

8 Foundation Class

8.1 General

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.

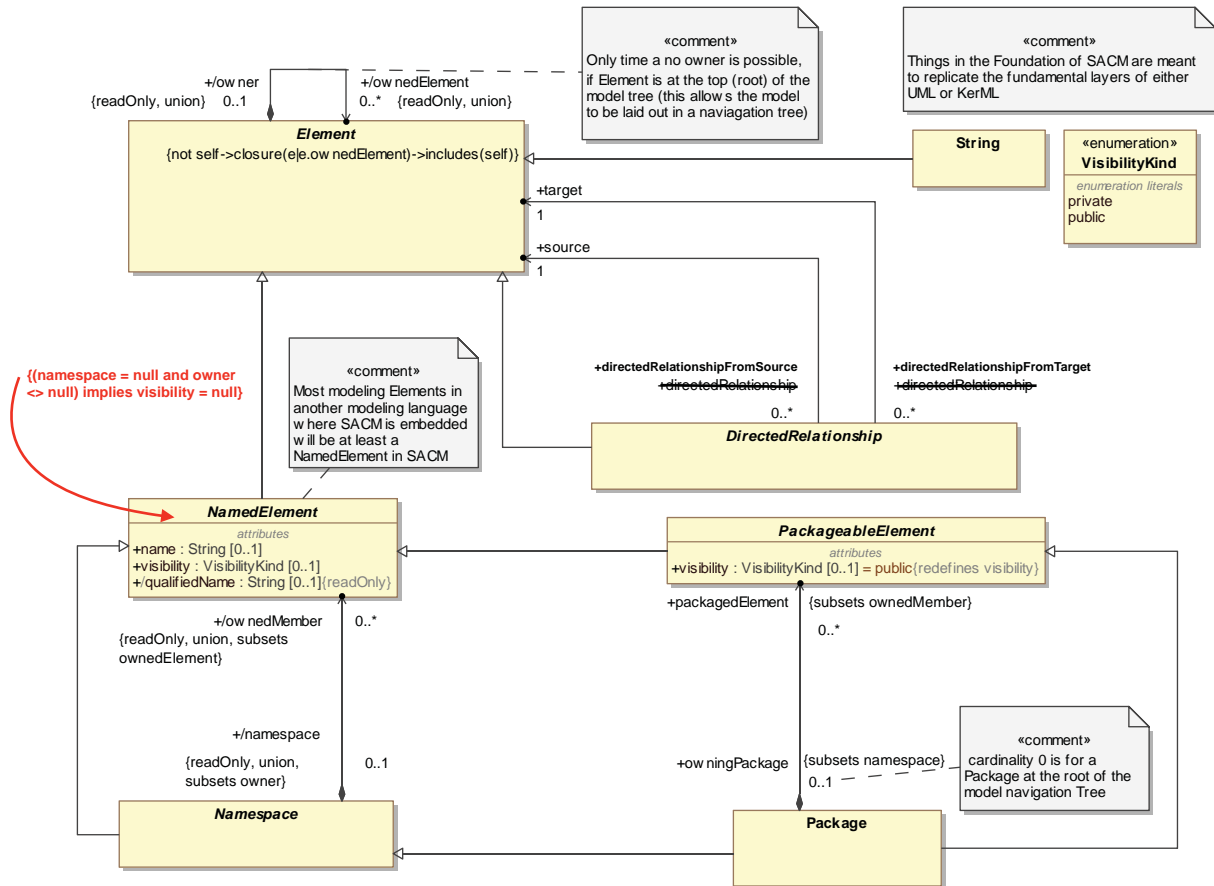


Figure 8.1 - Foundation Class

8.1 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
~~source~~ : Element [1]

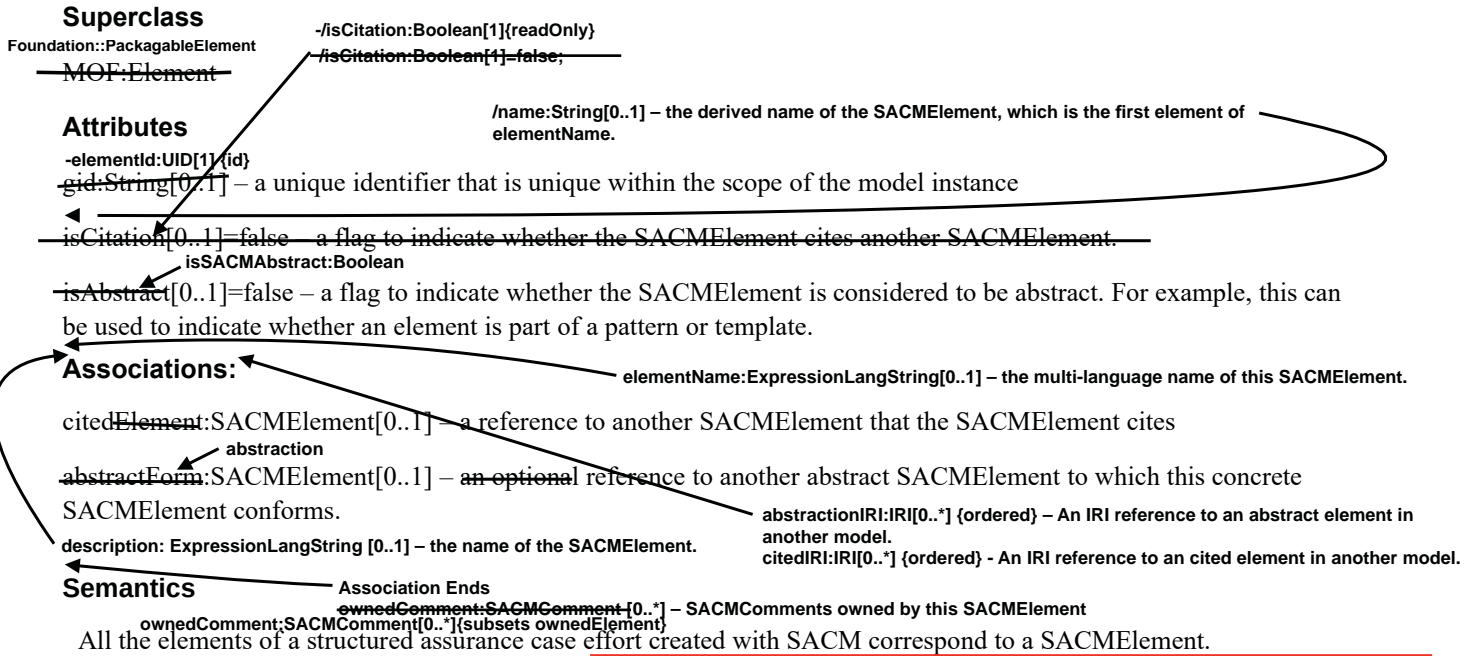
Specifies the source *Element* of the *DirectedRelationship*.

target
~~target~~ : Element [1]

Specifies the target *Element* of the *DirectedRelationship*.

9 8.2 SACMElement (abstract)

SACMElement is the base class for SACM.



Constraints:

If citedElement is populated, isCitation must be true.

When +abstractForm is used to refer to another SACMElement, +isAbstract of the referred SACMElement should be true.

If ImplementationConstraints are used, they should satisfy these ImplementationConstraints.

AbstractionIsAbstract
An abstraction for a SACMElement must have isSACMAbstract=true.
inv: abstraction->notEmpty() implies abstraction.isSACMAbstract=true

CitationKindOfCited
A citation must be of the same kind as the cited.
inv: cited->notEmpty() implies ocKindOf(cited)

DeriveCitation
isCitation is true if cited or citationIRI is not empty.
inv: isCitation = citedIRI->notEmpty() or cited->notEmpty()

DerivedName
If there is an elementName and it has content, then name is the first of that content.
inv: elementName->notEmpty() and elementName.content->notEmpty() implies name=elementName.content->first()

IfCitedAbstractCitationAbstract
If cited and cited is abstract, then citation is abstract.
inv: cited->notEmpty() and cited.isSACMAbstract=true implies isSACMAbstract=true

9 8.3 LangString

LangString is the format SACM uses for description 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.

Move to Terminology Class

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

Foundation::Package, Base::SACMMModel

AssociationEnds

+/element:SACMElement[0..*] {union,readOnly,ordered,redefines element, redefines packagedElement} - elements in this package

+baseElement:BaseElement[0..*] {subsets element} - BaseElement subset of element

10.2 Concept (abstract)

Abstraction for TerminologyConcept, ArtifactConcept, and ArugmmentConcept. A Concept includes assets and elements (packaging) from the various different domains.

Superclass

Base::ModelElement

10.3 ScopedPackage (abstract)

Cited elements in a scoped package are restricted to be either siblings to the scoped package (i.e. has the same parent package) or one of their parents (recursively) has the same parent package.

Superclass

SACMPackage

CitedElementsAreScoped

If elements in the package are citations, then the cited must be members (recursively) in the same package structure as the ScopePackage.

inv: element->forAll(e|e.cited->notEmpty() implies e.cited->closure(g|g.owningPackage).asSet()->one(p|p=self.owningPackage))

Constraints

~~inv CitedElementsAreScoped: element->forAll(c|c.cited->notEmpty() implies c.cited->closure(g|g.owningPackage).asSet()->one(p|p=self.owningPackage))~~

10.3 BindingPackage

A BindingPackage contains cited elements and additional elements that allow the stitching together or other domain packages.

Superclass

SACMPackageWithBinding, ScopedPackage

AssociationEnds

+concept:Concept[0..*] {subsets element} - concepts contained in this binding package.

10.3 SACMPackageWithBinding (abstract)

An abstract of packages which can contain a binding package.

Superclass

SACMPackage

AssociationEnds

+bindingPackage:BindingPackage[0..*] {subsets element} - binding packages contained in this package

10.3 InterfacePackage (abstract)

Abstraction for TerminologyInterfacePackage, ArtifactInterfacePackage, and ArugmmentInterfacePackage. Interface packages can have BaseElements and Diagrams as well as their respective domain concepts, but those domain concepts must be citations.

Superclass

ScopedPackage

Constraints

~~inv MustBeCited: element ->forAll(e|not e ->oclKindOf(SACMDiagram) and not e ->oclKindOf(BaseElement) implies e.isCitation=true)~~

MustBeCited

All elements in an interface except BaseElement and Diagrams must be citations.

inv: element->forAll(e|not e->oclKindOf(SACMDiagram) and not e->oclKindOf(BaseElement) implies e.isCitation=true)

Superclass
ModelElement
~~AssuranceCasePackage~~

Associations

participantPackage:AssuranceCasePackage[2..*] – references to AssuranceCasePackages which the AssuranceCasePackageBinding binds together. The Participant Package provides access to its elements through a Package Interface, if a Package Interface is named in the Package Binding, or to all elements if the Package is named in the Package Binding.

Semantics

AssuranceCasePackageBinding binds peer AssuranceCasePackages together to indicate the relationship between these AssuranceCasePackages. The bindings between AssuranceCasePackages consist of the bindings of the packages (i.e. ArgumentPackageBindings, ArtifactPackageBindings and TerminologyPackageBindings) contained in the AssuranceCasePackages, together with an optional ArgumentationPackage that asserts the relationship between +participantPackage. The AssuranceCasePackageBinding includes a reference to each participant Package which can be either an AssuranceCasePackage or an AssuranceCasePackageInterface and that are external to Packages to the AssuranceCasePackage that contains this AssuranceCasePackageBinding. Within the AssuranceCasePackageBinding, there might be zero or more ArgumentPackageBinding, ArtifactPackageBinding, or TerminologyPackageBinding. Each of these that exist would refer to at least two participant packages that could be a Package or PackageInterface of the same type.

~~**Constraints** 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~~

~~The participantPackages should be either AssuranceCasePackage or AssuranceCasePackageInterfaces.~~

~~**OCL:**~~

~~self.participantPackage->forall(pp|pp.oclsTypeOf(AssuranceCase::AssuranceCasePackage) or
pp.oclsTypeOf(AssuranceCase::AssuranceCasePackageInterface))~~

~~inv CitedElementsAreScoped: element->forall(e|e.cited->notEmpty() implies e.cited->closure(glg.owningPackage).asSet()->one(plp=self.owningPackage))~~

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).~~

IfConcreteEverythingConcrete

If Expression is not abstract, then none of its ExpressionElements can be abstract.

inv: isSACMAbstract = false implies expressionElement->forall(e|e.isSACMAbstract = false)

12 ~~11~~ 10.11/0 Term

Term is used to model both abstract and concrete terms in SACM. Abstract Terms can be considered placeholders for concrete terms and are denoted by the inherited isAbstract:Boolean attribute being set true. A concrete term is denoted by isAbstract:Boolean being false.

Superclass

ExpressionElement

Attributes externalReference:IRI[0..*]{ordered}

~~externalReference:String[0..1]~~ – an attribute recording an external reference (e.g., URI) to the object referred to by the Term

Associations Foundation::NamedElement

~~origin:Base::ModelElement~~[0..1] – a reference which points to the origin of the Term.

Semantics

Term class is used to model both abstract and concrete terms in SACM. Abstract Terms can be considered placeholders for concrete terms and are denoted by the inherited isAbstract:Boolean attribute being set true. A concrete term is denoted by isAbstract:Boolean being false.

The externalReference attribute enables the referencing of the object signified by the term (i.e., the signifier). It also provides a mechanism whereby terms can reference concepts and terms defined in other ontology and terminology models.

12.11 TerminologyConcept (abstract)

TerminologyConcept is an abstraction of all the packaging and elements in the terminology domain.

Superclass

Packaging::Concept

Constraints

OriginHasName

If a Term has an origin, then that origin must have a name.
inv: origin->notEmpty() implies origin.name->notEmpty()

OriginEmptyNameNotEmpty

If the origin is empty, then the Term must have a name.
inv: origin->isEmpty() implies name->notEmpty()

If AssertedInference has isCounter=false, then it should be interpreted as "if the inferrer is undefeated then the inferred is undefeated". If AssertedInference has isCounter=true, then it should be interpreted as "If the inferrer is undefeated then the inferred is defeated".

AssertedInference between two claims (A – the source – and B – the target) denotes that the truth of Claim A is said to infer the truth of Claim B.

13 14
~~12 11.15~~ **AssertedEvidence**

AssertedEvidence association records the declaration that one or more artifacts of Evidence (cited by ArtifactReference) provide information that helps establish the truth of a Claim. It is important to note that such a declaration is itself an assertion on behalf of the user. The artifact (cited by an ArtifactReference) may provide evidence for more than one Claim.

Superclass **Associations**
 AssertedRelationship +assertion:Assertion [1] - the assertion is that the evidence for this Assertion is enough to make the AssertedEvidence undefeated.
Semantics +evidence:ArtifactReference [1] - the assertion is that the evidence for this Assertion is enough to make the AssertedEvidence undefeated.

Where evidence (cited by ArtifactReference) exists that helps to establish the truth of a Claim in the argument, this relationship between the Claim and the evidence can be asserted by an AssertedEvidence association. An AssertedEvidence association between an artifact cited by an ArtifactReference and a Claim (A – the source evidence cited – and B – the target claim) denotes that the evidence cited by A is said to help establish the truth of Claim B.

Constraints If AssertedEvidence has isCounter=false, then it should be interpreted as "the evidence undefeats the Assertion". If AssertedEvidence has isCounter=true, then it should be interpreted as "the evidence defeats the Assertion".
 The source of AssertedEvidence relationships must be ArtifactReference.

~~OCL:~~

~~self.source -> forall(s|s.oellIsTypeOf(ArtifactReference))~~

AssertedContext can be used to declare that the Artifact or ArtifactReference provides the context for the interpretation and scoping of an ArgumentAsset (ArtifactReference, Artifact, Claim, ArgumentReasoning, or any AssertedRelationship specialization element).

13 15
~~12 11.16~~ **AssertedContext**

~~AssertedContext can be used to declare that the artifact cited by an ArtifactReference(s) provides the context for the interpretation and scoping of a Claim or ArgumentReasoning element. In addition, the AssertedContext can be used to declare a Claim asserted as necessary context (i.e. a precondition) for another Assertion or ArgumentReasoning.~~

Superclass **Associations**
 AssertedRelationship +context:ArtifactReference [1] - the context provides further clarification or constraint of the contextable
Semantics +contextable:ArgumentAsset [1] - the context provides further clarification or constraint of the contextable

~~Contextual information often needs to be cited in order to make clear the interpretation and scope of an Assertion and supporting argumentation. For example, a Claim can be said to be valid only in a defined context ("Claim A is asserted to be true only in a context as defined by the ArtifactReference B or conversely ArtifactReference B is the asserted context for Claim A").~~

~~Contextual Claims often need to be cited as preconditions for an Assertion. For example, a Claim may be asserted only in the context of another claim ("Claim A is asserted to be true only in a context where Claim B is true").~~

16
~~12 11.17~~ **AssertedArtifactSupport**

AssertedArtifactSupport records the assertion that ~~Contextual information often needs to be cited in order to make clear the interpretation and scope of an ArgumentAsset. For example, a Claim can be said to be valid only in a defined context ("Claim A is asserted to be true only in a context as defined by the ArtifactReference B or conversely ArtifactReference B is the asserted context for Claim A").~~

Superclass **Semantics**
 AssertedRelationship
 The truth of the assertions associated with an artifact are supported by the assertions that are associated with one or more other artifacts. Note: this can be declared as a constraint.
Constraints If AssertedContext has isCounter=false, then it should be interpreted as "the contextable must be interpreted in the context". If AssertedContext has isCounter=true, then it should be interpreted as "the contextable must not be interpreted in the context".

The source and target of AssertedArtifactSupport must be of type ArtifactReference.

13 17
~~12 11.18~~ **AssertedArtifactContext**

~~AssertedArtifactContext records the assertion that one or more artifacts provide context for another artifact.~~

Structured Assurance Case Metamodel, v2.3 **13.18 ArgumentConcept (abstract)**
 ArgumentConcept is an abstraction of all the packaging and elements in the argument domain.
 Superclass
 Packaging::Concept

internalReference : String [0..*] {ordered} - reference can be further restricted to some internal part for what the Artifact represents.
externalDocumentIRI : IRI [0..*] {ordered} - IRI to the external document that this Artifact element represents.
IRIs are ordered to allow preferred references appearing earlier in the ordering. All references should be to the same external document.

Attributes

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

~~13~~ ¹⁴ ~~12.8~~ ⁷ Property

~~Property enables the specification of the characteristics of an Artifact.~~

~~Superclass~~

~~ArtifactAsset~~

Constraints

OwnerIsArtifactConcept

Artifacts can only be owned in namespaces that are ArtifactConcepts.

inv: namespace->oclTypeOf(ArtifactConcept)

~~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).~~

~~13~~ ¹⁴ ~~12.9~~ ⁷ Event

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

Superclass

ArtifactAsset

Attributes

date: ^Ddate[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.

~~13~~ ¹⁴ ~~12.10~~ ⁸ Resource

Resource corresponds to the tangible objects representing an Artifact.

Superclass

ArtifactAsset

Attributes

location: ExpressionLangString[0..1]

~~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.