Date: March 2010

System Assurance Task Force: Argument Metamodel (ARM)

Version 1

OMG Document Number: Sysa/10-03-15

Standard document URL: http://www.omg.org/spec/arm/1.0/PDF

The material in this document details an Object Management Group specification in accordance with the terms, conditions and notices set forth below. This document does not represent a commitment to implement any portion of this specification in any company's products. The information contained in this document is subject to change without notice.

LICENSES

The companies listed above have granted to the Object Management Group, Inc. (OMG) a nonexclusive, royalty-free, paid up, worldwide license to copy and distribute this document and to modify this document and distribute copies of the modified version. Each of the copyright holders listed above has agreed that no person shall be deemed to have infringed the copyright in the included material of any such copyright holder by reason of having used the specification set forth herein or having conformed any computer software to the specification.

Subject to all of the terms and conditions below, the owners of the copyright in this specification hereby grant you a fully-paid up, non-exclusive, nontransferable, perpetual, worldwide license (without the right to sublicense), to use this specification to create and distribute software and special purpose specifications that are based upon this specification, and to use, copy, and distribute this specification as provided under the Copyright Act; provided that: (1) both the copyright notice identified above and this permission notice appear on any copies of this specification; (2) the use of the specifications is for informational purposes and will not be copied or posted on any network computer or broadcast in any media and will not be otherwise resold or transferred for commercial purposes; and (3) no modifications are made to this specification. This limited permission automatically terminates without notice if you breach any of these terms or conditions. Upon termination, you will destroy immediately any copies of the specifications in your possession or control.

PATENTS

The attention of adopters is directed to the possibility that compliance with or adoption of OMG specifications may require use of an invention covered by patent rights. OMG shall not be responsible for identifying patents for which a license may be required by any OMG specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. OMG specifications are prospective and advisory only. Prospective users are responsible for protecting themselves against liability for infringement of patents.

GENERAL USE RESTRICTIONS

DISCLAIMER OF WARRANTY

WHILE THIS PUBLICATION IS BELIEVED TO BE ACCURATE, IT IS PROVIDED "AS IS" AND MAY CONTAIN ERRORS OR MISPRINTS. THE OBJECT MANAGEMENT GROUP AND THE COMPANIES LISTED ABOVE MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS PUBLICATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF TITLE OR OWNERSHIP, IMPLIED WARRANTY OF MERCHANTABILITY OR WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE OR USE. IN NO EVENT SHALL THE OBJECT MANAGEMENT GROUP OR ANY OF THE COMPANIES LISTED ABOVE BE LIABLE FOR ERRORS CONTAINED HEREIN OR FOR DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL, RELIANCE OR COVER DAMAGES, INCLUDING LOSS OF PROFITS, REVENUE, DATA OR USE, INCURRED BY ANY USER OR ANY THIRD PARTY IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS MATERIAL, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

The entire risk as to the quality and performance of software developed using this specification is borne by you. This disclaimer of warranty constitutes an essential part of the license granted to you to use this specification.

RESTRICTED RIGHTS LEGEND

Use, duplication or disclosure by the U.S. Government is subject to the restrictions set forth in subparagraph (c) (1) (ii) of The Rights in Technical Data and Computer Software Clause at DFARS 252.227-7013 or in subparagraph (c)(1) and (2) of the Commercial Computer Software - Restricted Rights clauses at 48 C.F.R. 52.227-19 or as specified in 48 C.F.R. 227-7202-2 of the DoD F.A.R. Supplement and its successors, or as specified in 48 C.F.R. 12.212 of the Federal Acquisition Regulations and its successors, as applicable. The specification copyright owners are as indicated above and may be contacted through the Object Management Group, 140 Kendrick Street, Needham, MA 02494, U.S.A.

TRADEMARKS

MDA®, Model Driven Architecture®, UML®, UML Cube logo®, OMG Logo®, CORBA® and XMI® are registered trademarks of the Object Management Group, Inc., and Object Management Group™, OMG™ , Unified Modeling Language™, Model Driven Architecture Logo™, Model Driven Architecture Diagram™, CORBA logos™, XMI Logo™, CWM™, CWM Logo™, IIOP™ , MOF™ , OMG Interface Definition Language (IDL)™ , and OMG SysML™ are trademarks of the Object Management Group. All other products or company names mentioned are used for identification purposes only, and may be trademarks of their respective owners.

COMPLIANCE

The copyright holders listed above acknowledge that the Object Management Group (acting itself or through its designees) is and shall at all times be the sole entity that may authorize developers, suppliers and sellers of computer software to use certification marks, trademarks or other special designations to indicate compliance with these materials.

Software developed under the terms of this license may claim compliance or conformance with this specification if and only if the software compliance is of a nature fully matching the applicable compliance points as stated in the specification. Software developed only partially matching the applicable compliance points may claim only that the software was based on this specification, but may not claim compliance or conformance with this specification. In the event that testing suites are implemented or approved by Object Management Group, Inc., software developed using this specification may claim compliance or conformance with the specification only if the software satisfactorily completes the testing suites.

OMG's Issue Reporting Procedure

All OMG specifications are subject to continuous review and improvement. As part of this process we encourage readers to report any ambiguities, inconsistencies, or inaccuracies they may find by completing the Issue Reporting Form listed on the main web page http://www.omg.org, under Documents, Report a Bug/Issue (http://www.omg.org/technology/agreement.)

Table of Contents

Preface

OMG

Founded in 1989, the Object Management Group, Inc. (OMG) is an open membership, not-for-profit computer industry standards consortium that produces and maintains computer industry specifications for interoperable, portable, and reusable enterprise applications in distributed, heterogeneous environments. Membership includes Information Technology vendors, end users, government agencies, and academia.

OMG member companies write, adopt, and maintain its specifications following a mature, open process. OMG's specifications implement the Model Driven Architecture® (MDA®), maximizing ROI through a full-lifecycle approach to enterprise integration that covers multiple operating systems, programming languages, middleware and networking infrastructures, and software development environments. OMG's specifications include: UML® (Unified Modeling Language™); CORBA® (Common Object Request Broker Architecture); CWM™ (Common Warehouse Metamodel); and industry-specific standards for dozens of vertical markets.

More information on the OMG is available at http://www.omg.org/.

OMG Specifications

As noted, OMG specifications address middleware, modeling and vertical domain frameworks. A Specifications Catalog is available from the OMG website at:

http://www.omg.org/technology/documents/spec_catalog.htm

Specifications within the Catalog are organized by the following categories:

OMG Modeling Specifications

UML

MOF

XMI

CWM

Profile specifications

OMG Middleware Specifications

CORBA/IIOP

IDL/Language Mappings

Specialized CORBA specifications

CORBA Component Model (CCM)

Platform Specific Model and Interface Specifications

CORBAservices

CORBAfacilities

OMG Domain specifications

OMG Embedded Intelligence specifications

OMG Security specifications

All of OMG's formal specifications may be downloaded without charge from our website. (Products implementing OMG specifications are available from individual suppliers.) Copies of specifications, available in PostScript and PDF format, may be obtained from the Specifications Catalog cited above or by contacting the Object Management Group, Inc. at:

OMG Headquarters
140 Kendrick Street
Building A, Suite 300
Needham, MA 02494
USA
Tel: +1-781-444-0404
Fax: +1-781-444-0320
Email: *pubs@omg.org*

Certain OMG specifications are also available as ISO standards. Please consult http://www.iso.org

Typographical Conventions

The type styles shown below are used in this document to distinguish programming statements from ordinary English. However, these conventions are not used in tables or section headings where no distinction is necessary.

Times/Times New Roman - 10 pt.: Standard body text

Helvetica/Arial - 10 pt. Bold: OMG Interface Definition Language (OMG IDL) and syntax elements.

`Courier - 10 pt. Bold:` Programming language elements.

Helvetica/Arial - 10 pt: Exceptions

NOTE: Terms that appear in italics are defined in the glossary. Italic text also represents the name of a document, specification, or other publication.

1 Scope

This specification defines a metamodel for representing structured arguments. A convincing and valid argument that a system meets its assurance requirements is at the heart of an assurance case, which also may contain extensive references to evidence. ARM facilitates projects by allowing them to effectively and succinctly communicate in a structured way how their systems and services are meeting their assurance requirements. The scope of ARM is therefore to allow the interchange of structured arguments between diverse tools by different vendors. Each ARM instance represents the argument that is being asserted by the stakeholder that is offering the argument for consideration.

This specification is designed to stand alone, but eventually may be integrated with the Software Assurance Evidence Metamodel (SAEM) standard, also being developed by OMG. SAEM is designed to represent aspects of evidence and properties about evidence in further detail. In this ARM we have a simplified support to model the relation of evidence to a structured argument.

Standardization will ensure that end users are investing not just in individual tools but also rather into a coordinated strategy.

The metamodel for argumentation provides a common structure and interchange format that facilitates the exchange of system assurance arguments contained within individual tool models. The metamodel represents the core concepts for structured argumentation that underlie a number of existing argumentation notations.

2 Conformance

The metamodel presented here for structured argumentation is simple. A compliant tool shall implement all aspects of the metamodel.

3 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.

- OMG UML 2.2 Infrastructure Specification formal/2009-02-04
- OMG Meta-Object Facility (MOF) ver. 2.0 formal/2006-01-01

4 Terms and Definitions

For the purposes of this specification, the following terms and definitions apply.

Argument

A body of information presented with the intention to establish one or more claims through the presentation of related supporting claims, evidence and contextual information.

Structured argument

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

Evidence

Information or objective artifacts being offered in support of one or more claims.

Claim

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

5 Symbols

There are no symbols defined in this specification.

6 Additional Information

6.1 Changes to Adopted OMG Specifications

There are no changes to other OMG specifications.

6.2 How to Read this Specification

The rest of this document contains the technical content of this specification.

Chapter 7. Specification overview - Provides design rationale for the ARM specification.

Chapter 8. ARM – Provides the details of the ARM specification.

6.3 Acknowledgements and Contacts

All questions about this specification should be directed to:

Luke Emmet
Adelard LLP
Phone: +44 (020) 7490-9457
e-mail: luke.emmet@adelard.com

Contact information for representatives of other co-submitting organisations are:

Luke Emmet
Adelard LLP
Phone: +44 (020) 7490-9457
e-mail: luke.emmet@adelard.com

Dr. Tim Kelly
University of York
Phone: +44 (01904) 432-764
e-mail: tim.kelly@cs.york.ac.uk

Djenana Campara
KDM Analytics Inc
Phone: (613) 867-7918
e-mail: djenana@kdmanalytics.com

Nikolai Mansourov
KDM Analytics Inc
Phone: (613) 276-2323
e-mail: nick@kdmanalytics.com

Dr. Ben Calloni
Lockheed Martin
Phone: (817) 935-4482
e-mail: ben.a.calloni@lmco.com

Victor Harrison
Computer Sciences Corporation
Phone: (703) 876-1366
e-mail: vharris6@csc.com

Contact information for representatives of supporting organisations are:

Michael Kass
NIST
Phone: (301) 975-3266
e-mail: michael.kass@nist.gov

Robert Martin
MITRE
Phone: (781) 271-3001
e-mail: ramartin@mitre.org

Carol Woody
Carnegie Mellon University
Phone: (412) 268-9137
e-mail: cwoody@cert.org

Additional acknowledgment:

Sam Redwine
Sam Redwine Consulting
Phone: (540) 209-0316
e-mail: samredwine@ieee.org

7 ARM – background and rationale

7.1 Background – the need for assurance cases

All sectors of society are placing growing reliance on software-dependent systems, both information systems and embedded systems. Adequate functioning of many of these systems is critical to the well-being of organizations and society. Today, these numerous, large, complex systems provide increased benefits by connecting with others and generally directly or indirectly to the Internet.

However the societal and individual risks posed by attacks on, or in the maladaptive behaviour of such systems are significant enough to warrant a pro-active technology adoption approach whereby the emergent risks can be analysed, explored, communicated and ultimately accepted by those responsible for the assurance.

Thus, software suppliers face the task of engineering their products and services to meet these challenges and threats in such a way that users and other stakeholders can rationally possess the needed confidence in them – or at least judge their level of risk. This means that suppliers must not only ensure their delivery of adequate systems, but acquirers and users require the explicit, valid, well-reasoned, and evidence-supported grounds[1] for their confidence and decision making including related engineering conclusions and their uncertainty.

Historically assurance cases covering safety and security requirements for systems have been seen as an important tool for the interchange of assurance information.

To make software assurance more practical, automation and meaningful exchange of this assurance-related information is needed. Software suppliers, tool vendors, acquirers, users, and others would benefit from a flexible and extensible means for its representation and exchange.

The concept of an assurance case is one that provides a framework for analyzing and communicating the assurance arguments and evidence that relate to a system under consideration. Suppliers and customers can see how the system lifecycle products (system requirements, design, testing, field experience etc) relate to and satisfy the assurance requirements, enabling sufficient confidence to be gained in the behavior and integration of the system within its operational context.

Simply put, the assurance case comprises the arguments and evidence that a system will meet its assurance requirements over its lifecycle.

7.2 Structured arguments

Arguments have always been used – albeit informally – to communicate and persuade stakeholders that sufficient confidence can be had in a particular system. However these arguments are often spread over a range of system and management documentation, and it is difficult to see the argument as a whole in a clear way.

In the assurance domain an '**argument**' is defined as "a connected series of statements or reasons intended to establish a position…; a process of reasoning"[2]. In attempting to persuade others of a position, we cite reasons why a claim should be accepted as **true**. These reasons are described as the **premises** of the argument, and the claim they support as its **conclusion**. These terms can be used to define the 'normal form' of an argument as:

Premise
Premise
Premise
So, Conclusion

This form reduces argument to its most primitive building blocks, for example:

[1] Suppliers also need the same or similar case to justify release and deployment.
[2] *Shorter Oxford English Dictionary*, 6th Edition (2007)

Premise: All complex systems are susceptible to failure.
Premise: Failures can lead to accidents.
Therefore,
Conclusion: Accidents can occur in complex safety-critical systems.

The terms ‘premise’ and ‘conclusion’ are relative. The premise of one reasoning step (e.g. that “All complex systems are susceptible to failure”) may itself need further reasoning support and will become the conclusion of a subsequent supporting argument. This gives rise to hierarchical argument structures (‘chains of reasoning’) in which arguments are established by the composition of a number of (premise-conclusion) reasoning steps in order to support an overall conclusion, as illustrated in Figure 1:

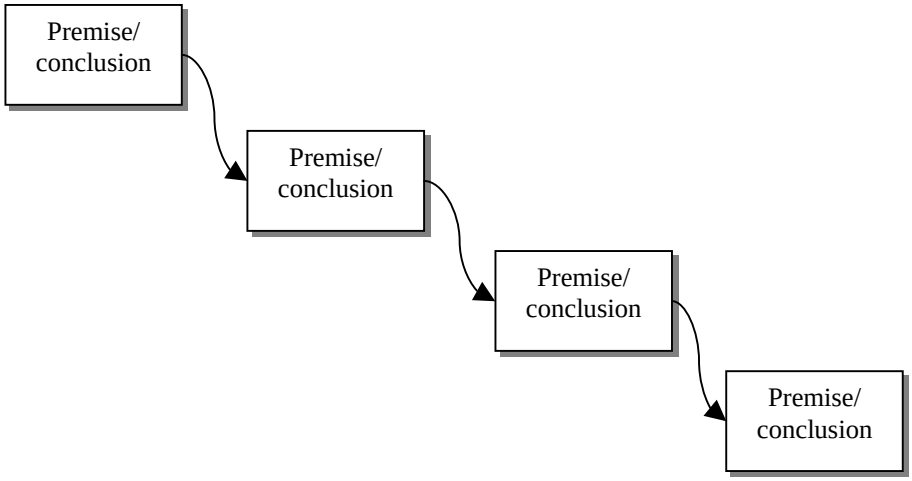

Figure 1: Argument Chain Structure

Structured arguments are therefore one way to allow the communication of how a series of claims can establish a conclusion.

7.3 Arguments as asserted positions

It is important to note that the representation of an argument is not the same as a valid argument. The process of argument representation and communication is separate from that of argument evaluation. For example, an argument may include invalid reasoning, or may have a reliance on irrelevant or false information.

Therefore representations of arguments should be seen as positions that are effectively asserted by the authors or organizations that are putting forward the argument.

Clearly professional ethics require that assurance stakeholders should present arguments that they believe to be correct, valid and relevant.

A key concept is that structured arguments allow users to express and declare what they consider the argument to be.

7.4 Structured arguments in ARM

ARM contains those elements presented as fundamental to the expression and exchange of structured arguments.

As noted above, a typical natural language dictionary definition of an argument is that an argument comprises a series of linked premises (propositions), leading to a conclusion. From this we can derive a set of practical modeling approach that allows users to link together propositions (claims) and to communicate how they consider that higher level claims to be supported or derived from the lower level claims. Since a claim can be used to support one or more other claims, the general form of a directed graph emerges.

ARM aims to provide a modeling framework to allow users to express and exchange their argument structures. The representation of an argument in ARM does not imply that the argument is complete, valid or correct. Similarly, the evaluation or acceptance of an argument by a separate party is not covered by the ARM.

In the ARM model, structured arguments comprise argument elements (primarily claims) that are being asserted by the author of the argument, together with relationships that are asserted to hold between those nodes.

8 ARM Specification

This section presents the normative specification for the ARM. It begins with an overview of the metamodel structure followed by a description of each element.

8.1 Overview

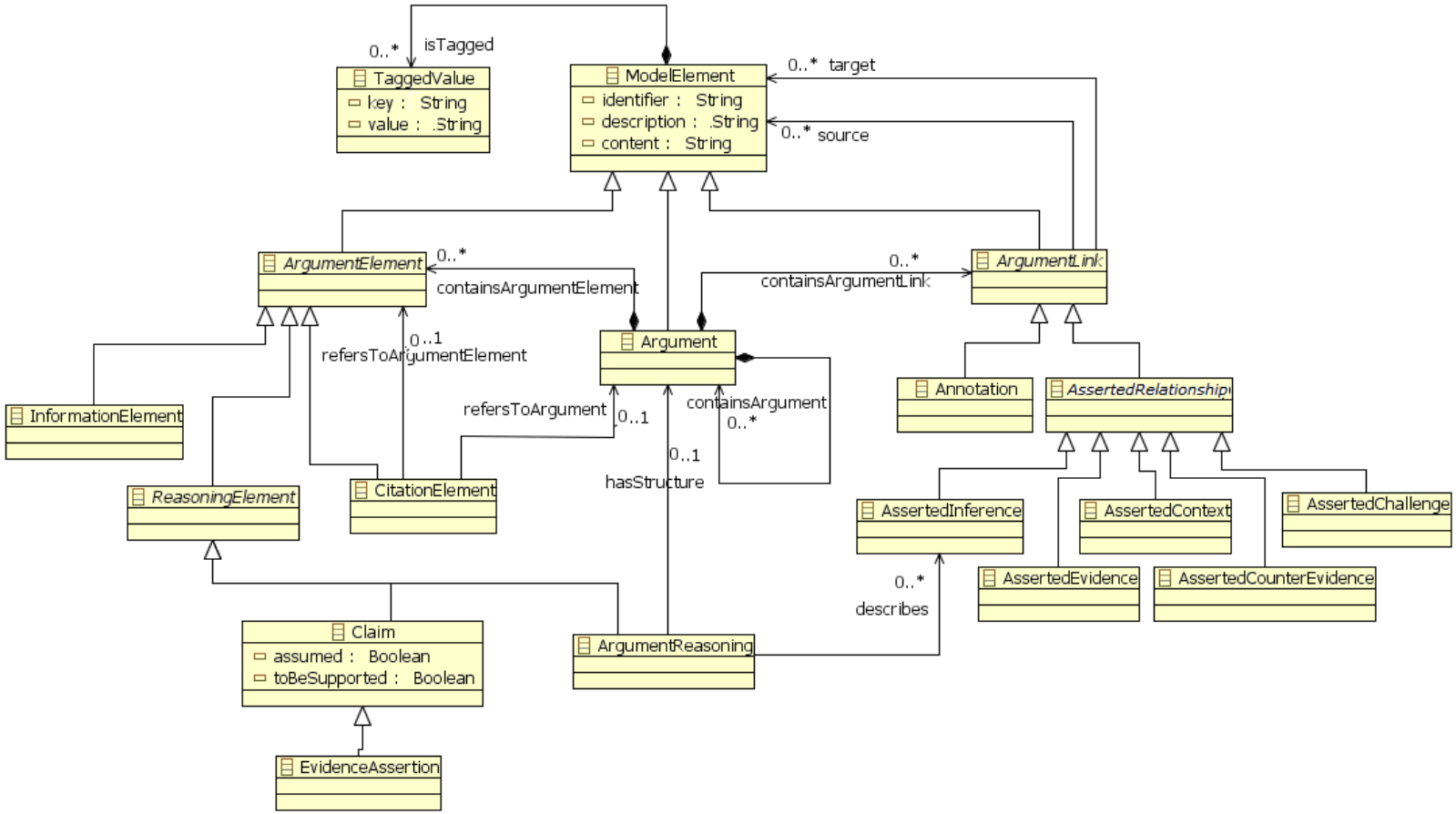

Figure 2 – Metamodel overview

8.2 Class definitions

In the following sections we describe the model elements.

8.2.1 ModelElement Class (Abstract)

A ModelElement is an atomic constituent of a structured argument represented using ARM. In the meta-model, ModelElement is the top meta-element in the ARM class hierarchy. ModelElement is an abstract meta-model element.

Attributes

identifier: String	A unique identifier for the ARM entity.
description: String	A description of the ARM entity.
content: String	Supporting content for the ARM entity.

Associations

isTagged:TaggedValue[0..*]	This association enables the association of one or more user defined TaggedValues to any ARM ModelElement.

Semantics

The ModelElement is a common class for all meta-model elements that represent some element of a structured argument.

Invariants

context ModelElement

inv UniqueIdentifier: ModelElement.allInstances()->select(me:ModelElement|me.identifier=self.identifier)->size()= 1

8.2.2 TaggedValue Class

A TaggedValue is an annotation that can be provided on any ModelElement in ARM.

Attributes

key: String	A key for the TaggedValue.
value: String	The value of the TaggedValue.

Semantics

It can be useful to be able to tag values onto the ARM ModelElements. For example, TaggedValues can record versioning information, ownership information, and external URI references. This is a deliberately general mechanism to allow users to associate tags that they find useful for any ARM instance.

8.2.3 Argument Class

The Argument Class is the container class for a structured argument represented using ARM.

Superclass

ModelElement

Associations

containsArgumentElement:ArgumentElement[0..*] The ArgumentElements contained in a given instance of an Argument.

containsArgumentLink:ArgumentLink[0..*] The ArgumentLinks (between ArgumentElements) contained in a given instance of an Argument.

Semantics

Structured arguments represented using ARM are composed of ArgumentElements and ArgumentLinks between ArgumentElements.

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

Example

<ARM:Argument xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ARM="ARM" xmi:id="0"> </ARM:Argument>

8.2.4 ArgumentElement Class (Abstract)

The ArgumentElement Class is the abstract class for the elements of any structured argument represented using ARM.

Superclass

ModelElement

Semantics

ArgumentElements represent the constituent building blocks of any structured Argument.

For example, ArgumentElements can represent the Claims made within a structured Argument.

8.2.5 ArgumentLink Class (Abstract)

The ArgumentLink Class is the abstract association class that enables the ArgumentElements of any structured argument to be linked together.

Superclass

ModelElement

Associations

source:ModelElement[0..*] Reference to the ModelElement(s) that are the source (start-point) of the relationship.
target:ModelElement[0..*] Reference to the ModelElement(s) that are the target (end-point) of the relationship.

Semantics

In ARM, the structure of an argument is declared through the linking together of primitive ArgumentElements. For example, links between ArgumentElements allow the declaration of the structure of the inferences of the argument, and the association between claims and evidence. In addition argument links can be used to associate ModelElements to ArgumentLinks (e.g. it could be necessary to add a supporting claim – backing – to an AssertedInference between two claims).

8.2.6 ReasoningElement Class (Abstract)

The ReasoningElement Class is the abstract class for the elements that comprise the core reasoning of any structured argument represented using ARM – namely, Claims and ArgumentReasoning (the description of inferential reasoning that exists between Claims.

Superclass

ArgumentElement

Attributes

toBeSupported: Boolean	An attribute recording whether further reasoning has yet to be provided to support the ReasoningElement (e.g. further evidence to be cited).

Semantics

The core of any argument is the reasoning that exists to connect individual claims of that argument. Reasoning is captured in ARM through the linking of fundamental claims, and the description of the links between claims. ReasoningElements represent these two elements.

When building an argument it can often be useful to create a ReasoningElement yet not have fully defined the element. When building an argument it is may be useful to denote that further (supporting) reasoning or evidence is necessary to support a ReasoningElement. This is done using the toBeSupported attribute.

8.2.7 InformationElement Class

The InformationElement Class enables the citation of a source of that *relates* to the structured argument. The citation is made by the InformationElement class. The declaration of relationship is made by the AssertedRelationship class.

Superclass

ArgumentElement

Semantics

It is necessary to be able to cite sources of information that support, provide context for, or provide additional description for the core reasoning of the recorded argument. InformationElements allow there to be an objectified citation of this information within the ARM structured argument, thereby allowing the relationship between this information and the argument to also be explicitly declared.

Example

<containsArgumentElement xsi:type="ARM:InformationElement" xmi:id="14" identifier="S2.1" description="" content="black box testing"/>

8.2.8 CitationElement Class

The CitationElement Class cites an Argument, or an ArgumentElement within another Argument, for use within an Argument.

Superclass

ArgumentElement

Associations

refersToArgumentElement:ArgumentElement[0..1]	References an ArgumentElement within another Argument.
refersToArgument:Argument[0..1]	References an Argument.

Semantics

Within an Argument (package) it can be useful to be able to cite elements of an Argument (i.e. ArgumentElements) to act as explicit proxies for those elements acting within the argument structure. For example, in supporting a Claim it may be useful to cite a Claim or InformationElement declared within another Argument. It can also be useful to be able to cite entire Arguments. For example, in supporting a Claim it may be useful to cite an existing (structured) Argument.

8.2.9 Claim Class

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

Superclass

ReasoningElement

Attributes

assumed: Boolean	An attribute recording whether the claim being made is declared as being assumed to be true rather than being supported by further reasoning.

Semantics

Structured arguments are declared by stating claims, and asserting how those claims relate to each other. 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 reasoning (in the recorded Argument) can be declared as being *assumed* to be true. It is an *assumption*. However, it should be noted that a Claim that is not 'assumed' (i.e. assumed = false) is not being declared as false.

Example

<containsArgumentElement xsi:type="ARM:Claim" xmi:id="5" identifier="C1.1" description="" content="Unintended opening of press (after PoNR) can only occur as a result of component failure"/>

8.2.10 EvidenceAssertion Class

A sub-type of Claim used to record propositions (assertions) made regarding an InformationElement being used as supporting evidence to the Argument. This is intended to be used as an interface element to external evidence. An evidence assertion is a minimal assertion (proposition) about an item of evidence, and there is no supporting argumentation being offered within the current structured argument.

Superclass

Claim

Semantics

Well supported arguments are those where evidence can be cited that is said to support the most fundamental claims of the argument. It is good practice that these fundamental claims of the argument state clearly the property that is said to exist in, be derived from, or be exhibited by the cited evidence. Where such claims are made these are said to be basic EvidenceAssertions.

Example

<containsArgumentElement xsi:type="ARM:EvidenceAssertion" xmi:id="12" identifier="C2.1.1" content="Failure 1 of PLC state machine includes BUTTON_IN remaining true"/>

8.2.11 ArgumentReasoning Class

ArgumentReasoning can be used to provide additional description or explanation of the asserted inference that connect one or more Claims (premises) to another Claim (conclusion). ArgumentReasoning elements are therefore related to AssertedInferences. 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.

Superclass

ReasoningElement

Associations

describes:AssertedInference[0..*]	Reference to the AssertedInference being described by the ArgumentReasoning.
hasStructure:Argument[0..1]	Optional reference to another structured Argument to provide the detailed structure of the Argument being described by the ArgumentReasoning.

Semantics

The argument step that relates one or more Claims (premises) to another Claim (conclusion) may not always be obvious. In such cases ArgumentReasoning can be used to provide further description of the reasoning steps involved.

Example

<containsArgumentElement xsi:type="ARM:ArgumentReasoning" xmi:id="2" identifier="RC1.1" content="Argument by omission of all identified software hazards" describes="5 6"/>

8.2.12 AssertedRelationship Class (Abstract)

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

Superclass

ArgumentLink

Semantics

In ARM, the structure of an argument is declared through the linking together of primitive ArgumentElements (and potentially ArgumentLinks). 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.

Example

8.2.13 AssertedInference Class

The AssertedInference association class records the inference that a user declares to exist between one or more Claims (premises) and another Claim (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 primitive Claims. For example, a AssertedInference can be said to exist between two claims ("Claim A implies Claim B"). A 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.

Example

<containsAssertedRelationship xsi:type="ARM:AssertedInference" xmi:id="16" identifier="C1.1.1" description="" target="5" source="1"/>

Invariants
context AssertedInference

inv SourceMustBeClaim : self.source->forAll(s|s.oclIsTypeOf(Claim))

inv TargetMustBeClaimOrAssertedRelationship : self.target->forAll(t|t.oclIsTypeOf(Claim) or t.oclIsTypeOf(AssertedRelationship))

8.2.14 AssertedEvidence Class

The AssertedEvidence association class records the declaration that one or more items of Evidence (cited by InformationItems) provides 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 information (cited by an InformationItem) may provide evidence for more than one Claim.

Superclass

AssertedRelationship

Semantics

Where evidence (cited by InformationItems) exists that helps to establish the truth of a Claim in the argument, this relationship between this Claim and the evidence can be asserted by a AssertedEvidence association. An AssertedEvidence association between some information cited by an InformationElement 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.

Example

<containsAssertedRelationship xsi:type="ARM:AssertedEvidence" xmi:id="22" identifier="S1.1" target="10" source="5 6"/>

Invariants

context AssertedEvidence

inv SourceMustBeInformationElement : self.source->forAll(s|s.oclIsTypeOf(InformationElement))

inv TargetMustBeClaimOrAssertedRelationship : self.target->forAll(t|t.oclIsTypeOf(Claim) or t.oclIsTypeOf(AssertedRelationship))

8.2.15 AssertedChallenge Class

The AssertedChallenge association class records the *challenge* (i.e. counter-argument) that a user declares to exist between one or more Claims and another Claim. It is important to note that such a declaration is itself an assertion on behalf of the user.

Superclass

AssertedRelationship

Semantics

An AssertedChallenge by Claim A (source) to Claim B (target) denotes that the truth of Claim A challenges the truth of Claim B (i.e. Claim A leads towards the conclusion that Claim B is false).

Invariants

context AssertedChallenge

inv SourceMustBeClaim : self.source->forAll(s|s.oclIsTypeOf(Claim))

inv TargetMustBeClaimOrAssertedRelationship : self.target->forAll(t|t.oclIsTypeOf(Claim) or t.oclIsTypeOf(AssertedRelationship))

8.2.16 AssertedCounterEvidence Class

AssertedCounterEvidence can be used to associate evidence (cited by InformationElements) to a Claim, where this evidence is being asserted to infer that the Claim is *false*. It is important to note that such a declaration is itself an assertion on behalf of the user.

Superclass

AssertedRelationship

Semantics

An AssertedCounterEvidence association between some evidence cited by an InformationNode and a Claim (A – the source evidence cited – and B – the target claim) denotes that the evidence cited by A is counter-evidence to the truth of Claim B (i.e. Evidence A suggests the conclusion that Claim B is false).

Invariants

context AssertedCounterEvidence

inv SourceMustBeInformationElement : self.source->forAll(s|s.oclIsTypeOf(InformationElement))

inv TargetMustBeClaimOrAssertedRelationship : self.target->forAll(t|t.oclIsTypeOf(Claim) or t.oclIsTypeOf(AssertedRelationship))

8.2.17 AssertedContext Class

The AssertedContext association class declares that the information cited by an InformationElement provides a context for the interpretation and definition of a Claim or ArgumentReasoning element.

Superclass

AssertedRelationship

Semantics

Claim and ArgumentReasoning often need contextual information to be cited in order for the scope and definition of the reasoning to be easily interpreted. For example, a Claim can be said to valid only in a defined context ("Claim A is asserted to be true only in a context as defined by the information cited by InformationItem B" or conversely "InformationItem B is the valid context for Claim A"). A declaration (AssertedContext) of context (InformationItem) for a ReasoningElement (A – the contextual InformationItem – and B – the ReasoningElement) denotes that A is asserted to be valid contextual information for B (i.e. A defines context where the reasoning presented by B holds true).

Example

<containsAssertedRelationship xsi:type="ARM:AssertedContext" xmi:id="21" identifier="CIRC1.1" target="4" source="2"/>

Invariants

context AssertedContext

inv SourceMustBeInformationElement :self.source->forAll(s|s.oclIsTypeOf(InformationElement))

inv TargetMustBeReasoningElement : self.target->forAll(t|t.oclIsTypeOf(ReasoningElement))

8.2.18 Annotates Class

The Annotates association class declares that the information cited by an InformationElement provides annotation of a ModelElement.

Superclass

ArgumentLink

Semantics

The Annotates association class allows the (informal) association of InformationElements to *any* ModelElement of the structured argument (i.e. both elements and links). For example, it can be useful to attach citations of information such as review comments, descriptive documents, and the text of assurance standards. Importantly. use of the Annotates association class does not assert any *logical* relationship between the InformationElement and the ModelElement (unlike AssertedRelationship and its subclasses), it is merely adding additional ancillary to the structured argument.

8.3 Examples

The section provides two examples of argument from the safety and the security domain. The safety argument refers to an industrial press, whereas the security example is a fragment from a Bluetooth security case.

8.3.1 Industrial Press Safety Argument


```
<?xml version="1.0" encoding="ASCII"?>
<ARM:Argument xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ARM="ARM" xmi:id="0">
  <containsArgumentElement xsi:type="ARM:Claim" xmi:id="1" identifier="C1" description="" content="C/S logic is fault
free"/>
  <containsArgumentElement xsi:type="ARM:ArgumentReasoning" xmi:id="2" identifier="RC1.1" content="Argument by
omission of all identified software hazards" describes="5 6"/>
  <containsArgumentElement xsi:type="ARM:ArgumentReasoning" xmi:id="3" identifier="RC1.2" content="Argument by
satisfaction of all C/S safety requirements" describes="7 8 9"/>
  <containsArgumentElement xsi:type="ARM:InformationElement" xmi:id="4" identifier="IRC1.1" description="Identified
software hazards"/>
  <containsArgumentElement xsi:type="ARM:Claim" xmi:id="5" identifier="C1.1" description="" content="Unintended
opening of press (after PoNR) can only occur as a result of component failure"/>
  <containsArgumentElement xsi:type="ARM:Claim" xmi:id="6" identifier="C1.2" description="" content="Unintended
closing of press can only occur as a result of component failure"/>
  <containsArgumentElement xsi:type="ARM:Claim" xmi:id="7" identifier="C2.1" content="Press controls being 'jammed
on' will cause press to halt"/>
  <containsArgumentElement xsi:type="ARM:Claim" xmi:id="8" identifier="C2.2" content="Release of controls prior to
press passing physical PoNR will cause press operation to abort"/>
  <containsArgumentElement xsi:type="ARM:Claim" xmi:id="9" identifier="C2.3" description="" content="C/S fails safe
(halts on) and annunciates (by sounding Klaxon) all component failures"/>
  <containsArgumentElement xsi:type="ARM:InformationElement" xmi:id="10" identifier="S1.1" content="Fault tree
analysis cutsets for event 'Hand trapped in press due to command error'"/>
  <containsArgumentElement xsi:type="ARM:InformationElement" xmi:id="11" identifier="S1.2" content="Hazard directed
test results"/>
  <containsArgumentElement xsi:type="ARM:EvidenceAssertion" xmi:id="12" identifier="C2.1.1" content="Failure 1 of PLC
state machine includes BUTTON_IN remaining true"/>
  <containsArgumentElement xsi:type="ARM:EvidenceAssertion" xmi:id="13" identifier="C2.2.1" content="Abort transition
of PLC state machine includes BUTTON_IN going false"/>
  <containsArgumentElement xsi:type="ARM:InformationElement" xmi:id="14" identifier="S2.1" description=""
content="black box testing"/>
  <containsArgumentElement xsi:type="ARM:InformationElement" xmi:id="15" identifier="S2.2.1" content="C/S state
machine"/>
  <containsAssertedRelationship xsi:type="ARM:AssertedInference" xmi:id="16" identifier="C1.1.1" description=""
target="5" source="1"/>
  <containsAssertedRelationship xsi:type="ARM:AssertedInference" xmi:id="17" identifier="C1.1.2" target="6"
source="1"/>
  <containsAssertedRelationship xsi:type="ARM:AssertedInference" xmi:id="18" identifier="C1.2.1" target="7"
source="1"/>
  <containsAssertedRelationship xsi:type="ARM:AssertedInference" xmi:id="19" identifier="C1.2.2" target="8"
source="1"/>
  <containsAssertedRelationship xsi:type="ARM:AssertedInference" xmi:id="20" identifier="C1.2.3" target="9"
source="1"/>
  <containsAssertedRelationship xsi:type="ARM:AssertedContext" xmi:id="21" identifier="CIRC1.1" target="4"
source="2"/>
<containsAssertedRelationship xsi:type="ARM:AssertedEvidence" xmi:id="22" identifier="S1.1" target="10" source="5 6"/>
  <containsAssertedRelationship xsi:type="ARM:AssertedEvidence" xmi:id="23" identifier="S1.2" target="11" source="5
6"/>
  <containsAssertedRelationship xsi:type="ARM:AssertedEvidence" xmi:id="24" identifier="SC2.1" target="14"
source="7"/>
  <containsAssertedRelationship xsi:type="ARM:AssertedEvidence" xmi:id="25" identifier="SC2.1.1" target="15"
source="12"/>
  <containsAssertedRelationship xsi:type="ARM:AssertedEvidence" xmi:id="26" identifier="SC2.2.1" target="15"
source="13"/>
  <containsAssertedRelationship xsi:type="ARM:AssertedInference" xmi:id="27" identifier="DI C2.1" target="12"
source="7"/>
```

```
  <containsAssertedRelationship xsi:type="ARM:AssertedInference" xmi:id="28" identifier="DI C2.2" target="13"
source="8"/>
</ARM:Argument>
```

8.3.2 Bluetooth Security Case

```
<?xml version="1.0" encoding="ASCII"?>
<ARM:Argument xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ARM="ARM" identifier="BSC11">
  <containsArgumentElement xsi:type="ARM:Claim" identifier="Bluetooth secure" content="A bluetooth enabled network
provides adequate security"/>
  <containsArgumentElement xsi:type="ARM:Claim" identifier="Availability" content="A bluetooth enabled network is
adequately available [1] Section 1 para 3"/>
  <containsArgumentElement xsi:type="ARM:Claim" identifier="Access" description="" content="A bluetooth enabled
network provides adequate control for access to services and data [1] Section 1 para 3"/>
  <containsArgumentElement xsi:type="ARM:Claim" identifier="Confidentiality" content="A bluetooth enabled network
provides adequate levels of confidentiality [1] Setion 1 para 3"/>
  <containsArgumentElement xsi:type="ARM:Claim" identifier="Integrity" content="A bluetooth enabled network provides
adequate levels of integrity [1] Section 1 para 3"/>
  <containsArgumentElement xsi:type="ARM:InformationElement" identifier="Context: security policy and scenario for use"
content="Definitions are required of the intented security policy and the scenario of use for the system, including what is
regarded as 'adequate'"/>
  <containsArgumentElement xsi:type="ARM:InformationElement" identifier="References" content="[1] Bluetooth security
white paper 19/4/02"/>
  <containsArgumentElement xsi:type="ARM:InformationElement" identifier="Definition: Availability" content="The system
is capable of providing requested services to authorised users, in an acceptable/defined time"/>
  <containsArgumentElement xsi:type="ARM:InformationElement" identifier="Definition: Access" content="Only users
permitted by the defined security policy have access to services and data"/>
  <containsArgumentElement xsi:type="ARM:InformationElement" identifier="Define: Confidentiality"
content="Unauthorised persons cannot intercept and understand information to which they are not entitled"/>
  <containsArgumentElement xsi:type="ARM:InformationElement" identifier="Define: Integrity" description=""
content="Services and data are provided to authorised users as intended and without corruption"/>
  <containsArgumentElement xsi:type="ARM:ArgumentReasoning" identifier="Argue over vulnerabilities" description=""
content="Argue for each security requirement identified in the security white paper" describes="AI1"/>
  <containsAssertedRelationship xsi:type="ARM:AssertedContext" identifier="AC1" target="References"
source="Bluetooth secure"/>
  <containsAssertedRelationship xsi:type="ARM:AssertedContext" identifier="AC2" target="Context: security policy and
scenario for use" source="Bluetooth secure"/>
  <containsAssertedRelationship xsi:type="ARM:AssertedInference" identifier="AI1" target="Integrity Confidentiality Access
Availability" source="Bluetooth secure"/>
</ARM:Argument>
```

8.3.3 Goal Structuring Notation (GSN) Examples

This section contains examples of arguments using the Goal Structuring Notation. The following table explains the relationship from the example to the modeling elements of ARM:

GSN element	ARM counterpart
Rectangle	Claim
Rounded rectangle	InformationElement
Parallelogram	ArgumentReasoning
Circle	InformationElement linked using an AssertedEvidence instance
Filled arrow	AssertedInference (or AssertedEvidence when linked to circle). The arrow head attaches to the source element.
Empty arrow	AssertedContext. The arrow head attaches to the source element.
Diamond decorator	ToBeSupported = true

Shaded triangle decorator	The current element is a citation element.

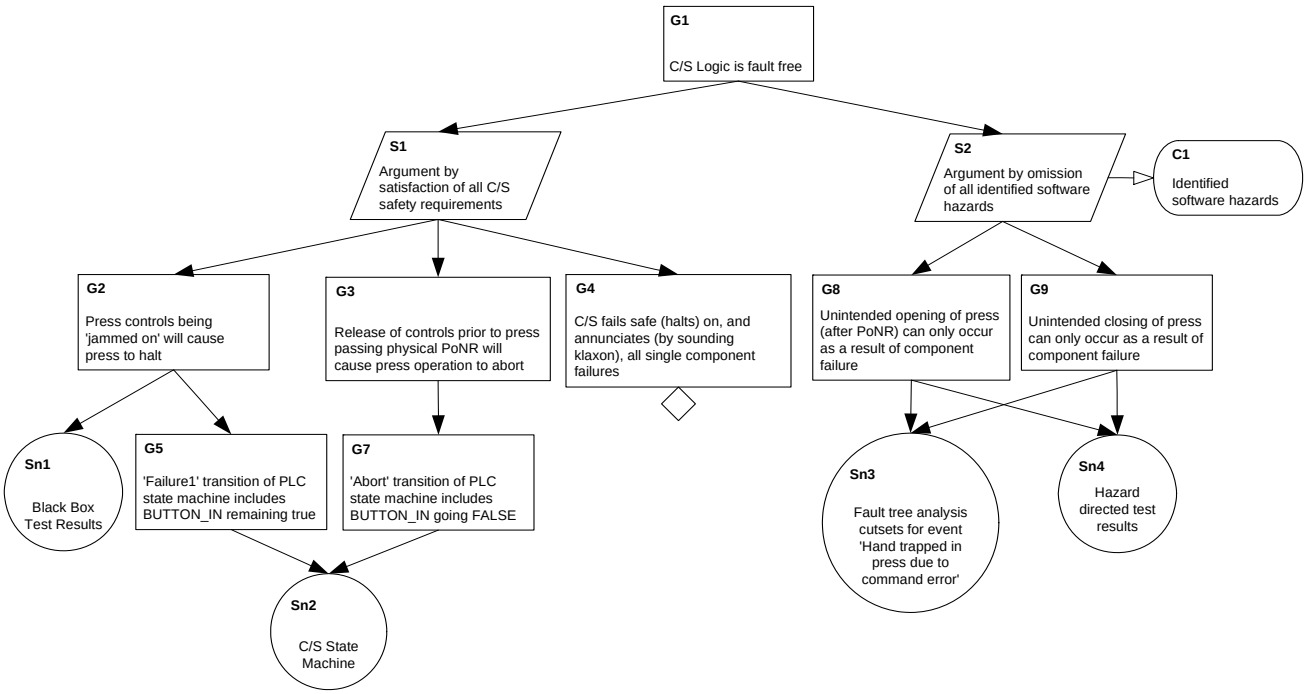

Figure 3 – Industrial Press Safety argument (§8.3.1)

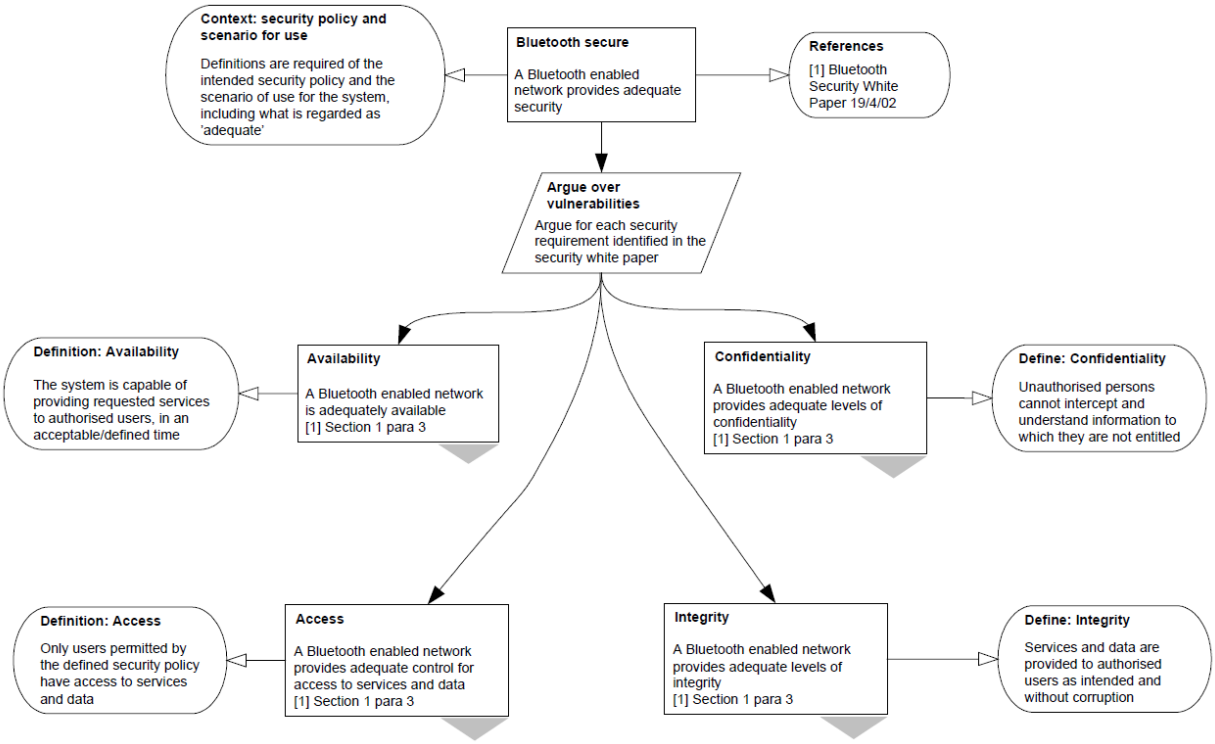

Figure 4 – GSN Bluetooth Security Case (§8.3.2)

8.3.4 Claims-Arguments-Evidence (CAE) Example

In CAE, contextual information can be represented either as visual nodes in a similar manner to GSN (see Figure 5), or alternatively as rich text associated with the node (see Figure 6)

The following table explains the relationship from the example to the modeling elements of ARM:

CAE element	ARM counterpart
Blue elipse	Claim
Green rounded box	ArgumentReasoning
Element with no border	InformationElement
Blue arrow	AssertedInference
Green arrow	AssertedInference (unless from InformationElement, in which case AssertedContext)
Rich narrative text	InformationElement attached using AssertedContext to the current element.

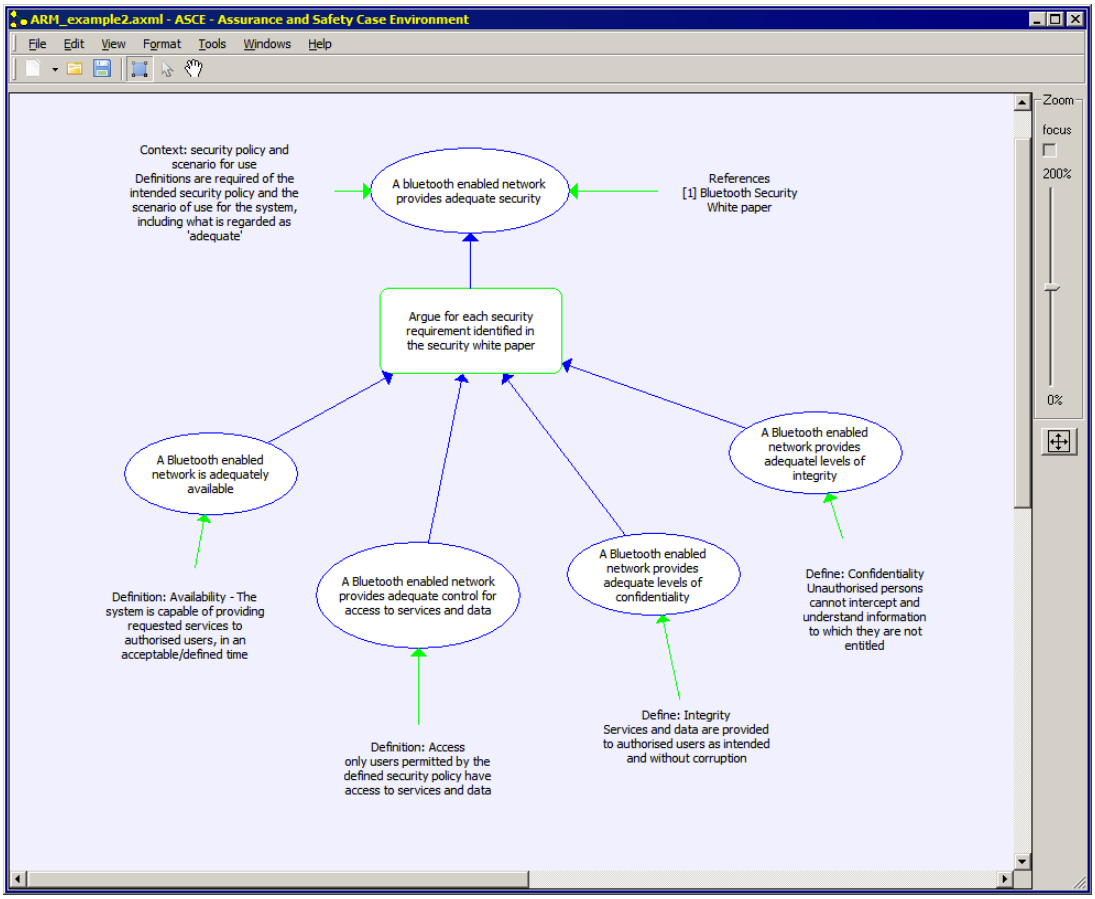

Figure 5: CAE of Bluetooth example – showing contextual information as visual nodes

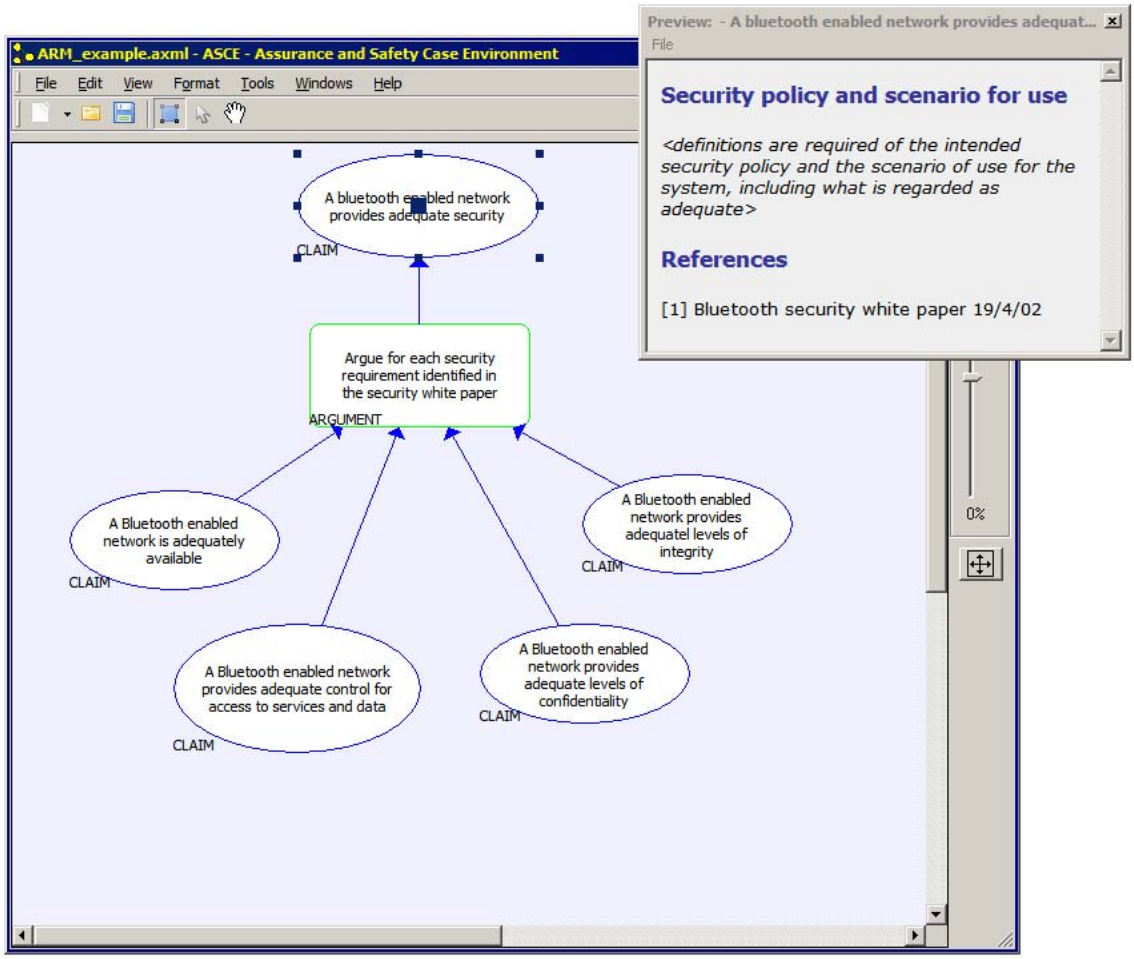

Figure 6: CAE representation of the Bluetooth example where contextual information held as rich text (top claim is selected)