

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.

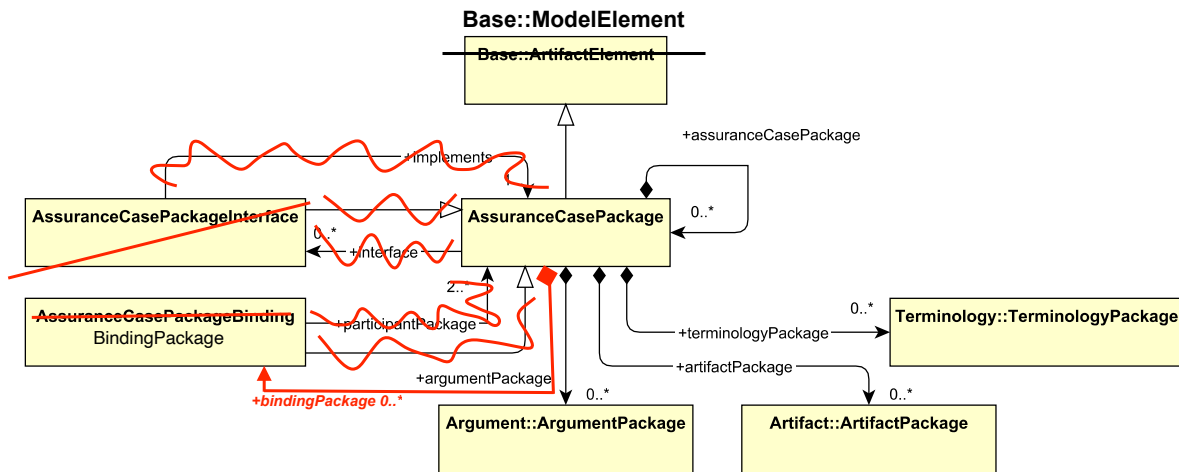


Figure 9.1 - Structured Assurance Case Packages Class Diagram

In SACM, the parent container element is AssuranceCasePackage. 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).

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
~~Base::ArtifactElement~~

Associations

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~~

bindingPackage: BindingPackage [0..*] Sub-packages within any BindingPackage can be bound together by means of a BindingPackage. A Binding can include any combination of reference to Package or InterfacePackage.

artifactPackage: ArtifactPackage [0..*] (composition) – a number of optional artifact sub-packages

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

argumentPackage: Argument::ArgumentPackage[0..*] (composition) – a number of optional argument packages.

Semantics

AssuranceCasePackage is the root class for creating structured assurance cases.

~~9.3 AssuranceCasePackageInterface~~

AssuranceCasePackageInterface is a kind of AssuranceCasePackage that defines an interface that may be exchanged between users. An AssuranceCasePackageInterface will consist of at least one of an ArgumentPackageInterface, ArtifactPackageInterface, and/or a TerminologyPackageInterface. Conversely, any combination of an ArgumentPackageInterface, ArtifactPackageInterface, and/or a TerminologyPackageInterface will be contained within an AssuranceCasePackageInterface. The combination depends on what parts of the assurance case are to be made “public.” An AssuranceCasePackage may declare one or more ArtifactPackageInterfaces.

Superclass

AssuranceCasePackage

Associations

implements:AssuranceCasePackage[1] – the AssuranceCasePackage that the AssuranceCasePackageInterface declares.

Semantics

AssuranceCasePackageInterface enables the declaration of the elements of an AssuranceCasePackage that might be referred to (cited) in another AssuranceCasePackage. These declarations are provided by containing AssuranceCasePackageInterface(s)/ArgumentPackageInterface(s)/ArtifactPackageInterface(s)/TerminologyPackageInterface(s) to the packages contained by the AssuranceCasePackage for which the interface is provided.

Constraints

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

OCL:

```
self.assuranceCasePackage->forall(acp|acp.oclIsTypeOf(AssuranceCasePackageInterface)) and
self.argumentPackage->forall(ap|ap.oclIsTypeOf(Argumentation::ArgumentPackageInterface))
and self.artifactPackage->forall(ap|ap.oclIsTypeOf(Artifact::ArtifactPackageInterface)) and
self.terminologyPackage->forall(tp|
tp.oclIsTypeOf(Terminology::TerminologyPackageInterface))
```

~~9.4 AssuranceCasePackageBinding~~

a BindingPackage.

Sub-packages within any ~~AssuranceCasePackage~~ **BindingPackage** can be bound together by means of ~~AssuranceCasePackageBindings~~ **SACMPackage types**. A Binding can include any combination of reference to Package or PackageInterface of the same type. An AssuranceCasePackageBinding can also include Package(s) or PackageInterface(s) for Argument, Artifact and/or Terminology. Each Package or PackageInterface referenced by a PackageInterface can be a participant Package in a Binding. AssuranceCasePackageBindings bind the participant packages by means of ArgumentPackageBindings/TerminologyPackageBindings/ArtifactPackageBindings elements that bind the contained packages of the participant packages.

Superclass

ModelElement

~~AssuranceCasePackage~~

Associations

participantPackage:AssuranceCasePackage[2..*] – references to AssuranceCasePackages which the AssuranceCasePackageBinding binds together. The Participant Package provides access to its elements through a Package Interface, if a Package Interface is named in the Package Binding, or to all elements if the Package is named in the Package Binding.

Semantics

AssuranceCasePackageBinding binds peer AssuranceCasePackages together to indicate the relationship between these AssuranceCasePackages. The bindings between AssuranceCasePackages consist of the bindings of the packages (i.e. ArgumentPackageBindings, ArtifactPackageBindings and TerminologyPackageBindings) contained in the AssuranceCasePackages, together with an optional ArgumentationPackage that asserts the relationship between +participantPackage. The AssuranceCasePackageBinding includes a reference to each participant Package which can be either an AssuranceCasePackage or an AssuranceCasePackageInterface and that are external to Packages to the AssuranceCasePackage that contains this AssuranceCasePackageBinding. Within the AssuranceCasePackageBinding, there might be zero or more ArgumentPackageBinding, ArtifactPackageBinding, or TerminologyPackageBinding. Each of these that exist would refer to at least two participant packages that could be a Package or PackageInterface of the same type.

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

~~The participantPackages should be either AssuranceCasePackage or AssuranceCasePackageInterfaces.~~

OCL:

~~self.participantPackage->forall(pp|pp.oclsTypeOf(AssuranceCase::AssuranceCasePackage) or
pp.oclsTypeOf(AssuranceCase::AssuranceCasePackageInterface))~~

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

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

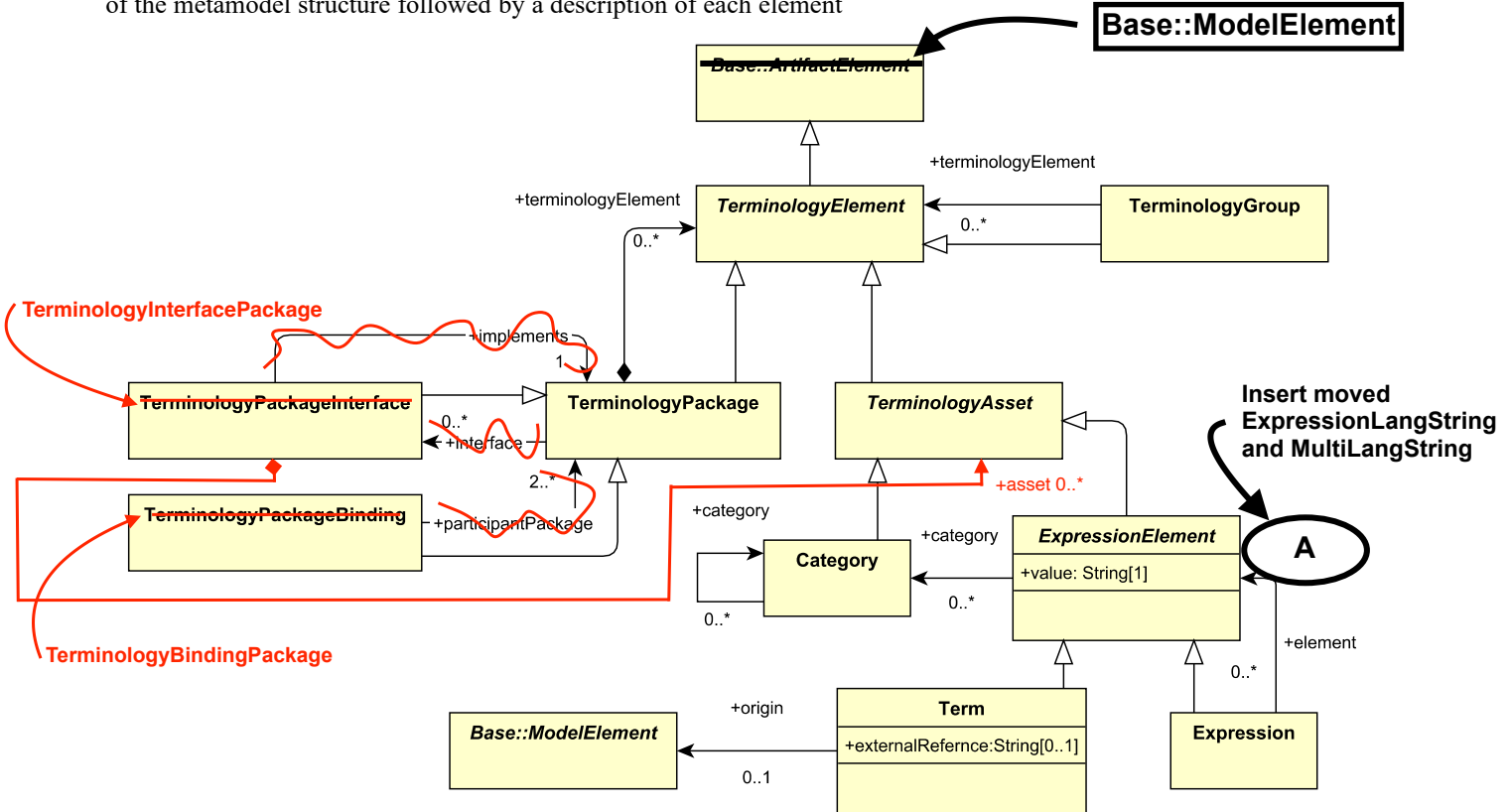


Figure 10.1 - Terminology Class Diagram

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

Associations

~~+asset : terminologyAsset[0..*]~~

~~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.

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.

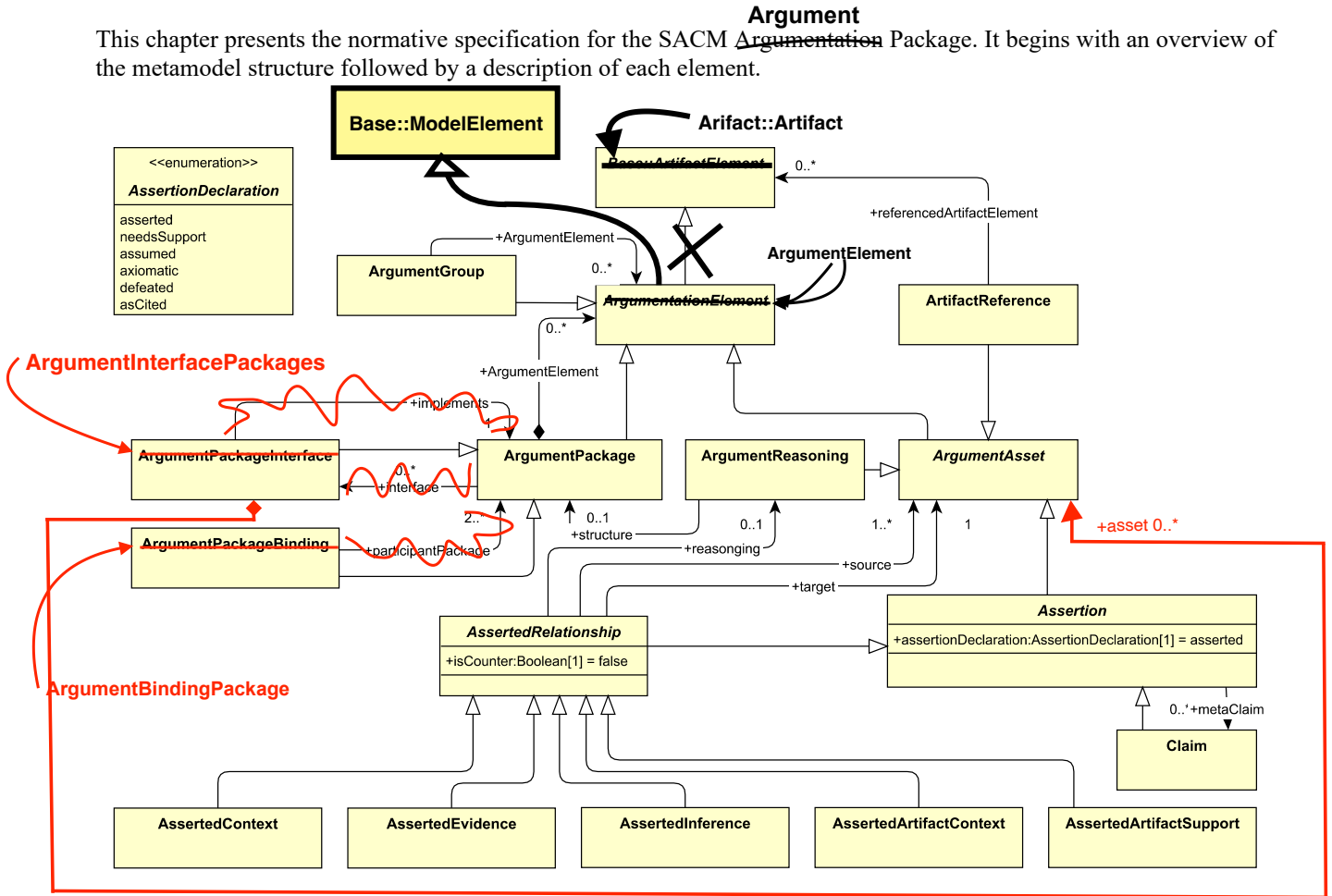


Figure 11.1 - ~~Argumentation~~ Package Diagram

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**).

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 **argumentation** 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

ArgumentationElement

Associations

argumentationElement:ArgumentationElement[0..*] – an optional collection of ArgumentationElements 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)

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

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

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.

11.5 ArgumentPackageBinding

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

argument

Argument

Argumentation

ArgumentBindingPackage

ArgumentBindingPackages

ArgumentBindingPackage

ArgumentElements

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

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.argumentElement

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.

11.6 ArgumentPackageInterface

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

Superclass

ArgumentPackage

Associations

+asset : ArgumentAsset[0..*]
implements:ArgumentPackage[1] - a reference to the ArgumentPackage which the ArgumentPackageInterface declares.

Semantics

ArgumentPackageInterfaces 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

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.

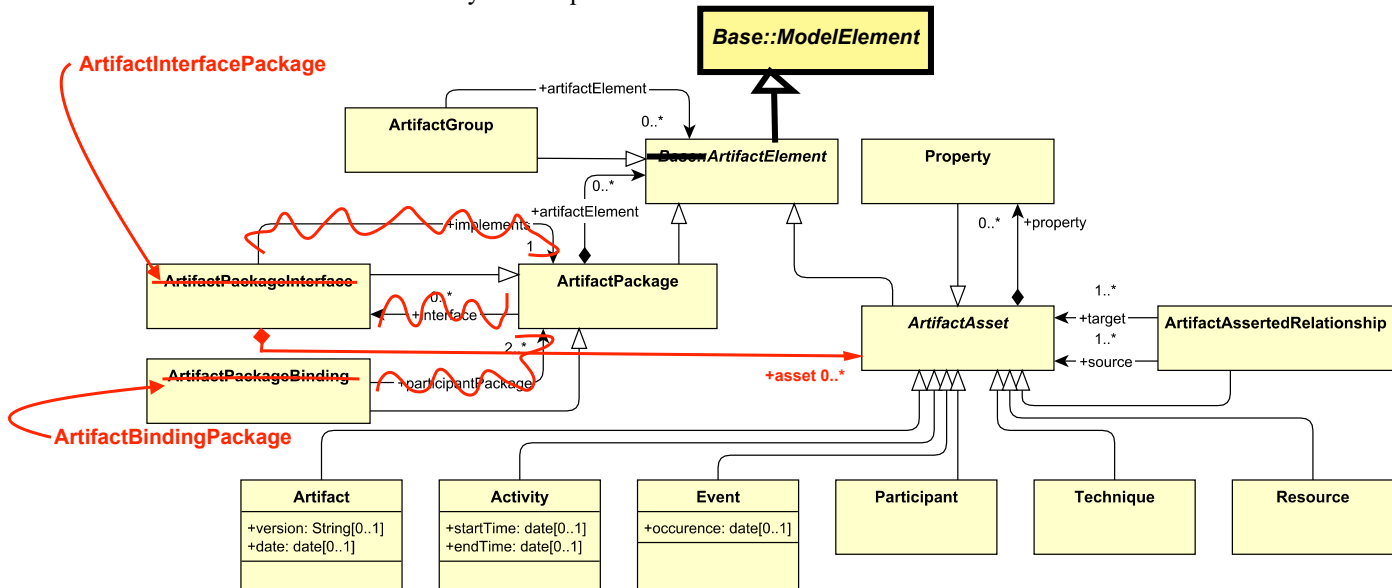


Figure 12.1 - Artifact Package Diagram

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 ~~ArtifactPackageBinding~~

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

Superclass

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 ~~ArtifactPackageInterface~~ ^{ArtifactInterfacePackage}

ArtifactInterfacePackage

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

Superclass

ArtifactPackage

Associations

+asset : ArtifactAsset[0..*]

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.

12.6 ArtifactAsset (abstract)

ArtifactAsset represents the artifact-specific pieces of information of an assurance case, in contrast to the argument-specific pieces of information.

Superclass

Base::ArtifactElement

Association

property:Property[0..*] (composition) – an optional collection of Propert(ies) which enable the specification of the 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.

12.7 Artifact

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

Superclass

ArtifactAsset