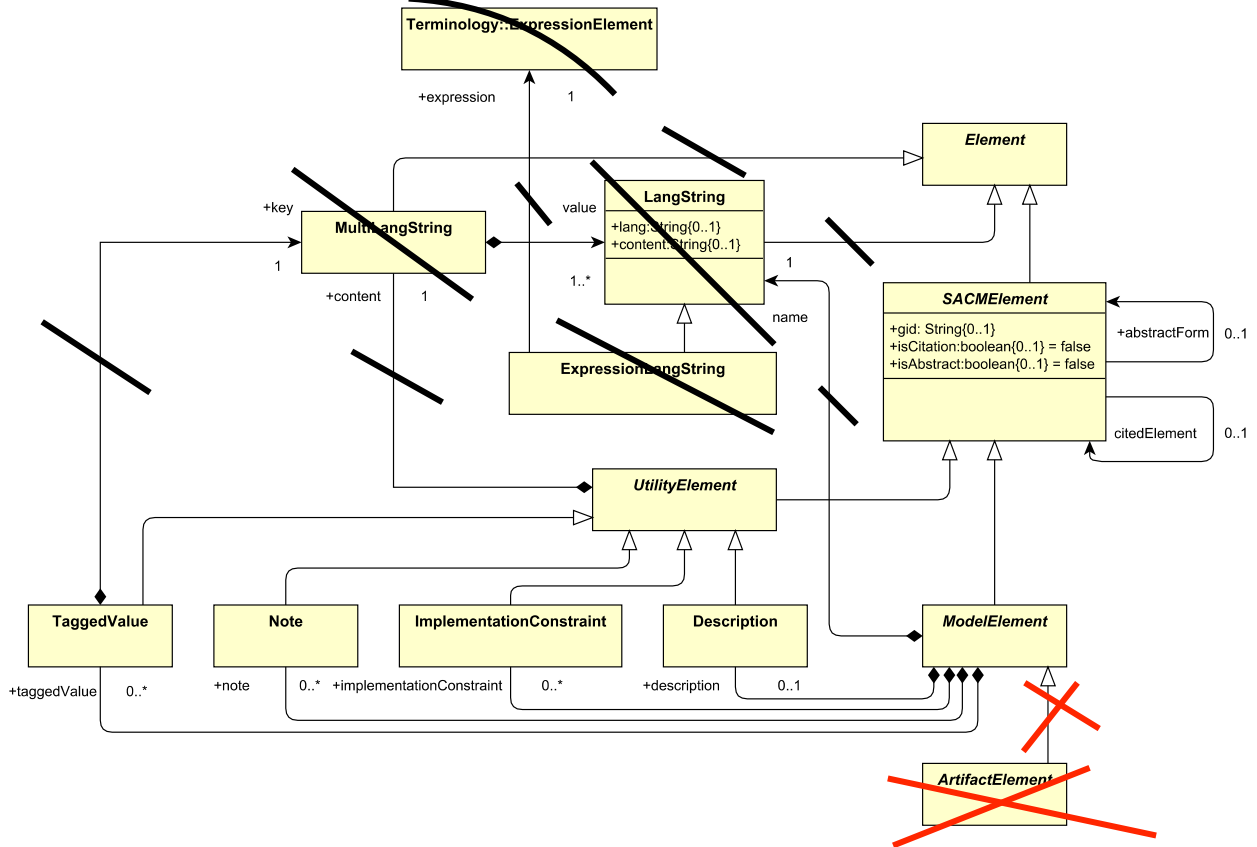


~~9~~ ~~8~~ **Structured Assurance Case Base Classes**

~~9~~ ~~8.1~~ **General**

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



~~9~~ ~~8.1~~ **9** Figure 8.1 - Structured Assurance Case Base Classes Diagram

The Structured Assurance Case Base Classes express the foundational concepts and relationships of the base elements of the SACM metamodel and are utilized, through inheritance, by the bulk of the rest of the Structured Assurance Case Metamodel.

Constraints

ImplementationConstraints should only be specified if +isAbstract is true

OCL:

self.implementationConstraint->size() > 0 implies self.isAbstract = true

9 ~~8.7~~ 4 UtilityElement (abstract)

UtilityElement is the base element for a number of auxiliary elements which can be added to ModelElements.

Superclass

SACMElement

Associations

content:MultiLangString[0..1] (composition) – a MultiLangString to describe the content of the UtilityElement in (possibly) multiple languages

Semantics

UtilityElement supports the specification of additional information for a ModelElement.

9 ~~8.8~~ 5 ImplementationConstraint

ImplementationConstraint specifies details of any implementation constraints that must be satisfied whenever a referencing ModelElement is to be converted from *isAbstract = true* to *isAbstract = false*. For example in the context of a SACM pattern fragment, an element will need to satisfy the implementation rules of the pattern.

Superclass

UtilityElement

Semantics

ImplementationConstraints indicate the conditions to fulfill in order to allow an abstract ModelElement (*isAbstract = true*) to become non-abstract (*isAbstract = false*).

9 ~~8.9~~ 6 Description

Description is used to specify a description that may be associated with a ModelElement. In many cases Description is used to provide the ‘content’ of a SACM element. For example, it would be used to provide the text of a Claim.

Superclass

UtilityElement

Semantics

A Description provides details about ModelElements in relation to aspects such as their content or purpose. Therefore, Descriptions can be used to both characterize ModelElements and facilitate their understanding.

Move to Artifact Class

9 ~~8.10~~ 7 ArtifactElement (abstract)

ArtifactElement acts as the base class for elements in other SACM packages. Essentially, all elements which extend ArtifactElement is considered to be an artifact, and therefore can be referenced using Argument:ArtifactReference.

Superclass

ModelElement

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~~⁷ Note

This class specifies a generic note that may be associated with a ModelElement. For example a note may include a number of explanatory comments.

Superclass

UtilityElement

Semantics

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.

9 ~~8.12~~⁸ TaggedValue

This class represents a simple key/value pair 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

UtilityElement

Associations

key:MultiLangString[1] (composition) – the key of the TaggedValue.

Semantics

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

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.

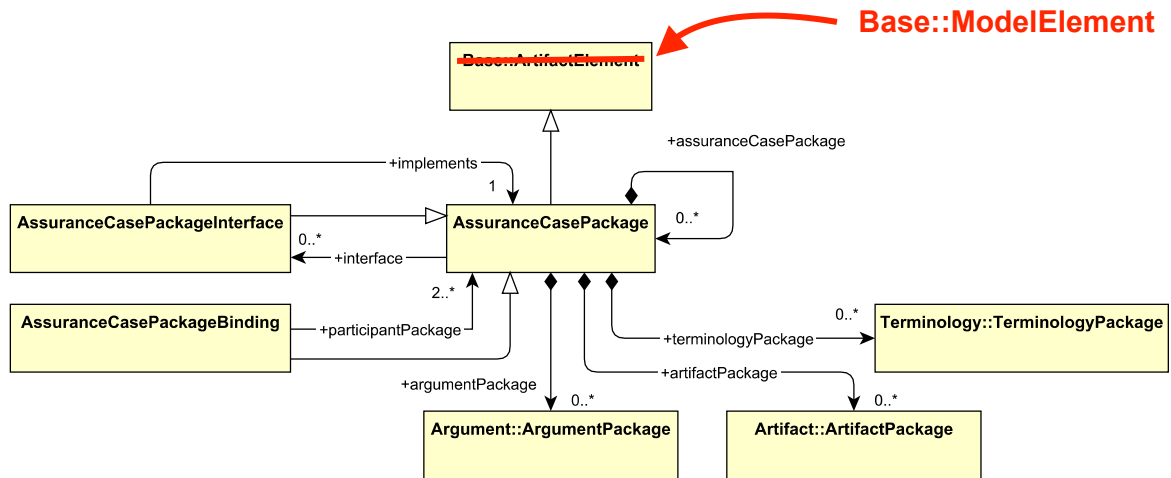


Figure 9.1 - Structured Assurance Case Packages Class Diagram

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

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

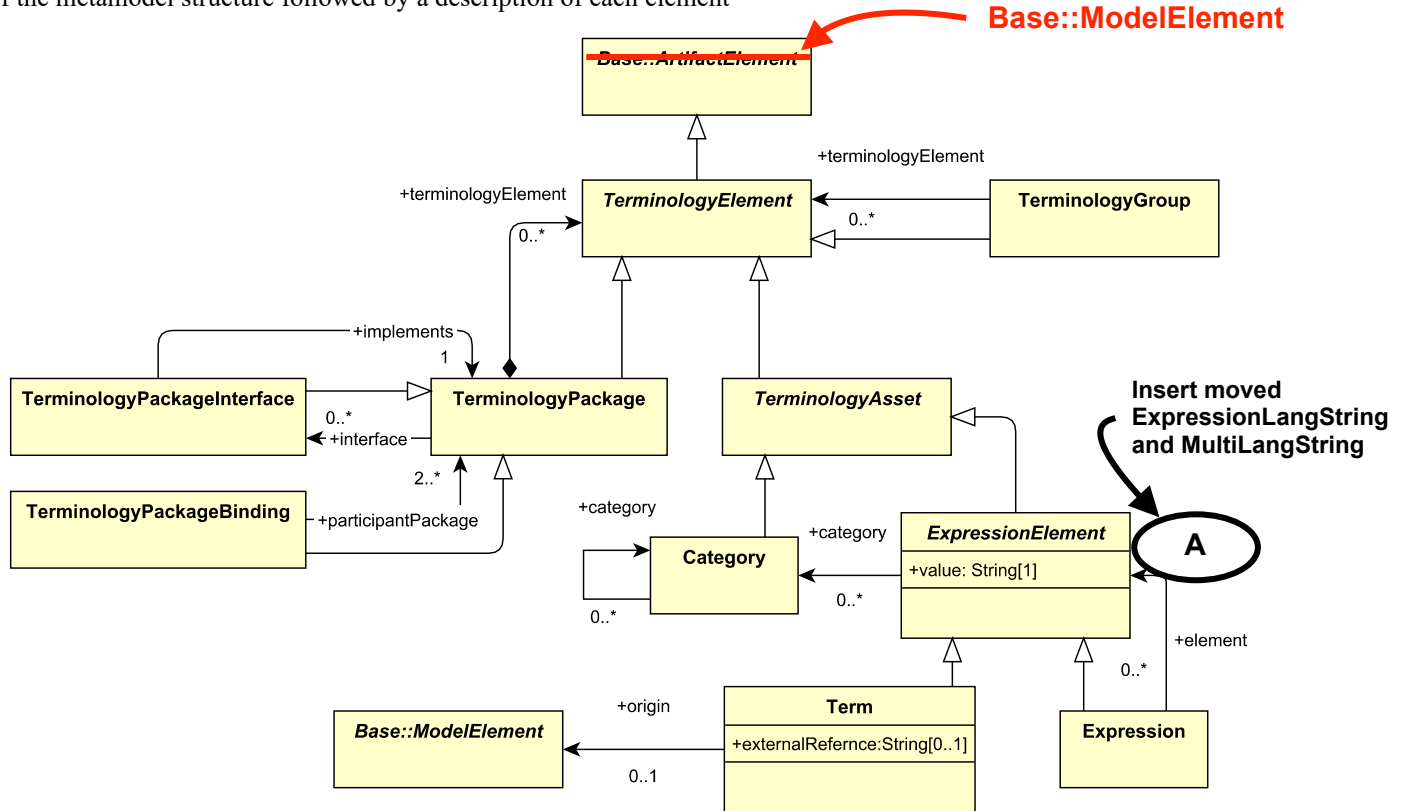


Figure 10.1 - Terminology Class Diagram

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).

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 ArgumentPackages 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.

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

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.

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

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).

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., assertionDeclaration = 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”.

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.

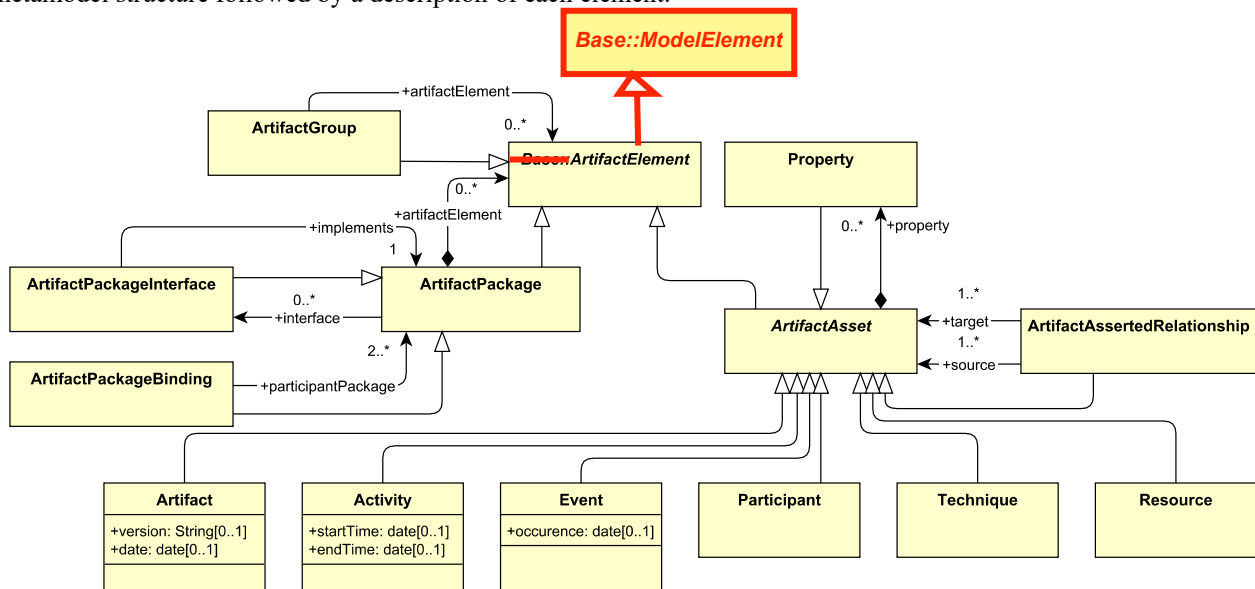


Figure 12.1 - Artifact Package Diagram

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

Attributes

version: String[0..1] - the version of the artifact

date: date[0..1] - the date on which the artifact was created.

Semantics

Artifacts correspond to the main evidentiary support for the arguments and claims of an assurance case: an Artifact can play the role of evidence of a Claim (AssertedEvidence), or of counterevidence (AssertedCountedEvidence with isCounter = true). An Artifact can take several forms, such as a diagram, a plan, a report, or a specification, both in electronic (e.g., a pdf file) or physical (e.g., a paper document) formats. Typical examples of Artifacts include system lifecycle plans, dependability (e.g., safety) analysis results, system specifications, and V&V results.

12.8 Property

Property enables the specification of the characteristics of an Artifact.

Superclass

ArtifactAsset

Semantics

An Artifact can have different, specific characteristics independent of the argumentation structure in which the Artifact is used. Some can be objective (e.g., the result of a test case execution, as passed or not passed) and others can be based on a person's judgement (e.g., regarding a quality aspect of a report).

12.9 Event

Event enables the specification of the events in the lifecycle of an Artifact.

Superclass

ArtifactAsset

Attributes

date: date[0..1] - the date on which the Event occurred.

Semantics

Artifacts change during their lifecycle, and different types of happenings can occur at different moments: creation, modification, revocation... Events serve to maintain a history log of an Artifact, and can be consulted to know how an Artifact has evolved and to develop confidence in its adequate management.

12.10 Resource

Resource corresponds to the tangible objects representing an Artifact.

Superclass

ArtifactAsset

Attributes

location:Base::MultiLangString (composition) – the path or URL specifying the location of the Resource, can be in multiple languages.

Semantics

Artifacts are located and accessible somewhere, usually in the form of some electronic file for an assurance case. Such information is specified by means of Resources.

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.

Moved from Artifact Class

12.15 ArtifactElement (abstract)

ArtifactElement acts as the base class for elements in other SACM packages. Essentially, all elements which extend ArtifactElement is considered to be an artifact, and therefore can be referenced using Argument:ArtifactReference.

Superclass

ModelElement

Semantics

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