

# 8 Foundation Class

## 8.1 General

This Foundation provides a semantic backing to the SACM Model from UML/KerML like models to make mapping (e.g. through Profiles) into UML/SysMLv1 or SysMLv2 more regular. Many of the textual entries in this section come directly from the UML 2.5.1 metamodel.

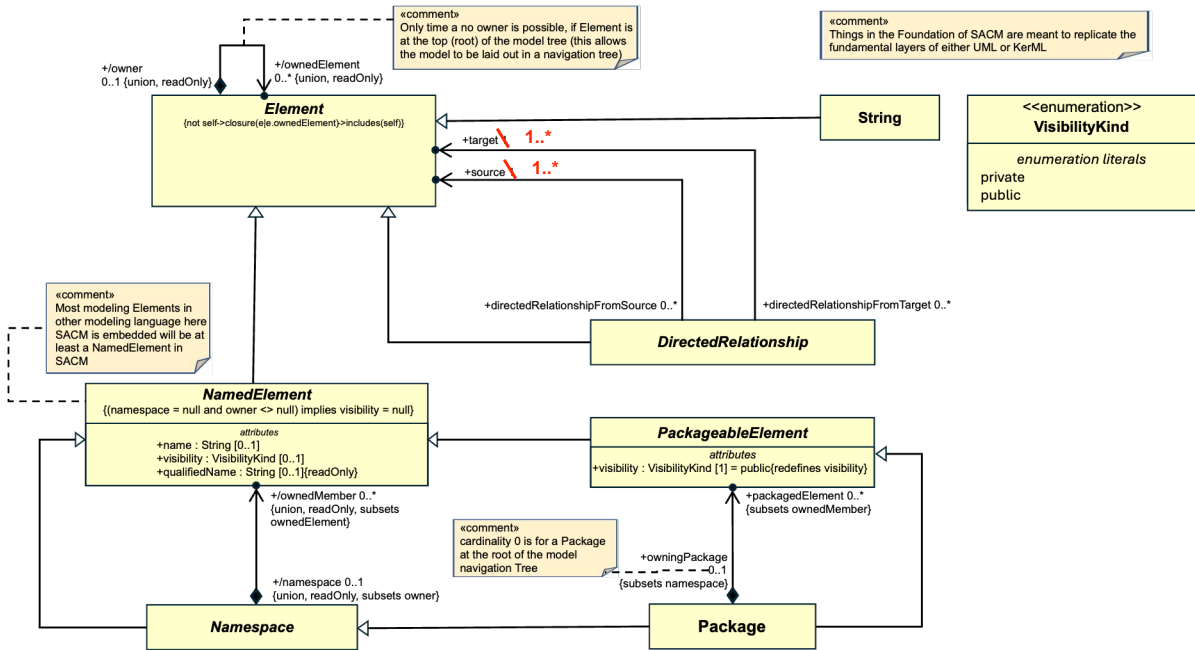


Figure 8.1 – Foundation Class

## 8.2 DirectedRelationship [Abstract Class]

A DirectedRelationship represents a relationship between a source model Element and target model Element. (This represents a more restrictive, than UML, Binary Relationship.)

his Foundation provides a semantic backing to the SACM Model from UML/KerML like models to make mapping (e.g. through Profiles) into UML/SysMLv1 or SysMLv2 more regular. Many of the textual entries in this section come directly from the UML 2.5.1 metamodel.

### Association Ends

- source : Element [1] - Specifies the source Element of the DirectedRelationship.
- target : Element [1] - Specifies the target Element of the DirectedRelationship.

# 13 SACM Argument Metamodel

## 13.1 General

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

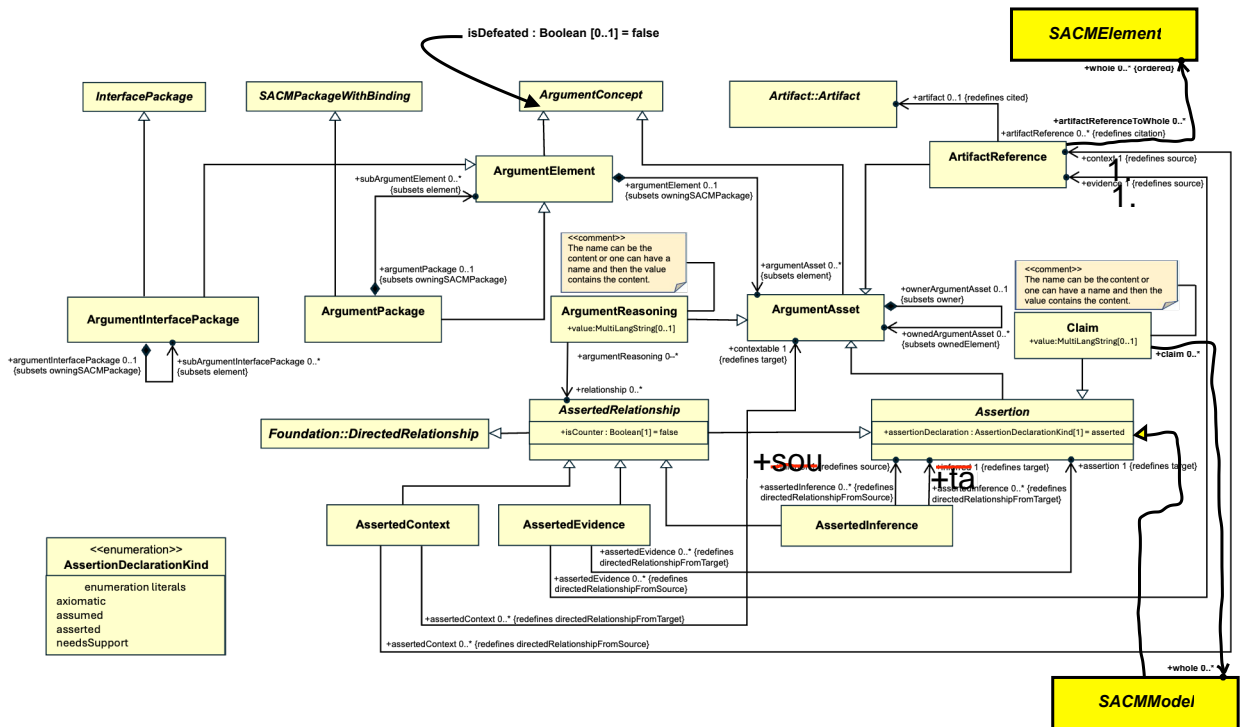


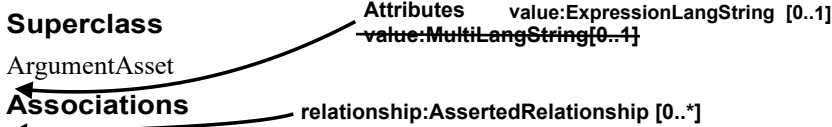
Figure 13.1 – Argument Package Diagram

This portion of the SACM model describes and defines the concepts required to model structured arguments. Arguments are represented in SACM through explicitly representing the Claims and citation of artifacts (e.g., as evidence) (ArtifactReference), and the ‘links’ between these elements – e.g., how one or more Claims are asserted to infer another Claim, or how one or more artifacts (referenced by ArtifactReference) are asserted as providing evidence for a Claim (AssertedEvidence). In addition to these core elements, in SACM it is possible to provide additional description of the ArgumentReasoning associated with inferential and evidential relationships, represent counter-arguments and counter-evidence (through isCounter: Boolean), and represent how artifacts provide the context in which arguments should be interpreted (through AssertedContext).

The packaging of structured arguments into ‘modular’ argument packages is enabled through ArgumentPackages. Users are able to declare interfaces for their packages through the use of ArgumentInterfacePackage. Within an ArgumentInterfacePackage, users create citations of the argument elements they select to disclose to external parties. Users are able to integrate ArgumentPackages through the use of ArgumentBindingPackage. An ArgumentBindingPackage binds ArgumentPackages together by including the declared ArgumentInterfacePackages for the ArgumentPackages, it may contain additional argument structures to provide the rationale of the binding. It is also possible within a package to cite elements contained within other argument packages (through ArtifactReference).

13 11  
~~12~~ ~~11~~ ~~12~~ **ArgumentReasoning**

ArgumentReasoning can be used to provide additional description or explanation of the asserted relationship. For example, it can be used to provide description of an AssertedInference that connects one or more Claims (premises) to another Claim (conclusion). ArgumentReasoning elements are therefore related to AssertedInferences, AssertedContexts, and AssertedEvidence. 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, contexts, and evidence.



**Semantics**

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

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

13 12  
~~12~~ ~~11~~ ~~13~~ **AssertedRelationship (abstract)**

AssertedRelationship 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.



**Attributes**

isCounter:Boolean[1] = false – a flag indicating whether the AssertedRelationship counters its declared purposes (e.g. setting isCounter = true for an AssertedEvidence indicates that the relationship is a counter-evidence).

~~**Associations**~~

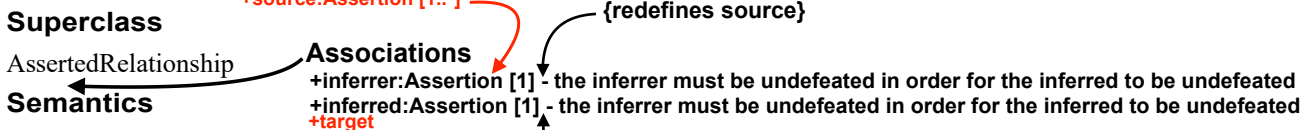
~~source:ArgumentAsset[1..\*] - reference to the ArgumentAsset(s) that are the source (starting point) of the relationship.~~  
~~target:ArgumentAsset[1] - reference to the ArgumentAsset(s) that are the target (ending point) of the relationship.~~  
~~reasoning:ArgumentReasoning[0..1] - an optional reference to the a description of the reasoning underlying the AssertedRelationship.~~

**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.

13 13  
~~12~~ ~~11~~ ~~14~~ **AssertedInference**

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



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

If AssertedInference has isCounter=false, then it should be interpreted as "if the inferrer is undefeated then the inferred is undefeated". If AssertedInference has isCounter=true, then it should be interpreted as "If the inferrer is undefeated then the inferred is defeated".

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.

~~12~~ ~~11~~ ~~15~~ <sup>13</sup> <sup>14</sup> **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. [1..\*] {redefines source}  
 +evidence:ArtifactReference [1] - the assertion is that the evidence for this Assertion is enough to make the AssertedEvidence undefeated. {redefines target}

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** If AssertedEvidence has isCounter=false, then it should be interpreted as "the evidence undefeats the Assertion". If AssertedEvidence has isCounter=true, then it should be interpreted as "the evidence defeats the Assertion".  
 The source of AssertedEvidence relationships must be ArtifactReference.

~~OCL:~~

~~self.source > forall(s|s.oellIsTypeOf(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).

~~12~~ ~~11~~ ~~16~~ <sup>13</sup> <sup>15</sup> **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 [1..\*]  
 +context:ArtifactReference [1] - the context provides further clarification or constraint of the contextable {redefines source}  
 +contextable:ArgumentAsset [1] - the context provides further clarification or constraint of the contextable {redefines target}

~~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").~~

~~12~~ ~~11~~ ~~17~~ <sup>16</sup> **AssertedArtifactSupport**

~~AssertedArtifactSupport records the assertion that~~

~~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").~~

**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 used to declare that an artifact is supported by other artifacts.~~

**Constraints** If AssertedContext has isCounter=false, then it should be interpreted as "the contextable must be interpreted in the context". If AssertedContext has isCounter=true, then it should be interpreted as "the contextable must not be interpreted in the context".

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

~~12~~ ~~11~~ ~~18~~ <sup>13</sup> <sup>17</sup> **AssertedArtifactContext**

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

Structured Assurance Case Metamodel, v2.3  
 Attributes isDefeated : Boolean [0..1] = false - specified whether this ArgumentConcept is defeated or not (or with cardinality 0, has not been determined yet).

**13.18 ArgumentConcept (abstract)**  
 ArgumentConcept is an abstraction of all the packaging and elements in the argument domain.  
 Superclass Packaging::Concept

# 14 Artifact Classes

## 14.1 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.

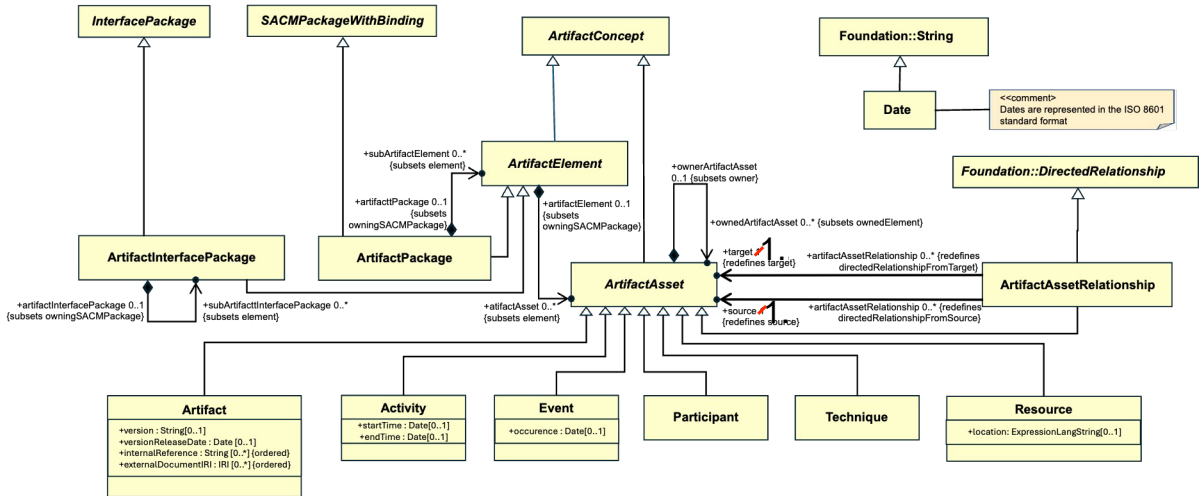


Figure 14.1 - Artifact Package Diagram

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 'isSACMAbstract'. 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

## 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~~ 12.14<sup>3</sup> 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