Part I - Common Elements

The first part of the specification defines the common elements of the Structured Assurance Case Metamodel, including the Base Classes, the Structured Assurance Case Terminology Classes, and the Structured Assurance Case Packages. Subsequent parts define the Argumentation Metamodel and the Artefact Metamodel.

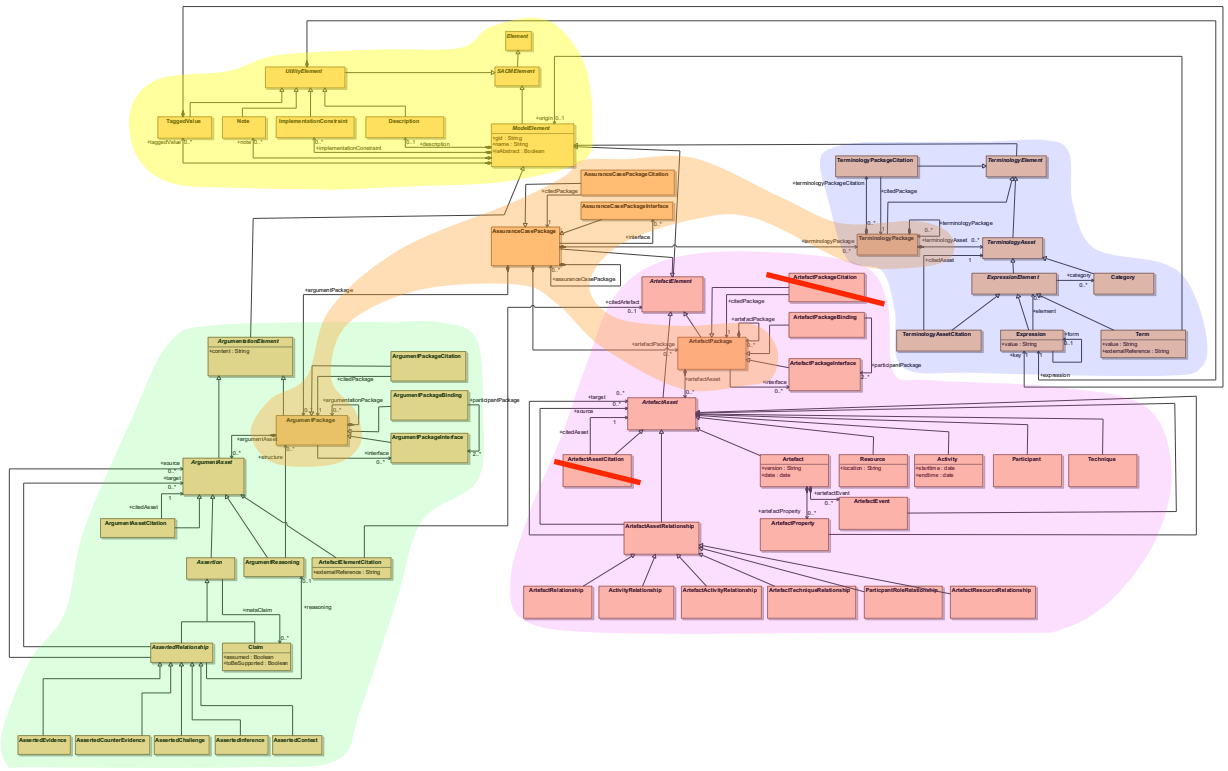


Figure 7.2 - Overall SACM Class Diagram

Yellow denotes items covered in Clause 8, Structured Assurance Case Base Classes.

Orange denotes items covered in Clause 9, Structured Assurance Case Packages.

Blue denotes items covered in Clause 10, Structured Assurance Case Terminology Classes.

Green denotes items covered in Clause 11, Argumentation Metamodel.

Purple denotes items covered in Clause 12, Artefact Metamodel.

8 Structured Assurance Case Base Classes

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.

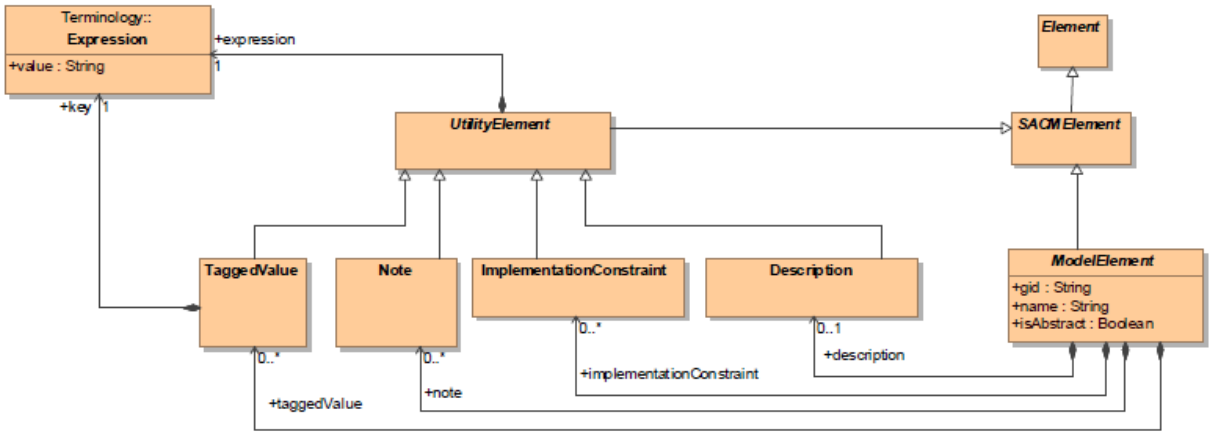


Figure 8.1 - Overall SACM Class Diagram

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.

8.2 SACMElement (abstract)

SACMElement is the base class for SACM.

Superclass

MOF:Element

Attributes

None

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

isAbstract[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.

Semantics

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

8.3 ModelElement (abstract)

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

Superclass

SACMElement

Attributes

gid: String – a unique identifier that is unique within the scope of the model instance

name: String – the name of the element

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.

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.

This follows the existing practice of considering an assurance case when fully completed to comprise both argumentation and evidence, although each may be exchanged individually.

AssuranceCasePackage is a sub-class of ArtefactElement. Semantically an AssuranceCasePackage can be considered as an artefact of evidence (e.g. from the perspective of another AssuranceCasePackage).

Superclass

ArtefactElement

Associations

assuranceCasePackageCitation: AssuranceCasePackageCitation [0..*] – a collection of optional citations to other AssuranceCasePackages

assuranceCasePackage: AssuranceCasePackage [0..*] – a number of optional sub-packages

interface: AssuranceCasePackageInterface [0..*] – a number of optional assurance case package interfaces that the current package may implement

artefactPackage: ArtefactPackage [0..*] – a number of optional artefact sub-packages

terminologyPackage: TerminologyPackage [0..*] – a number of optional terminology sub-packages

Semantics

AssuranceCasePackage is the root class for creating structured assurance cases.

9.4 AssuranceCasePackageInterface

AssuranceCasePackageInterface is a kind of AssuranceCasePackage that defines an interface that may be exchanged between users. An AssuranceCasePackage may declare one or more ArtefactPackageInterfaces.

Superclass

AssuranceCasePackage

Semantics

AssuranceCasePackageInterface enables the declaration of the elements of an AssuranceCasePackage that might be referred to (cited) in another AssuranceCasePackage, thus the elements can be used for assurance in the scope of the latter AssuranceCasePackage.

Constraints

AssuranceCasePackageInterface are only allowed to contain the following: ArgumentPackageInterfaces, ArtefactPackageInterfaces, and TerminologyPackages.

9.5 AssuranceCasePackageCitation

AssuranceCasePackageCitation is used to cite another AssuranceCasePackage. The citation can be used where an assurance case author wishes to refer to an AssuranceCasePackage outside of the current AssuranceCasePackage hierarchy.

Superclass

ArtefactElement

Associations

citedPackage: AssuranceCasePackage – the existing AssuranceCasePackage being referenced.

Constraints

The citedPackage referred to by a AssuranceCasePackageCitation must be outside of the containment hierarchy containing the citation.

9.6 ArgumentPackage

ArgumentPackage is a container for the structured argument aspect of the assurance case. It contains the structure of assertions which comprise the structured argument.

Superclass

ArgumentationElement

Associations

argumentPackageCitation: ArgumentPackageCitation [0..*] – an optional set of citations to other ArgumentPackages

argumentPackage: ArgumentPackage [0..*] – an optional set of sub ArgumentPackages, allowing for recursive containment argumentAsset: ArgumentAsset [0..*] an optional set of ArgumentAssets

Semantics

ArgumentPackage is the base class for specifying the results of the argumentation efforts for a structured assurance case (i.e., an AssuranceCase).

9.7 TerminologyPackage

TerminologyPackage is a container element for terminology that may be exchanged. Terminology can define terms, expressions or categories, used elsewhere in the assurance case.

Superclass

TerminologyElement

Associations

terminologyPackageCitation: TerminologyPackageCitation [0..*] – an optional set of citations to other TerminologyPackage elements

terminologyAsset: TerminologyAsset [0..*] – an optional set of terminology assets (expressions, terms and categories)

terminologyPackage: TerminologyPackage [0..*] – an optional set of contained TerminologyPackage elements, allowing for recursive containment.

Semantics

TerminologyPackage is the base class for specifying all the terminology needs and constraints (via TerminologyAssets) for a structured assurance case (i.e., an AssuranceCase).

9.8 ArtefactPackage

ArtefactPackage is a container element for the assets that are used as evidence or cited in support of a structured argument. These assets form the evidential basis for the assurance case.

Superclass

ArtefactElement

Associations

artefactPackageCitation: ArtefactPackageCitation [0..*] – an optional set of citations to other ArtefactPackage elements artefactAsset:

10 Structured Assurance Case Terminology Classes

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.

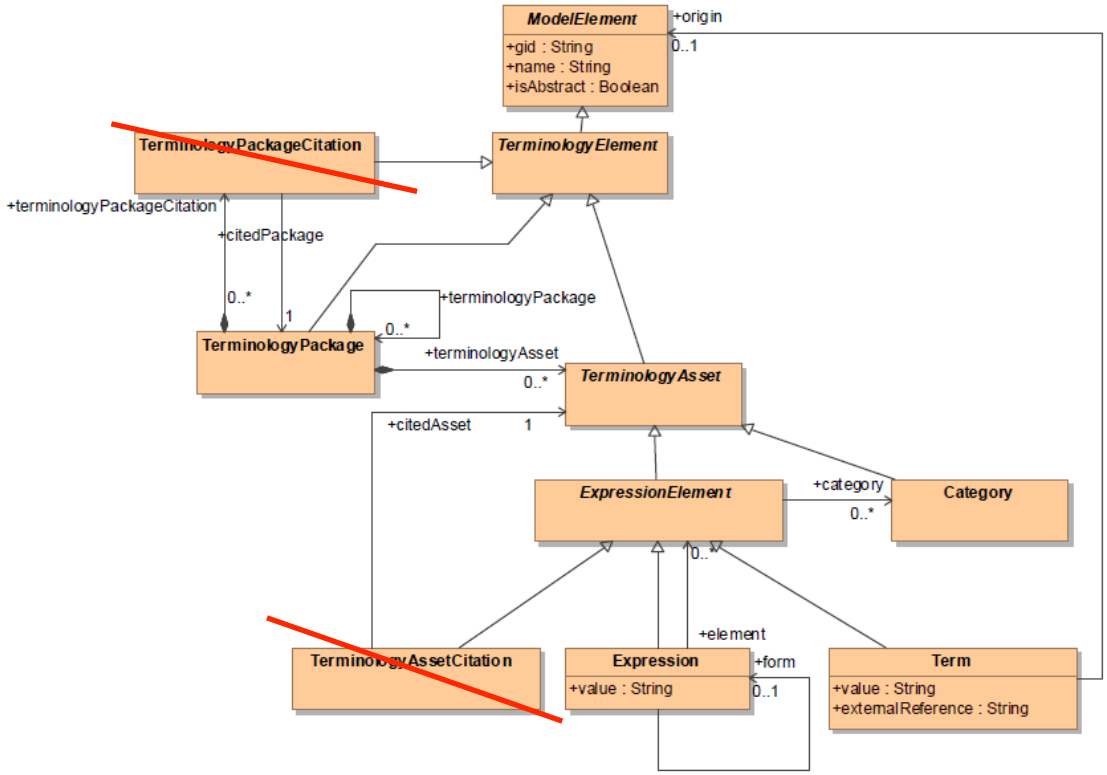


Figure 10.1 - Terminology Class Diagram

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.

10.2 TerminologyElement (abstract)

TerminologyElement is an abstract class that serves as a parent class for all SACM terminology assets (TerminologyAsset) and the packaging of these assets (TerminologyPackage ~~and TerminologyPackageCitation~~).

Superclass

ModelElement

Semantics

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

10.3 TerminologyPackage

The TerminologyPackage Class is the container class for SACM terminology assets.

Superclass

TerminologyElement

Associations

TerminologyAsset:TerminologyAsset[0..*]

The TerminologyAssets contained in a given instance of a TerminologyPackage.
terminologyPackage:TerminologyPackage[0..*]

The nested terminologyPackage contained in a given instance of a TerminologyPackage
terminologyPackageCitation:TerminologyPackageCitation[0..*]

The nested terminologyPackageCitation contained in a given instance of a TerminologyPackage

Semantics

TerminologyPackages contain the TerminologyAssets that can be used within the naming and description of SACM arguments and artefacts. TerminologyPackage elements can be nested, and can contain citations (references) to other TerminologyPackages.

10.4 TerminologyPackageCitation

The TerminologyPackageCitation is a citation (reference) to another TerminologyPackage.

Superclass

TerminologyElement

Associations

citedPackage: TerminologyPackage[0..1]

The TerminologyPackage being cited by the TerminologyPackageCitation.

Semantics

TerminologyPackageCitations make it possible to cite other TerminologyPackages.For example, within a TerminologyPackage it can be useful to refer to another TerminologyPackage (to reference terminology) that is not contained with the same TerminologyPackage and is defined elsewhere.

Constraints

The citedPackage referred to by a TerminologyPackageCitation must be outside of the containment hierarchy containing the citation.

10.5 TerminologyAsset (abstract)

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

Superclass

TerminologyElement

Semantics

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

10.9 Term

The 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 attribute being set true. A concrete term is denoted by isAbstract being false.

Attributes

value: String – An attribute recording the value of the Term

externalReference: String – An attribute recording an external reference (e.g., URI) to the object referred to by the Term

Superclass

ExpressionElement

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 attribute being set true. A concrete term is denoted by isAbstract being false.

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

10.10 TerminologyAssetCitation

The TerminologyAssetCitation is a citation (reference) to an ExpressionElement contained in another TerminologyPackage.

Superclass

ExpressionElement

Associations

citedElement:TerminologyAsset [1] The TerminologyAsset being cited by the TerminologyAssetCitation.

Semantics

TerminologyAssetCitations make it possible to cite TerminologyAssets from other TerminologyPackages when forming TerminologyPackages or Expressions.

For example, within a TerminologyPackage it can be useful to refer to TerminologyAssets within another TerminologyPackage (to reference terminology) that are not contained with the same TerminologyPackage and is defined elsewhere. Within an Expression it can also be useful to refer to TerminologyAssets within another TerminologyPackage that are not contained with the same TerminologyPackage and is defined elsewhere.

Constraints

The citedElement referred to by a TerminologyAssetCitation must be outside of the containment hierarchy containing the citation.

The packaging of structured arguments into 'modular' argument packages is enabled through ArgumentPackages, an optional declaration of an interface for the package (ArgumentPackageInterface) that cites a specific selection of the ArgumentElements contained within the package, and the ability to link (by means of an argument) two or more argument packages (through an ArgumentPackageBinding). It is also possible within a package to cite elements contained within other argument packages (through using ArgumentElementCitation).

In the following sections we describe these model elements in detail.

11.2.1 ArgumentationElement class (abstract)

An ArgumentationElement is the top level element of the hierarchy for argumentation elements.

Semantics

The ArgumentationElement is a common class for all elements within a structured argument.

11.2.2 ArgumentPackage Class

The ArgumentPackage Class is the container class for a structured argument represented using the SACM Argumentation Metamodel.

Superclass

ArgumentationElement

Associations

argumentAsset:ArgumentAsset[0..*]

The ArgumentAssets contained in a given instance of an ArgumentPackage.
argumentPackage:ArgumentationPackage[0..*]

The nested argumentPackage contained in a given instance of an ArgumentPackage
interface:ArgumentationPackage[0..*]

Reference to the declared interface for the ArgumentPackage.

Semantics

ArgumentPackages contain structured arguments. These arguments are composed of ArgumentAssets. ArgumentPackages elements can be nested, and can contain citations (references) to other ArgumentPackages.

For example, arguments can be established through the composition of Claims (propositions) and the AssertedInferences between those Claims.

~~11.2.3 ArgumentPackageCitation Class~~

~~The ArgumentPackageCitation is a citation (reference) to another ArgumentPackage.~~

~~**Superclass**~~

~~ArgumentPackage~~

~~**Associations**~~

~~citedPackage:ArgumentPackage[1]~~

~~The ArgumentPackage being cited by the ArgumentPackageCitation.~~

~~**Semantics**~~

~~ArgumentPackageCitations make it possible to cite other ArgumentPackages.~~

For example, within an ArgumentPackage it can be useful to refer to another ArgumentPackage that is not contained within the same ArgumentPackage.

Constraints

ArgumentPackageCitations have no contents other than the association to the citedPackage.

The citedPackage referred to by an ArgumentPackageCitation must be outside of the containment hierarchy containing the citation.

11.2.4 ArgumentPackageBinding Class

The ArgumentPackageBinding is a sub type of ArgumentPackage used to record the mapping (agreement) between two or more ArgumentPackages.

Superclass

ArgumentPackage

Associations

participantPackage:ArgumentPackageInterface[2..*]

The ArgumentPackages being mapped together by the ArgumentPackageBinding.

Semantics

ArgumentPackageBindings can be used to map resolved dependencies between the Claims of two or more ArgumentPackages.

For example, one ArgumentPackage may contain a claim that is toBeSupported (i.e. currently has no supporting argument). An ArgumentPackageBinding can be used to record the mapping (by means of containing a structured argument linking ArgumentAssetCitations to the claims in question) between this claim and a supporting claim in another ArgumentPackage.

An ArgumentPackageInterface is a sub type of ArgumentPackage that can be used to create an explicit interface to an existing ArgumentPackage.

Constraints

The ‘root’ ArgumentAssets contained by an ArgumentPackageBinding (i.e. the ArgumentAssets only associated as target of an AssertedRelationship) and ‘leaf’ ArgumentAssets (i.e. the ArgumentAssets only associated as source of an AssertedRelationship) must be ArgumentAssetCitations to Claims or ArtefactElementCitations contained within the ArgumentPackages associated by the participantPackage association.

11.2.5 ArgumentPackageInterface Class

Superclass

ArgumentPackage

Semantics

ArgumentPackageInterfaces can be used to declare (by means of containing ArgumentAssetCitations) the ArgumentAssets contained in an ArgumentPackage that form part of the explicit, declared, interface of the ArgumentPackage.

For example, whilst an ArgumentPackage may contain many Claims, it may be desirable to create an 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.

Constraints

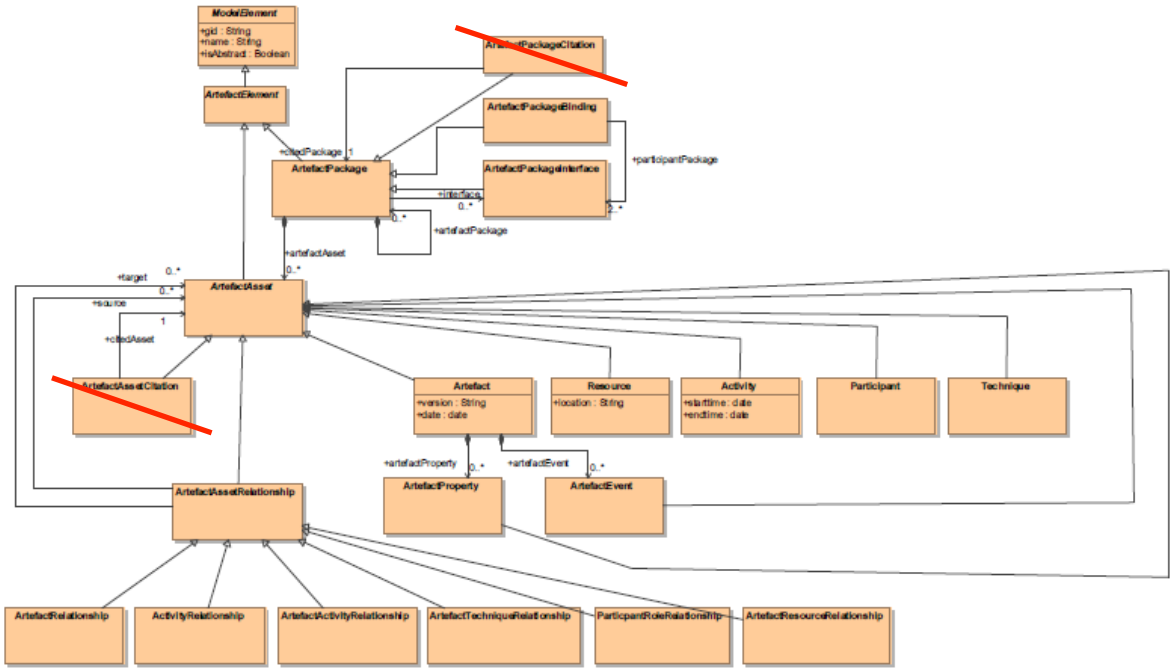
ArgumentPackageInterfaces are only allowed to contain ArgumentAssetCitations to ArgumentAssets within the ArgumentPackage with which this ArgumentPackageInterface is associated (by the interface association).

12 Artefact Classes

12.1 General

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

Figure 12.1 - Artefact Class Diagram



Artefacts correspond to the main evidentiary elements of an assurance case. By means of assertions (AssertedEvidence and AssertedCounterEvidence), artefacts are used for supporting claims and arguments.

In general, artefacts 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, artefact management might also require the specification of patterns (or templates) in order to allow a user, for instance, to indicate that a given artefact 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 artefacts that the supplier will have to provide as assurance evidence for a system. As a result of this process, artefact patterns could be specified, and such patterns would need to be made concrete during the lifecycle of the system. Artefact patterns are specified by mean of the attribute 'isAbstract' (ModelElement). 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 Artefact that is abstract. Once created, the value of this attribute of the hazard log would be 'false'. The specification of artefact patterns also facilitates their reuse, as the corresponding artefacts 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 Artefact.

When made concrete, an Artefact can relate to many different types of information necessary for developing confidence in the Artefact and thus for assurance purposes. Such information can be regarded as meta-data or provenance information about an Artefact, provides information about its management, and is specified with the rest of specializations of ArtefactAsset ~~(different to ArtefactAssetCitation)~~. Using a design specification as an example, properties (ArtefactProperty) 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 be the owner of the design specification, which would also relate to other artefacts: the requirements specification that satisfies, the architecture that implements, its verification report, etc. Further relationships might be specified between other artefact assets, such precedence between activities ('Specify system design' precedes 'Verify system design') and the participants in an Activity.

~~12.2 ArtefactPackageCitation~~

~~ArtefactPackageCitation is used to cite another ArtefactPackage. The citation can be used where an assurance case author wishes to refer to an existing ArtefactPackage.~~

~~Superclass~~

~~ArtefactPackage~~

~~Associations~~

~~citedPackage: ArtefactPackage [1] -- the ArtefactPackage cited by the ArtefactPackageCitation~~

~~Semantics~~

~~ArtefactPackageCitations enable the reference, in a given ArtefactPackage, to another ArtefactPackage.~~

~~Constraints~~

~~ArtefactPackageCitations have no contents other than the association to the citedPackage.~~

12.3 ArtefactPackageBinding

The ArtefactPackageBinding is a sub type of ArtefactPackage used to record ArtefactAssetRelationships between the ArtefactAssets of two or more ArtefactPackages.

Superclass

ArtefactPackage

Associations

participantPackage:ArtefactPackageInterface[2..*]

The ArtefactPackages containing the ArtefactAssets being related together by the ArtefactPackageBinding.

Semantics

ArtefactPackageBindings can be used to map dependencies between the cited ArtefactAssets of two or more ArtefactPackages. For example, a binding could be used to record a 'derivedFrom' ArtefactAssetRelationship between the ArtefactAsset of one package to the ArtefactAsset of another.

Contraints

~~ArtefactPackageBindings must only contain ArtefactAssetRelationships with source and target ArtefactAssetCitations citing ArtefactsAssets contained within the ArtefactPackageInterfaces associated by participantPackage.~~

12.4 ArtefactPackageInterface

ArtefactPackageInterface is a kind of ArtefactPackage that defines an interface that may be exchanged between users. A typical use case might be for a component supplier to provide its customers with ArtefactPackageInterfaces that contain the relevant supplier's ArtefactElements for the customers' ArtefactPackages. An ArtfefactPackage may also declare

that it implements or conforms to a particular ArtefactPackageInterface.

Superclass

ArtefactPackage

Associations

artefactPackageCitation: ArtefactPackageCitation [0..*] – an optional set of citations to other ArtefactPackage elements

artefactAsset: ArtefactAsset [0..*] – an optional set of ArtefactAsset elements, such as citations, artefacts, resources, activities, etc.

artefactPackage: ArtefactPackage [0..*] - an optional set of contained ArtefactPackage elements, allowing for recursive containment.

Semantics

ArtefactPackageInterface enables the declaration of the elements of an ArtefactPackage that might be referred to (cited) in another ArtefactPackage, thus the elements can be used for assurance in the scope of the latter ArtefactPackage.

Constraints

ArtefactPackageInterfaces are only allowed to contain ArtefactAssetCitations to ArtefactAssets within the ArtefactPackage with which this ArtefactPackageInterface is associated (by the interface association).

12.5 ArtefactAsset class (abstract)

The ArtefactAsset class represents the artefact-specific pieces of information of an assurance case, in contrast to the argument-specific pieces of information.

Superclass

ArtefactElement

Semantics

Information about artefacts is essential for any assurance case. The artefacts 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 artefact, its lifecycle, and its properties. All this information might have to be consulted for developing confidence in the validity of an assurance case.

12.5.1 ArtefactAssetCitation class

The ArtefactAssetCitation class allows an ArtefactPackage to refer to the components of another ArtefactPackage.

Superclass

ArtefactAsset

Associations

citedAsset:ArtefactAsset[1]

The ArtefactAsset that the ArtefactAssetCitation cites

Constraints

The citedAsset of an ArtefactAssetCitation must be part of an ArtefactPackageInterface.

The citedAsset of an ArtefactAssetCitation must be part of a different ArtefactPackage.

The citedAsset of an ArtefactAssetCitation cannot be an ArtefactAssetCitation.

The citedAsset of an ArtefactAssetCitation cannot be an ArtefactAssetRelationship.

Semantics

~~ArtefactAssets belong to single ArtefactPackages. Nonetheless, the ArtefactAssets can be referred to in other ArtefactPackages in order to, for instance, specify that a relationship exists between ArtefactAssets of different ArtefactPackages. For example, an ArtefactPackage might be specified for all the V&V results of an assurance case, and another for the requirements specifications. The fist ArtefactPackage might refer to the second for further specifying that a given V&V result corresponds to the validation of a given requirement.~~

12.5.2 Artefact class

The Artefact class represents the distinguishable units of data used in an assurance case.

Superclass

ArtefactAsset

Attributes

version: String
The version of the Artefact

date: Date
The date on which the artefact was created.

Associations

artefactProperty::ArtefactProperty[0..*]
The ArtefactProperties of the Artefact

artefactEvent::ArtefactEvent[0..*]
The set of ArtefactEvents that represent the lifecycle of the Artefact

Semantics

Artefacts correspond to the main evidentiary support for the arguments and claims of an assurance case: an Artefact can play the role of evidence of a Claim (AssertedEvidence), or of counterevidence (AssertedCountedEvidence). An Artefact 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 Artefacts include system lifecycle plans, dependability (e.g., safety) analysis results, system specifications, and V&V results.

12.5.3 ArtefactProperty class

The ArtefactProperty class enables the specification of the characteristics of an Artefact.

Semantics

An Artefact can have different, specific characteristics independent of the argumentation structure in which the Artefact 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).

12.5.4 ArtefactEvent class

The ArtefactEvent class enables the specification of the events in the lifecycle of an Artefact.

Attributes

date: Date
The date on which the ArtefactEvent occurred.

characteristics for the Artefacts. For example, the use of UML (as a Technique) for designing a system results in a design specification with a set of UML diagrams that could represent static and dynamic internal aspects of the system.

12.5.8 Participant class

The Participant class enables the specification of the parties involved in the management of ArtefactAssets.

Superclass

ArtefactAsset

Semantics

Different parties can participate in an assurance case effort, such as specific people, organizations, and tools.

12.5.9 ArtefactAssetRelationship class

The ArtefactAssetRelationship class enables the ArtefactAssets of an AssuranceCase to be linked together. The linking together of ArtefactAssets allows a user to specify that a relationship exists between the assets.

Superclass

ArtefactAsset

Associations

source:ArtefactAsset[0..*]
The source of the ArtefactRelationship

target:ArtefactAsset[0..*]
The target of the ArtefactRelationship

Constraints

The source or target of an ArtefactAssetRelationship cannot be another ArtefactAssetRelationship.

Semantics

An ArtefactAsset can be related to other ArtefactAssets. This kind of information is specified by means of ArtefactAssetRelationships, which can also have a specific type depending on the ArtefactAssets being linked together.

12.5.10 ArtefactRelationship class

The ArtefactRelationship class enables two Artefacts to be linked together.

Superclass

ArtefactAssetRelationship

Constraints

The source and target of an ArtefactRelationhsip must be Artefacts~~, or ArtefactAssetCitations citing an Artefact~~.

Semantics

The Artefacts managed during a system's lifecycle do not exist in isolation, but relationships typically exist between them: the test cases that validate some requirement, the design standard followed in a design specification, etc. These relationships are specified by means of ArtefactRelationships.

12.5.11 ActivityRelationship class

The ActivityRelationship class enables two Activities to be related together.

Superclass

ArtefactAssetRelationship

Constraints

The source and target of an ActivityRelationship must be Activities ~~or ArtefactAssetCitations citing an Activity~~.

Semantics

ActivityRelationships aim to support the specification of how Activities, and citations to them, relate each other: an Activity that precedes another, an Activity decomposed into others, etc.

12.5.12 ArtefactActivityRelationship class

The ArtefactActivityRelationships class enables an Artefact and an Activity to be linked together.

Superclass

ArtefactAssetRelationship

Constraints

The source of an ArtefactActivityRelationship must be an Artefact~~, or an ArtefactAssetCitation citing an Artefact~~.

The target of an ArtefactActivityRelationship must be an Activity~~, or an ArtefactAssetCitation citing an Activity~~.

Semantics

Artefacts are managed in the scope of Activities, which usually use the Artefact as input and output. Such information is specified by means of ArtefactActivityRelationships.

12.5.13 ArtefactTechniqueRelationship class

The ArtefactTechniqueRelationship class enables an Artefact and a Technique to be linked together.

Superclass

ArtefactAssetRelationship

Constraints

The source of an ArtefactActivityRelationship must be an Artefact~~, or an ArtefactAssetCitation citing an Artefact~~.

The target of an ArtefactActivityRelationship must be a Technique~~, or an ArtefactAssetCitation citing a Technique~~.

Semantics

Artefacts result from the application of Techniques, such as the application of UML for a design specification. ArtefactTechniqueRelationships are used to specify such a kind of information.

12.5.14 ParticipantRoleRelationship class

The ParticipantRoleRelationships class enables a Participant to be linked to other ArtefactAssets.

Superclass

ArtefactAssetRelationship

Constraints

The source of an ParticipantRoleRelationship must be a Participant ~~or an ArtefactAssetCitation citing a Participant~~.

Semantics

The information about the roles and functions that a Participant plays with regard to other ArtefactAssets is specified by means of ParticipantRoleRelationships. Examples of roles and functions include the owner of an Artefact, the executor of an Activity, and possible relationships between Participants (e.g., supervisor).

12.5.15 ArtefactResourceRelationship class

The ArtefactResourceRelationship class enables an Artefact and a Resource to be linked together.

Superclass

ArtefactAssetRelationship

Constraints

The source of an ArtefactActivityRelationship must be an Artefact~~, or an ArtefactAssetCitation citing an Artefact~~.

The target of an ArtefactActivityRelationship must be a Resource~~, or an ArtefactAssetCitation citing a Resource~~.

Semantics

The specific Resources where an Artefact is located are specified by means of ArtefactResourceRelationships.