GSN Metamodel Specification

DISCLAIMER

LICENSE

[1] SCSC : Safety-Critical Systems Club C.I.C. A Community Interest Company registered in England (Company number 13084663)

1. GSN Metamodel Overview

This document provides a metamodel for GSN as defined in version2 of the GSN standard[2]. The GSN metamodel is defined by extending the Structured Assurance Case Metamodel (SACM), a standard specified and maintained by the Object Management Group (OMG). The definition of the GSN metamodel allows modelers to create GSN models that can be exchanged among modelling tools. It should be noted however that the class diagrams shown in this document (such as Figure 1) depict the abstract syntax of GSN. This abstract syntax needs to be made concrete in order to be comprehended by humans (or computers). The graphical notations defined in the GSN standard are the concrete syntax of the abstract syntax defined in this document. The concrete syntax enables GSN models (created using this metamodel) to be represented as GSN diagrams. A detailed specification of the metamodel is provided in section 2.

The definition of the GSN metamodel involves extending elements (concepts) in SACM, as well as defining additional constraints to disable elements that are not used in the context of GSN and are therefore not to be included in GSN models. Figure 1 shows a simplified class diagram describing the GSN metamodel. Elements rendered in blue in Figure 1 are elements (i.e. concepts) that are defined in the GSN standard that have been created by extending existing SACM elements. The constraints that accompany the GSN metamodel are discussed in Section 2.

It should be noted that the constraints discussed in this document are only those applicable to models created using the GSN metamodel. There is an additional set of constraints related to creating valid GSN arguments (i.e. through the use of GSN graphical notations); these constraints are discussed in the GSN standard and not in this document as they are not related to the definition of the GSN metamodel itself.

[2] GSN Community Standard Version 2, SCSC-141B, https://scsc.uk/r141B

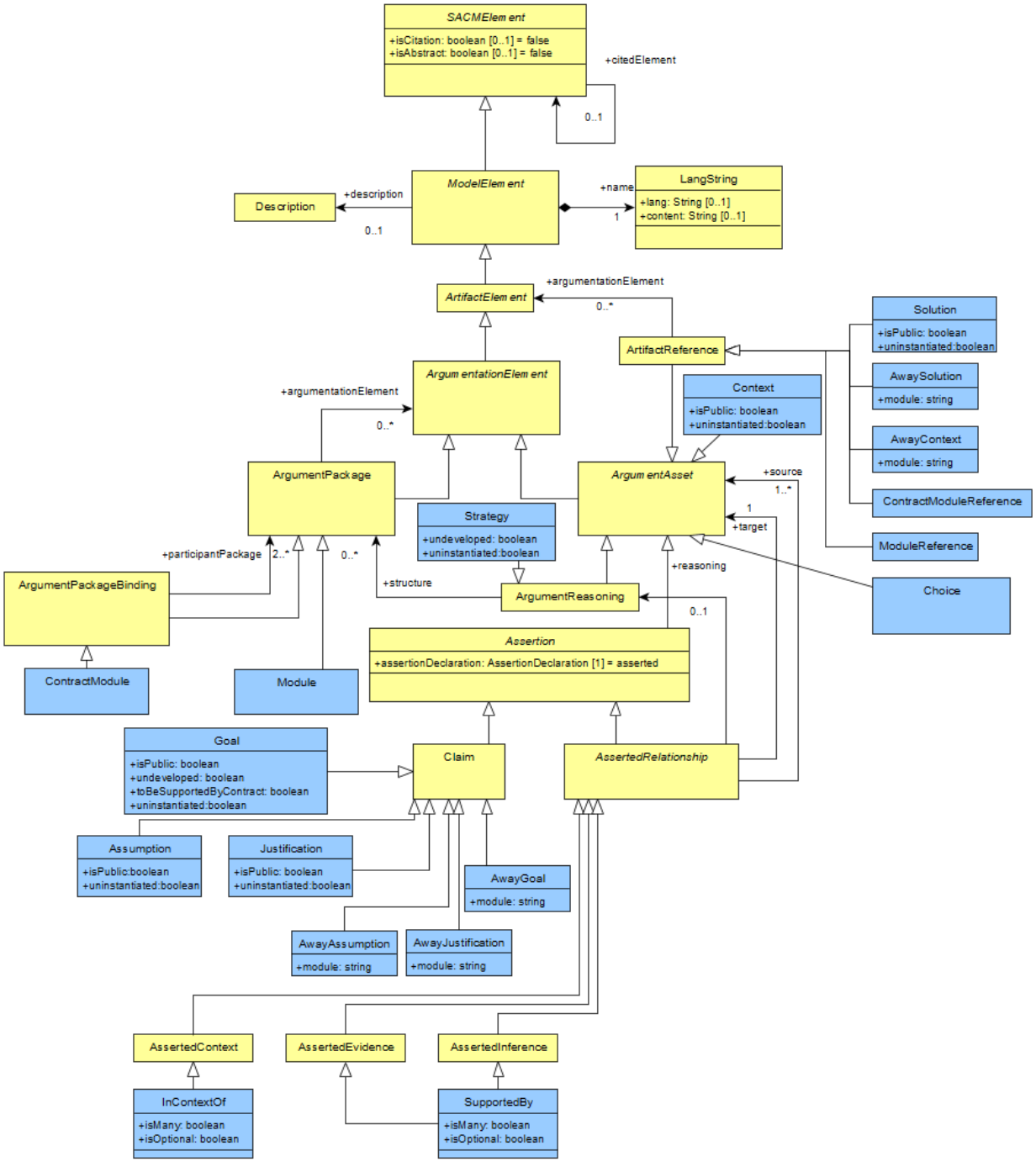

Figure 1 – Simplified Class Diagram of GSN Metamodel [3]

[3] Note that the term “element” is used differently in SACM from its use in the GSN standard. In SACM, “ArgumentationElement” includes all node entities (such as Claim, ArtifactReference, etc.) as well as relationship entities (such as AssertedContext, AssertedEvidence, etc.). In the GSN standard “GSN Element” is used to refer exclusively to node entities (Goal, Solution, etc.).

2. GSN Metamodel Specification

The GSN metamodel has been specified through the definition of extensions to, and constraints upon, the existing SACM[4] specification.

The GSN metamodel is described in two separate and related components:

- Argumentation component – Specifies all the GSN argument elements
- Base component – Specifies the foundational concepts and relationships utilised by all elements of the GSN metamodel

Each component is described by a class diagram that depicts the elements and their relationships. Elements represented in blue in the class diagrams are meta-elements for GSN (their concrete syntax are the graphical notations with the same name in the GSN notation).

Elements or attributes that are depicted with a red line through them represent elements of SACM that are not valid as part of GSN models i.e. GSN models should not contain those elements or attributes. These elements or attributes have been excluded through the definition of constraints, which when executed/applied, check the validity of GSN models. The constraints for each component are specified using OCL (Object Constraint Language)5, but tool implementors are free to choose an alternative constraint language to implement the constraints within their own tool if they so wish. These constraints can be enforced during model creation or used to check the validity of a GSN model once created. It should be noted that abstract classes in SACM are not instantiated in a GSN model.

[4] https://www.omg.org/spec/SACM/About-SACM/
[5] https://www.omg.org/spec/OCL/2.4/PDF

2.1 Argumentation Component

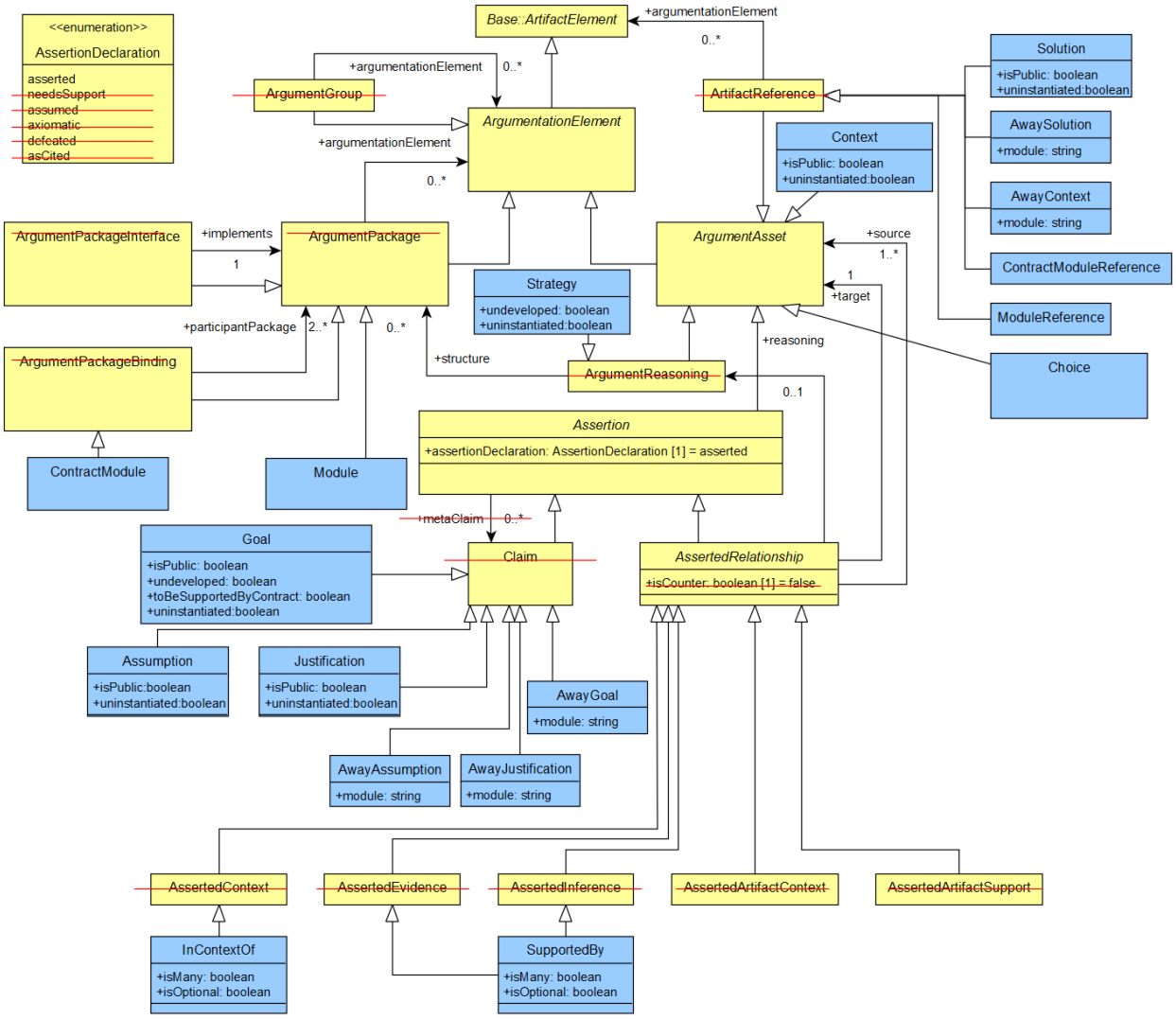

Figure 2 - GSN Argumentation Component Metamodel Class Diagram

Figure 2 shows a class diagram for the GSN argumentation component. The GSN elements are specified as extensions of elements of SACM. The constraints defined below ensure that GSN models cannot include SACM elements.

In Figure 2 “Context” is a sub-class of “ArgumentAsset” since in GSN context elements may be either of type Axiomatic Claim or of type ArtifactReference. The type of the Context element is determined by the nature of its content. For example, the context “The contractors for this project are BTR construction” would be considered an Axiomatic Claim. The context “System design description (Ref Y)” would be considered an ArtifactReference.

Note that all sub-classes of Claim in GSN are assigned the value of “asserted” for AssertionDeclaration. This ensures that attributes of elements of the SACM metamodel do not need to be considered when creating GSN models. If a model is transformed from GSN to SACM then the appropriate AssertionDeclaration must be applied to each SACM Claim to reflect the nature of the GSN element e.g. a GSN Assumption would be transformed to a SACM claim with AssertionDeclaration assigned the value of “assumed”. These transformation rules are not included in this metamodel specification.

Similarly, providing attributes of Goal allows types of Goal to be declared without the use of SACM AssertionDeclaration.

“AwayGoal” is modelled as a sub-class of “Claim” since it is type of SACM Claim, but does not inherit all of the attributes of Goal. A similar approach has been adopted for AwayJustification and AwayAssumption.

ModuleReferences are modelled as sub-classes of ArtifactReference since Modules are modelled as Artifacts within the GSN metamodel.

Choice is modelled a sub-class of ArgumentAsset to enable its connection to multiple SupportedBy relationships.

2.1.1 Argumentation Component Constraints

ArgumentPackage

```
Context:: GSN!ArgumentPackage

self.argumentationElement->forAll(e|
not e.oclIsTypeOf(GSN!Claim) and
not e.oclIsTypeOf(GSN!ArtifactReference) and
not e.oclIsTypeOf(GSN!AssertedInference) and
not e.oclIsTypeOf(GSN!AssertedContext) and
not e.oclIsTypeOf(GSN!ArgumentReasoning) and
not e.oclIsTypeOf(GSN!AssertedEvidence) and
not e.oclIsTypeOf(GSN!AssertedArtifactContext) and
not e.oclIsTypeOf(GSN!AssertedArtifactSupport) and
not e.oclIsTypeOf(GSN!ArgumentPackage) and
not e.oclIsTypeOf(GSN!ArgumentPackageInterface) and
not e.oclIsTypeOf(GSN!ArgumentPackageBinding) and
not e.oclIsTypeOf(GSN!ArgumentGroup))
```

SupportedBy

```
Context::SupportedBy

self.source.forAll(s|s.oclIsTypeOf(GSN::Strategy) or
s.oclIsTypeOf(GSN::Goal))

self.isCounter = false

self.target.forAll(s|s.oclIsTypeOf(GSN::Strategy) or
s.oclIsTypeOf(GSN::Solution) or
s.oclIsTypeOf(GSN::AwayGoal) or
s.oclIsTypeOf(GSN::AwaySolution) or
s.oclIsTypeOf(GSN::ModuleReference) or
s.oclIsTypeOf(GSN::ContractModuleReference))
```

InContextOf

```
Context::InContextOf

self.source.forAll(s|s.oclIsTypeOf(GSN::Strategy) or
s.oclIsTypeOf(GSN::Goal))

self.isCounter = false

self.target.forAll(s|s.oclIsTypeOf(GSN::Context) or
s.oclIsTypeOf(GSN::Justification) or
s.oclIsTypeOf(GSN::Assumption) or
s.oclIsTypeOf(GSN::AwayContext) or
s.oclIsTypeOf(GSN::AwayGoal) or
s.oclIsTypeOf(GSN::ModuleReference))
```

Assertion

```
Context::Assertion
self.assertionDeclaration = asserted
```

2.2 Base Component

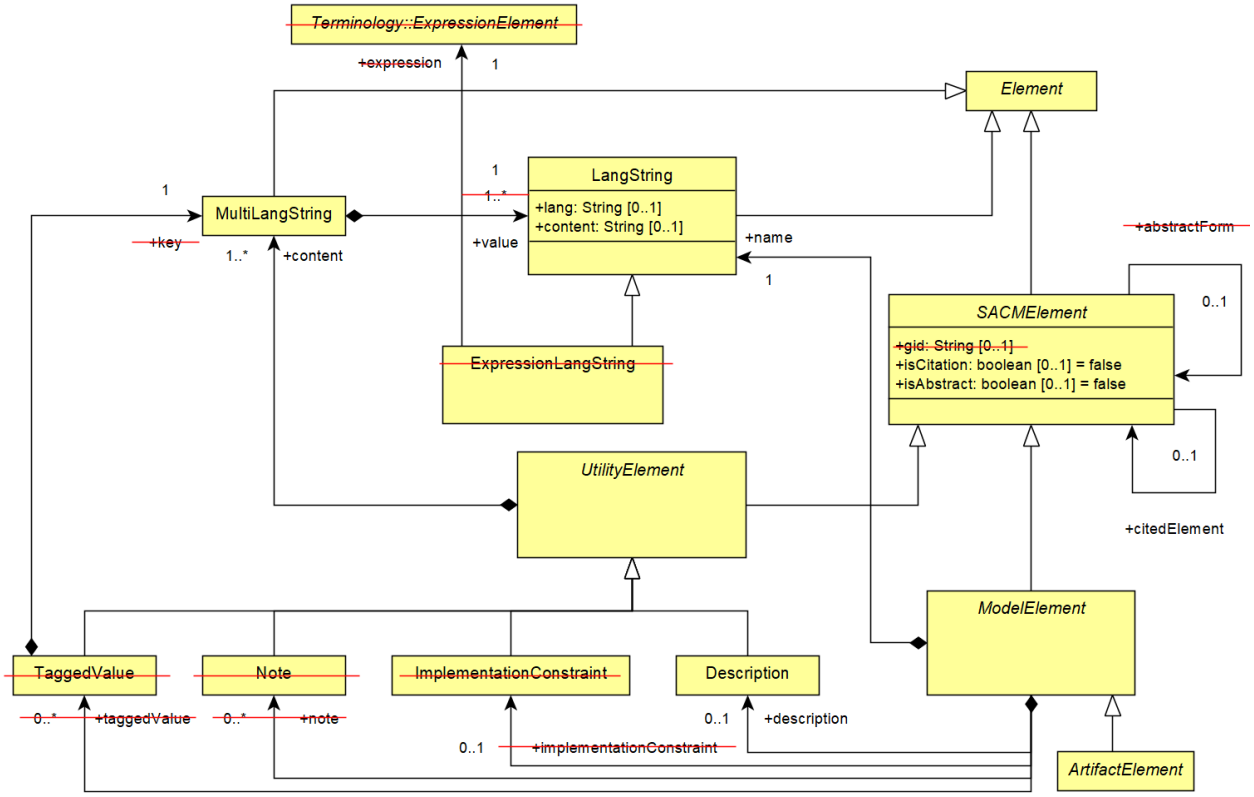

Figure 3 - GSN Base Component Metamodel Class Diagram

Figure 3 shows a class diagram for the GSN Base component. The GSN elements are specified as extensions of elements of SACM. The constraints below can be used to ensure that GSN models cannot include the “disabled” SACM elements.

2.2.1 Base Component Constraints

ModelElement

```
Context::GSN!ModelElement

self.implementationConstraint.isUndefined() = true
self.note.isUndefined() = true
self.taggedValue.isUndefined() = true
self.abstractForm.isUndefined() = true
self.gid.isUndefined() = true
not self.name.oclIsTypeOf(ExpressionLangString)
self.description.content.forAll(e|e.value.
forAll(v|v.oclIsTypeOf(SACM!LangString)))
```

MultiLangString

```
Context::GSN!ModelElement

self.value.size() == 1
```