

# 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

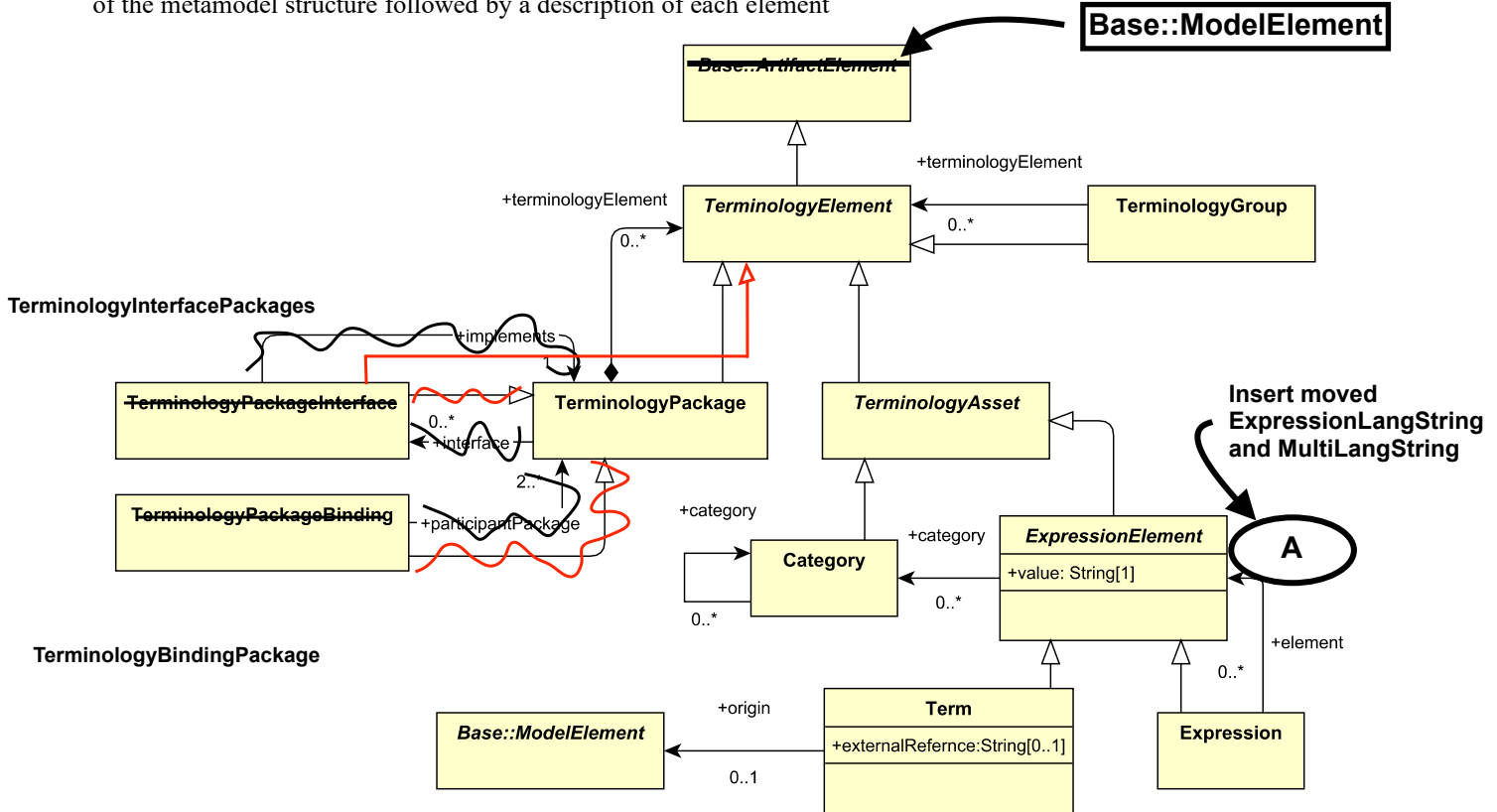


Figure 10.1 - Terminology Class Diagram

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 grouping of TerminologyElements (TerminologyGroup). ~~TerminologyElement extends Base::ArtifactElement, this implies that all elements in the Terminology package are artifacts.~~

### Superclass

Base::~~ArtifactElement~~ ← ModelElement

### Semantics

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

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

TerminologyInterfacePackage

## 10.5 ~~TerminologyPackageInterface~~

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~~  
~~TerminologyElement~~

### ~~Associations~~

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

### Semantics

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.

28

#### Constraints

All Elements that are cited in a BindingPackage must be contained in either the owner of that BindingPackage or a (recursively) sibling Package of that BindingPackage

inv CitedElementsAreScoped: element->forAll(e|e.cited->notEmpty() implies e.cited->closure(g|g.owningPackage).asSet()->one(p|p=self.owningPackage))

Elements that are not either a Diagram or a BaseElement must be a citation.

inv MustBeCited: element->forAll(e|not e->oclKindOf(SACMDiagram) and not e->oclKindOf(BaseElement) implies e.isCitation=true)

## 10.6 TerminologyPackageBinding

Elements within the TerminologyPackage can be bound together by means of TerminologyPackageBindings. TerminologyPackageBindings bind the participant packages by means of terminology elements that connect the cited elements of the participant packages.

### Superclass

TerminologyPackage

### Semantics

TerminologyPackageBinding binds TerminologyPackages together to indicate the relationship between two TerminologyPackages. A TerminologyPackageBinding resides within an AssuranceCasePackageBinding. It contains references, using isCitation=True to each TerminologyElement and connects TerminologyElements from different TerminologyPakages.

### Constraints

The participantPackages should be either TerminologyPackage or TerminologyPackageInterface

### OCL:

```
self.participantPackage->forall(pp|pp.ocIsKindOf(Terminology::TerminologyPackage))
```

## 10.7 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).

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

# 11 SACM Argumentation Metamodel

## 11.1 General

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

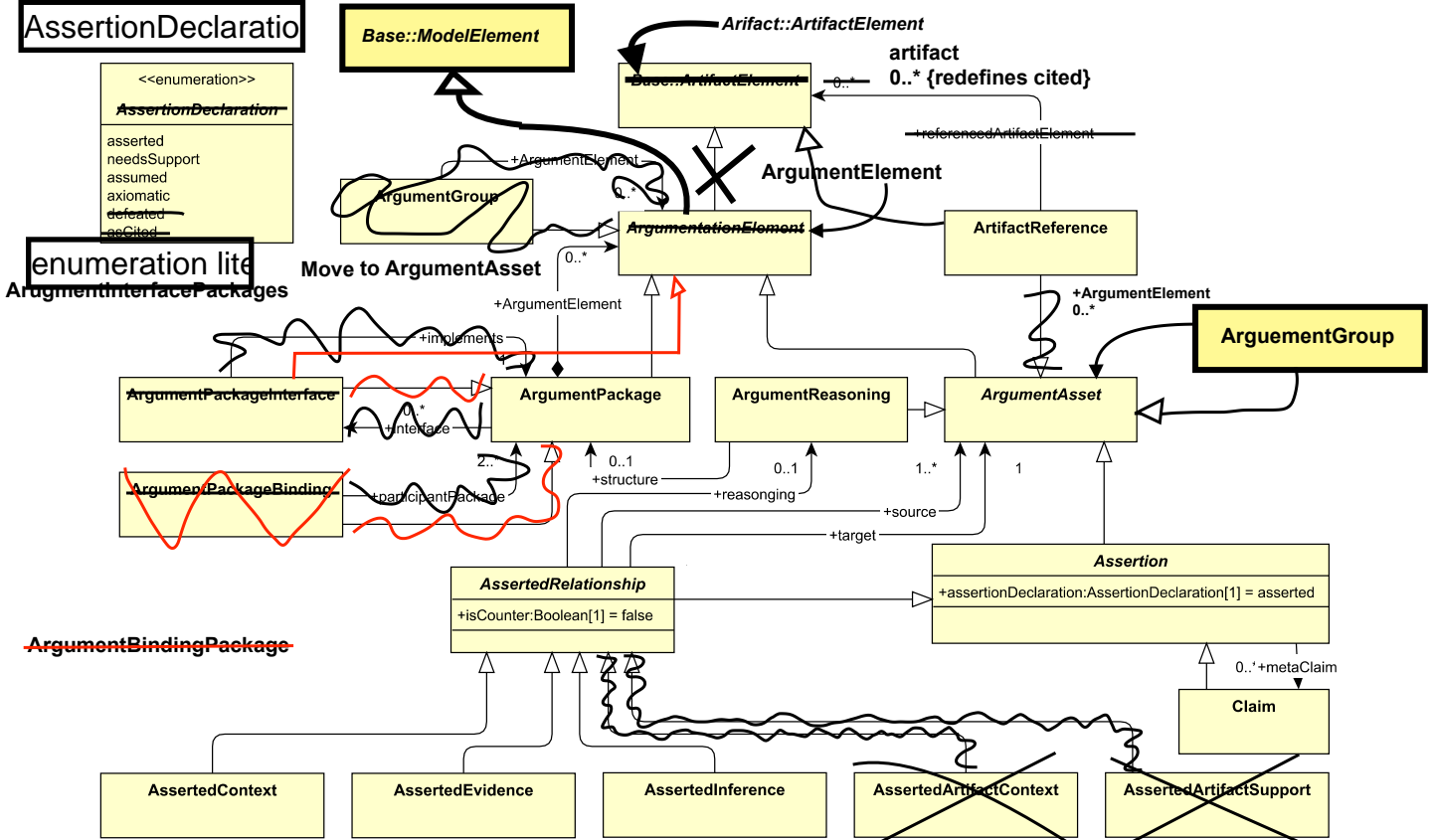


Figure 11.1 - Argumentation Package Diagram

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).

### ArgumentInterfacePackages

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 ArgumentPackageInterface. Within an ArgumentPackageInterface, users create citations of the argument elements they select to disclose to external parties. Users are able to integrate ArgumentPackages through the use of ArgumentPackageBinding. An ArgumentPackageBinding binds ArgumentPackages together by including the declared ArgumentPackageInterfaces 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).

### ArgumentBindingPackage

## 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..\*]~~ <sup>a</sup> an optional collection of ~~ArgumentationElements~~ <sup>ArgumentAss</sup> 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).

## 11.3 ArgumentationElement (abstract)

argument

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::ArtifactElement

### Semantics

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

## 11.4 ArgumentPackage Class

Argument  
Argumentation

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

### Superclass

~~ArgumentationElement~~

### Associations

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

### 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.

ArgumentBindingPackage

## 11.5 ArgumentPackageBinding

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

ArgumentElements

ArgumentInterfacePackages

ArgumentPackage). An ArgumentPackageBinding binds the participant packages (i.e., ArgumentPackageInterfaces) by means of structured arguments, with ArgumentationElements citing the contents inside the participant packages.

### Superclass

ArgumentPackage

### Associations

participantPackage:ArgumentPackage[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 needsSupport (i.e. currently has no supporting argument). An ArgumentPackageBinding can be used to record the mapping by means of containing a structured argument linkingArgumentElements that cite the claims in question.

**ArgumentBindingPackage**  
ArgumentPackageBinding is a sub type of ArgumentPackage, it is used to record the argument that connects the arguments of two or more ArgumentPackages.

**ArgumentBindingPackage**  
An ArgumentPackageBinding resides within an AssuranceCasePackageBinding. It contains references, using isCitation=True to each ArgumentationElement used in its argument structure that relates ArgumentationElements from different ArgumentPackages.

### Constraints

The participantPackages should be only ArgumentPackages

OCL: self.argumentEle

self.participantPackage->forall(pp|pp.ocllsTypeOf(Argument::ArgumentPackageInterface)) and self.argumentationElement->forall(e|e.isCitation = true and e.citedElement <> null)

The ArgumentElements contained by an ArgumentPackageBinding must be ArgumentElement citations to ArgumentElements contained within the ArgumentPackages associated by the participantPackage association.

ArgumentInterfacePackage

## 11.6 ArgumentPackageInterface

ArgumentInterfacePackage  
~~ArgumentPackageInterface~~ is a kind of ArgumentPackage that defines an interface that may be exchanged between users. An ArgumentPackage may declare one or more ~~ArgumentPackageInterface~~.  
**ArgumentInterfacePackage**

### Superclass

~~ArgumentElement~~  
~~ArgumentPackage~~

### Associations

implements:ArgumentPackage[1] - a reference to the ArgumentPackage which the ~~ArgumentPackageInterface~~ declares.  
**ArgumentInterfacePackage**

### Semantics

**ArgumentInterfacePackages**  
~~ArgumentPackageInterface~~ can be used to declare (by means of containing ArgumentElement based citations) 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.

An ArgumentPackageInterface resides inside the ArgumentPackage to which it refers. It refers to ArgumentationElements using isCitation=True that reside within the same ArgumentPackage as itself. Similar relationships exist for an ArtifactPackageInterface and for a TerminologyPackageInterface.

### Constraints

~~ArgumentPackageInterfaces are only allowed with isCitation=true and +citedElement refer to ArgumentAssets within the ArgumentPackage implementation referred to by implements.~~

## 11.7 ArgumentAsset (abstract)

ArgumentAsset is the abstract base element for the element

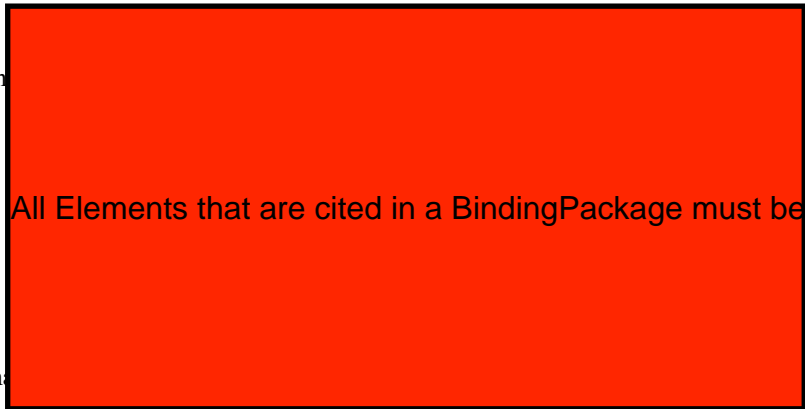
### Superclass

ArgumentationElement

### Semantics

ArgumentAssets represent the constituent building blocks of an ArgumentPackage.

For example, ArgumentAssets can represent the Claims made in an ArgumentPackage.



AssertionDeclarationKind

## 11.8 AssertionDeclaration (Enumeration)

AssertionDeclaration provides a list of declarations which can be used to declare the state of an Assertion.

### Superclass

N/A

### Enumeration Literals

asserted – the default enumeration literal, indicating that an Assertion is asserted.

needsSupport – a flag indicating that further argumentation has yet to be provided to support the Assertion.

assumed – a flag indicating that the Assertion being made is declared by the author as being assumed to be true rather than being supported by further argumentation.

axiomatic – a flag indicating that the Assertion being made by the author is axiomatically true, so that no further argumentation is needed.

~~defeated – a flag indicating that the Assertion is defeated by counter evidence and/or argumentation.~~

~~asCited – a flag indicating that because the Assertion is cited, the AssertionDeclaration should be transitively derived from the value of the AssertionDeclaration of the cited Assertion.~~

### Semantics

~~AssertionDeclaration~~ provides a list of declarations which indicate the state of an Assertion.

## 11.9 ArtifactReference

ArtifactReference enables the citation of an artifact as information that relates to the structured argument.

# 12 Artifact Classes

## 12.1 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.

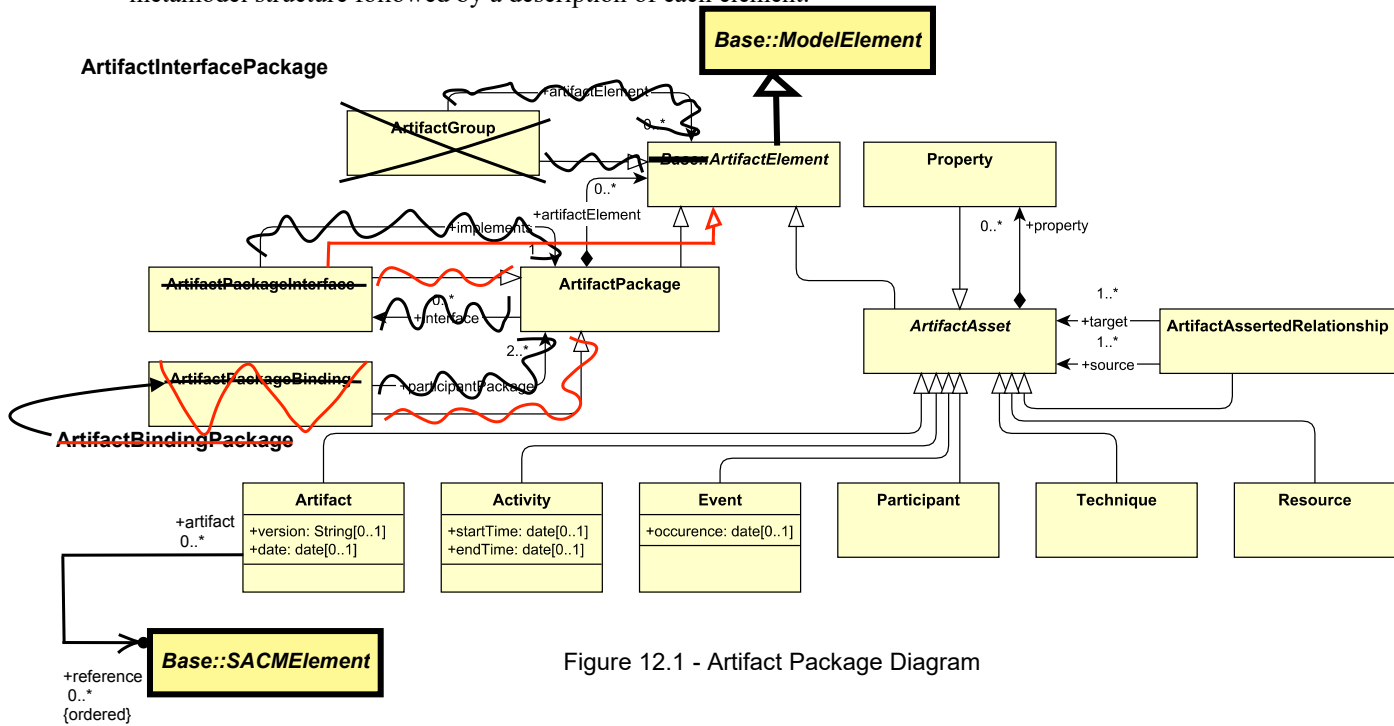


Figure 12.1 - Artifact Package Diagram

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 '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.

## 12.2 ArtifactPackage

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

### Superclass

Base::ArtifactElement

### Associations

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).~~

## ~~12.4.3 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

~~AssuranceCasePackageBinding. It contains references, using isCitation=True to each ArtifactAsset needed and defines relationships among ArtifactAssets from different ArtifactPackages.~~

### Constraints

~~ArtifactBindingPackages  
ArtifactPackageBindings must only contain ArtifactAssetRelationships with source and target Artifacts, with isCitation = true citing ArtifactAssets contained within the ArtifactPackages associated by participantPackage.~~

## 12.5/4 <sup>ArtifactInterfacePackage</sup> ArtifactPackageInterface

~~ArtifactPackageInterface is a kind of ArtifactPackage that defines an interface that may be exchanged between users. An ArtifactPackage may define one or more ArtifactPackageInterfaces.~~

### Superclass

~~ArtifactElement  
ArtifactPackage~~

### Associations

implements:ArtifactPackage[1] - a reference to the ArtifactPackage which the ArtifactPackageInterface declares.

### Semantics

~~ArtifactPackageInterface enables the declaration of the elements of an ArtifactPackage that might be referred to (cited) in another ArtifactPackage. An ArtifactPackageInterface resides inside the ArtifactPackage to which it refers. It refers to ArtifactAssets using isCitation=True that reside within the same ArtifactPackage as itself.~~

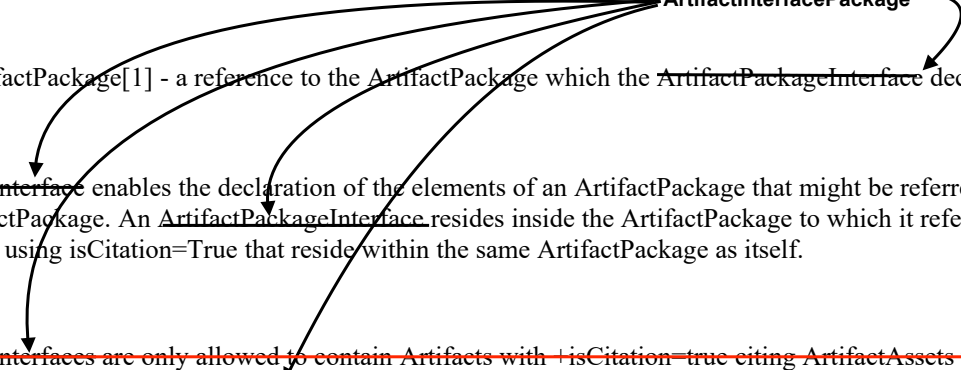
### Constraints

~~ArtifactPackageInterfaces are only allowed to contain Artifacts with isCitation=true citing ArtifactAssets within the ArtifactPackage with which this ArtifactPackageInterface is associated.~~

ArtifactInterfacePackage

ArtifactInterfacePackages

ArtifactInterfacePackage



## 12.6/5 ArtifactAsset (abstract)

ArtifactAsset represents the artifact-specific pieces of information.

### Superclass

Base::ArtifactElement

### Association

property:Property[0..\*] (composition) – an optional collection of characteristics of an ArtifactAsset.

### Semantics

Information about artifacts is essential for any assurance case. The artifacts 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 artifact, its lifecycle, and its properties. All this information might have to be consulted for developing confidence in the validity of an assurance case.

All Elements that are cited in a BindingPackage must be contained in either the owner of that BindingPackage or a (recursively) sibling Package of that BindingPackage

```
inv CitedElementsAreScoped: element->forAll(e|e.cited->notEmpty() implies e.cited->closure(g|g.owningPackage).asSet()->one(p|p=self.owningPackage))
```

Elements that are not either a Diagram or a BaseElement must be a citation.

```
inv MustBeCited: element->forAll(e|not e->oclKindOf(SACMDiagram) and not e->oclKindOf(BaseElement) implies e.isCitation=true)
```

## 12.7/6 Artifact

Artifact represents the distinguishable units of data used in a structured assurance case.

### Superclass

ArtifactAsset