13 SACM Argument Metamodel

13.1 General

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

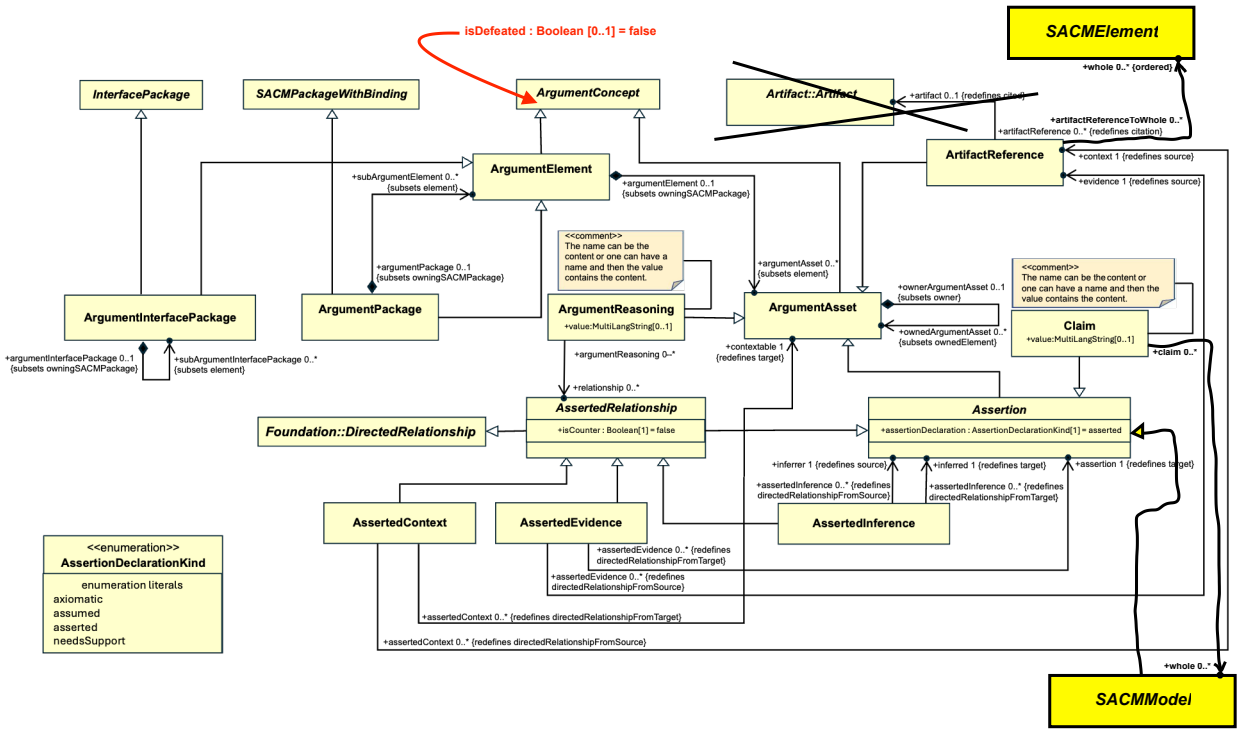

Figure 13.1 – Argument Package Diagram

This portion of the SACM model describes and defines the concepts required to model structured arguments. Arguments are represented in SACM through explicitly representing the Claims and citation of artifacts (e.g., as evidence) (ArtifactReference), and the 'links' between these elements – e.g., how one or more Claims are asserted to infer another Claim, or how one or more artifacts (referenced by ArtifactReference) are asserted as providing evidence for a Claim (AssertedEvidence). In addition to these core elements, in SACM it is possible to provide additional description of the ArgumentReasoning associated with inferential and evidential relationships, represent counter-arguments and counter-evidence (through isCounter:Boolean), and represent how artifacts provide the context in which arguments should be interpreted (through AssertedContext).

The packaging of structured arguments into 'modular' argument packages is enabled through ArgumentPackages. Users are able to declare interfaces for their packages through the use of ArgumentInterfacePackage. Within an ArgumentInterfacePackage, users create citations of the argument elements they select to disclose to external parties. Users are able to integrate ArgumentPackages through the use of ArgumentBindingPackage. An ArgumentBindingPackage binds ArgumentPackages together by including the declared ArgumentInterfacePackages for the ArgumentPackages, it may contain additional argument structures to provide the rationale of the binding. It is also possible within a package to cite elements contained within other argument packages (through ArtifactReference).

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 12 11.15 14 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 ArtifactReferemce) may provide evidence for more than one Claim.

Superclass

AssertedRelationship

Associations

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

Semantics

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.

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".

Constraints

The source of AssertedEvidence relationships must be ArtifactReference.

~~OCL:~~

~~self.source >forall(s\|s.oclIsTypeOf(ArtifactReference))~~

13 12 11.16 15 AssertedContext

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).

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

AssertedRelationship

Associations

+context:ArtifactReference [1] - the context provides further clarification or constraint of the contextable
+contextable:ArgumentAsset [1] - the context provides further clarification or constraint of the contextable

Semantics

Contextual information often needs to be cited in order to make clear the interpretation and scope of an Assertion and supporting arguementation. 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".

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").

12 11.17 16 AssertedArtifactSupport

AssertedArtifactSupport records the assertion th

Superclass

AssertedRelationship

Semantics

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 car
cases, it would be clearer to declar

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".

Constraints

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

13 12 11.18 17 AssertedArtifactContext

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

13.18 ArgumentConcept (abstract)

ArgumentConcept is an abstraction of all the packaging and elements in the argument domain.

Superclass
Packaging::Concept

Attributes
isDefeated : Boolean [0..1] = false - specified whether this ArgumentConcept is defeated or not (or with cardinality 0, has not been determined yet).

15.4 Assessment

Assurance Cases, in general, can be assessed by multiple subject matter experts (e.g. an Assessor) who may have differing ideas about the correctness of the argumentation or support of that argumentation with evidence. The first "assessment" made on the Assurance Cases comes from the author of the Assurance Case who puts their best effort forward in their description of the model. But, this "assessment" may be further augmented by another Assessor. This section describes additional parts to Assurance Cases to support multiple Assessments and changes to the model to support those assessments. Also, Assurance Cases may go through many changes from it beginning to the point of publication of the "final" model, and a record of those changes may want to be kept (e.g. so wrong avenues of thinking that have been explored and found to be unwanted can be documented in a tool). These changes can be kept through this mechanism as well.

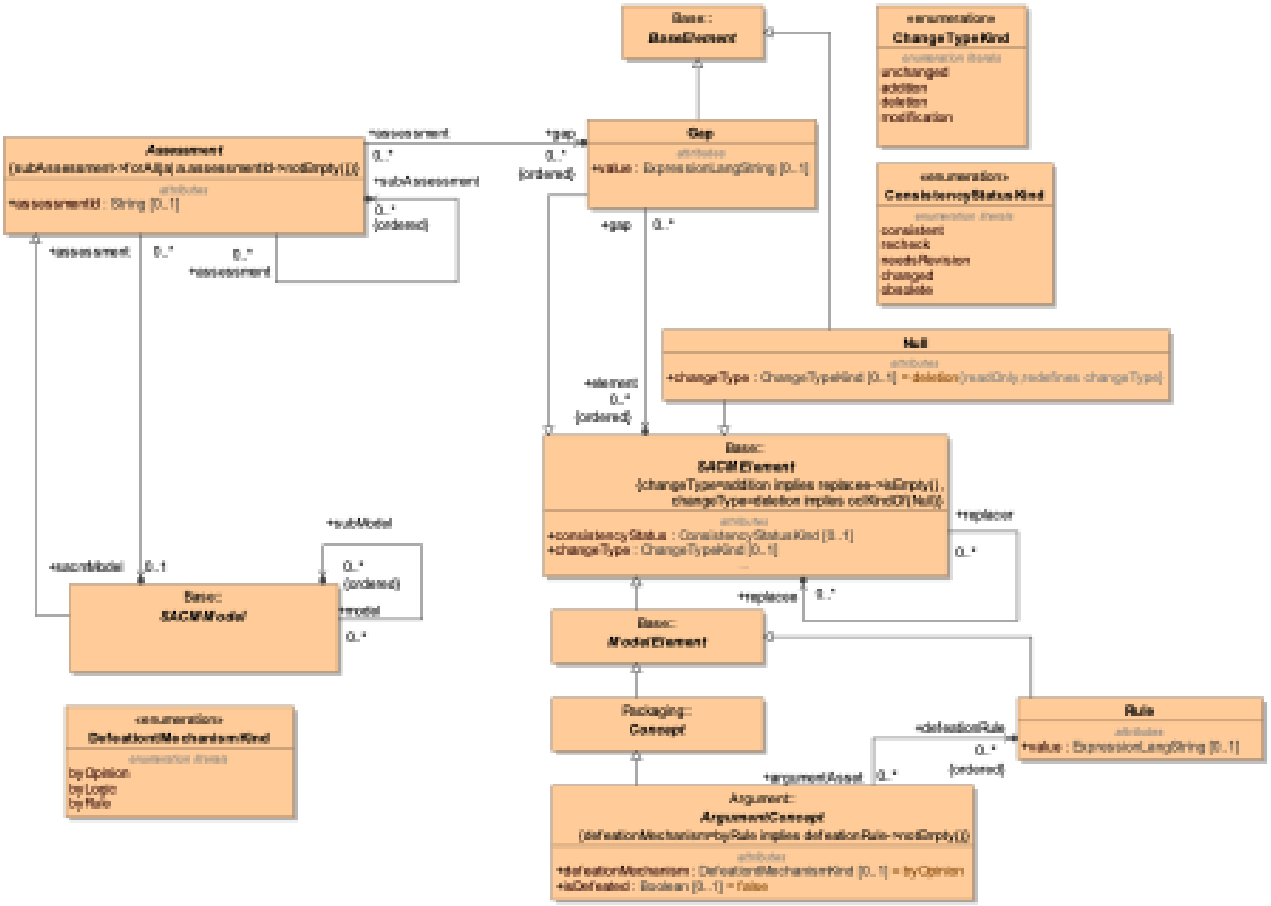

Figure 15.4.1 – SACM Assessments

15.4.1 Assessment (abstract)

Assessment is an additional metaclass generalization for SACMModel to allow a SACMModel to be defined as an Assessment.

Attributes

assessmentId : String [0..1] - an assessmentId that could be used as a unique key in a database that defines the nature of this assessment (e.g. who the Assessor was). Any assessmentId (not necessarily unique) means that the concrete SACMModel is considered an Assessment. No assessmentId means that this SACMModel is not considered an Assessment.

AssociationEnds

subAssessment : Assessment [0..*] {ordered} - A sub assessment (e.g. a previous assessment) of this Assessment. Direct judgements on an Assessment override any judgement in a subAssessment. Judgements in earlier assessments are overridden by judgements in later Assessments in the order list of subAssessments.

Constraints

SubAssessmentAreAssessments
subAssessment must be Assessments (i.e. assessmentId have a value).
inv: subAssessment->forAll(a| a.assessmentId->notEmpty())

15.4.2 Gap

A Gap can be specified by an Assessor as something missing from the Assurance Case. Gaps can be further detailed by specifying specific replacer SACMElements.

Supertype

Base::BaseElement

Attributes

value : ExpressionLangString [0..1] - statement of the Gap. The name can be the content or one can have a name and then the value contains the content.

AssociationEnds

element : SACMElement [0..*] {ordered} - Replacer SACMElements that are related to this Gap.

15.4.3 Null

Null is a replacer which allows an Assessor to remove any replacee SACMElement in an Assurance Case in their Assessment.

Supertype

Base::BaseElement, Base::SACMElement

Attributes

changeType : ChangeTypeKind [0..1] = deletion {readOnly, redefines changeType}

15.4.4 DefeationMechanismKind

DefeationMechanismKind defines how a defeation (or no defeation) has been decided.

EnumerationLiteral

- byOpinion - state of the defeation is by the opinion of the Assessor
- byLogic - state of the defeation is by the understood logic of the inferences
- byRule - state of the defeation is by the Rules specified

15.4.5 ChangeTypeKind

ChangeTypeKind defines how a replacer SACMElement changes in this assessment from its replacee.

EnumerationLiteral

- unchanged - The replacer SACMElement does not change from the replacee.
- addition - The replacer SACMElement is an addition (there will be no replacee).
- deletion - The replacer Null is a deletion of the replacee.
- modification - The replacer SACMElement modifies the replacee.

15.4.6 ConsistencyStatusKind

ConsistencyStatusKind allows marking of replacer SACMElements during the process and at the end of the process of an Assessment.

EnumerationLiteral

- consistent - the replacer SACMElement is consistent with what is needed to fix the Assurance Case.
- recheck - the replacer SACMElement is considered consistent now, but want to recheck it later.
- needsRevision - the replacer SACMElement needs revision, but it has not been changed as yet.

- changed - the replacer SACMElement has been changed to fix the Assurance Case.
- obsolete - The replacer SACMElement is no longer needed to fix the Assurance Case.

Note: ChangeTypeKind and ConsistencyStatusKind are based on Figure 6.3 from the whitepaper: Checkable Safety Arguments – A Modeling Framework Supporting the Maintenance of Safety Arguments Consistent with System Development Artifacts, Carmen Cârlan, 2024.

15.4.7 **SACMElement** [Additions to 9.2]

Attributes

consistencyStatus : ConsistencyStatusKind [0..1] - Defines the consistency for the SACMElement on this particular Assessment.
changeType : ChangeTypeKind [0..1] - Defines what the change type is for this SACMElement is in the Assessment.

AssociationEnds

replacee : SACMElement [0..*] - Specifies the SACMElement which this SACMElement replaces in this Assessment.

Constraints

DeletionsAreNulls
If changeType=deletion then the replacer must be of type Null (or one of its specializations).
inv: changeType=deletion implies oclKindOf(Null)

AdditionHasNoReplacee
If changeType=addition then the replacee cardinality is 0.
inv: changeType=addition implies replacee->isEmpty()

15.4.8 **ArgumentConcept** [Additions to 13.18]

Attributes

defeationMechanism : DefeationMechanismKind [0..1] = byOpinion - Defines the defeation mechanism used on this ArgumentConcept.

AssociationEnds

defeationRule : Rule [0..*] {ordered} - if defeationMechanism=byRule, then these are the (possible multiple) Rule(s) that define the reason why defeation (or no defeation) happened.

Constraints

ByRuleImpliesDefeationRule
If defeationMechanism=byRule then defeationRule must have at least one reference.
inv: defeationMechanism=byRule implies defeationRule->notEmpty()

15.4.9 Examples

Assessment is not normally modeled in graphical form, but does need to be represented in the model. In this Assessment Example, Assessment1 with Gap1, replaced AssertedInference p with Null and ClaimB with ClaimC. In Addition, Null was marked as consistent, but ClaimC is marked to be rechecked.

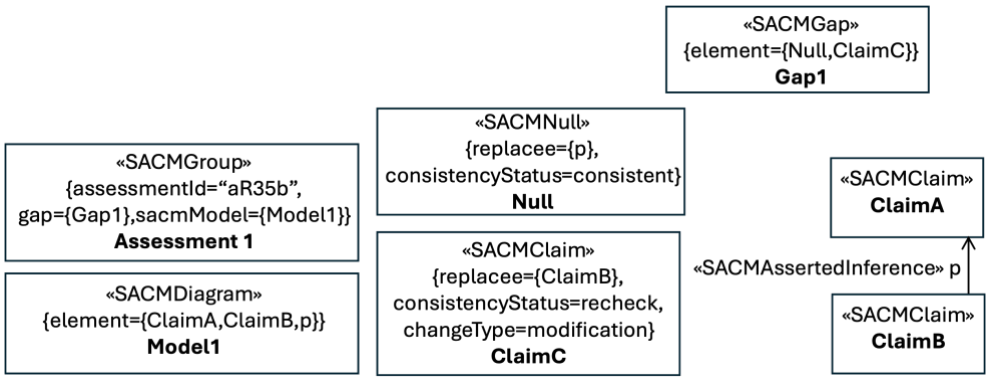

Figure 15.4.9 – Assessment Example