

## 10.9 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.

## 10.10 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

### Associations

element: ExpressionElement [0..\*] – an optional reference to other ExpressionElements forming part of the structured 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).

# Argument

## 11 SACM Argumentation Metamodel

### 11.1 General

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

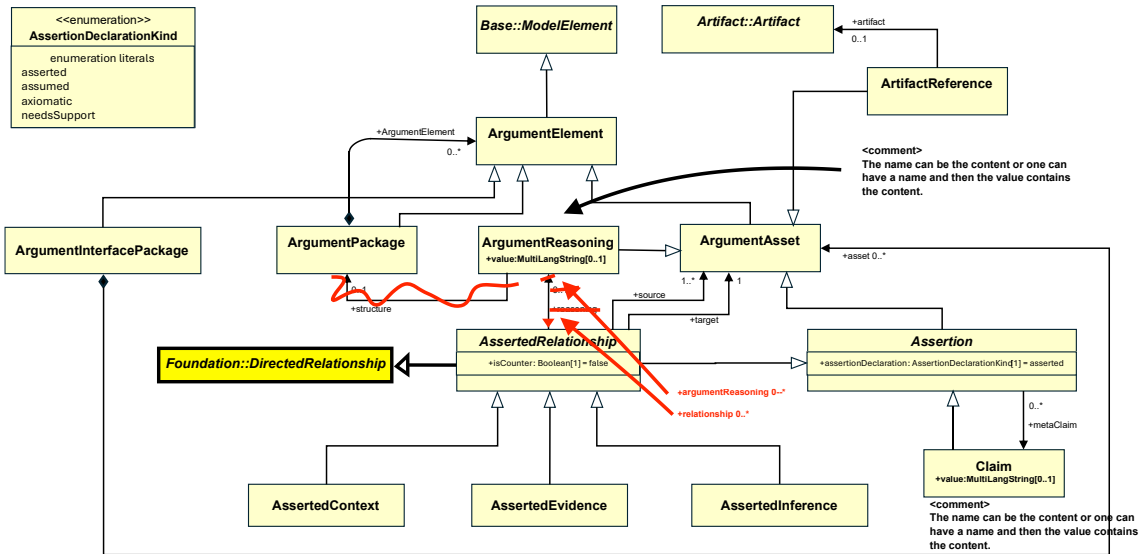


Figure 11.1 - Argumentation Package Diagram

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 ArgumentPackageInterface. Within an ArgumentPackageInterface, users create citations of the argumentation elements they select to disclose to external parties. Users are able to integrate ArgumentPackages through the use of ArgumentPackageBinding. An ArgumentPackageBinding binds ArgumentPackages together by including the declared ArgumentPackageInterfaces 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).

ArgumentInterfacePackages  
 ArgumentPackageBinding

argument

## Superclass

ArgumentAsset

## Associations

referencedArtifactElement:Base::ArtifactElement[0..\*] – reference to a collection of ArtifactElements.

## Semantics

It is necessary to be able to cite artifacts that provide supporting evidence, context, or additional description within an argument structure. ArtifactReferences allow there to be an objectified citation of this information within the structured argument, thereby allowing the relationship between this artifact and the argument to also be explicitly declared.

## 11.10 Assertion (abstract)

Assertions are used to record the propositions of Argumentation (including both the Claims about the subject of the argument and the structure of the Argumentation being asserted). Propositions can be true or false, but cannot be true and false simultaneously.

## Superclass

ArgumentAsset

## Attributes

assertionDeclaration:AssertionDeclaration[1] = asserted – the declaration indicating the state of the Assertion.

## ~~Associations~~

~~metaClaim:Claim[0..\*] - references Claims concerning (i.e., about) the Assertion (e.g., regarding the confidence in the Assertion)~~

## Semantics

Structured arguments are declared by stating claims, citing evidence and contextual information, and asserting how these elements relate to each other.

## 11.11 Claim

Claims are used to record the propositions of any structured argument contained in an ArgumentPackage. Propositions are instances of statements that could be true or false, but cannot be true and false simultaneously.

## Superclass

Assertion

Attributes  
value:MultiLangString[0..1]

## Semantics

The core of any argument is a series of claims (premises) that are asserted to provide sufficient reasoning to support a (higher- level) claim (a conclusion).

A Claim that is intentionally declared without any supporting evidence or argumentation can be declared as being assumed (i.e., assertionDeclared = assumed). It is an assumption. However, it should be noted that a Claim that is not ‘assumed’ (i.e., assertionDeclaration = asserted) is not being declared as false. However, there is the expectation of the provision of a supporting argument structure (e.g., it may represent part of an incomplete structure).

A Claim that is intentionally declared as requiring further evidence or argumentation can be denoted by setting +assertionDeclaration to “needsSupport”.

A Claim that is being declared as axiomatically true can be denoted by setting +assertionDeclaration to “axiomatic”.

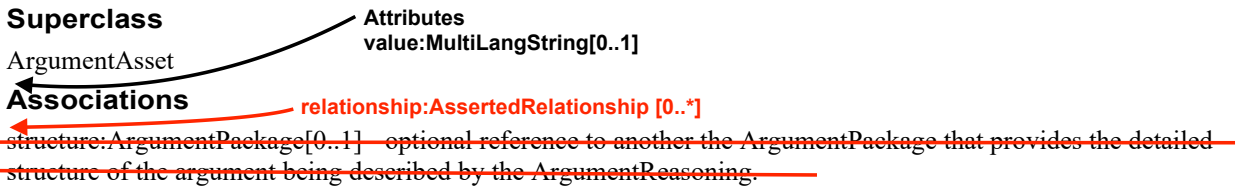
A Claim that is defeated by counter evidence or counter argument can be denoted by setting +assertionDeclaration to “defeated”.

A Claim which cites another claim and supported by the cited claim can be denoted by setting +assertionDeclaration to “asCited”.

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

## 11.12 ArgumentReasoning

ArgumentReasoning can be used to provide additional description or explanation of the asserted relationship. For example, it can be used to provide description of an AssertedInference that connects one or more Claims (premises) to another Claim (conclusion). ArgumentReasoning elements are therefore related to AssertedInferences, AssertedContexts, and AssertedEvidence. It is also possible that ArgumentReasoning elements can refer to other structured Arguments as a means of documenting the detail of the argument that establishes the asserted inferences, contexts, and evidence.



### Semantics

The AssertedRelationship that relates one or more Claims (premises) to another Claim (conclusion), or evidence cited by an ArtifactReasoning to a Claim, may not always be obvious. In such cases ArgumentReasoning can be used to provide further description of the reasoning involved.

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

## 11.13 AssertedRelationship (abstract)

AssertedRelationship is the abstract association class that enables the ArgumentAssets of any structured argument to be linked together. The linking together of ArgumentAssets allows a user to declare the relationship that they assert to hold between these elements.

### Superclass

Assertion ← , Foundation::AssertedRelationship

### Attributes

isCounter:Boolean[1] = false – a flag indicating whether the AssertedRelationship counters its declared purposes (e.g. setting isCounter = true for an AssertedEvidence indicates that the relationship is a counter-evidence).

### Associations

source:ArgumentAsset[1..\*] - reference to the ArgumentAsset(s) that are the source (starting point) of the relationship.

target:ArgumentAsset[1] - reference to the ArgumentAsset(s) that are the target (ending point) of the relationship.

~~reasoning:ArgumentReasoning[0..1] - an optional reference to the a description of the reasoning underlying the AssertedRelationship.~~

### Semantics

In SACM, the structure of an argument is declared through the linking together of primitive ArgumentAssets. For example, a sufficient inference can be asserted to exist between two claims (“Claim A implies Claim B”) or sufficient evidence can be asserted to exist to support a claim (“Claim A is evidenced by Evidence B”). An inference asserted 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.

## 11.14 AssertedInference

AssertedInference association records the inference that a user declares to exist between one or more Assertion (premise) and another Assertion (conclusion). It is important to note that such a declaration is itself an assertion on behalf of the user.

### Superclass

AssertedRelationship

### Semantics

The core structure of an argument is declared through the inferences that are asserted to exist between Assertions (e.g., Claims). For example, an AssertedInference can be said to exist between two claims (“Claim A implies Claim B”). An