Classes

10.1 General

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

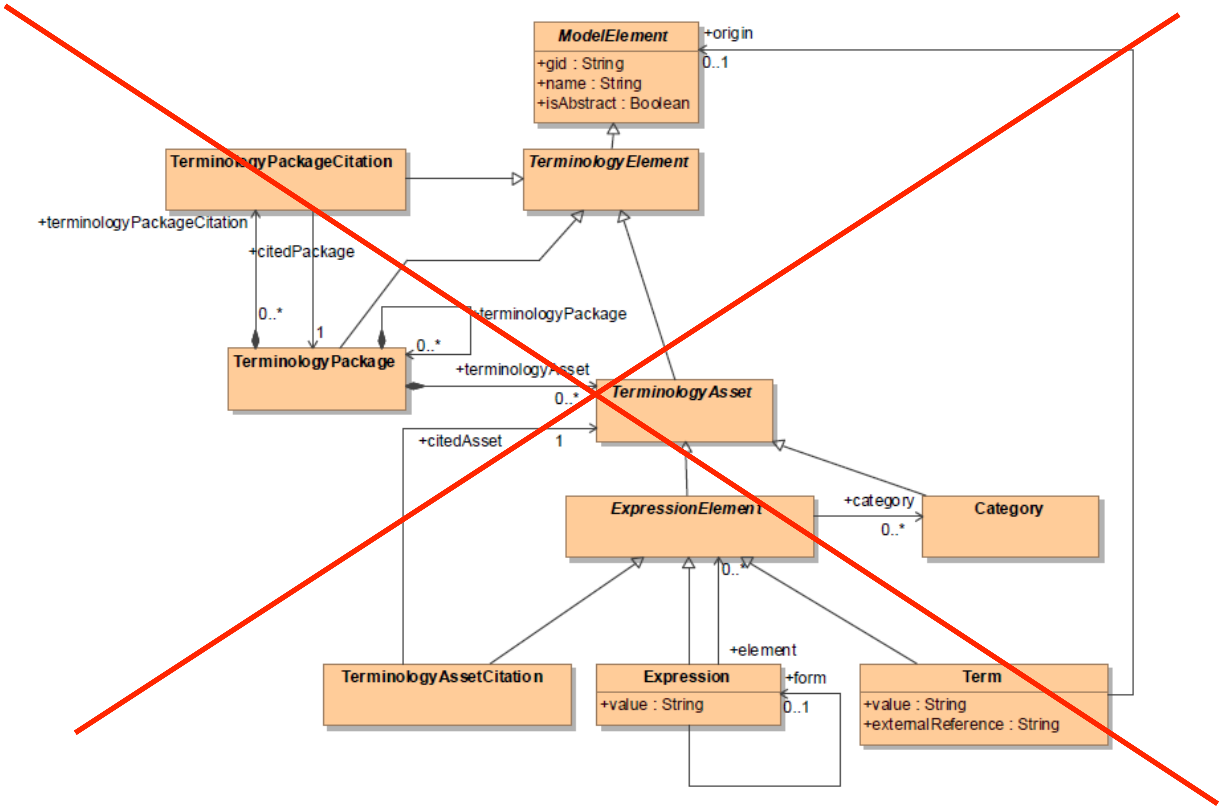


Figure 10.1 - Terminology Class Diagram

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

10.2 TerminologyElement (abstract)

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

Superclass

ModelElement

Semantics

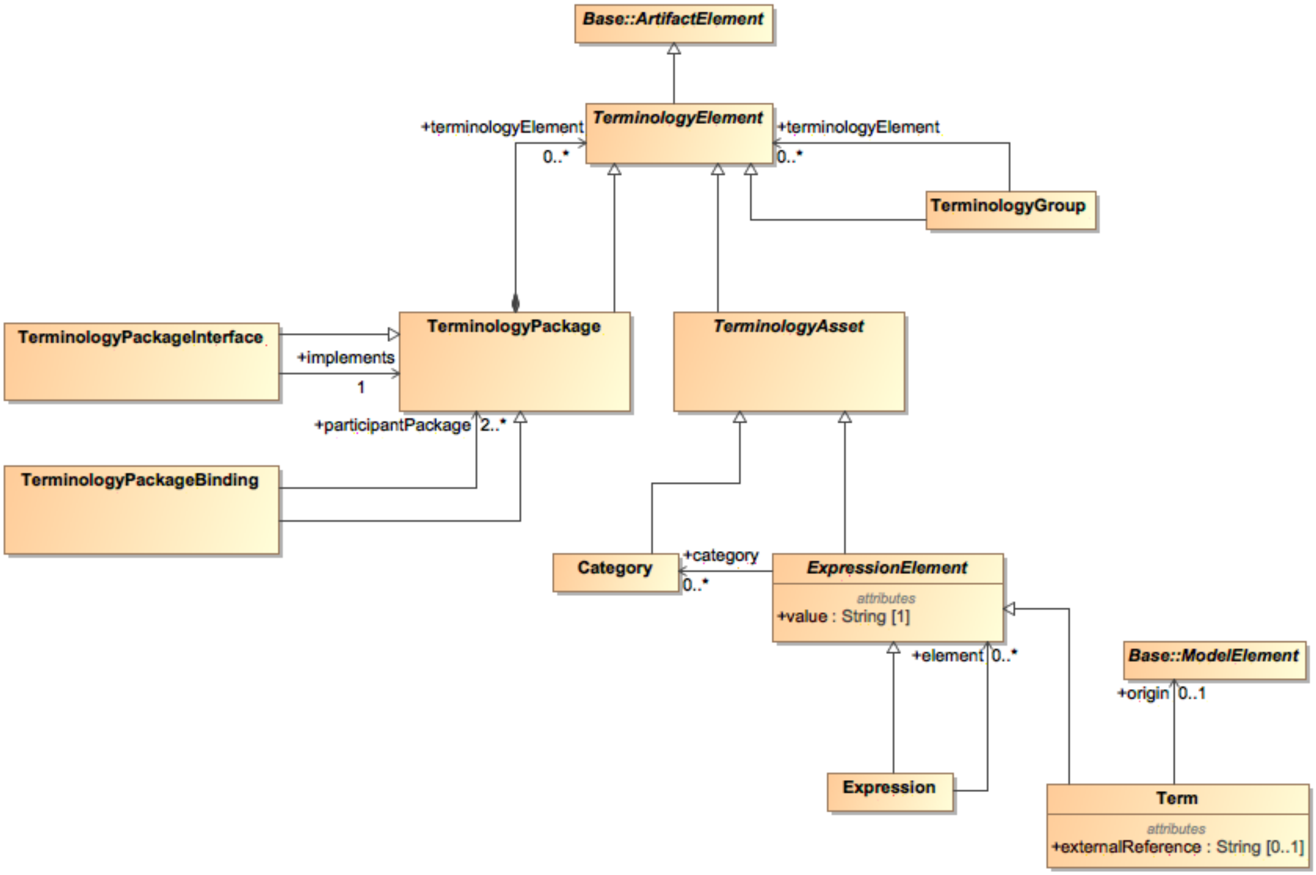
TerminologyElement is the base class for specifying the terminology aspects of an assurance case (AssuranceCasePackage).

10.3 TerminologyPackage

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

Superclass

Figure 10.1 – Terminology Class Diagram



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).

~~TerminologyAsset:TerminologyAsset[0..*]~~

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

~~The nested terminologyPackage contained in a given instance of a TerminologyPackage~~

Semantics

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

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

10.4 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).

10.5 Category

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

Superclass

TerminologyAsset

Semantics

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

10.6 ExpressionElement (abstract)

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

Superclass

TerminologyAsset

Associations

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

Semantics

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

10.7 Expression

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

Superclass

ArtifactElement

Attributes

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

Associations

11 SACM Argumentation Metamodel

11.1 General

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

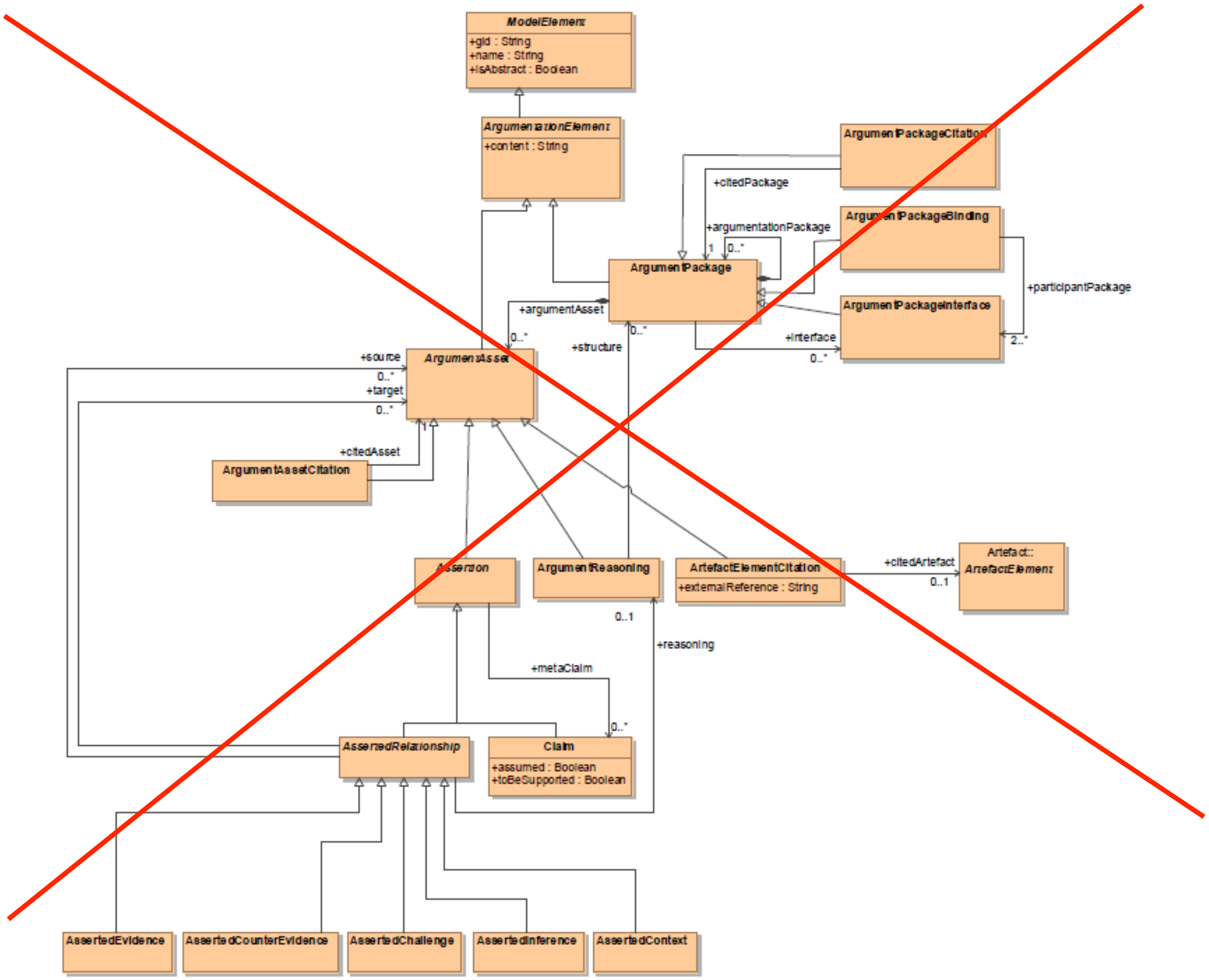
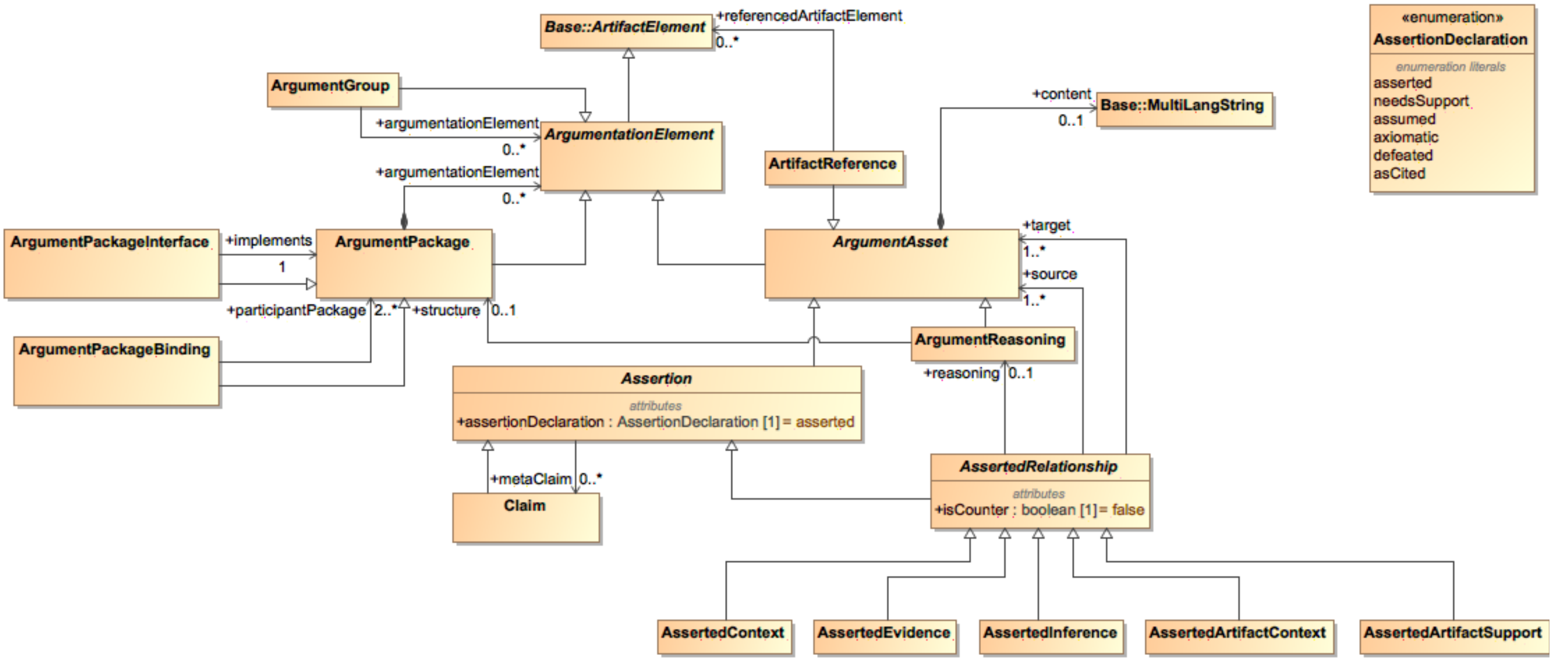


Figure 11.1 - Argumentation ~~Class~~ Package Diagram

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 (referenced by ArtifactReference) (e.g., as evidence) (~~ArtifactElementCitation~~ 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 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 ~~(through AssertedChallenge),~~ counter-evidence (through ~~AssertedCounterEvidence~~ 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, an optional declaration of an interface for the package (ArgumentPackageInterface)

Figure 11.1 – Argumentation Package Diagram



organizes

AssertedContext

that ~~cites~~ a specific selection of the ArgumentElements contained within the package, and the ability to link (by means of an argument) two or more argument packages (through an ArgumentPackageBinding). It is also possible within a package to cite elements contained within other argument packages (through ~~using ArgumentElementCitation~~).

~~In the following sections we describe these model elements in detail.~~

11.2.1 ArgumentationElement class (abstract)

An ArgumentationElement is the top level element of the hierarchy for argumentation elements.

Semantics

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

11.2.2 ArgumentPackage Class

The ArgumentPackage Class is the container class for a structured argument represented using the SACM Argumentation Metamodel.

The packaging of structured arguments into 'modular' argument packages is enabled through ArgumentPackages, an optional declaration of an interface for the package (ArgumentPackageInterface) that organises a specific selection of the ArgumentElements contained within the package, and the ability to link (by means of an argument) two or more argument packages (through an ArgumentPackageBinding). It is also possible within a package to cite elements contained within other argument packages (through ArtifactReference).

Superclass

ArgumentationElement

Associations

argumentAsset:ArgumentAsset[0..*]

The ArgumentAssets contained in a given instance of an ArgumentPackage.
argumentPackage:ArgumentationPackage[0..*]

The nested argumentPackage contained in a given instance of an ArgumentPackage
interface:ArgumentationPackage[0..*]

Reference to the declared interface for the ArgumentPackage.

Semantics

ArgumentPackages contain structured arguments. These arguments are composed of ArgumentAssets. ArgumentPackages elements can be nested, and can contain citations (references) to other ArgumentPackages.

For example, arguments can be established through the composition of Claims (propositions) and the AssertedInferences between those Claims.

11.2.4 ArgumentPackageBinding Class

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

Superclass

ArgumentPackage

Associations

participantPackage:ArgumentPackageInterface[2..*]

The ArgumentPackages being mapped together by the ArgumentPackageBinding.

Semantics

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

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

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

Constraints

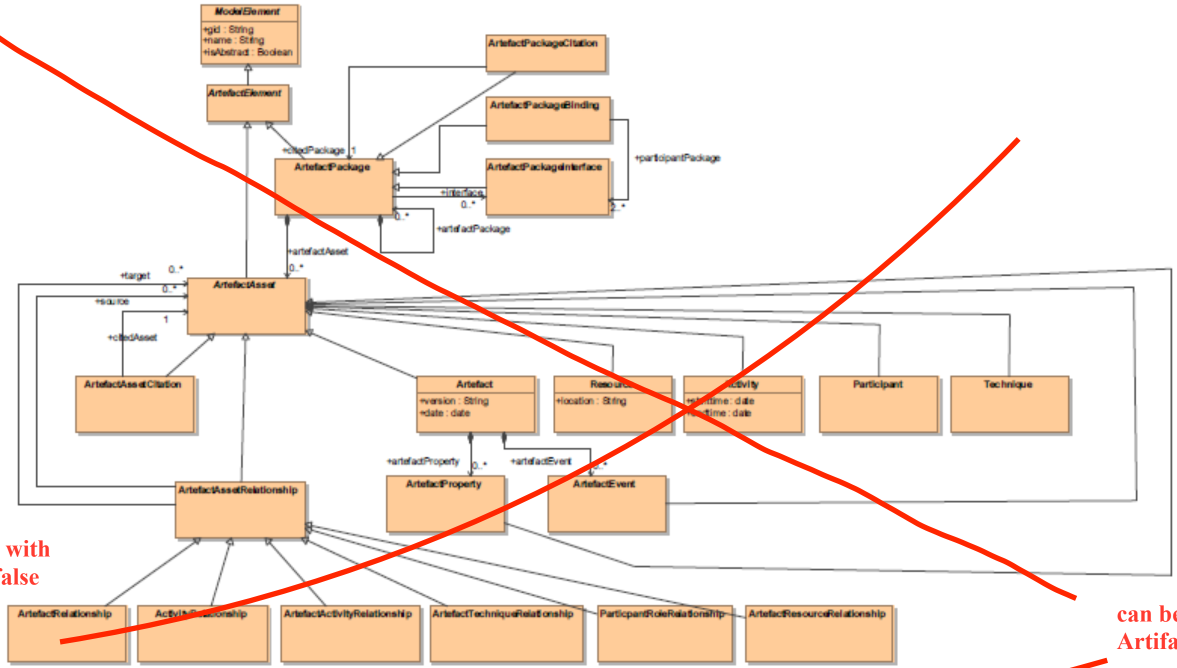
The 'root' ArgumentAssets contained by an ArgumentPackageBinding (i.e. the ArgumentAssets only associated as target of an AssertedRelationship) and 'leaf' ArgumentAssets (i.e. the ArgumentAssets only associated as source of an

12 Artifact Classes

12.1 General

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

Figure 12.1 - Artifact ~~Class~~ Package Diagram

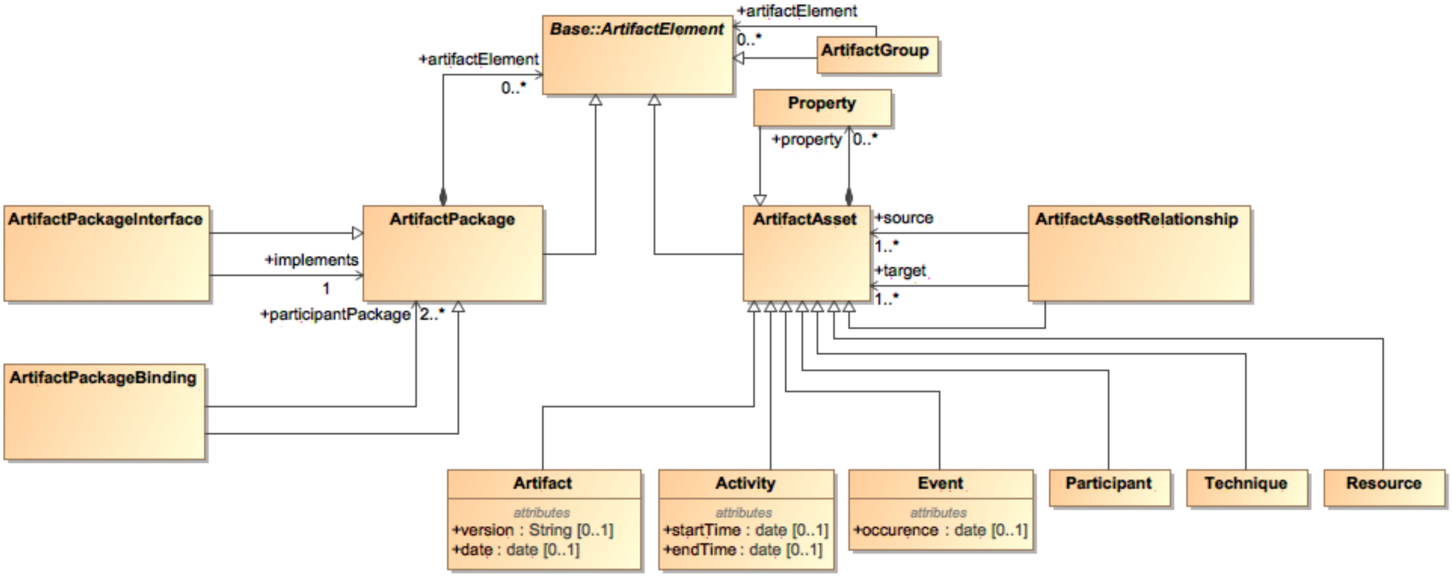


Artifacts correspond to the main evidentiary elements of an assurance case. By means of assertions (~~AssertedEvidence and AssertedCounterEvidence~~ AssertedEvidence with isCounter = true/false), artifacts ~~are used for~~ can be referenced (using ArtifactReferences) as supporting claims and arguments.

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 ~~mean~~ means of the attribute 'isAbstract' (ModelElement). For example, a supplier and a certifier might agree upon the need for maintaining a hazard log during a system's lifecycle. Such a hazard log would initially be modeled as an 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 (ArtifactProperty) 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

Figure 12.1 – Artifact Package Diagram



12.2 ArtifactPackage

Associations between Artifacts and Activities /Events/Participants/ Resources/ Techniques, and between Aritfacts and Activities /Events/Participants/ Resources/ Techniques Participants can be recorded by means ArtifactAssetRelationships.

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. ~~Further relationships might be specified between other artifact assets, such precedence between activities ('Specify system design' precedes 'Verify system design') and the participants in an Activity.~~

12.3 ~~12.2~~ 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:ArtifactPackageInterface[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.

Contraints

12.4 ~~12.3~~ ArtifactPackageInterface

ArtifactPackageInterface is a kind of ArtifactPackage that defines an interface that may be exchanged between users. A typical use case might be for a component supplier to provide its customers with ArtifactPackageInterfaces that contain the relevant supplier's ArtifactElements for the customers' ArtifactPackages. An ArtfefactPackage may also declare that it implements or conforms to a particular ArtifactPackageInterface.

Superclass

ArtifactPackage

Associations

artifactAsset: ArtifactAsset [0..*] – an optional set of ArtifactAsset elements, such as citations, artifacts, resources, activities, etc.

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

Semantics

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

Constraints

12.5 ~~12.4~~ ArtifactAsset class (abstract)

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

Superclass

ArtifactElement

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

12.2 ArtifactPackage

ArgumentPackage 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 a 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.