8.6 ModelElement (abstract)

ModelElement is the base element for the majority of modeling elements.

**Superclass**
SACMElement

**Associations**

- implementationConstraint: ImplementationConstraint [0..*] (composition) – a collection of implementation constraints.
- description: Description[0..1] (composition) – the description of the ModelElement.
- note:Note[0..*] (composition) – a collection of notes for the ModelElement.
- taggedValue: TaggedValue [0..*] (composition) – a collection of TaggedValues, TaggedValues can be used to describe additional features of a ModelElement

**Semantics**

All the individual and identifiable elements of a SACM model correspond to a ModelElement.

**Constraints**

ImplementationConstraints should only be specified if +isAbstract is true OCL: self.implementationConstraint->size() > 0 implies self.isAbstract = true

8.7 UtilityElement (abstract)

UtilityElement is an abstract element for a number of utility elements.

**Superclass**
SACMElement

**Associations**

- expression: Expression [1] – the expression object containing the value of the UtilityElement (see Terminology section 10)

**Semantics**

UtilityElement supports the specification of additional information for a ModelElement.

8.8 ImplementationConstraint

This class 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).

**Constraints**

ImplementationConstraints should only specified if isAbstract is true.

8.9 Description

This class specifies 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.

8.10 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.

8.11 ArtifactElement (abstract)
8.10 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 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 ArtifactElement (abstract)

ArtifactElement is an abstract class that serves as a parent class for Artifacts and AssuranceCasePackage elements.

Superclass

ModelElement

Semantics

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

9.3 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 ofArtifactElement. Semantically an AssuranceCasePackage can be considered as an artifact of evidence (e.g. from the perspective of another AssuranceCasePackage).
argumentPackage:Argument::ArgumentPackage[0..*] (composition) – a number of optional argument packages.

**Superclass**
ArtifactElement

**Associations**
- assuranceCasePackage: AssuranceCasePackage [0..*] – a number of optional sub-packages
- interface: AssuranceCasePackageInterface [0..*] – a number of optional assurance case package interfaces that the current package may implement
- artifactPackage: ArtifactPackage [0..*] – a number of optional artifact sub-packages
- terminologyPackage: TerminologyPackage [0..*] – a number of optional terminology sub-packages

**Semantics**
AssuranceCasePackage is the root class for creating structured assurance cases.

### 9.4 AssuranceCasePackageInterface
AssuranceCasePackageInterface is a kind of AssuranceCasePackage that defines an interface that may be exchanged between users. An AssuranceCasePackage may declare one or more ArtifactPackageInterfaces.

**Superclass**
AssuranceCasePackage

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

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

### 9.5 ArgumentPackage
ArgumentPackage is a container for the structured argument aspect of the assurance case. It contains the structure of assertions which comprise the structured argument.

**Superclass**
ArgumentationElement

**Associations**
- argumentPackage: ArgumentPackage [0..*] – an optional set of sub ArgumentPackages, allowing for recursive containment
- argumentAsset: ArgumentAsset [0..*] an optional set of ArgumentAssets

**Semantics**
ArgumentPackage is the base class for specifying the results of the argumentation efforts for a structured assurance case (i.e., an AssuranceCase).

### 9.6 TerminologyPackage
TerminologyPackage is a container element for terminology that may be exchanged. Terminology can define terms, expressions or categories, used elsewhere in the assurance case.

**Superclass**
TerminologyElement

**Associations**
- terminologyAsset: TerminologyAsset [0..*] – an optional set of terminology assets (expressions, terms and categories)

**Semantics**
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.

![Terminology Class Diagram]

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).

TerminologyElement extends Base::ArtifactElement, this implies that all elements in the Terminology package are artifacts.

10.3 TerminologyPackage

The TerminologyPackage is the container element for SACM terminology assets.

**Superclass**
11.3 **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.

11.4 **ArgumentPackage Class**

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

**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 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.

11.5 **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 AssertedRelationship) must be ArgumentAssetCitations to Claims or ArtifactElementCitations contained within the ArgumentPackages associated by the participantPackage association.

11.6 **ArgumentPackageInterface Class**

**Superclass**
ArgumentPackage
ArtifactPackage Interface 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.

### 12.5 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 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.

### 12.6 Artifact class

The Artifact class represents the distinguishable units of data used in an assurance case.

**Superclass**
ArtifactAsset

**Attributes**

- **version**: String
  The version of the Artifact
- **date**: Date
  The date on which the artifact was created.

**Associations**

- **artifactProperty::ArtifactProperty[0..*]**
  The ArtifactProperties of the Artifact
- **artifactEvent::ArtifactEvent[0..*]**
  The set of ArtifactEvents that represent the lifecycle of the Artifact

**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). 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.

### 12.7 ArtifactProperty class

The ArtifactProperty class enables the specification of the characteristics of an Artifact.

**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).

### 12.8 ArtifactEvent class

The ArtifactEvent class enables the specification of the events in the lifecycle of an Artifact.