

# 12 Artifact Classes

## 12.1 General

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

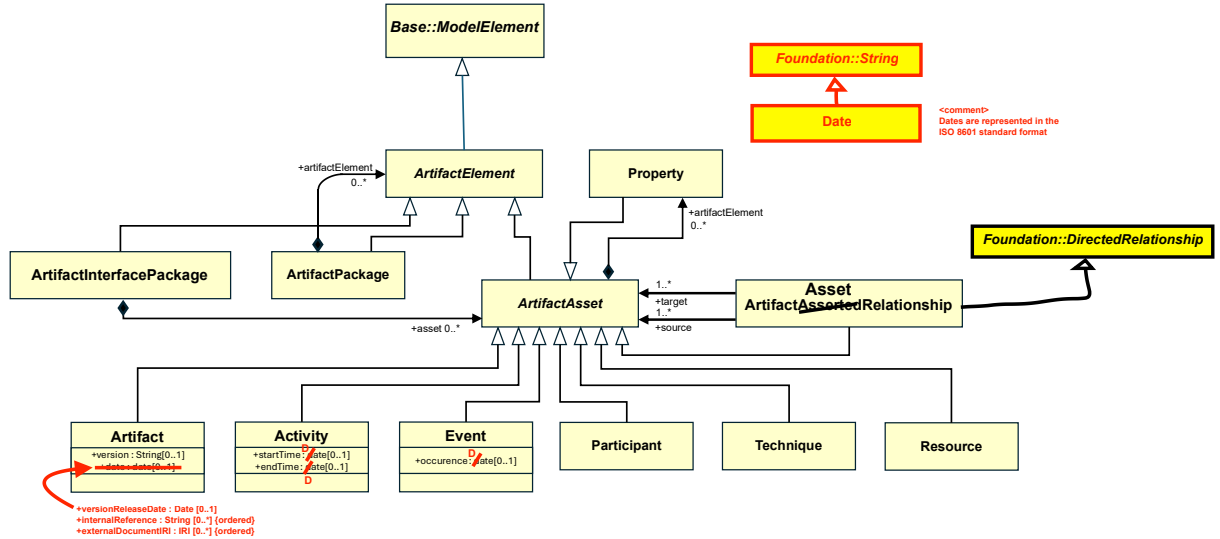


Figure 12.1 - Artifact Package Diagram

Artifacts correspond to the main evidentiary elements of an assurance case. By means of assertions (`AssertedEvidence` with `isCounter = true/false`), artifacts can be referenced (using `ArtifactReferences`) as supporting claims and arguments.

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

When made concrete, an `Artifact` can relate to many different types of information necessary for developing confidence in the `Artifact` and thus for assurance purposes. Such information can be regarded as meta-data or provenance information about an `Artifact`, provides information about its management, and is specified with the rest of specializations of `ArtifactAsset`. Using a design specification as an example, properties (`Property`) 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

internalReference : String [0..\*] {ordered} - refence 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.

## 12.8<sup>7</sup> Property

Property enables the specification of the characteristics of an Artifact.

### Superclass

ArtifactAsset

### 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).

## 12.9<sup>8</sup> Event

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

### Superclass

ArtifactAsset

### Attributes

date: ~~date~~<sup>D</sup>[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.

## 12.10<sup>9</sup> Resource

Resource corresponds to the tangible objects representing an Artifact.

### Superclass

ArtifactAsset

### Attributes

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.

## 12.11<sup>0</sup> Activity

Activity represents units of work related to the management of ArtifactAssets.

### Superclass

ArtifactAsset

### Attributes

startTime: ~~D~~ date[0..1] - time when the activity started.

endTime: ~~D~~ date[0..1] - time when the activity ended.

### Semantics

The Artifacts used in an assurance case are the result of and managed via the execution of processes, which consist of Activities: specification of requirements, design of the system, integration of system components, etc. ArtifactActivityRelationships can be used to specify the relationship between Activities and Artifacts. Activities can, for instance, be described as using a given Artifact as input or producing an Artifact as output. Activities can be related to one another using ActivityRelationships (e.g., 'preceding'). The purpose of an activity can be specified in its description.

## 12.12<sup>1</sup> Technique

Technique represents techniques associated with Artifacts (e.g., associated with the creation, inspection, review or analysis of an Artifact).

### Superclass

ArtifactAsset

### Semantics

Artifacts are created, or managed from a more general perspective, via some method whose use results in specific characteristics for the Artifacts. 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.

## 12.13<sup>2</sup> Participant

Participant enables the specification of the parties involved in the management of ArtifactAssets.

### Superclass

ArtifactAsset

### Semantics

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

## 12.14<sup>3</sup> ArtifactAssetRelationship

ArtifactAssetRelationship enables the ArtifactAssets of a structured assurance case to be linked together. The linking together of ArtifactAssets allows a user to specify that a relationship exists between the assets.

### Superclass

ArtifactAsset ← , Foundation::DirectedRelationship

## Associations

source:ArtifactAsset[1..\*] - the source of the ArtifactAssetRelationship

target:ArtifactAsset[1..\*] - the target of the ArtifactAssetRelationship

## Semantics

An ArtifactAsset can be related to other ArtifactAssets. This kind of information is specified by means of ArtifactAssetRelationships name and description of the ArtifactAssetRelationship can be used to describe the semantics of the ArtifactAssetRelationship.

### 12.14 Date

**Date is a type, whose primitive type is String and represented in the ISO 8601 standard format**