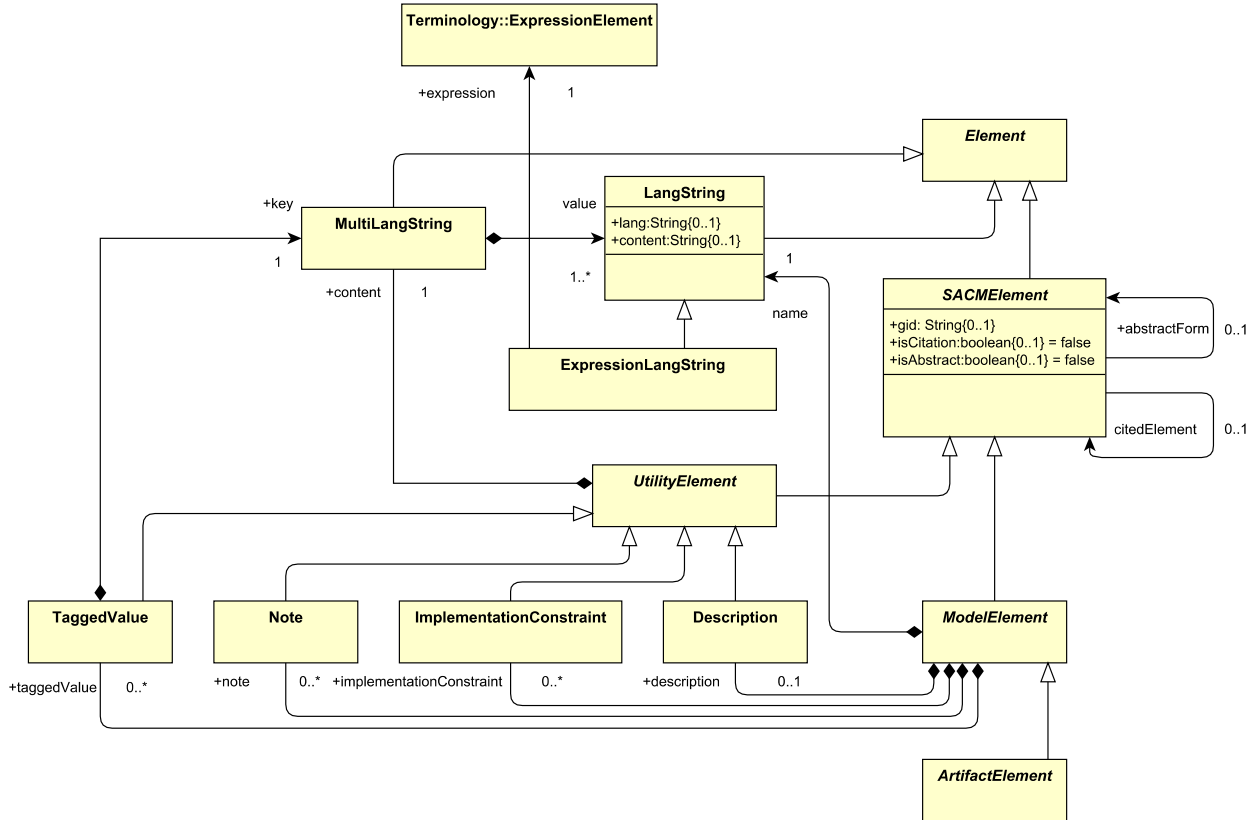


9 ~~8~~ Structured Assurance Case Base Classes

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
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.

9 ~~8.2~~ SACMElement (abstract)

SACMElement is the base class for SACM.

Superclass

MOF:Element

Attributes

gid:String[0..1] – a unique identifier that is unique within the scope of the model instance

isCitation[0..1]=false – a flag to indicate whether the SACMElement cites another SACMElement.

isAbstract[0..1]=false – a flag to indicate whether the SACMElement is considered to be abstract. For example, this can be used to indicate whether an element is part of a pattern or template.

Associations:

citedElement:SACMElement[0..1] – a reference to another SACMElement that the SACMElement cites

abstractForm:SACMElement[0..1] – an optional reference to another abstract SACMElement to which this concrete SACMElement conforms.

Semantics

All the elements of a structured assurance case effort created with SACM correspond to a SACMElement.

Constraints:

If citedElement is populated, isCitation must be true. OCL: self.citedElement <> null implies self.isCitation = true

When +abstractForm is used to refer to another SACMElement, +isAbstract of the SACMElement is false, and the +isAbstract of the referred SACMElement should be true. The referred SACMElement should be of the same type of the SACMElement. If ImplementationConstraints are expressed on the referred SACMElement, the SACMElement should satisfy these ImplementationConstraints.

9 ~~8.3~~ LangString

LangString is the format SACM uses for description. It serves the same purpose as String but with the additional specification of the language used for the content.

Superclass

MOF:Element

Attributes

lang:String[0..1] – a field to indicate the language used in the string.

content:String[0..1] – the content of the string

Semantics

LangString serves the same purpose as String, SACM uses LangString for description, which containing the information of the language it uses in the content.

9 ~~8.4~~ ExpressionLangString

ExpressionLangString is used to denote a structured expression, it contains a description (LangString) and it also (optionally) points to an ExpressionElement in the Terminology Package.

Superclass

LangString

Attributes

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

Associations

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

Semantics

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

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

9 ~~8.6~~ ModelElement (abstract)

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

Superclass

SACMElement

Associations

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

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

9 ~~8.7~~ 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~~ 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~~ 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.

9 ~~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 ~~8.11~~ 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~~ 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.

Foundation Class

Foundation provides a semantic backing to the SACM Model from UML/KerML like models to make mapping (e.g. through Profiles) into UML/SysMLv1 or SysMLv2 more regular. Many of the textual entries in this section come directly from the UML 2.5.1 metamodel.

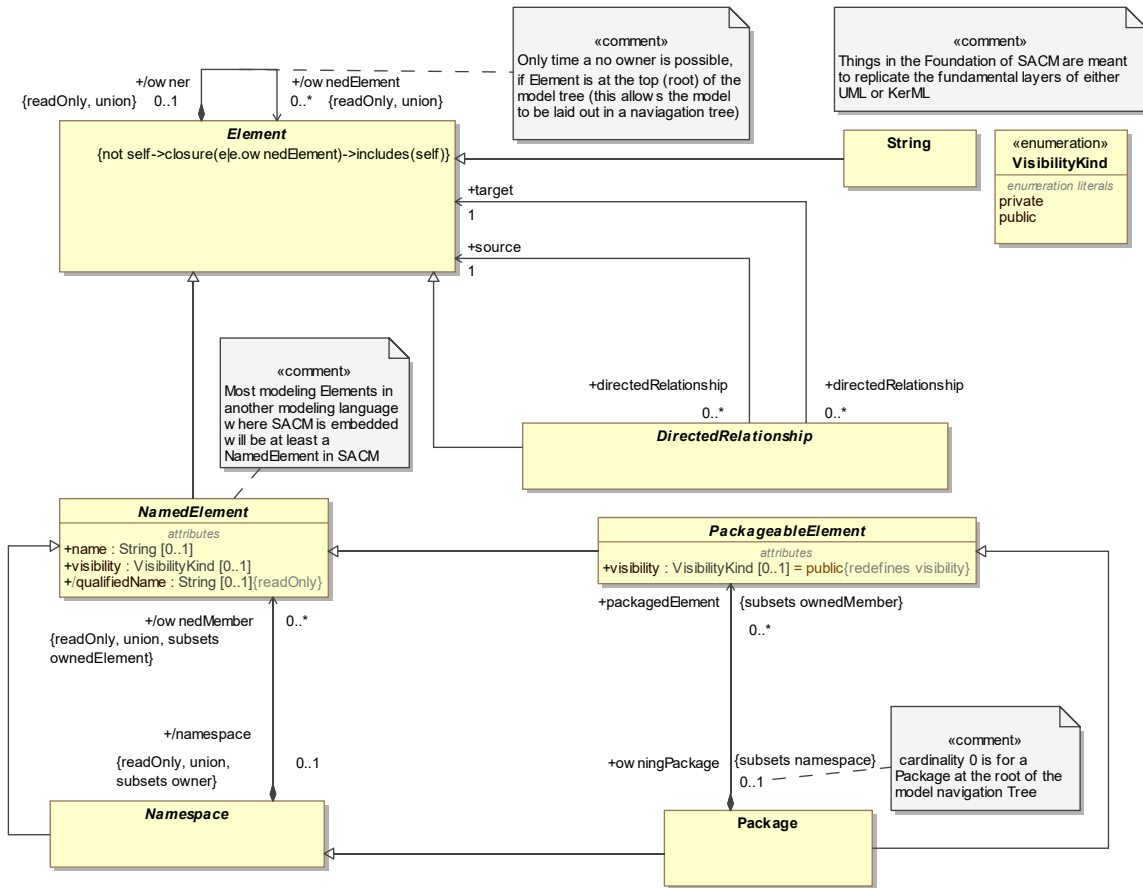


Figure 8.1 - Foundation Class

DirectedRelationship [Abstract Class]

A *DirectedRelationship* represents a relationship between a source model Element and target model Element. (This represents a more restrictive, than UML, Binary Relationship.)

Association Ends

/source : Element [1]

Specifies the source Element of the DirectedRelationship.

/target : Element [1]

Specifies the target Element of the DirectedRelationship.

8.2 Element [Abstract Class]

An Element is a constituent of a model. As such, it has the capability of owning other Elements.

Association Ends

/ownedElement : Element [0..*]{union}

The Elements owned by this Element.

/owner : Element [0..1]{union}

The Element that owns this Element.

Constraints

not_own_self

An element may not directly or indirectly own itself.

inv: not self->closure(e | e->ownedElement)->includes(self)

8.3 NamedElement [Abstract Class]

A NamedElement is an Element in a model that may have a name.

Attributes

name : String [0..1]

The name of the NamedElement.

/qualifiedName : String [0..1] {readOnly}

A name that allows the NamedElement to be identified within a hierarchy of nested Namespaces. It is constructed from the names of the containing Namespaces starting at the root of the hierarchy and ending with the name of the NamedElement itself. (Separator of names is "::".)

visibility : VisibilityKind [0..1]

Determines whether and how the NamedElement is visible outside its owning Namespace.

Association Ends

/namespace : Namespace [0..1]{union}

Specifies the Namespace that owns the NamedElement.

Constraints

visibility_needs_ownership

If a NamedElement is owned by something other than a Namespace, it does not have a visibility.

inv: (namespace = null and owner <> null) implies visibility = null

8.4 Namespace [Abstract Class]

A Namespace is an Element in a model that owns and/or imports a set of NamedElements that can be identified by name.

Association Ends

/ownedMember : NamedElement [0..*]{union, subsets Element::ownedElement}

A collection of NamedElements owned by the Namespace.

Package [Class]

A package can have one or more profile applications to indicate which profiles have been applied. Because a profile is a package, it is possible to apply a profile not only to packages, but also to profiles.

attributes

packageableElement : PackageableElement [0..*]{subsets Namespace::ownedMember}

Specifies the packageable elements that are owned by this Package.

8.5 PackageableElement [Abstract Class]

A PackageableElement is a NamedElement that may be owned directly by a Package.

Attributes

visibility : VisibilityKind [0..1] = public

A PackageableElement must have a visibility specified if it is owned by a Namespace. The default visibility is public.

VisibilityKind [Enumeration]

VisibilityKind is an enumeration type that defines literals to determine the visibility of Elements in a model.

Literals

public

A Named Element with public visibility is visible to all elements that can access the contents of the Namespace that owns it.

private

A NamedElement with private visibility is only visible inside the Namespace that owns it.