 ArtifactElement (abstract)

~~ArtifactElement is the packaging notions in the artifact domain containing both ArtifactPackage and ArtifactInterfacePackage.~~
ArtifactElement is a common class for the packages in the Artifact domain.

superclass
ArtifactConcept

Association
artifactAsset:Artifact[0..*] {subsets element} - ArtifactAssets contained in this ArtifactElement.

14.5 ArtifactConcept (abstract)

ArtifactConcept is an abstraction of all the packaging and elements in the artifact domain.

Superclass
Packaging::Concept