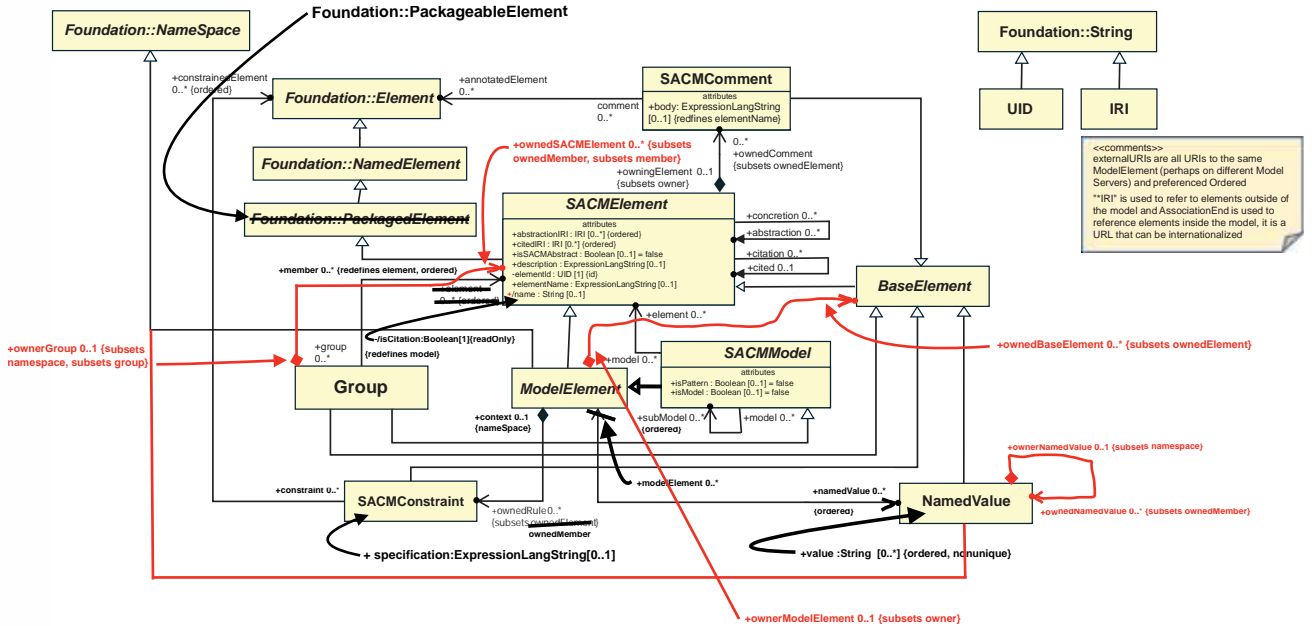


9 ~~8~~
9 ~~8.1~~

Structured Assurance Case Base Classes

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 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.

Superclass

MultiLangString

Attributes

expression:ExpressionEleme

~~expression:Terminology::ExpressionElement[1] (composition) – a reference to an ExpressionElement in the TerminologyPackage~~

Semantics

~~ExpressionLangString provides a means for description, it can also be used to link to an ExpressionElement in the Terminology package.~~

~~**Constraints**~~

~~If expression is not empty, then content should be empty.~~

9 ~~8.5~~ **MultiLangString**

MultiLangString, as its name suggests, provides a means to describe things using different languages.

Superclass

~~Element~~ Foundation::Element

Associatiions

~~value:LangString[1..*] (composition) – contains the descriptions which bear the same meaning but in different languages~~

content:String[0..*] {ordered,nonunique} – the content of the string.

Semantics

lang:String[0..*] {ordered} – a field to indicate the language used in the string.

MultiLangString provides a means to describing things using different languages. It contains a ~~list of LangString, which the user can specify their languages and the descriptions in the languages.~~

Constraints

set of contents each of which is in a different language represented by the lang attribute.

~~For each of the LangString in the value feature, their lang must be unique.~~

9 ~~8.6~~ **3 ModelElement (abstract)**

LangIfContentMoreThan1
If content has more than one value, then lang must have string representing the languages of the values.
inv: content->size() > 1 implies lang->size() = content->size()

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

Superclass

, Foundation::NameSpace

SACMElement

Associations

Associations
Attributes

~~ownedRule:SACMConstraint[0..*] (subsets ownedElement) – {subsets ownedMember}~~

~~name:LangString[1] (composition) – the name of the ModelElement.~~

~~ownedRole: SACMConstraint [0..*] – SACMConstraints owned by this ModelElement.~~

~~implementationConstraint: ImplementationConstraint [0..*] (composition) – a collection of implementation constraints.~~

~~ownedBaseElement [0..*] {subsets ownedElement} - BaseElements owned by this ModelElement~~

~~description: Description[0..1] (composition) – the description of the ModelElement.~~

~~note:Note[0..*] (composition) – a collection of notes for the ModelElement.~~

namedValue:NamedValue

{ordered}

NamedValues NamedValues

~~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.

Moved to, and revised in, 8.2

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~~ 8 **Note** SACMComment

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

Superclass BaseElement
~~SACMCoreElement~~
~~UtilityElement~~

Attributes
~~body:ExpressionLangString[0..1]~~
~~body:ExpressionLangString[0..1]~~
 AssociationEnds
 annotatedElement:Element[0..*] – Elements to which the SACMComment applies

Semantics
 SACMComments
 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~~ 9 **TaggedValue** NamedValue

This class represents a simple key/value pair ^{attributes} 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
 BaseElement, Foundation::Namespace
~~UtilityElement~~
~~Attributes~~

Associations
 value :String [0..*] {ordered, nonunique}

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

~~key:MultiLangString[1] (composition) – the key of the TaggedValue.~~
 Associations
 ownedNamedValue [0..*] {subsets ownedMember} - NamedValues owned by this NamedValue

9.10 Group

Group can be used to associate a number of SACMElements to a common group (e.g., representing a common type or purpose, or being of interest to a particular stakeholder).

Supertype BaseElement, SACMModel
~~SACMElement~~

Association End
~~element: SACMElement [0..*] {ordered} – a collection of SACMElements that comprise a group~~
~~member: SACMElement [0..*] {redefines element, ordered} - a collection of SACMElements that compromise a group~~

ownedSACMElement [0..*] {subsets ownedMember, subsets member} - SACMElements that are owned by this Group

~~9.11~~ 9.11 IRI

IRI is a international resource identifier is a String and is formatted according to RFC 3987.

Supertype String
~~String~~

Supertype ModelElement
 Foundation::String

Attributes
 isPattern : Boolean [0..1] = false - if true elements in SACMModel are part of a pattern.
 isModel : Boolean [0..1] = false - if true elements in SACMModel are part of a model.

Association
 +subModel:SACMModel [0..*] - subModel is a recursive containment of elements in the model.
 +elements:SACMElement [0..*] - elements are items in the model.

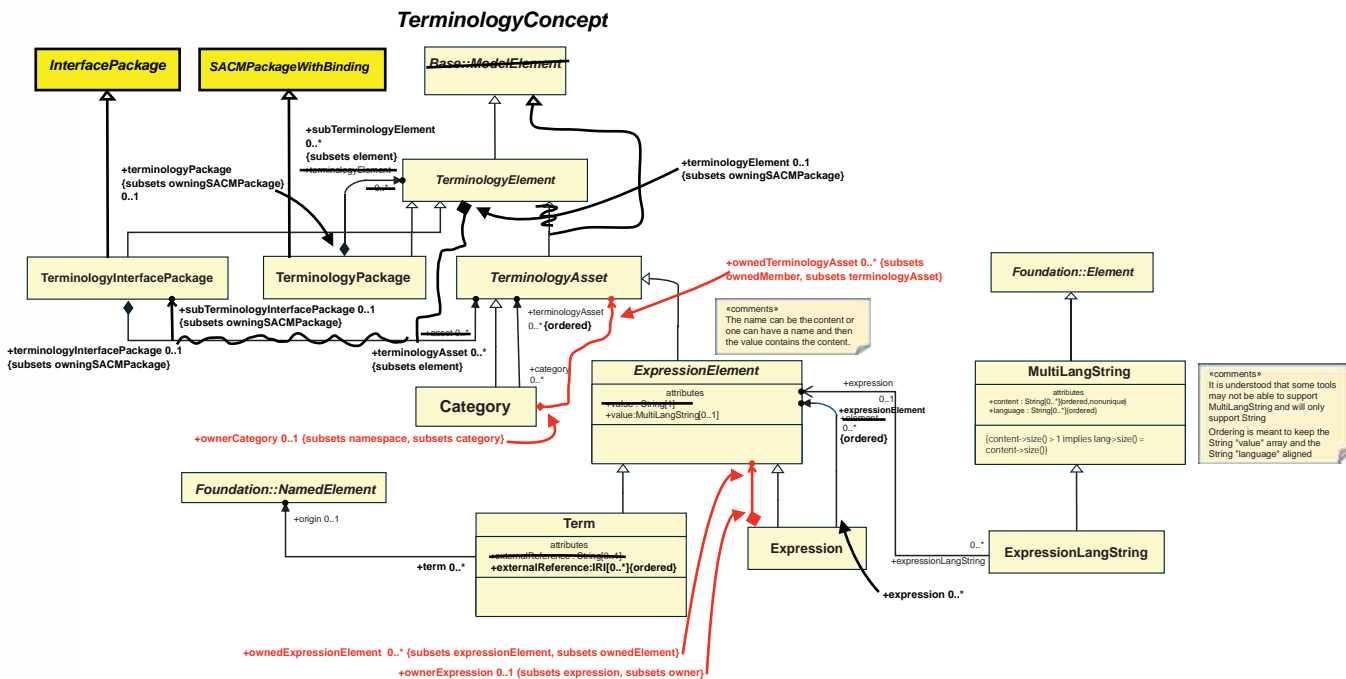
12
11-10

Structured Assurance Case Terminology Classes

12
11-10.1

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



12
11-10.1
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.

12
11-10.2

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::ModelElement TerminologyConcept

Base::ArtifactElement

Semantics

Association terminologyAsset [0..*] {subsets element} - TerminologyAssets that are contained in this TerminologyElement

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

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))
```

12 11 10.7⁶ TerminologyAsset (abstract)

The TerminologyAsset Class is the abstract class for the different types of terminology elements represented in SACM.

Superclass

~~TerminologyConcept~~
~~TerminologyElement~~

Semantics

TerminologyAssets represent all of the elements required to model and categorize expressions in SACM (expressions and terminology categories).

12 11 10.8⁷ Category

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

Superclass

TerminologyAsset

Semantics

Association Ends
~~expressionElement:ExpressionElement~~[0..*]{ordered} – elements contained in the Category
~~terminologyAsset:TerminologyAsset~~ +ownedTerminologyAsset 0..* {subsets ownedMember, subsets terminologyAsset} - TerminologyAssets owned by this TerminologyAsset.

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.

12 ~~11~~10.9⁸ ExpressionElement (abstract)

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

Superclass

TerminologyAsset

Attributes

~~value:String[1] the value of the expression.~~

~~value:MultiLangString[0..1]~~

~~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.

← The name can be the content or one can have a name and then the value contains the content.

12 ~~11~~10.10⁹ Expression

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

Superclass

ExpressionElement expressionElement

Associations

~~element:ExpressionElement[0..*]{ordered}~~

~~element:ExpressionElement [0..*] - an optional reference to other ExpressionElements forming part of the structured Expression.~~

ownedExpressionElement [0..*] {subsets expressionElement, subsets ownedElement} - ExpressionElements owned by this Expression

Semantics

Expressions are used to model phrases and sentences. These are defined using the value feature. Alternatively, the expression can also be defined (using the value feature) as a production rule involving other ExpressionElements. In this case, the value must use a suitable (string) form for denoting the position of involved ExpressionElements (e.g. “\$<ExpressionElement.name>\$”) within the production rule, and expressing production rule operators (e.g. Extended Backus-Naur Form operators).

Constraints

Where an Expression has associated ExpressionElements (the +element feature), these should be referenced by name within the +value feature.

Where the +value feature references ExpressionElement by name, these ExpressionElements should be associated (using the +element feature) with Expression. A concrete expression should have references to only concrete ExpressionElements

OCL:

self.isAbstract = false implies self.element->forall(expr|expr.isAbstract = false).

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(glg.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)

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.~~

13 12 11.7 6 ArgumentAsset (abstract)

ArgumentAsset is the abstract base element for the elements of any structured argument represented in SACM.

Superclass

~~Argument~~ ~~ArgumentationElement~~ ~~ArgumentConcept~~

Semantics

Association
 ownedArgumentAsset:ArgumentAsset[0..*] {subsets ownedElement} -
 ArgumentAssets owned by this ArgumentAsset.

ArgumentAssets represent the constituent building blocks of any structured argument contained in an ArgumentPackage.

For example, ArgumentAssets can represent the Claims made within a structured argument contained in an ArgumentPackage.

13 12 11.8 7 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~~ (default), 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.

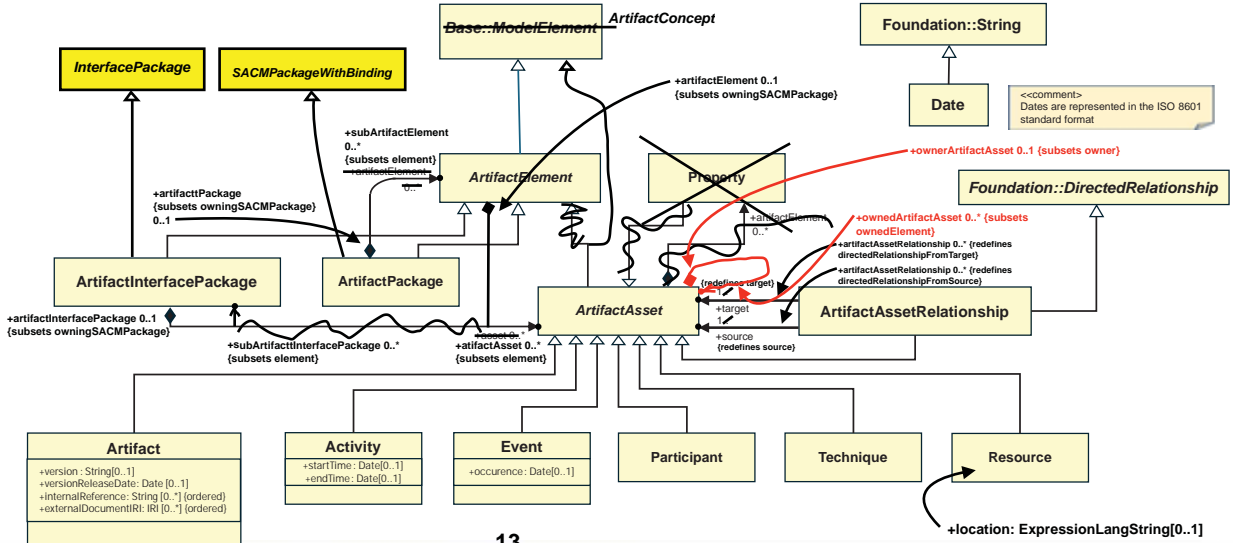
13 12 11.9 8 ArtifactReference

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

Artifact Classes

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.



13
Figure 13.1 - 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

~~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.~~

14
~~13~~12.5⁴ **ArtifactPackageInterface**

~~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.~~

Superclass

~~ArtifactElement
ArtifactPackage~~

ArtifactInterfacePackage

subArtifactInterfacePackage [0..1] {subsets owningSACMPackage} -
ArtifactInterfacePackages that are contained in this ArtifactInterfacePackage

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
ArtifactPackage with which this ArtifactPackageInterface is associated.~~

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(glg.owningPackage).asSet()->one(plp=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)

14
~~13~~12.6⁵ **ArtifactAsset (abstract)**

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

Superclass

~~ArtifactConcept
Base::ArtifactElement~~

Association

Association

ownedArtifactAsset:ArtifactAsset[0..*] {subsets ownedElement} - ArtifactAssets owned by this ArtifactAsset.

~~property:Property[0..*] (composition) an optional collection of Property(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.

14
~~13~~12.7⁶ **Artifact**

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

Superclass

ArtifactAsset