change for a point release. The final SACM 1.1, released in July 2015, was scaled back to address some of the issues and it cleaned up some terminology and logical issues but it did not substantially alter the underlying metamodel.

During this same timeframe other efforts in the OMG (the Dependability Assurance Framework for Safety-Sensitive Consumer Devices (DAF)) and in The Open Group (the Dependability Assurance Framework (O-DA), as well as the work of the Food and Drug Administration (FDA) in the U.S., started making use of the assurance case concept and articulated implicit requirements/needs for tools that would work with assurance case models and their exchange.

Additionally, the Open Platform for EvolutioNary Certification of Safety-critical Systems (OPENCOSS) effort in Europe was exploring different uses of assurance cases, including the creation of a Common Certification Language, and the OMG's Architecture Driven Modernization Task Force crafted a Structured Pattern Metamodel Standard (SPMS) that provided a method for describing patterns within models. Together these new needs and the new openly available capabilities represented in OPENCOSS and SPMS offered a way forward.

This version 2.0 of SACM has been created as a major version release since pursuing another point release revision of SACM would appear to be incompatible with achieving the integration and harmonization that is critical to obtain wide-spread adoption and implementation within the tooling market and allow that market to deliver on some of the potential capabilities they could provide to address the emerging and evolving need for assurance case tools, such as:

- Improving the Understandability of an Assurance Case to a 3rd Party
- Improving Rigor of Assurance Cases through Modeling
- Allowing for Reexamination of Assumptions, Argument Structuring, and the Appropriateness of Evidence
- Allowing for Reuse of Sub-Claim/Evidence Constructs That “Work”
- Authoring/Sharing Libraries of Sub-Claims/Supporting Evidence
- Providing for Assurance Case Analytics/Validation
- Providing for Exchange of Assurance Cases (Import/Export)
- Providing for Enforcing Community of Interest Norms of Practice

2.0 came

The resulting metamodel in this version of SACM comes from the ideas in the 2013 Berlin metamodel, along with the approaches utilized for modeling artifact- and process-related concepts in OPENCOSS Common Certification Language and the pattern metamodel and concepts from the SPMS.

In SACM 2.1, the concrete syntax for the Argumentation metamodel is defined. The concrete syntax is designed based on visual notation design theories such as semantic transparency and structure visual inheritance. Furthermore, the existing notations in the domain such as GSN and CAE are also considered in the design process.

2 Conformance

2.1 Introduction

The Structured Assurance Case Metamodel (SACM) specification defines the following compliance points:

- Argumentation Model
- Artifact Model
- Assurance Case Model
- Terminology Model

• Concrete Syntax Graphical Notation

2.2 Argumentation Model Compliance Point

Software that conforms to the SACM specification at the Argumentation Model compliance point shall be able to import and export XMI documents that conform with the SACM XML Schema produced by applying XMI rules to the normative MOF metamodel defined in the Argumentation subpackage of the SACM specification, including the

common elements defined in the Common and Predefined diagrams of the SACM. The top object of the Argumentation package as a unit of interchange shall be the Argumentation::ArgumentPackage element of the SACM.

Conformance to the Argumentation Model compliance point does not entail support for the Evidence subpackage of SACM, or the terminology sub package of the SACM.

This compliance point facilitates interchange of the structured argumentation documents produced by existing tools supporting existing structured argument notations such as the Goal Structuring Notation (GSN) and the Claims-Arguments-Evidence (CAE) notation which provide their own mapping onto SACM argumentation aspects. Further details of these mappings are given in Annex A.

2.3 Artifact Model Compliance Point

Software that conforms to the specification at the Artifact Model compliance point shall be able to import and export XMI documents that conform with the SACM XML Schema produced by applying XMI rules to the normative MOF metamodel defined in this Artifact subpackage of the SACM specification, including the common elements defined in the Common and Predefined diagrams of the SACM. The top object of the Evidence package as a unit of interchange shall be the ArtifactModel::ArtifactPackage element of the SACM.

Conformance to the Artifact Model compliance point does not entail support for the Argumentation subpackage of SACM, or the terminology diagram of the SACM. This compliance point facilitates interchange of the packages of evidence. In particular, this compliance point facilitates development of evidence repositories in support of software assurance and regulatory compliance.

2.4 Assurance Case Model Compliance Point

This complaince point is mandatory. Software that conforms to the specification at the Assurance Case Model compliance point shall be able to import and export XMI documents that conform with the SACM XML Schema produced by applying XMI rules to the normative MOF metamodel defined in this entire specification. The top object of the Assurance Case package as a unit of interchange shall be the SACM::AssuranceCasePackage element.

The Conformance clause identifies which clauses of the specification are mandatory (or conditionally mandatory) and which are optional in order for an implementation to claim conformance to the specification.

2.5 Terminology Model Compliance Point

Software that conforms to the specification at the Terminology Model compliance point shall be able to import and export XMI documents that conform with the SACM XML Schema produced by applying XMI rules to the normative MOF metamodel defined in this entire specification. The top object of the Terminology package as a unit of interchange shall be the SACM::AssuranceCasePackage element.

The Conformance clause identifies which clauses of the specification are mandatory (or conditionally mandatory) and which are optional in order for an implementation to claim conformance to the specification.

2.6 Concrete Syntax Graphical Notation compliance point

Software that conforms to the Concrete Syntax Graphical Notation compliance point shall be able produce the notational graphics defined in the Concrete Syntax elements for the Argumentation Model.

3 References

3.1 Normative References

The following normative documents contain provisions which, through reference in this text, constitute provisions of this specification. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply.

- ISO/IEC 15026-1:2013 Systems and software engineering - Systems and software assurance - Part 1: Concepts and vocabulary, 2013. <http://www.iso.org/iso/catalogue_detail.htm?csnumber=62526>

Assurance Case

A collection of auditable claims, arguments, and evidence created to support the contention that a defined system/service will satisfy its assurance requirements.

Claim

A proposition being asserted by the author or utterer that is a true or false statement.

Evidence

Objective artifacts being offered in support of one or more claims.

Evidence Repository

A software service providing access to, and information about, a collection of evidence items, such as records, documents, and other exhibits together with related information that facilitates management of evidence, the interpretation of evidence, and understanding the evidentiary support provided to claims.

Structured argument

A particular kind of argument where the relationships between the asserted claims, and from the evidence to the claims are explicitly represented.

5 Symbols

In SACM 2.1, a number of symbols (concrete syntax) are defined for the elements in the Argumentation Metamodel, which are detailed in Section 11. The usage of these symbols are illustrated through examples in Annex C. Note: the concrete syntax for other packages are not currently defined.

~~There are no symbols defined in this specification.~~

6 Additional Information

6.1 Changes to Adopted OMG Specifications [optional]

This specification completely replaces the SACM 1.1 specification.

6.2 Acknowledgements

The following companies submitted this specification:

- MITRE Corporation
- Adelard LLP
- KDM Analytics
- Lockheed Martin
- Benchmark Consulting

The following companies supported this specification:

11.2 ArgumentGroup

ArgumentGroup can be used to associate a number of ArgumentElements to a common group (e.g., representing a common type or purpose, or being of interest to a particular stakeholder).

Superclass

ArgumentationElement

Associations

argumentationElement:ArgumentationElement[0..*] – an optional collection of ArgumentationElements organized within the ArgumentGroup.

Semantics

ArgumentGroup can be used to associate a number of ArgumentElements to a common group (e.g., representing a common type or purpose, or being of interest to a particular stakeholder). The name and the description of the ArgumentGroup should provide the semantic for understanding the ArgumentGroup. ArgumentGroups serve no structural purpose in the formation of the argument network, nor are they meant as a structural packaging mechanism (this should be done using ArgumentPackages).

11.3 ArgumentationElement (abstract)

An ArgumentationElement is the top level element of the hierarchy for argumentation elements. ArgumentationElement extends Base::ArtifactElement. Subsequently, all argument elements are considered artifacts.

Superclass

Base::ArtifactElement

Semantics

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

11.4 ArgumentPackage Class

ArgumentPackage is the containing element for a structured argument represented using the SACM Argumentation Metamodel.

Superclass

ArgumentationElement

Associations

argumentationElement:ArgumentationElement[0..*] (composition) – a collection of ArgumentationElements forming a structured argument.

Semantics

ArgumentPackages contain structured arguments. These arguments are composed of ArgumentAssets. ArgumentPackages elements can also be nested.

Constraints

If an ArgumentPackage has nested ArgumentPackages, then it is only allowed to contain ArgumentPackages.

11.5 ArgumentPackageBinding

ArgumentElement within the ArgumentPackage can be bound together by means of ArgumentPackageBinding. ArgumentPackageBinding bind the participant packages by means of argument elements that connect the cited elements of the participant packages.

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 needsSupport (i.e., currently has no supporting argument). An ArgumentPackageBinding can be used to record the mapping by means of containing a structured argument linkingArgumentElements that cite the claims in question.

ArgumentPackageBinding is a sub type of ArgumentPackage, it is used to record the argument that connects the arguments of two or more ArgumentPackages.

Add Concrete Syntax for ArgumentPackageBinding

Constraints

The participantPackages should be only ArgumentPackages

OCL: self.participantPackage->forall(pp|pp.oclIsTypeOf(Argument::ArgumentPackage))

The ArgumentElements contained by an ArgumentPackageBinding must be ArgumentElement citations to ArgumentElements contained within the ArgumentPackages associated by the participantPackage association.

11.6 ArgumentPackageInterface

ArgumentPackageInterface is a kind of ArgumentPackage that defines an interface that may be exchanged between users. An ArgumentPackage may declare one or more ArgumentPackageInterface(s).

Superclass

ArgumentPackage

Associations

implements:ArgumentPackage[1] – a reference to the ArgumentPackage which the ArgumentPackageInterface declares.

Semantics

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

For example, while 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.

Add Concrete Syntax for ArgumentPackageInterface

Superclass

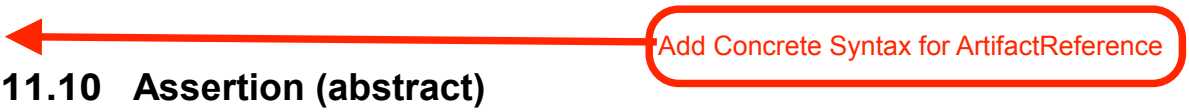
ArgumentAsset

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

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 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.”

Constraints

Self.assumed and self.toBeSupported cannot both be true simultaneously.

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.

Superclass

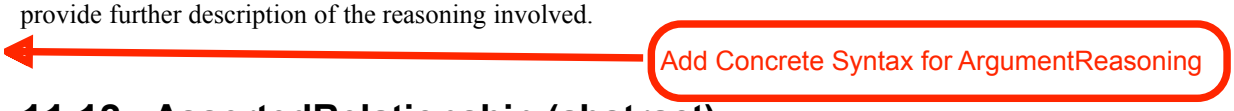
ArgumentAsset

Associations

structure:ArgumentPackage[0..1] - optional reference to another the ArgumentPackage that provides the detailed structure of the argument being described by the ArgumentReasoning.

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.



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

Attributes

isCounter:Boolean[1] = false – a flag indicating that 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 AssertedInference between two claims (A – the source – and B – the target) denotes that the truth of Claim B is said to infer the truth of Claim A.

Add Concrete Syntax for AssertedInference

11.15 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

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.

Constraints

Add Concrete Syntax for AssertedEvidence

The source of AssertedEvidence relationships must be ArtifactReference.

OCL

self.source->forall(s|s.oclIsTypeOf(ArtifactReference))

11.16 AssertedContext

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

Semantics

Contextual information often needs to be cited in order to make clear the interpretation and scope of a Claim or ArgumentReasoning description. 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 information cited by Artifact B" or conversely "InformationItem 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."

Add Concrete Syntax for AssertedContext

11.17 AssertedArtifactSupport

AssertedArtifactSupport records the assertion that one or more artifacts support another artifact.

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 can be an ambiguous relationship if the nature of these Assertions is unclear. In such cases, it would be clearer to declare explicit AssertedInferences between Claims drawn out from the ArtifactReference.

Constraints

Add Concrete Syntax for AssertedArtifactSupport

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

11.18 AssertedArtifactContext

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

Superclass

AssertedRelationship

Semantics

One or more other artifacts provide the necessary context in which the assertions associated with another artifact should be understood. Note: this can be an ambiguous relationship if the nature of these Assertions is unclear. In such cases, it would be clearer to declare explicit AssertedContext between Claims drawn out from the ArtifactReference.

Constraints

Add Concrete Syntax for AssertedArtifactContext

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

provide the semantic for understanding the ArgumentGroup. ArgumentGroups serve no structural purpose in the formation of the argument network, nor are they meant as a structural packaging mechanism (this should be done using ArgumentPackages).

11.3 ArgumentationElement (abstract)

An ArgumentationElement is the top level element of the hierarchy for argumentation elements. ArgumentationElement extends Base::ArtifactElement. Subsequently, all argument elements are considered artifacts.

Superclass

Base::ArtifactElement

Semantics

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

11.4 ArgumentPackage Class

ArgumentPackage is the containing element for a structured argument represented using the SACM Argumentation Metamodel.

Superclass

ArgumentationElement

Associations

argumentationElement:ArgumentationElement[0..*] (composition) – a collection of ArgumentationElements forming a structured argument

Semantics

ArgumentPackages contain structured arguments. These arguments are composed of ArgumentAssets. ArgumentPackages elements can also be nested.

Concrete Syntax

The concrete syntax for ArgumentPackage is defined in Figure 11.2.

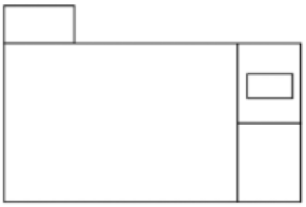


Figure 11.2 - Concrete Syntax for ArgumentPackage

Constraints

If an ArgumentPackage has nested ArgumentPackages, then it is only allowed to contain ArgumentPackages..

11.5 ArgumentPackageBinding

ArgumentElement within the ArgumentPackage can be bound together by means of ArgumentPackageBinding. ArgumentPackageBinding bind the participant packages by means of argument elements that connect the cited elements of the participant packages.

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 needsSupport (i.e. currently has no supporting argument). An ArgumentPackageBinding can be used to record the mapping by means of containing a structured argument linkingArgumentElements that cite the claims in question.

ArgumentPackageBinding is a sub type of ArgumentPackage, it is used to record the argument that connects the arguments of two or more ArgumentPackages.

Concrete Syntax

The concrete syntax for ArgumentPackageBinding is defined in Figure 11.3.

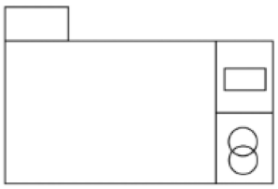


Figure 11.3 - Concrete Syntax for ArgumentPackageBinding

Constraints

The participantPackages should be only ArgumentPackages

OCL: self.participantPackage->forall(pp|pp.oclIsTypeOf(Argument::ArgumentPackage))

The ArgumentElements contained by an ArgumentPackageBinding must be ArgumentElement citations to ArgumentElements contained within the ArgumentPackages associated by the participantPackage association.

11.6 ArgumentPackageInterface

ArgumentPackageInterface is a kind of ArgumentPackage that defines an interface that may be exchanged between users. An ArgumentPackage may declare one or more ArgumentPackageInterface.

Superclass

ArgumentPackage

Associations

implements:ArgumentPackage[1] – a reference to the ArgumentPackage which the ArgumentPackageInterface declares.

Semantics

ArgumentPackageInterfaces can be used to declare (by means of containing ArgumentElement based citations) 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.

Concrete Syntax

The concrete syntax for ArgumentPackageInterface is defined in Figure 11.4.

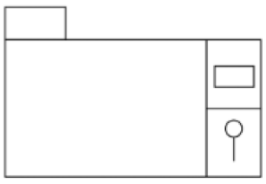


Figure 11.4 - Concrete Syntax for ArgumentPackageInterface

Constraints

ArgumentPackageInterfaces are only allowed with isCitation=true and +citedElement refer to ArgumentAssets within the ArgumentPackage implementation referred to by implements.

11.7 ArgumentAsset (abstract)

ArgumentAsset is the abstract bsse element for the elements of any structured argument represented in SACM.

Superclass

ArgumentationElement

Associations

content:Base::MultiLangString[0..1] (composition) – the content of the ArgumentAsset defined in possibly multiple languages

Semantics

ArgumentAssets represent the constituent building blocks of any structured argument contained in an ArgumentPackage.

For example, ArgumentAssets can represent the Claims made within a structured argument contained in an ArgumentPackage.

11.8 AssertionDeclaration (Enumeration)

AssertionDeclaration provides a list of declarations which can be used to declare the state of an Assertion.

Superclass

N/A

Enumeration Litterals

asserted – the default enumeration literal, indicating that an Assertion is asserted. needsSupport – a flag indicating that further argumentation has yet to be provided to support the Assertion.

assumed – a flag indicating that the Assertion being made is declared by the author as being assumed to be true rather than being supported by further argumentation.

axiomatic – a flag indicating that the Assertion being made by the author is axiomatically true, so that no further argumentation is needed.

defeated – a flag indicating that the Assertion is defeated by counter-evidence and/or argumentation.

asCited – a flag indicating that because the Assertion is cited, the AssertionDeclaration should be transitively derived from the value of the AssertionDeclaration of the cited Assertion.

Semantics

AssertionDeclaration provides a list of declarations which indicate the state of an Assertion.

11.9 ArtifactReference

ArtifactReference enables the citation of an artifact as information that relates to the structured 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 with in 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.

Concrete Syntax

The concrete syntax for an ArtifactReference is defined in Figure 11.5.

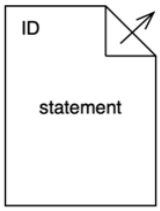


Figure 11.5 - Concrete Syntax for ArtifactReference

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)

Concrete Syntax

MetaClaim can be used as references Claims concerning (i.e., about) the Assertion (e.g., regarding the confidence in the Assertion). The concrete syntax for the +metaClaim reference is defined as below.

Figure 11.6 - Concrete Syntax for the +metaClaim reference

Examples of using the +metaClaim reference can be found in Appendix C.

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

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 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”.

Concrete Syntax

By default the AssertionDeclaration of a Claim is set to asserted, the concrete syntax for an asserted Claim is defined as below.

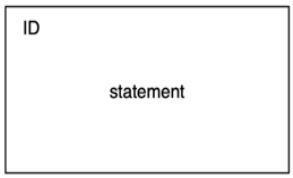


Figure 11.7 - Concrete Syntax for asserted Claim

An assumed Claim indicates that an assumption is declared without any supporting evidence or argumentation. The concrete syntax for an assumed Claim is defined as below.

Figure 11.8 - Concrete Syntax for assumed Claim

A needsSupport Claim indicates that a Claim is declared as requiring further evidence or argumentation. The concrete syntax for a needsSupport Claim is defined as below.

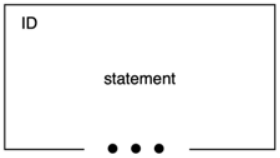


Figure 11.9 - Concrete Syntax for needsSupport Claim

An axiomatic Claim indicates that a Claim is intentionally declared to be axiomatically true. The concrete syntax of an axiomatic Claim is defined as below.

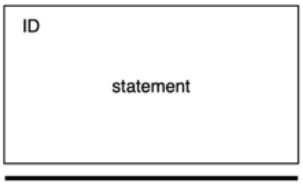


Figure 11.10 - Concrete Syntax for axiomatic Claim

A defeated Claim indicates that a Claim is defeated by counter-evidence. The concrete syntax of a defeated Claim is defined as below.

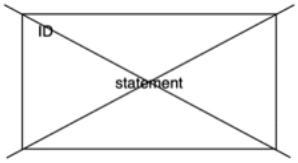


Figure 11.11 - Concrete Syntax for defeated Claim

An asCited Claim indicates that a Claim cites another claim and is hence supported by the cited Claim. The concrete syntax of an asCIted Claim is defined as below.

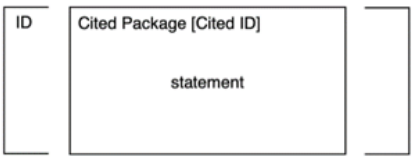


Figure 11.12 - Concrete Syntax for asCited Claim

An abstract Claim indicates that a Claim is part of a pattern or a template. The concrete syntax for an Abstract Claim is to render the Claim with dash lines, below is an example of an abstract asserted Claim.

Figure 11.13 - Concrete Syntax for abstract asserted Claim

For other types of Claims, they should be rendered in dash lines, should their +isAbstract attribute is true.

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.

Superclass

ArgumentAsset

Associations

structure:ArgumentPackage[0..1] - optional reference to another the ArgumentPackage that provides the detailed structure of the argument being described by the ArgumentReasoning.

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.

Concrete Syntax

The concrete syntax of ArgumentReasoning is defined as below (note: the right hand side of the notation).

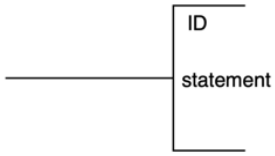


Figure 11.14 - Concrete Syntax for ArgumentReasoning

11.13 AssertedRelationship (abstract)

AssertedRelationship is the abstract association class whether 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

Attributes

isCounter:Boolean[1] = false – a flag indicating that 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 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.

Concrete Syntax

The concrete syntax of AssertedInference is defined as below, where the dot represents the AssertedInference instance, the edge without an arrow represents the +source reference of the AssertedInference, and the edge with an arrow represents the +target reference of the AssertedInference.

Figure 11.15 - Concrete Syntax for asserted AssertedInference

An assumed AssertedInference indicates that the inference is assumed without any supporting evidence or argumentation. The concrete syntax of an assumed AssertedInference is defined as below (note: the change is applied to the +target reference edge of an AssertedInference).

Figure 11.16 - Concrete Syntax for assumed AssertedInference

A needsSupport AssertedInference indicates that the inference is declared as requiring further evidence or argumentation. The concrete syntax of a needsSupport AssertedInference is defined as below (note: the change is applied to the +target reference edge of an AssertedInference).

Figure 11.17 - Concrete Syntax for needsSupport AssertedInference

An axiomatic AssertedInference indicates that the inference is declared to be axiomatically true. The concrete syntax of an axiomatic AssertedInference is defined as below (note: the change is applied to the +target reference edge of an AssertedInference).

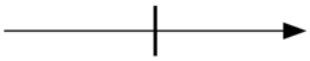


Figure 11.18 - Concrete Syntax for axiomatic AssertedInference

A defeated AssertedInference indicates that the inference is defeated by counter-evidence. The concrete syntax of a defeated AssertedInference is defined as below (note: the change is applied to the +target reference edge of an AssertedInference).

Figure 11.19 - Concrete Syntax for defeated AssertedInference

A asCited AssertedInference indicates that the inference cites another AssertedInference and is hence supported by the cited AssertedInference. The concrete syntax of an asCited AssertedInference is defined as below (note: the change is applied to the +target reference edge of an AssertedInference).

Figure 11.20 - Concrete Syntax for asCited AssertedInference

An abstract AssertedInference indicates that the inference is part of a pattern or template. The concrete syntax of an abstract AssertedInference is defined as below (note: the change is applied to the +target reference edge of an AssertedInference).

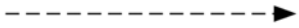


Figure 11.21 - Concrete Syntax for abstract asserted AssertedInference

For other types of AssertedInference, they should be rendered in dash lines, should their +isAbstract attribute is true.

An isCounter AssertedInference indicates that the inference counters its declared purposes. The concrete syntax of an isCounter AssertedInference is defined as below (note: the change is applied to the +target reference edge of an AssertedInference).

Figure 11.22 - Concrete Syntax for counter asserted AssertedInference

11.15 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

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.

Concrete Syntax

The concrete syntax of AssertedEvidence is defined as below, where the dot represents the AssertedEvidence instance, the edge without an arrow represents the +source reference of the AssertedEvidence, and the edge with an arrow represents the +target reference of the AssertedEvidence

Figure 11.23 - Concrete Syntax for asserted AssertedEvidence

An assumed AssertedEvidence indicates that the inference is assumed without any supporting evidence or argumentation. The concrete syntax of an assumed AssertedEvidence is defined as below (note: the change is applied to the +target reference edge of an AssertedEvidence).

Figure 11.24 - Concrete Syntax for assumed AssertedEvidence

A needsSupport AssertedEvidence indicates that the inference is declared as requiring further evidence or argumentation. The concrete syntax of a needsSupport AssertedEvidence is defined as below (note: the change is applied to the +target reference edge of an AssertedEvidence).

Figure 11.25 - Concrete Syntax for needsSupport AssertedEvidence

An axiomatic AssertedEvidence indicates that the inference is declared to be axiomatically true. The concrete syntax of an axiomatic AssertedEvidence is defined as below (note: the change is applied to the +target reference edge of an AssertedEvidence).

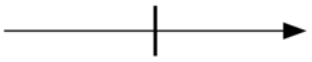


Figure 11.26 - Concrete Syntax for axiomatic AssertedEvidence

A defeated AssertedEvidence indicates that the inference is defeated by counter-evidence. The concrete syntax of a defeated AssertedEvidence is defined as below (note: the change is applied to the +target reference edge of an AssertedEvidence).

Figure 11.27 - Concrete Syntax for defeated AssertedEvidence

A asCited AssertedEvidence indicates that the inference cites another AssertedEvidence and is hence supported by the cited AssertedEvidence. The concrete syntax of an asCited AssertedEvidence is defined as below (note: the change is applied to the +target reference edge of an AssertedEvidence).

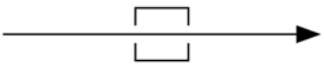


Figure 11.28 - Concrete Syntax for asCited AssertedEvidence

An abstract AssertedEvidence indicates that the inference is part of a pattern or template. The concrete syntax of an abstract AssertedEvidence is defined as below (note: the change is applied to the +target reference edge of an AssertedEvidence).

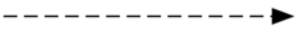


Figure 11.29 - Concrete Syntax for abstract asserted AssertedEvidence

For other types of AssertedEvidence, they should be rendered in dash lines, should their +isAbstract attribute is true.

An isCounter AssertedEvidence indicates that the inference counters its declared purposes. The concrete syntax of an isCounter AssertedEvidence is defined as below (note: the change is applied to the +target reference edge of an AssertedEvidence).

Figure 11.30 - Concrete Syntax for counter asserted AssertedEvidence

Constraints

The source of AssertedEvidence relationships must be ArtifactReference.

OCL:

self.source->forall(s|s.oclIsTypeOf(ArtifactReference))

11.16 AssertedContext

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

Semantics

Contextual information often needs to be cited in order to make clear the interpretation and scope of a Claim or ArgumentReasoning description. 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 information cited by Artifact B" or conversely "InformationItem 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".

Concrete Syntax

The concrete syntax of AssertedContext is defined as below, where the dot represents the AssertedContext instance, the edge without an arrow represents the +source reference of the AssertedContext, and the edge with an arrow represents the +target reference of the AssertedContext.

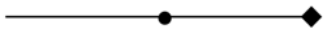


Figure 11.31 - Concrete Syntax for asserted AssertedContext

An assumed AssertedContext indicates that the inference is assumed without any supporting evidence or argumentation. The concrete syntax of an assumed AssertedContext is defined as below (note: the change is applied to the +target reference edge of an AssertedContext).

Figure 11.32 - Concrete Syntax for assumed AssertedContext

A needsSupport AssertedContext indicates that the inference is declared as requiring further evidence or argumentation. The concrete syntax of a needsSupport AssertedContext is defined as below (note: the change is applied to the +target reference edge of an AssertedContext).

Figure 11.33 - Concrete Syntax for needsSupport AssertedContext

An axiomatic AssertedContext indicates that the inference is declared to be axiomatically true. The concrete syntax of an axiomatic AssertedContext is defined as below (note: the change is applied to the +target reference edge of an AssertedContext).

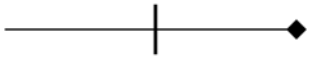


Figure 11.34 - Concrete Syntax for axiomatic AssertedContext

A defeated AssertedContext indicates that the inference is defeated by counter-evidence. The concrete syntax of a defeated AssertedContext is defined as below (note: the change is applied to the +target reference edge of an AssertedContext).

Figure 11.35 - Concrete Syntax for defeated AssertedContext

A asCited AssertedContext indicates that the inference cites another AssertedContext and is hence supported by the cited AssertedContext. The concrete syntax of a defeated AssertedInference is defined as below (note: the change is applied to the +target reference edge of an AssertedContext).

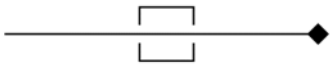


Figure 11.36 - Concrete Syntax for asCited AssertedContext

An abstract AssertedContext indicates that the inference is part of a pattern or template. The concrete syntax of a defeated AssertedContext is defined as below (note: the change is applied to the +target reference edge of an AssertedContext).

Figure 11.37 - Concrete Syntax for abstract asserted AssertedContext

For other types of AssertedContext, they should be rendered in dash lines, should their +isAbstract attribute is true.

An isCounter AssertedContext indicates that the inference counters its declared purposes. The concrete syntax of an isCounter AssertedContext is defined as below (note: the change is applied to the +target reference edge of an AssertedContext).

Figure 11.38 - Concrete Syntax for counter asserted AssertedContext

11.17 AssertedArtifactSupport

AssertedArtifactSupport records the assertion that one or more artifacts support another artifact.

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 can be an ambiguous relationship if the nature of these Assertions is unclear. In such cases, it would be clearer to declare explicit AssertedInferences between Claims drawn out from the ArtifactReference.

Concrete Syntax

The concrete syntax of AssertedArtifactSupport is defined as below, where the dot represents the AssertedArtifactSupport instance, the edge without an arrow represents the +source reference of the AssertedArtifactSupport, and the edge with an arrow represents the +target reference of the AssertedArtifactSupport.

Figure 11.39 - Concrete Syntax for asserted AssertedArtifactSupport

An assumed AssertedArtifactSupport indicates that the inference is assumed without any supporting evidence or argumentation. The concrete syntax of an assumed AssertedArtifactSupport is defined as below (note: the change is applied to the +target reference edge of an AssertedArtifactSupport).

Figure 11.40 - Concrete Syntax for assumed AssertedArtifactSupport

A needsSupport AssertedArtifactSupport indicates that the inference is declared as requiring further evidence or argumentation. The concrete syntax of a needsSupport AssertedArtifactSupport is defined as below (note: the change is applied to the +target reference edge of an AssertedArtifactSupport).

Figure 11.41 - Concrete Syntax for needsSupport AssertedArtifactSupport

An axiomatic AssertedArtifactSupport indicates that the inference is declared to be axiomatically true. The concrete syntax of an axiomatic AssertedArtifactSupport is defined as below (note: the change is applied to the +target reference edge of an AssertedArtifactSupport).

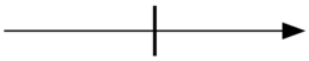


Figure 11.42 - Concrete Syntax for axiomatic AssertedArtifactSupport

A defeated AssertedArtifactSupport indicates that the inference is defeated by counter-evidence. The concrete syntax of a defeated AssertedArtifactSupport is defined as below (note: the change is applied to the +target reference edge of an AssertedArtifactSupport).

Figure 11.43 - Concrete Syntax for defeated AssertedArtifactSupport

A asCited AssertedArtifactSupport indicates that the inference cites another AssertedArtifactSupport and is hence supported by the cited AssertedArtifactSupport. The concrete syntax of a defeated AssertedInference is defined as below (note: the change is applied to the +target reference edge of an AssertedArtifactSupport).

Figure 11.44 - Concrete Syntax for asCited AssertedArtifactSupport

An abstract AssertedArtifactSupport indicates that the inference is part of a pattern or template. The concrete syntax of a defeated AssertedArtifactSupport is defined as below (note: the change is applied to the +target reference edge of an AssertedArtifactSupport).

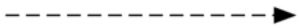


Figure 11.45 - Concrete Syntax for abstract asserted AssertedArtifactSupport

For other types of AssertedArtifactSupport, they should be rendered in dash lines, should their +isAbstract attribute is true.

An isCounter AssertedArtifactSupport indicates that the inference counters its declared purposes. The concrete syntax of an isCounter AssertedArtifactSupport is defined as below (note: the change is applied to the +target reference edge of an AssertedArtifactSupport).

Figure 11.46 - Concrete Syntax for counter asserted AssertedArtifactSupport

Note: although the graphical notation of AssertedArtifactSupport is similar to AssertedInference/AssertedEvidence, they are distinguishable through the types of elements that the +source and +target references connect to.

Constraints

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

11.18 AssertedArtifactContext

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

Superclass

AssertedRelationship

Semantics

One or more other artifacts provide the necessary context in which the assertions associated with another artifact should be understood. Note: this can be an ambiguous relationship if the nature of these Assertions is unclear. In such cases, it would be clearer to declare explicit AssertedContext between Claims drawn out from the ArtifactReference.

Concrete Syntax

The concrete syntax of AssertedArtifactContext is defined as below, where the dot represents the AssertedArtifactContext instance, the edge without an arrow represents the +source reference of the AssertedArtifactContext, and the edge with an arrow represents the +target reference of the AssertedArtifactContext.

Figure 11.47 - Concrete Syntax for asserted AssertedArtifactContext

An assumed AssertedArtifactContext indicates that the inference is assumed without any supporting evidence or argumentation. The concrete syntax of an assumed AssertedArtifactContext is defined as below (note: the change is applied to the +target reference edge of an AssertedArtifactContext).

Figure 11.48 - Concrete Syntax for assumed AssertedArtifactContext

A needsSupport AssertedArtifactContext indicates that the inference is declared as requiring further evidence or argumentation. The concrete syntax of a needsSupport AssertedArtifactContext is defined as below (note: the change is applied to the +target reference edge of an AssertedArtifactContext).

Figure 11.49 - Concrete Syntax for needsSupport AssertedArtifactContext

An axiomatic AssertedArtifactContext indicates that the inference is declared to be axiomatically true. The concrete syntax of an axiomatic AssertedArtifactContext is defined as below (note: the change is applied to the +target reference edge of an AssertedArtifactContext).

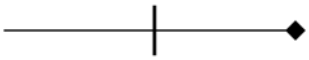


Figure 11.50 - Concrete Syntax for axiomatic AssertedArtifactContext

A defeated AssertedArtifactContext indicates that the inference is defeated by counter-evidence. The concrete syntax of a defeated AssertedArtifactContext is defined as below (note: the change is applied to the +target reference edge of an AssertedArtifactContext).

Figure 11.51 - Concrete Syntax for defeated AssertedArtifactContext

A asCited AssertedArtifactContext indicates that the inference cites another AssertedArtifactContext and is hence supported by the cited AssertedArtifactContext. The concrete syntax of a defeated AssertedInference is defined as below (note: the change is applied to the +target reference edge of an AssertedArtifactContext).

Figure 11.52 - Concrete Syntax for asCited AssertedArtifactContext

An abstract AssertedArtifactContext indicates that the inference is part of a pattern or template. The concrete syntax of a defeated AssertedArtifactContext is defined as below (note: the change is applied to the +target reference edge of an AssertedArtifactContext).

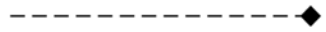


Figure 11.53 - Concrete Syntax for abstract asserted AssertedArtifactContext

For other types of AssertedArtifactContext, they should be rendered in dash lines, should their +isAbstract attribute is true.

An isCounter AssertedArtifactContext indicates that the inference counters its declared purposes. The concrete syntax of an isCounter AssertedArtifactContext is defined as follows (note: the change is applied to the +target reference edge of an AssertedArtifactContext).

Note: although the graphical notation of AssertedArtifactContext is similar to AssertedContext, they are distinguishable through the types of elements that the +source and +target references connect to.

Constraints

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

Part III - Artifact Metamodel

This part of the specification defines the Artifact Metamodel.

Annex C: Examples of Argumentation Elements

(informative)

C.1 Claims

In some cases, it is necessary to state explicitly the assumption to support the declared Assertion in an argumentation. For example, Claims G2 and G3 are asserted to support Claim G1, the relationship between them is declared using an AssertedInference. In this case, an assumed Claim A1 is declared to explicitly describe the assumption that is being made to support the AssertedInference between Claim G2, G3, and G1.

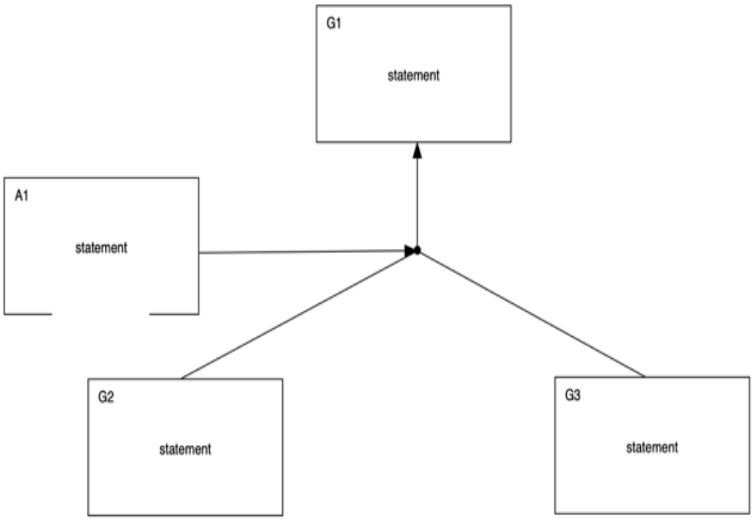


Figure C1 – Example of Claim Assumptions

A needsSuport Claim indicates a Claim is intentionally declared as requiring further evidence or argumentation. For example, Claim G11 is supported by Claim G12 and Claim G13. However, both Claim G12 and Claim 13 is declared as needsSupport Claims, indicates that both Claims required further evidence or argumentation.

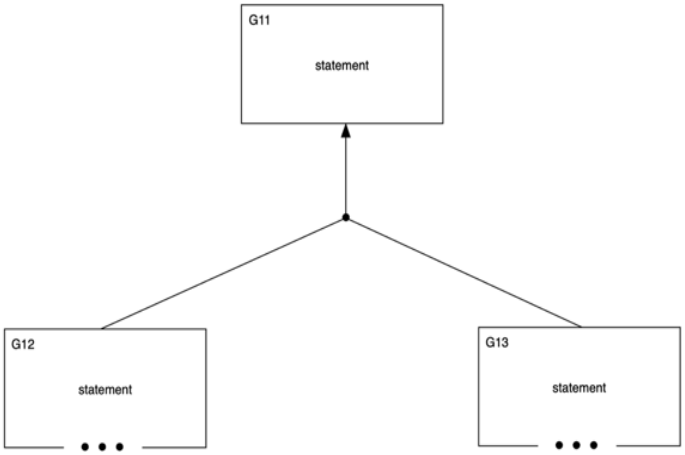


Figure C2 – Example of a Claim needing support

An axiomatic indicates a Claim is intentionally declared as axiomatically true. In some cases, an axiomatic Claim can be used to support the assertion that is made in an argumentation. For example, an axiomatic Claim AC1 is declared to support the inference (using AssertedInference) that is asserted from the Claim G8 and Claim G9 to support Claim G6.

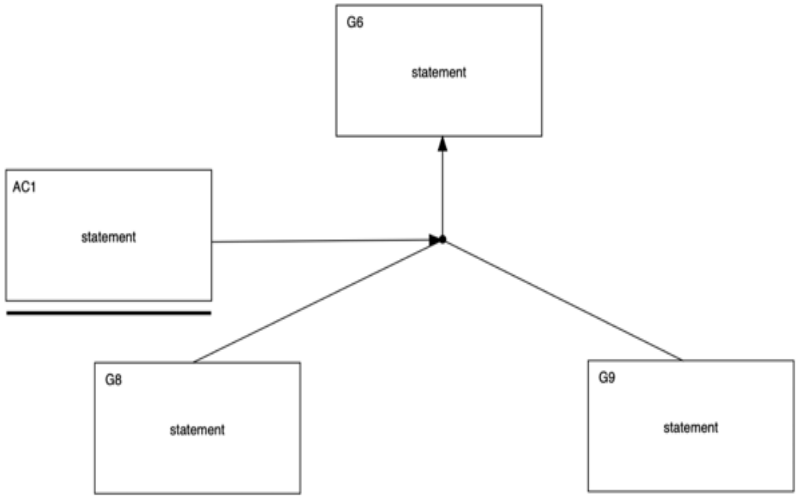


Figure C3 – Example of an Axiomatic Claim

A defeated Claim indicates a Claim is defeated by counter-evidence. For example, Claim G9 is defeated by evidence E3 (cited using ArtifactReference) that is declared using the counter-evidence relationship. Therefore, the Claim G9 is further declared as Defeated Claim.

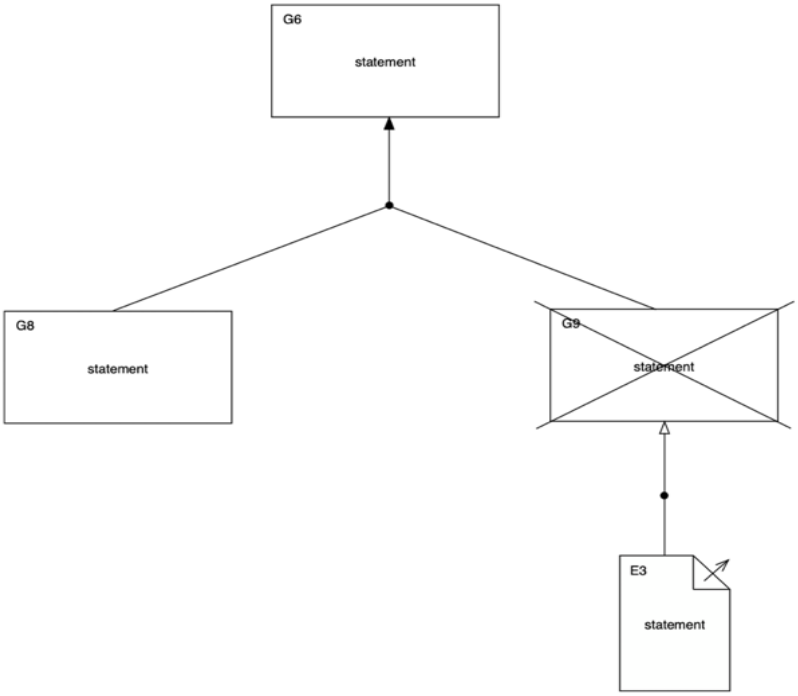


Figure C4 – Example of Defeating a Claim

An asCited Claim indicates a Claim which cites another claim and supported by the cited claim. The identifier of the Claim is placed in the top-left corner of the square brackets. The identifier of the cited Claim is placed in the top-left corner of the cited Claim and is written within a square bracket. An optional identifier of the cited package where the cited claim is located, can be written before the cited claim identifier. For example, Claim G3 is supported by Claim G6

and Claim G7. Claim G7 is declared as asCited Claim that is a Claim that cited another Claim, in this case is Claim G10.

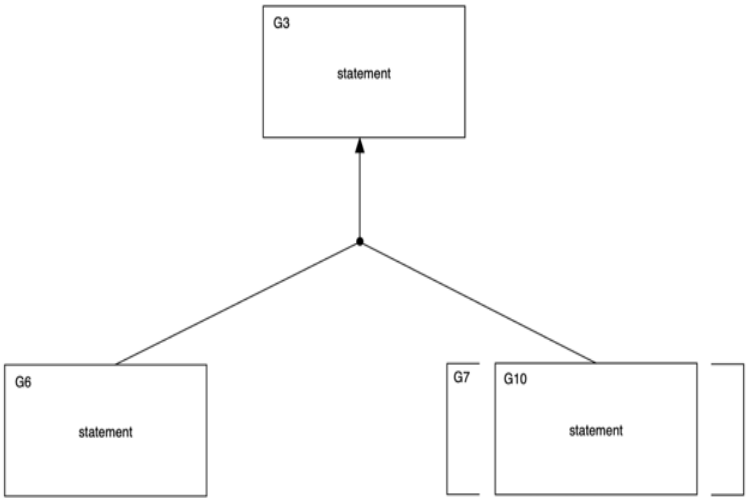


Figure C5 – Example of Claim citation

An abstract Claim indicates a Claim is part of a pattern or template. For example, Claim G1, G2, and G3 are declared as an abstract Claim that indicates that abstract Claim G1, G2, and G3 are part of argument pattern.

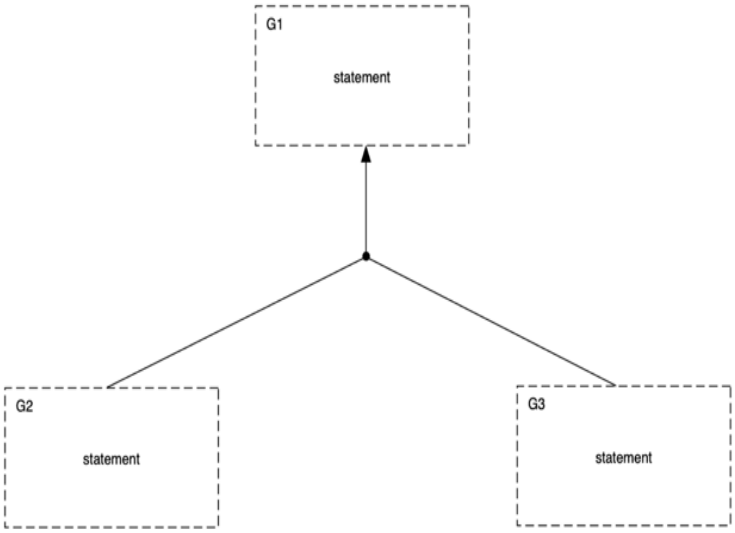


Figure C6 – Example of abstract Claims forming an argument pattern

C.2 MetaClaim

When used in a diagram, the source element of the MetaClaim must be type of Claim and the targeted element can be type of Assertion. The location of the source element of the MetaClaim must be located on the left and rigt side of the targeted element and the relationship between them is declared using the MetaClaim.

For example, Claim MC1 that is connected to Claim G1. The relationship between MC1 and G1 is declared using the MetaClaim, indicates Claim MC1 is concerning (i.e. about) Claim G1.

Figure C7 – Example of Claim and MetaClaim Relationship

C.3 AssertedInference

One or more assertions (e.g. Claims) can be linked together using the AssertedInference relationship. The direction of the AssertedInference relationship starts from the supporting element to the supported element. When used in a diagram, a connected dot is used as a connection point when more than one AssertedInferences are connected.

For example, Claim G1 is supported by Claim G2 and G3. The direction of the AssertedInference relationship is start from the supporting elements, Claim G2 and G3, to the supported element, Claim G1.

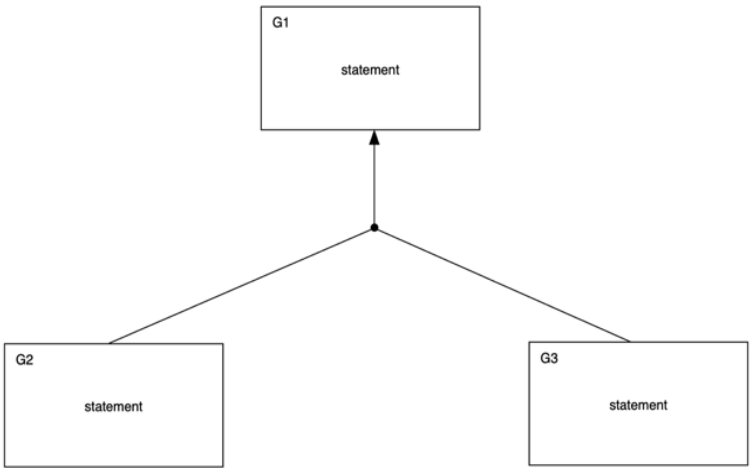


Figure C8 – Example of AssertedInference of Supporting Claims

C.4 ArtifactReference and AssertedEvidence

AssertedEvidence can be used to records the declaration that one or more artifacts of Evidence (cited by ArtifactReference) provide supporting information that helps establish the truth of a Claim. When used in a diagram, the

direction of the AssertedEvidence relationship starts from the evidence (cited by ArtifactReference) to the supported element. The position of the ArtifactReference as evidence must be located below the supported element.

For example, Claim G4 is supported by Evidence E1 (cited by ArtifactReference), connected via AssertedEvidence relationship.

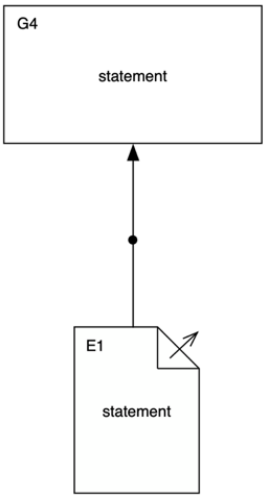


Figure C9 – Example of ArtifactReference Citation via AssertedEvidence

In another case, ArtifactReference as evidence can be used to support another ArtifactReference, for example ArtifactReference as context, to provides evidential information. In this case, ArtifactReference as evidence E2 is declared to support ArtifactReference as context C1. The ArtifactReference as evidence E2 is located below ArtifactReference C1. The relationship between them is declared using the AssertedArtifactSupport.

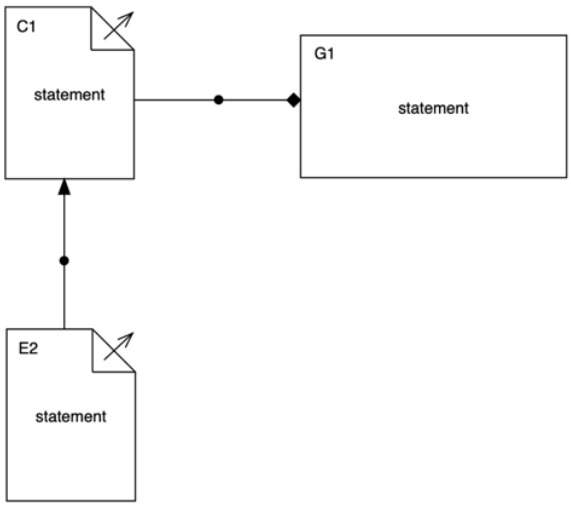


Figure C10 – Example of ArtifactReference Support of Another ArtifactReference

C.5 AssertedContext

AssertedContext can be used to declare that the artifact (cited by an ArtifactReference) provides the context for the interpretation and scoping of a Claim. When used in a diagram, the source element of the AssertedContext must be an ArtifactRefence element, and the targeted element can be the Assertion type element (e.g. Claim). The location of the ArtifactReference as a context must be located on the left and right side of the targeted element.

For example, ArtifactReference C1 as a context provides contextual information to the Claim G1 that is connected using AssertedContext relationship.

In another case, ArtifactReference as context can be used to provides contextual information to another ArtifactReference (as evidence). In this case, ArtifactReference as context C2 is located on the right side of the ArtifactReference as evidence E1. The relationship between them is declared using the AssertedArtifactContext relationship.

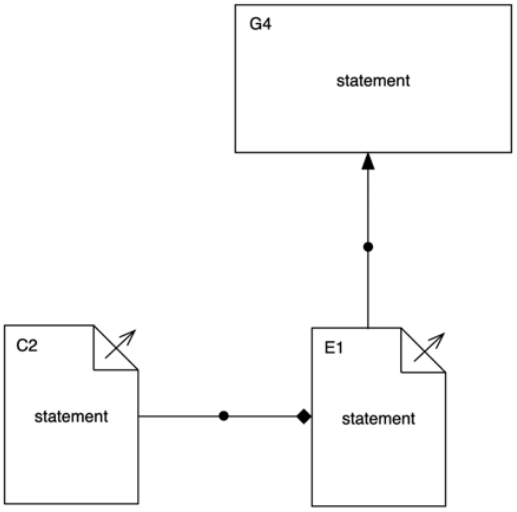


Figure C11 – Example of AssertedContext

This page intentionally left blank.