|  |
| --- |
|  |
| |  |  |  | | --- | --- | --- | | 75 | Date: | November 2014 | |
| Archetype Modeling Language (AML)  Version: 0.5  **OMG Document Number: health/2014-10-01**  **Standard document URL: http://www.omg.org/spec/AML/1.0**  Original File: N/A |

Copyright © 2014, Mayo Clinic  
Copyright © 2014, Object Management Group, Inc.  
Copyright © 2014, Visumpoint, LLC

USE OF SPECIFICATION - TERMS, CONDITIONS & NOTICES

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

Any unauthorized use of this specification may violate copyright laws, trademark laws, and communications regulations and statutes. This document contains information which is protected by copyright. All Rights Reserved. No part of this work covered by copyright herein may be reproduced or used in any form or by any means--graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems--without permission of the copyright owner.

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™ , IMM™ , 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

1 Scope 10

2 Conformance 10

3 Normative References 10

4 Terms and Definitions 11

5 Symbols 15

5.1 Graphical Symbols 15

5.2 Abbreviations 15

6 Additional Information 16

6.1 Changes to Adopted OMG Specifications 16

6.2 Acknowledgements 16

7 AML Object Model 16

7.1 <Package> Archetype Object Model 17

7.1.1 <DataType> ArchetypeId 17

7.1.2 <DataType> ArchetypeVersionId 18

7.1.3 <Class> Archetype 18

7.1.4 <Class> ArchetypeLibrary 19

7.1.5 <Class> ArchetypeVersion 20

7.1.6 <Class> AuthoredResource 21

7.1.7 <Class> Class 22

7.1.8 <Class> ComplexObjectConstraint 24

7.1.9 <Class> FlatArchetype 24

7.1.10 <Class> Package 25

7.1.11 <Class> RuleStatement 26

7.1.12 <Class> SourceArchetype 26

7.1.13 <Enumeration> ArchetypeType 26

7.1.14 <Class> Classifier 27

7.1.15 <Class> NamedElement 28

7.1.16 <Class> Namespace 28

7.2 <Package> Reference Object Model 29

7.2.1 <Package> Primitive Data Types 29

7.2.2 <DataType> AnyPrimitiveType 30

7.2.3 <DataType> BooleanPrimitive 30

7.2.4 <DataType> DateMatchPattern 30

7.2.5 <DataType> DatePrimitive 30

7.2.6 <DataType> DateTimeMatchPattern 31

7.2.7 <DataType> DateTimePrimitive 31

7.2.8 <DataType> DurationMatchPattern 31

7.2.9 <DataType> DurationPrimitive 31

7.2.10 <DataType> IntegerPrimitive 31

7.2.11 <DataType> RealPrimitive 32

7.2.12 <DataType> StringPrimitive 32

7.2.13 <DataType> TerminologyCode 32

7.2.14 <DataType> TimeMatchPattern 32

7.2.15 <DataType> TimePrimitive 33

7.2.16 <Class> Interval 33

7.2.17 <Package> Reference Metamodel 34

7.2.18 <Class> DataType 34

7.2.19 <Class> Enumeration 34

7.2.20 <Class> Property 35

7.2.21 <Class> RMMCollectionProperty 36

7.2.22 <Class> RMMSingularProperty 37

7.2.23 <Package> Attribute Constraint References 38

7.2.24 <Class> AttributeCollectionConstraint 38

7.2.25 <Class> AttributeConstraint 39

7.2.26 <Class> SingularAttributeConstraint 42

7.2.27 <Package> Object Constraint References 44

7.2.28 <Class> EnumerationConstraint 44

7.2.29 <Class> NamedObjectConstraint 45

7.2.30 <Class> PrimitiveObjectConstraint 46

7.2.31 <Package> Template Metamodel 47

7.2.32 <Class> ClassifierTemplateParameter 47

7.2.33 <Class> RedefinableTemplateSignature 47

7.2.34 <Class> TemplateBinding 48

7.2.35 <Class> TemplateParameterSubstitution 48

7.2.36 <Package> Instance Metamodel 49

7.2.37 <Class> InstanceSpecification 49

7.2.38 <Class> PrimitiveType 49

7.2.39 <Class> RMClass 50

7.2.40 <Class> RMClassInstance 50

7.2.41 <Class> RMPrimitiveDataTypeInstance 51

7.2.42 <Package> Package Metamodel 51

7.2.43 <Package> Enumeration Metamodel 52

7.2.44 <Class> EnumeratedValueDomain 52

7.2.45 <Class> Enumerated\_Value\_Domain 53

7.2.46 <Class> EnumerationLiteral 53

7.2.47 <Class> PermissibleValue 54

7.3 <Package> Constraint Object Model 55

7.3.1 <Package> Primitive Type Constraints 56

7.3.2 <Class> BooleanConstraint 56

7.3.3 <Class> DateConstraint 56

7.3.4 <Class> DateTimeConstraint 57

7.3.5 <Class> DurationConstraint 57

7.3.6 <Class> IntegerConstraint 58

7.3.7 <Class> RealConstraint 58

7.3.8 <Class> StringConstraint 58

7.3.9 <Class> TimeConstraint 59

7.3.10 <Package> Terminology Constraints 60

7.3.11 <DataType> ValueSetDefinitionReference 60

7.3.12 <Class> ObjectConstraint 60

7.3.13 <Class> TBValueSetReference 62

7.3.14 <Class> TerminologyConstraint 62

7.3.15 <Class> URIAndEntityName 63

7.3.16 <Enumeration> ValidityKind 63

7.3.17 <Package> Enumeration Constraints 64

7.3.18 <Class> LocalEnumerationConstraint 65

7.3.19 <Class> ValueSet 65

7.3.20 <Package> Attribute Constraints 66

7.3.21 <Class> AttributeCollectionMember 66

7.3.22 <Class> AttributeTuple 67

7.3.23 <Class> AttributeTupleConstraint 68

7.3.24 <Class> MultiplicityInterval 68

7.3.25 <Enumeration> AttributeExistence 68

7.3.26 <Enumeration> CollectionType 69

7.3.27 <Package> Object Constraints 70

7.3.28 <Class> ArchetypeRootProxy 70

7.3.29 <Class> ArchetypeSlot 71

7.3.30 <Class> ObjectConstraintProxy 71

7.3.31 <Enumeration> Redefinability 72

7.4 <Package> Terminology Object Model 73

7.4.1 <Package> Common Terminology Services Components 74

7.4.2 <DataType> CodeSystemAndVersionReference 74

7.4.3 <DataType> CodeSystemReference 75

7.4.4 <DataType> CodeSystemVersionReference 75

7.4.5 <DataType> ExternalURI 75

7.4.6 <DataType> NameAndMeaningReference 76

7.4.7 <DataType> NamespaceIdentifier 76

7.4.8 <DataType> RenderingURI 77

7.4.9 <DataType> URI 77

7.4.10 <DataType> ValueSetAndDefinitionReference 77

7.4.11 <DataType> ValueSetReference 77

7.4.12 <Class> ConceptReference 78

7.4.13 <Class> ScopedEntityName 78

7.4.14 <Enumeration> ReferenceType 79

7.4.15 <Package> ISO Model Components 79

7.4.16 <DataType> Sign 80

7.4.17 <DataType> Text 80

7.4.18 <Class> Definition 81

7.4.19 <Class> DescribedItem 81

7.4.20 <Class> Designatable\_Item 82

7.4.21 <Class> Designation 83

7.4.22 <Class> Identified\_Item 83

7.4.23 <Class> ItemDescription 84

7.4.24 <Class> Language\_Identification 84

7.4.25 <Class> Namespace 85

7.4.26 <Class> Scoped\_Identifier 86

7.4.27 <Class> Concept 87

7.4.28 <Class> Conceptual\_Domain 88

7.4.29 <Class> Concept\_System 88

7.4.30 <Class> Enumerated\_Conceptual\_Domain 89

7.4.31 <Class> Permissible\_value 89

7.4.32 <Class> Value\_Domain 90

7.4.33 <Class> Value\_Meaning 91

7.4.34 <Class> Data\_Element 93

7.4.35 <Class> Data\_Element\_Concept 93

7.4.36 <Package> AML Described Items 94

7.5 <Package> Rules Object Model 95

7.5.1 <Class> Assertion 95

7.5.2 <Class> BuiltinVariable 96

7.5.3 <Class> ExprArchetypeIdConstraint 96

7.5.4 <Class> ExprbinaryOperator 96

7.5.5 <Class> ExprConstant 97

7.5.6 <Class> ExprConstraint 97

7.5.7 <Class> ExprItem 98

7.5.8 <Class> ExprLeaf 98

7.5.9 <Class> ExprModelRef 98

7.5.10 <Class> ExprOperator 99

7.5.11 <Class> ExprUnaryOperator 99

7.5.12 <Class> ExprVariable 100

7.5.13 <Class> ExprVariableRef 100

7.5.14 <Class> QueryVariable 101

7.5.15 <Class> RuleElement 101

7.5.16 <Class> VariableDeclaration 101

7.5.17 <Enumeration> OperatorKind 102

7.5.18 <Enumeration> ReferenceType 104

7.6 <Package> Metadata Object Model 105

8 AML Profiles 105

8.1 <Package> Reference Model Profile 107

8.1.1 <Stereotype> ConcreteInstanceSpecification 107

8.1.2 <Class> Date 108

8.1.3 <Class> DateTime 108

8.1.4 <Class> Duration 109

8.1.5 <Class> Time 109

8.1.6 <Stereotype> AMLDataType 109

8.1.7 <Stereotype> DataBinding 110

8.1.8 <Stereotype> Infrastructure 110

8.1.9 <Stereotype> MappedDataType 111

8.1.10 <Stereotype> ReferenceModel 111

8.1.11 <Stereotype> Runtime 112

8.2 <Package> Terminology Profile 112

8.2.1 <Stereotype> ArchetypeType 112

8.2.2 <Stereotype> DescribedItem 113

8.2.3 <Stereotype> DesignatableItem 113

8.2.4 <Stereotype> IdentifiedItem 114

8.2.5 <Stereotype> KnownNamespace 114

8.2.6 <Stereotype> Language 115

8.2.7 <Stereotype> NamespaceInstance 115

8.2.8 <Stereotype> ScopedIdentifier 116

8.2.9 <Class> ConceptReferenceClass 117

8.2.10 <Stereotype> CodeSystemReference 117

8.2.11 <Stereotype> CodeSystemReferenceInstance 118

8.2.12 <Stereotype> CodeSystemVersionReference 118

8.2.13 <Stereotype> ConceptReference 118

8.2.14 <Stereotype> ConceptReferenceInstance 119

8.2.15 <Stereotype> ResourceReference 119

8.2.16 <Stereotype> ResourceReferenceInstance 120

8.2.17 <Stereotype> ValueSetDefinitionReference 121

8.2.18 <Stereotype> ValueSetReference 121

8.2.19 <Stereotype> ValueSetReferenceInstance 121

8.2.20 <Stereotype> EnumeratedValueDomain 122

8.2.21 <Stereotype> PermissibleValue 123

8.3 <Package> Constraint Profile 124

8.3.1 <Package> Archetypes 124

8.3.2 <Stereotype> Archetype 124

8.3.3 <Stereotype> ArchetypeLibrary 126

8.3.4 <Stereotype> ArchetypeType 126

8.3.5 <Stereotype> ArchetypeVersion 127

8.3.6 <Stereotype> AuthoredResource 127

8.3.7 <Stereotype> ComplexObjectConstraint 128

8.3.8 <Stereotype> ReferenceModelImport 129

8.3.9 <Stereotype> AMLType 130

8.3.10 <Stereotype> ArchetypeId 130

8.3.11 <Stereotype> ArchetypeVersionId 131

8.3.12 <Stereotype> ResourceDescription 131

8.3.13 <Stereotype> ResourceTranslation 131

8.3.14 <Stereotype> TranslationDetails 132

8.3.15 <Package> Data Type Constraints 132

8.3.16 <Stereotype> BooleanConstraint 132

8.3.17 <Stereotype> ConceptReferenceConstraint 133

8.3.18 <Stereotype> DateConstraint 133

8.3.19 <Stereotype> DateTimeConstraint 134

8.3.20 <Stereotype> DurationConstraint 135

8.3.21 <Stereotype> IntegerConstraint 135

8.3.22 <Stereotype> PrimitiveObjectConstraint 136

8.3.23 <Stereotype> RealConstraint 136

8.3.24 <Stereotype> StringConstraint 136

8.3.25 <Stereotype> TimeConstraint 137

8.3.26 <Class> DateInterval 138

8.3.27 <Class> DateTimeInterval 138

8.3.28 <Class> DurationInterval 139

8.3.29 <Class> IntegerInterval 139

8.3.30 <Class> Interval 139

8.3.31 <Class> RealInterval 139

8.3.32 <Class> TimeInterval 140

8.3.33 <Package> Object and Property Constraints 140

8.3.34 <Stereotype> ArchetypeRootProxy 140

8.3.35 <Stereotype> Constrains 141

8.3.36 <Stereotype> EnumerationConstraint 141

8.3.37 <Stereotype> ObjectConstraint 142

8.3.38 <Stereotype> ObjectConstraintProxy 142

8.3.39 <Stereotype> AttributeCollectionConstraint 144

8.3.40 <Stereotype> AttributeConstraint 144

8.3.41 <Stereotype> SingularAttributeConstraint 144

8.3.42 <Class> Enumeration 145

8.3.43 <Class> EnumerationLiteral 145

8.3.44 <Stereotype> EnumeratedValueDomainConstraint 146

8.3.45 <Stereotype> ArchetypeRootConstraint 147

8.3.46 <Stereotype> TargetConstraint 147

8.3.47 <Package> Terminology Constraints 148

8.3.48 <Stereotype> ConstrainsConceptReference 148

**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

OMG Headquarters   
 109 Highland Ave,   
 Needham, MA 02494 USA  
 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.

# Scope

The objective of this RFP is to provide a standard for modeling Archetype Models (AMs) using UML, to support the representation of Clinical Information Modeling Initiative (CIMI) artifacts in UML. Archetypes are Platform Independent Models (PIMs), which are developed as a set of constraints on a specific Reference Model (RM).

The goal of the Archetype Modeling Language (AML) is to provide a common approach to modeling Archetype Models (AMs) using UML, thereby supporting the representation of Clinical Information Modeling Initiative (CIMI) artifacts in UML. Archetypes are Platform Independent Models (PIMs), which are specified by applying a set of constraints on a specific Reference Model (RM).

AML is specified as a UML Profile comprised of three sub-profiles, which collectively meet the requirements of archetype modeling:

* *Reference Model Profile (RMP)*: Enables the specification of reference models upon which archetypes can be based.
* *Constraint Model Profile (CMP)*: Supports the specification of constraints on a given reference model, to enable the development of archetypes, including Clinical Information Models (CIMs).
* *A Terminology Binding Profile (TBP)*: Supports the binding of information models to terminology[, with optional support for binding to CTS2] [TBR]. Terminology bindings include:
  1. *Value Bindings*: Support linking the data model to value domains, which restrict the valid value of an attribute to a set of values that corresponds to a set of meanings recorded in an external terminology;
  2. *Semantic Bindings:* Define the meaning of model elements, using concepts in an external terminology; and
  3. *Constraint Bindings:* Specify constraints on the information model, using concepts and relationships defined in an external terminology.

This set of UML sub-profiles enables the specification of CIMI clinical model content (using the CIMI Reference Model), and the generation of CIMI clinical model artifacts, such as the Archetype Definition Language (ADL). While the transformation of AML models to an instance of the Archetype Object Model v1.5 (AOM-1.5) is optional, this transformation must be possible. The Archetype Definition Language (ADL) is a serialization of the Archetype Object Model.

# Conformance

[TBD – After completion of Sections 7 and 8.]

# 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.

NOTE 1: The openEHR references have invalid hyperlinks – openEHR was moved from an SVN repository to GitHub – correction of references is in progress.

NOTE 2: Required versions of some references were not specified in the RFP.

[ADL] openEHR *Archetype Definition Language Version 1.4 (ADL)*, <http://www.openehr.org/releases/1.0.2/architecture/am/adl.pdf>

[AOM] openEHR *Archetype Object Model* (corresponding to ADL version 1.4), <http://www.openehr.org/releases/1.0.2/architecture/am/aom.pdf>

[AOMT] openEHR Archetype Templates, <http://www.openehr.org/svn/specification/TRUNK/publishing/architecture/am/tom.pdf>

[ARCH] openEHR *Archetypes: Constraint-based Domain Models for Future-proof Information Systems*, <http://www.openehr.org/publications/archetypes/archetypes_beale_oopsla_2002.pdf>

[CIMI] CIMI Reference Model Requirements, <http://informatics.mayo.edu/CIMI/index.php/CIMI_Reference_Model_Requirements>

[CTS2] OMG *Common Terminology Service 2 (CTS2)*, <http://www.omg.org/spec/CTS2/1.1/>

[HLV7v3] *HL7 Version 3 Standard: Core Principles and Properties of Version 3 Models*, <http://www.hl7.org/implement/standards/product_brief.cfm?product_id=58>

[MDMI] OMG *Model Driven Message Interoperability (MDMI), Version 1.0*, <http://www.omg.org/spec/MDMI/1.0/>

[MDR] *ISO/IEC 11179, Information Technology, -- Metadata registries*, <http://metadata-standards.org/11179/>

[NIEM] OMG *UML Profile for NIEM Version 1.0*, <http://www.omg.org/spec/NIEM-UML/1.0/>

[OCL] OMG *Object Constraint Language (OCL), Version 2.4*, <http://www.omg.org/spec/OCL/2.4/>

[ODM] OMG *Ontology Definition Metamodel (ODM) Version 1.1*, <http://www.omg.org/spec/ODM/1.1/>

[QVT] OMG *Meta Object Facility (MOF) 2.0 Query/View/Transformation, V1.2 (Beta)*, <http://www.omg.org/spec/QVT/1.2/Beta/>

[UML] OMG Unified Modeling Language (UML) Version 2.5 – Beta 2, <http://www.omg.org/spec/UML/2.5/Beta2/>

# Terms and Definitions

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

Archetype

An archetype is a re-usable, formal model of a concept. An archetype is expressed as a computable set of constraint statements, on an underlying reference model (URM). Concepts that can be modeled using archetypes include weight measurement, blood pressure, microbiology results, discharge referral, prescription, diagnosis. CIMI archetypes will be represented as an instance of the ‘Archetype Object Model’.

Archetype Definition Language

ADL is a formal language for expressing archetypes. It provides a formal, textual syntax for describing constraints on any domain entity whose data is described by an information model (also known as the 'underlying reference model'). The ADL syntax is semantically equivalent to the AOM, and represents one possible serialisation of the AOM. The current version of ADL is known as 'ADL 1.5'.

Archetype Instance

A single instantiation of data conforming to a specific archetype. In the context of CIMI this data will typically be clinical.

Archetype Model

An archetype model [TBD].

Archetype Object Model

The AOM is the definitive expression of archetype semantics, and is independent of any particular syntax. The AOM is defined as an object model, using a UML class diagram. It is a generic model, meaning that it can be used to express archetypes for any reference model in a standard way. Version 1.4 of the AOM was standardized in ISO-13606:2. The current version is known as 'AOM 1.5'.

Archetype Query Language

Archetype Query Language (AQL) is a declarative query language developed specifically for expressing queries used for searching and retrieving the clinical data found in archetype-based EHRs. AQL expresses the queries at the archetype level, i.e. semantic level, other than at the data instance level. This is the key in achieving sharing queries across system boundaries or enterprise boundaries.

Clinical Data Repository

A data store that holds and manages clinical data collected from service encounters at the point-of-service locations, for example: hospitals, clinics, etc.

Clinical Document Architecture

The HL7 Clinical Document Architecture (CDA) is an XML-based markup standard intended to specify the encoding, structure and semantics of clinical documents for exchange.

Clinical Information Model

A representation of the structured clinical information (including relationships, constraints and terminology), that describes a specific clinical concept - e.g. a blood pressure observation, a Discharge Summary, or a Medication Order.

Clinical Information Modeling Initiative

An initiative established to “improve the interoperability of healthcare information systems through shared implementable clinical information models.”

Clinical Information Modeling Initiative (CIMI) Reference Model

The CIMI Reference Model is the Underlying Reference Model on which CIMI's clinical models (i.e. archetypes) are defined. This reference model defines a rigorous and stable set of modeling patterns, including a set of complex data types, information patterns (e.g. data, qualifier, state), and structural patterns (e.g. composition, entry, tree). All CIMI clinical models (i.e. archetypes) will be defined by constraining the CIMI reference model. The reference model is intended to be instantiated with patient data, which conforms to the constraints defined by the associated clinical model.

Clinical Model Governance

A set of policies and processes through which the high clinical quality of all clinical artifacts (including clinical models and-or archetypes) is maintained during creation, storage, verification, maintenance, and distribution, by, for and on behalf of CIMI.

Clinical Model Repository

A clinical model repository is a data store that holds clinical information models and associated artifacts in an agreed sharable format.

Clinical Model Verification

Clinical model verification is the act of reviewing, inspecting or testing, in order to establish that a specification meets appropriate clinical safety and quality standards.

Clinical Modeling Language

A clinical modeling language is used to define clinical information models.

Clinical Requirement

Clinical requirements articulate clinical needs, including clinical practices, standards, guidelines, principles and other clinical concepts.

Code System

A managed collection of uniquely identifiable concepts with associated representations. A code system may also form an ontological system for representing a set of concepts, e.g. SNOMED-CT, LOINC, ICD-10, etc.

Common Terminology Services 2 (CTS2)

CTS2 is an OMG specification that provides a standard interface to disparate terminology sources. The Information Model specifies the structural definition, attributes and associations of Resources common to structured terminologies such as Code Systems, Binding Domains and Value Sets. The Computational Model specifies the service descriptions and interfaces needed to access and maintain structured terminologies.

Concept

In information modeling, a concept represents an “idea” as a word or phrase in order to support human understanding, but may also be represented with a concept identifier in order to bind it to a controlled terminology or ontology.

Concept Domain

A named category of like concepts that will be bound to one or more coded elements in an information model. Concept Domains exist to constrain the intent of the coded element and are independent of any specific vocabulary, code system, or Realm. A Concept Domain provides a high level grouping for all things possible in a given domain from which value sets will be constructed.

Concept Domain Binding

The association of a value set with a concept domain in a given context.

Conceptual Information Model

A representation of real-world objects and their relationships and constraints as understood by domain experts. A conceptual model should include no implementation-specific details.

Conformance

The requirement that those who participate in CIMI by contributing data components or creating and sharing ADL artifacts are following the agreed-upon procedures for doing so and that all documentation meets minimum criteria and the CIMI Naming and Design Rules where applicable.

Constraint Model

A formal specification used for describing constraints on an Underlying Reference Model. The Constraint Model is used to express clinical information models (i.e. archetypes). Not to be confused with the clinical information models that are instances of the constraint model.

Detailed Clinical Model

A relatively small, standalone information model designed to express a precise clinical concept in a standardized and reusable manner.

Fully Defined Concept

A concept that is uniquely defined by a set of defining relationships.

Information Model

A structured representation of the information requirements of a domain including the classes of information required and their attributes, relationships and constraints.

Node

A named part of an information model.

Ontology

A formal representation of knowledge as a set of concept identifiers, terms describing the concepts so identified, and the relationships among them.

Reference Model

A Reference Model is an information model which defines a set of modelling patterns upon which the clinical models are defined.

Reference Terminology

A terminology designed to provide common semantics for diverse implementations.

Semantic Binding

The association of a node in an information model with a concept from a controlled terminology that represents its meaning.

Terminology

A vocabulary of technical terms used in a particular field, subject, science, or art.

Terminology Binding

The assertion of a relationship between an information model and a terminology.

Value Binding

The association of a given node in a clinical model with the set of valid concepts that may populate it.

Value Set

A set of concept identifiers deemed valid for use in a specific context, especially to define the domain of a data element.

# Symbols

## Graphical Symbols

[TBD]

## Abbreviations

ADL Archetype Definition Language

AM Archetype Model

AML Archetype Modeling Language

AOM Archetype Object Model

AQL Archetype Query Language

CDA Clinical Document Architecture

CDL Clinical Document Language

CDR Clinical Data Repository

CIM Clinical Information Model

CIMI Clinical Information Modeling Initiative

CMP Constraint Model Profile

CRM Clinical Reference Model

CTS2 Common Terminology Services 2

EHR Electronic Health Record

HL7 Health Level Seven

LOINC Logical Observation Identifiers Names and Codes

MDA Model Driven Architecture

OMG Object Management Group

OpenEHR Open Electronic Health Record

PIM Platform Independent Model

PSM Platform Specific Model

RM Reference Model

RMP Reference Model Profile

SNOMED CT Systematized Nomenclature of Medicine – Clinical Terms

TBP Terminology Binding Profile

UML Unified Modeling Language

URI Uniform Resource Identifier

URM Underlying Reference Model

# Additional Information

## Changes to Adopted OMG Specifications

No changes to adopted OMG specifications are required to adopt this specification.

## 6.2 Acknowledgements

The following companies submitted this specification:

1. Mayo Clinic
2. Visumpoint, LLC

The following companies supported this specification:

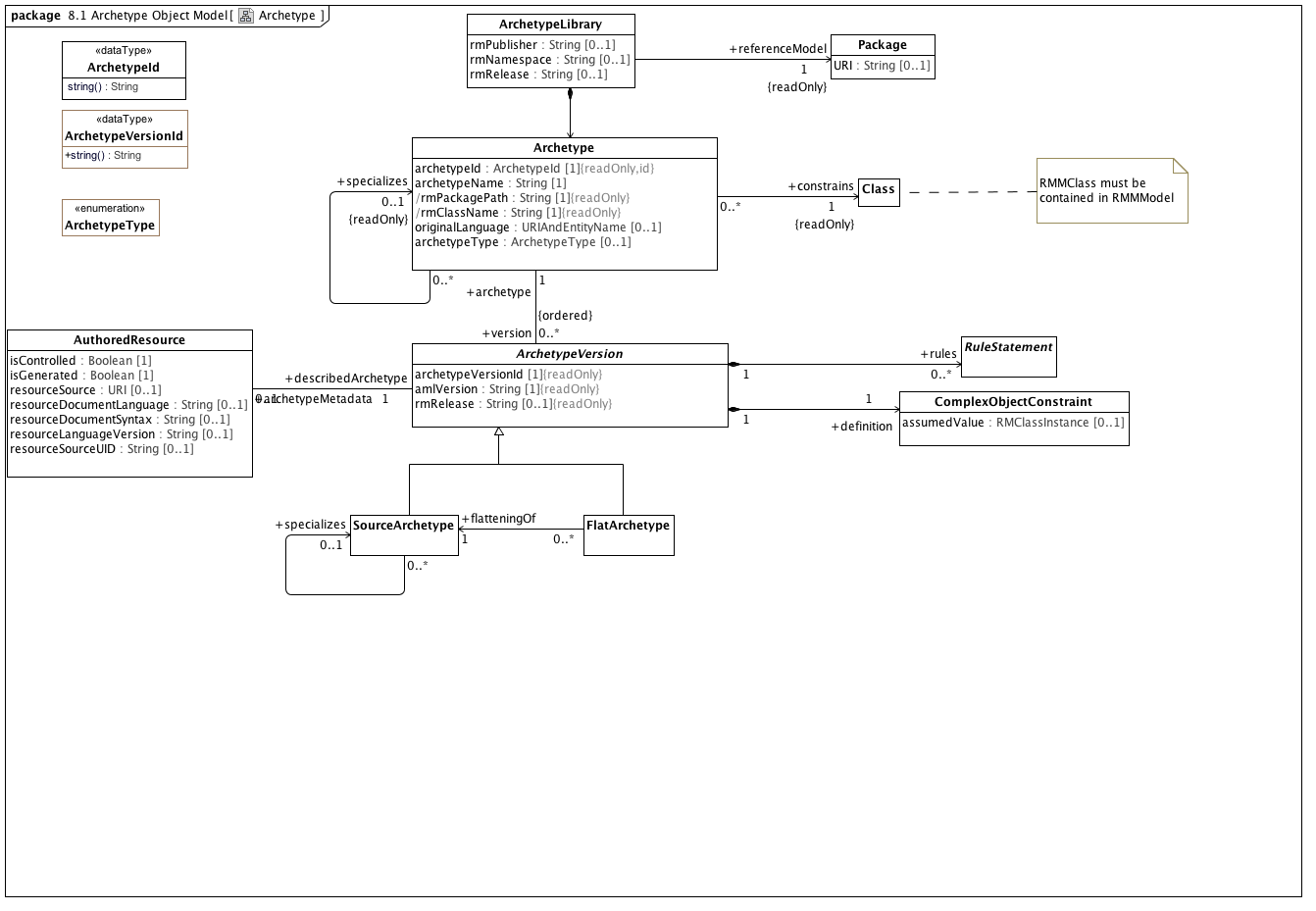
1. Escape Velocity, LLC

The following companies and organizations provided assistance aiding the development of this specification.

1. Intermountain Health [TBR]
2. The openEHR Foundation

# AML Object Model

## <Package> Archetype Object Model



**Archetype**

The Archetype package, shows the compositions rules, definition, archetypeId, and terminology; the concrete kinds of archetypes; and lineage by the recursive parent relationship.

### <DataType> ArchetypeId

**Description**

An artifact that uniquely identifies an *Archetype* within a given community of use. The actual syntax and structure of the *ArchetypId* type should be established by a community of use, but all *ArchetypeId* implementations must support a *String* representation.

**Diagrams**

[Archetype](#_7412d314307bc61dfa0287deae4ee4ce)

**Operations**

• string () : String

The stringified representation of the identifier.

### <DataType> ArchetypeVersionId

**Description**

An artifact that uniquely identifies a version of the Archetype within a given community of use. The actual syntax and structure of the ArchetypVersionId type should be established by a community of use, but all ArchetypeId implementations must support a String representation.

**Diagrams**

[Archetype](#_7412d314307bc61dfa0287deae4ee4ce)

**Operations**

• public string () : String

### <Class> Archetype

**Description**

A versioned constraint on a target UML class or other Archetype.

**Diagrams**

[Archetype](#_7412d314307bc61dfa0287deae4ee4ce)

**Attributes**

• public archetypeId : [ArchetypeId](#_abe68de6d7b599f5e4ea361caee12c81) [1]

The unique archetype identifier. However constructed, this uniquely identifies the Archetype across its entire life cycle. No other Archetype can have this identifier and a different identifier designates a different Archetype.

• public archetypeName : String [1]

The human readable name of the Archetype. This is typically derived from the other archetype details (See: openEHR Knowledge Artefact Identification - Revision 0.7.0 for an example). It is possible for this attribute to change over the life of an Archetype.

• public rmPackagePath : String [1]

The qualifiedName of a package in the target reference model that has the root rmClass as a visible member (there can be more than one possibility in a reference model).

• public rmClassName : String [1]

Name of the root class of this archetype. rmClass must match the name of the constrained RMMClass.

• public originalLanguage : [URIAndEntityName](#_4d1f571ab5e9384786ffe39444e822b4) [0..1]

The original language in which the archetype was authored.

• public archetypeType : [ArchetypeType](#_115a19030ad491571447653210109e56) [0..1]

**Associations**

• public constrains : [Class](#_a75c06fc93e516ccf92a1e38e18c46f3)[1]

The purpose of a Class is to specify a classification of objects and to specify the Features that characterize the structure and behavior of those objects.

• public version : [ArchetypeVersion](#_1de96fa71501cf96b27b14f3f9f1bb99)[0..\*]

An Archetype is a set of constraints that can be applied as a predicate against instances of the constrained Reference Model class as referenced by the Archetype.

• public specializes : [Archetype](#_1de96fa71501cf96b27b14f3f9f1bb99)[0..1]

An Archetype is a set of constraints that can be applied as a predicate against instances of the constrained Reference Model class as referenced by the Archetype.

**Constraints**

* **uri**

[OCL]

rmURI = rmUMLModel.URI

* **package**

[OCL]

rmPackagePath = rmPackage.qualifiedName

* **class**

[OCL]

rmClassName = constrains.name

* **classpackage**

[OCL]

rmPackage.member->exists(c|c=constrains)

* **differentArchetype**

An archetype cannot specialize itself or any of its descendants (note - only self is included in formal OCL)

[OCL]

not (self = self.specializes)

### <Class> ArchetypeLibrary

**Description**

A collection of archetypes that apply constrain one or more classes within a single reference model.

**Diagrams**

[Archetype](#_7412d314307bc61dfa0287deae4ee4ce)

**Attributes**

• public rmPublisher : String [0..1]

The name of the Reference Model publisher.

• public rmNamespace : String [0..1]

A human-readable organization identifier corresponding to the current custodian of the artefact. In ADL this is represented as the reverse domain name of the organization.

• public rmRelease : String [0..1]

The release id of the reference model. In ADL this would be in semver.org format.

**Associations**

• public : [Archetype](#_f45a7b68ecac449e953ff8a65d6eff75)

A versioned constraint on a target UML class or other Archetype.

• public referenceModel : [Package](#_a0a843d7d41881592e31e887cebd6da4)[1]

A Package is a namespace for its members, which comprise those elements associated via packagedElement (which are said to be owned or contained), and those imported.

### <Class> ArchetypeVersion

**Description**

An Archetype is a set of constraints that can be applied as a predicate against instances of the constrained Reference Model class as referenced by the Archetype.

**Diagrams**

[Archetype](#_7412d314307bc61dfa0287deae4ee4ce)

**Direct Known Subclasses (Specialization)**

[FlatArchetype](#_166addccec3fe63279b3f9eeb9930ab8), [SourceArchetype](#_dcaf9716e9bc2255b93c20393e8712f2)

**Attributes**

• public archetypeVersionId [1]

The identifier of the specific ArchetypeVersion. archetypeVersionId must, at a minimum, be unique within the context of the referencing archetype although some tools such as ADL use globally unique identifiers.

• public amlVersion : String [1]

The version of the AML specification in which the current archetype is expressed.

• public rmRelease : String [0..1]

The specific version of the reference model that was current when the ArchetypeVersion instance was last updated.

**Associations**

• public definition : [ComplexObjectConstraint](#_abfab8c8e983a73b4981f6fcfdd16134)[1]

A constraint on a complex object, which will typically consist of other constraints.

• public rules : [RuleStatement](#_f8740e8d27529166da46265bd8521c94)[0..\*]

Abstract parent of all statement types.

• public archetypeMetadata : [AuthoredResource](#_47dea9d0676ad6870be946fa52e870ad)[0..1]

*AuthoredResource* carries a minimal set of information about the source and origin of an *Archetype*. Its intent is to be a "connection point" to attach additional workflow and other provenance information to the target *Archetype.*

• public archetype : [Archetype](#_f45a7b68ecac449e953ff8a65d6eff75)[1]

A versioned constraint on a target UML class or other Archetype.

### <Class> AuthoredResource

**Description**

*AuthoredResource* carries a minimal set of information about the source and origin of an *Archetype*. Its intent is to be a "connection point" to attach additional workflow and other provenance information to the target *ArchetypeVersion.*

**Diagrams**

[Archetype](#_7412d314307bc61dfa0287deae4ee4ce)

**Attributes**

• public isControlled : Boolean [1]

A flag indicating whether the ArchetypeVersion is change-controlled or not can be included after the version. Archetypes that include the “controlled” flag should have the revision history section included, while those with the “uncontrolled” flag, or no flag at all, may omit the revision history. This enables archetypes to be privately edited in an early development phase without generating large revision histories of little or no value.

• public isGenerated : Boolean [1]

A flag indicating whether the archetype was generated or authored. This marker is used to support the migration to differential archetype representation introduced in ADL 1.5, to enable proper representation of specialised archetypes.

• public resourceSource : [URI](#_887928f30f99c8a1ca89ed7a082356aa) [0..1]

A URI that references the source document (if any) from which the original resource was derived.

• public resourceDocumentLanguage : String [0..1]

The language (e.g. AOM, CEM, ...) of the source of the constraints, if any.

• public resourceDocumentSyntax : String [0..1]

The syntax of the resource document (ADL, XML, XMI, ...)

• public resourceLanguageVersion : String [0..1]

The version of the resourceDocumentLanguage (e.g. ADL 1.5, XMI 2.1, etc)

• public resourceSourceUID : String [0..1]

An external identifier that uniquely identifies this *Archetype*. The format and structure of this identifier are determined by the rules of the *resourceDocumentLanguage* and/or *resourceDocumentSyntax.* This identifier cannot be used as an identifier within AML itself as it may not always be present. It must be preserved, however, for export to external resources.

**Associations**

• public describedArchetype : [ArchetypeVersion](#_1de96fa71501cf96b27b14f3f9f1bb99)[1]

An Archetype is a set of constraints that can be applied as a predicate against instances of the constrains Reference Model class as referenced by the Archetype.

### <Class> Class

**Description**

The purpose of a Class is to specify a classification of objects and to specify the Features that characterize the structure and behavior of those objects.

**Diagrams**

[Archetype](#_7412d314307bc61dfa0287deae4ee4ce)

**Direct Known Superclasses (Generalization)**

[Classifier](#_31f3bed9860f1a34043799bd12ffe873)

**Associations**

• public ownedAttribute : [Property](#_652433ba6af347b5ab3d4b0b4b2931c9)[0..\*]

A Reference Model Property (RMProperty) is a proper subset of a UML property. The aspects of a property that can be addressed by Archetype include:

The property name. Only named properties may be constrained within a reference model. Referring to section 7.4.3 in the ptc/2013-09-05, a non-hidden *RMProperty* is always referred to in its unqualified form. If it is necessary to reference hidden elements within an archetype, the qualified name (N::x) form should be used. Qualification should be the minimum sufficient to render the name unique.

The lower and upper bounds. The UML MultiplicityElement shows lower and upper as derived properties. The AML specification assumes that, if present, these properties have been computed and it is up to the implementer to correctly interpret MultiplicityElement lowerValue and upperValue properties to determine these results.

Derived properties cannot be constrained using AML -- the rationale being that (a) derived properties may or may not be present in object instances and (b) the primary constraints need to be applied to the parameters of the derivation rather than the result.

UML::Associations are not used in AML -- the model is traversed via the *ownedAttribute* association. AssociationClasses are not differentiated from any other RMMClass in the model, and may be referenced and traversed via. whatever *ownedAttribute*/type links that are available.

While default values can be specified in the Reference Model, they are ignored in AML. Note, however, that AML can specify default values (with tighter semantics) in an archetype.

All other UML::Property links, including aggregation, isComposite, isID, association, qualifier, opposite, defaultVelue, redefines, subsettedProperty, and interface are ignored within the AML profile.

• public ownedTemplateSignature : [RedefinableTemplateSignature](#_6afdd25f5589999ef5ae78a4eab8563d)[0..1]

A *RMRedefinableTemplateSignature* associates an ordered list of *RMClassifierTemplateParameters* with an owning *RMClass.* The owning *RMClass* typically, but not always, has one or more ownedAttributes that reference one of the ownedParameters of the *RMRedefinableTemplateSignature.*

• public templateBinding : [TemplateBinding](#_039ec0a61521832e985575d3d9688234)[0..\*]

*RMTemplateBinding* is a subtype of the UML::TemplateBinding class. It represents a set of parameter substitutions that are to be applied to a *RMRedefinableTemplateSignature* defined by a parent or ancestor *superClass.* A *RMTemplateBinding* contains one or more parameter substitutions to be applied to one or more types referenced by an *ownedAttribute* of the parent or ancestor class.

• public attribute : [Property](#_652433ba6af347b5ab3d4b0b4b2931c9)[0..\*]

A Reference Model Property (RMProperty) is a proper subset of a UML property. The aspects of a property that can be addressed by archetype includes:

The property name. Only named properties may be constrained within a reference model. Referring to section 7.4.3 in the ptc/2013-09-05, a non-hidden *RMProperty* is always referred to in its unqualified form. If it is necessary to reference hidden elements within an archetype, the qualified name (N::x) form should be used. Qualification should be the minimum that sufficient to render the name unique.

The lower and upper bounds. The UML MultiplicityElement shows lower and upper as derived properties. The AML specification assumes that, if present, these properties have been computed and it is up to the implementer to correctly interpret MultiplicityElement lowerValue and upperValue properties to determine these results.

Derived properties cannot be constrained using AML -- the rationale being that (a) derived properties may or may not be present in object instances and (b) the primary constraints need to be applied to the parameters of the derivation rather than the result.

UML::Associations are not used in AML -- the model is traversed via the *ownedAttribute* association. AssociationClasses are not differentiated from any other RMMClass in the model, and may be referenced and traversed via. whatever *ownedAttribute*/type links that are available.

While default values can be specified in the Reference Model, they are ignored in AML. Note, however, that AML can specify default values (with tighter semantics) in an archetype.

All other UML::Property links, including aggregation, isComposite, isID, association, qualifier, opposite, defaultVelue, redefines, subsettedProperty, and interface are ignored within the AML profile.

• public superClass : [Property](#_652433ba6af347b5ab3d4b0b4b2931c9)[0..\*]

A Reference Model Property (RMProperty) is a proper subset of a UML property. The aspects of a property that can be addressed by archetype includes:

The property name. Only named properties may be constrained within a reference model. Referring to section 7.4.3 in the ptc/2013-09-05, a non-hidden *RMProperty* is always referred to in its unqualified form. If it is necessary to reference hidden elements within an archetype, the qualified name (N::x) form should be used. Qualification should be the minimum that sufficient to render the name unique.

The lower and upper bounds. The UML MultiplicityElement shows lower and upper as derived properties. The AML specification assumes that, if present, these properties have been computed and it is up to the implementer to correctly interpret MultiplicityElement lowerValue and upperValue properties to determine these results.

Derived properties cannot be constrained using AML -- the rationale being that (a) derived properties may or may not be present in object instances and (b) the primary constraints need to be applied to the parameters of the derivation rather than the result.

UML::Associations are not used in AML -- the model is traversed via the *ownedAttribute* association. AssociationClasses are not differentiated from any other RMMClass in the model, and may be referenced and traversed via. whatever *ownedAttribute*/type links that are available.

While default values can be specified in the Reference Model, they are ignored in AML. Note, however, that AML can specify default values (with tighter semantics) in an archetype.

All other UML::Property links, including aggregation, isComposite, isID, association, qualifier, opposite, defaultVelue, redefines, subsettedProperty, and interface are ignored within the AML profile.

### <Class> ComplexObjectConstraint

**Description**

A constraint on a complex object, which will typically consist of other constraints.

**Diagrams**

[Archetype](#_7412d314307bc61dfa0287deae4ee4ce)

**Direct Known Superclasses (Generalization)**

[NamedObjectConstraint](#_ab5b3b01964560abb1047edd9efa4eb9)

**Attributes**

• public assumedValue : [RMClassInstance](#_935cd7de4b22d47dd2c6aef93bed5c7a) [0..1]

Value to be assumed in instances in which no value is provided.

**Associations**

• public attributeTuple : [AttributeTupleConstraint](#_f6da15c71717330ae1b56f8b41e3dd51)[0..\*]

An AttributeTupleConstraint presents a set of two or more alternative tuples, each of which consists of two or more attributes. The containing ComplexObjectConstraint is satisfied when all of the constraints in one of the AttributeTuples are satisfied.

• private targetObject : [ObjectConstraintProxy](#_6da4a9bc7db41a2b89064f79f0c4ed36)[0..\*]

A constraint defined by reference to a node defined elsewhere in the same archetype.

• public constrains : [Class](#_a75c06fc93e516ccf92a1e38e18c46f3)[1]

The purpose of a Class is to specify a classification of objects and to specify the Features that characterize the structure and behavior of those objects.

**Constraints**

* **instanceOfConstraint**

[English]

If assumedValue exists, assumedValue.classifier must be equal to or a specialization of self.parent

### <Class> FlatArchetype

**Description**

A *FlatArchetype* is generated from one or more *SourceArchetypes* via the flattening process. The flattening operation:

Replaces *ComplexObjectConstraintProxies* with *ComplexObjectConstraints* that contain copies of the subtrees to which they point.

Applies *SourceArchetype* overlays to the parent structure resulting in a full archetype structure.

**Diagrams**

[Archetype](#_7412d314307bc61dfa0287deae4ee4ce)

**Direct Known Superclasses (Generalization)**

[ArchetypeVersion](#_1de96fa71501cf96b27b14f3f9f1bb99)

**Associations**

• public flatteningOf : [SourceArchetype](#_dcaf9716e9bc2255b93c20393e8712f2)[1]

The source form of an archetype, potentially including references to other archetypes whose contents are not explicitly reproduced in the source form

**Constraints**

* **noSpecialization**

A fFatArchetype cannot specialize another archetype

[OCL]

not exists(archetype.specializes)

### <Class> Package

**Description**

A Package is a namespace for its members, which comprise those elements associated via packagedElement (which are said to be owned or contained), and those imported.

**Diagrams**

[Archetype](#_7412d314307bc61dfa0287deae4ee4ce)

**Direct Known Superclasses (Generalization)**

[Namespace](#_f762e4ef59f1948849a49d421126c16b)

**Attributes**

• public URI : String [0..1]

### <Class> RuleStatement

**Description**

Abstract parent of all statement types

**Diagrams**

[Archetype](#_7412d314307bc61dfa0287deae4ee4ce)

**Direct Known Superclasses (Generalization)**

[RuleElement](#_aca03c36f52ec3dbae79ae94604d6df9)

**Direct Known Subclasses (Specialization)**

[Assertion](#_f9e7c553caf3e674732fe386e3d45466), [VariableDeclaration](#_8a634b04f92ff4c449cdcaaae16ba015)

### <Class> SourceArchetype

**Description**

The source form of an archetype, potentially including references to other archetypes whose contents are not explicitly reproduced in the source form.

**Diagrams**

[Archetype](#_7412d314307bc61dfa0287deae4ee4ce)

**Direct Known Superclasses (Generalization)**

[ArchetypeVersion](#_1de96fa71501cf96b27b14f3f9f1bb99)

**Associations**

• public specializes : [SourceArchetype](#_dcaf9716e9bc2255b93c20393e8712f2)[0..1]

The source form of an archetype, potentially including references to other archetypes whose contents are not explicitly reproduced in the source form.

**Constraints**

* **sameArchetypes**

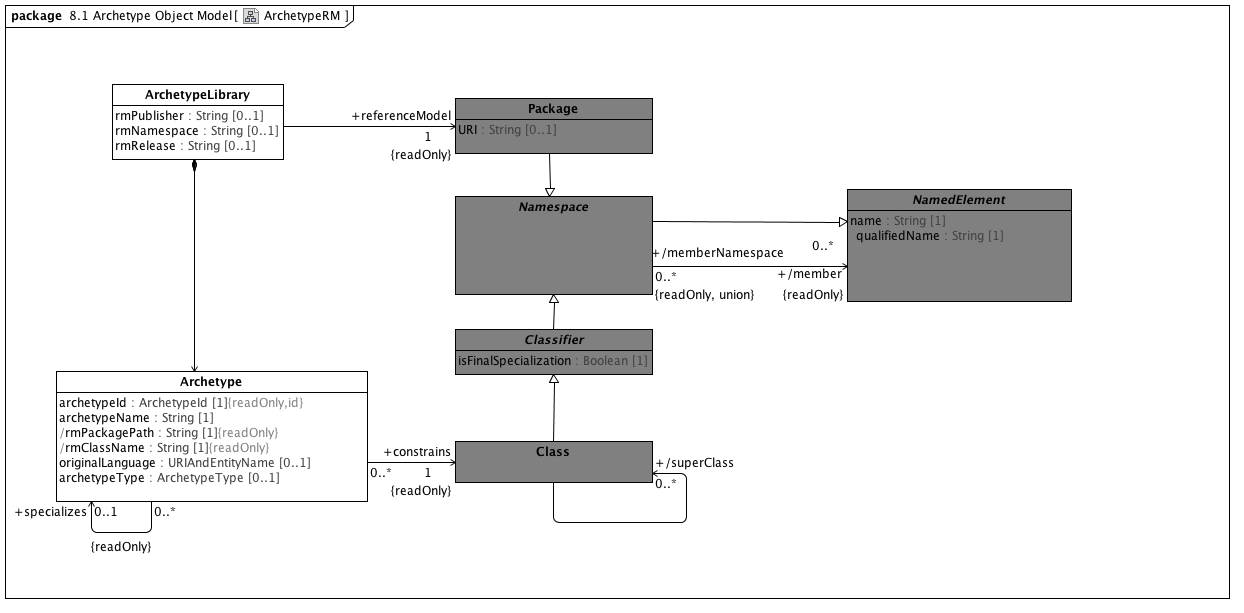
[OCL]

exists(self.specializes) implies self.archetype.specializes = self.specializes.archetype.specializes

### <Enumeration> ArchetypeType

**Diagrams**

[Archetype](#_7412d314307bc61dfa0287deae4ee4ce)



**ArchetypeRM**

An Archetype references (or constrains) a single UML Class. The constrained Class must be a member of the UML Package that is constrained by the Archetype Library.

### <Class> Classifier

**Description**

A Classifier has a set of Features, some of which are Properties called the attributes of the Classifier.

**Diagrams**

[ArchetypeRM](#_8371278cdefade515c9d54e183ad347b)

**Direct Known Superclasses (Generalization)**

[Namespace](#_f762e4ef59f1948849a49d421126c16b)

**Direct Known Subclasses (Specialization)**

[Class](#_a75c06fc93e516ccf92a1e38e18c46f3), [DataType](#_d5914eb0da42172989bbe57f23fc4310)

**Attributes**

• public isFinalSpecialization : Boolean [1]

If true, the *RMMClassifier* instance cannot be constrained within an archetype.

**Associations**

• public templateParameter : [ClassifierTemplateParameter](#_3d9b09fe9052c8305d90ab92bc37d26b)[0..1]

### <Class> NamedElement

**Description**

NamedElement is the superclass of all named elements in the Reference Model, and represents the subset of UML::NamedElements that are referenced by the AML profile. While a Reference Model may contain UML::NamedElements without names, Archetypes can only constrain those that have names and are of type PrimitiveDataType, Class or Property.

**Diagrams**

[ArchetypeRM](#_8371278cdefade515c9d54e183ad347b)

**Direct Known Subclasses (Specialization)**

[EnumerationLiteral](#_41faf6a7041d7068bbbf4f9ff3924d22), [Namespace](#_f762e4ef59f1948849a49d421126c16b), [Property](#_652433ba6af347b5ab3d4b0b4b2931c9)

**Attributes**

• public name : String [1]

The name of the Reference Model element. Name must be unique within the context of the owning *namespace.*

• public qualifiedName : String [1]

**Associations**

• public namespace : [Namespace](#_f762e4ef59f1948849a49d421126c16b)[0..1]

An element in a model that owns and/or imports a set of NamedElements that can be identified by name.

### <Class> Namespace

**Description**

An element in a model that owns and/or imports a set of NamedElements that can be identified by name.

**Diagrams**

[ArchetypeRM](#_8371278cdefade515c9d54e183ad347b)

**Direct Known Superclasses (Generalization)**

[NamedElement](#_527fd9eb1e787c36a3748854a9431816)

**Direct Known Subclasses (Specialization)**

[Classifier](#_31f3bed9860f1a34043799bd12ffe873), [Package](#_a0a843d7d41881592e31e887cebd6da4)

**Associations**

• public member : [NamedElement](#_527fd9eb1e787c36a3748854a9431816)[0..\*]

NamedElement is the superclass of all named elements in the Reference Model, and represents the subset of UML::NamedElements that are referenced by the AML profile. While a Reference Model may contain UML::NamedElements without names, Archetypes can only constrain those that have names and are of type PrimitiveDataType, Class or Property.

• public ownedMember : [NamedElement](#_527fd9eb1e787c36a3748854a9431816)[0..\*]

NamedElement is the superclass of all named elements in the Reference Model, and represents the subset of UML::NamedElements that are referenced by the AML profile. While a Reference Model may contain UML::NamedElements without names, Archetypes can only constrain those that have names and are of type PrimitiveDataType, Class or Property.

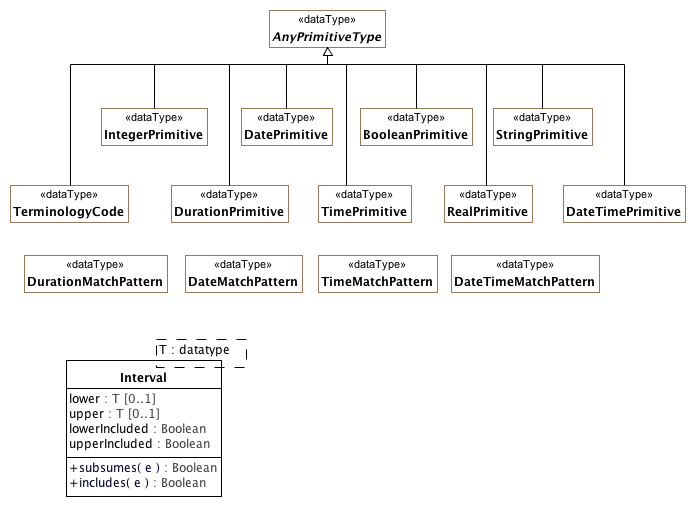
**Known other classes**

[Archetype](#_f45a7b68ecac449e953ff8a65d6eff75), [ArchetypeLibrary](#_214dd64d9ba4a03a70f9f9f80f8c47d2), [Class](#_a75c06fc93e516ccf92a1e38e18c46f3), [Package](#_a0a843d7d41881592e31e887cebd6da4)

## <Package> Reference Object Model

### <Package> Primitive Data Types

In the AML/ADL context, the term "primitive data type" is used to indicate "leaf nodes" -- data elements that are treated as being atomic and are only constrained in terms of their possible value ranges. AML data types, like UML data types, are "model Types whose instances are distinguished only by their value" but, unlike the UML definition of "Primitive Type", AML primitive types can embody the notion of substructure. When a AML profile is applied to a UML Reference Model, it may be necessary to map one or more of the AML types to corresponding types in the target model.



**PrimitiveDataTypes**

The set of primitive data types that can appear on a reference model and can be constrained using AML primitive constraints.

### <DataType> AnyPrimitiveType

**Description**

An AnyPrimitiveType is an abstract supertype of data types. This type usually maps to a type like "Any" or "Object" in an object system; it is defined here to provide the value and reference equality semantics.

**Diagrams**

[PrimitiveDataTypes](#_1b4f676a5a33d3d5b1673552f8bd8ac1)

**Direct Known Subclasses (Specialization)**

[BooleanPrimitive](#_19737cdaaaee2179c50553b52361808b), [DatePrimitive](#_db5d020506d0af7a330f0b4fe1cb870a), [DateTimePrimitive](#_8347879bb381db17040637cc3ba0a25c), [DurationPrimitive](#_64b0026498682fb721ccdb38c186bba2), [IntegerPrimitive](#_89e2b2b9de405e6d05c4c5259fc8ffd6), [RealPrimitive](#_c596f10fb93cb4f697f2f1b0b64b43ce), [StringPrimitive](#_6a90be7cfa784ea4b4e8cad8f4a47e82), [TerminologyCode](#_3828b4174a837b16c5b4b5dcadf76a23), [TimePrimitive](#_05e0f2a221f0733d693058e9253b0017)

### <DataType> BooleanPrimitive

**Description**

A BooleanPrimative is a primitive type with two values supporting binary logic.

**Diagrams**

[PrimitiveDataTypes](#_1b4f676a5a33d3d5b1673552f8bd8ac1)

**Direct Known Superclasses (Generalization)**

[AnyPrimitiveType](#_72b7d3db018aa6b2ac9fd5dcdf816a12)

### <DataType> DateMatchPattern

**Description**

ADateMatchPattern is a pattern supporting the specification of dates by partial match.

**Diagrams**

[PrimitiveDataTypes](#_1b4f676a5a33d3d5b1673552f8bd8ac1)

### <DataType> DatePrimitive

**Description**

A DatePrimitive is a primitive type specifying a calendar date.

**Diagrams**

[PrimitiveDataTypes](#_1b4f676a5a33d3d5b1673552f8bd8ac1)

**Direct Known Superclasses (Generalization)**

[AnyPrimitiveType](#_72b7d3db018aa6b2ac9fd5dcdf816a12)

### <DataType> DateTimeMatchPattern

**Description**

A DateTimeMatchPattern is a pattern supporting the specification of DateTimes by partial match.

**Diagrams**

[PrimitiveDataTypes](#_1b4f676a5a33d3d5b1673552f8bd8ac1)

### <DataType> DateTimePrimitive

**Description**

A DateTimePrimitive is a primitive type specifying a character string.

**Diagrams**

[PrimitiveDataTypes](#_1b4f676a5a33d3d5b1673552f8bd8ac1)

**Direct Known Superclasses (Generalization)**

[AnyPrimitiveType](#_72b7d3db018aa6b2ac9fd5dcdf816a12)

### <DataType> DurationMatchPattern

**Description**

A DurationMatchPattern is a pattern supporting the specification of durations by partial match.

**Diagrams**

[PrimitiveDataTypes](#_1b4f676a5a33d3d5b1673552f8bd8ac1)

### <DataType> DurationPrimitive

**Description**

A DurationPrimitive is a primitive type specifying a length of time.

**Diagrams**

[PrimitiveDataTypes](#_1b4f676a5a33d3d5b1673552f8bd8ac1)

**Direct Known Superclasses (Generalization)**

[AnyPrimitiveType](#_72b7d3db018aa6b2ac9fd5dcdf816a12)

### <DataType> IntegerPrimitive

**Description**

An IntegerPrimative is a primitive type specifying an integer.

**Diagrams**

[PrimitiveDataTypes](#_1b4f676a5a33d3d5b1673552f8bd8ac1)

**Direct Known Superclasses (Generalization)**

[AnyPrimitiveType](#_72b7d3db018aa6b2ac9fd5dcdf816a12)

### <DataType> RealPrimitive

**Description**

A RealPrimitive is a primitive type specifying a real number.

**Diagrams**

[PrimitiveDataTypes](#_1b4f676a5a33d3d5b1673552f8bd8ac1)

**Direct Known Superclasses (Generalization)**

[AnyPrimitiveType](#_72b7d3db018aa6b2ac9fd5dcdf816a12)

### <DataType> StringPrimitive

**Description**

A StringPrimitive is a primitive type specifying a character string.

**Diagrams**

[PrimitiveDataTypes](#_1b4f676a5a33d3d5b1673552f8bd8ac1)

**Direct Known Superclasses (Generalization)**

[AnyPrimitiveType](#_72b7d3db018aa6b2ac9fd5dcdf816a12)

### <DataType> TerminologyCode

**Diagrams**

[PrimitiveDataTypes](#_1b4f676a5a33d3d5b1673552f8bd8ac1)

**Direct Known Superclasses (Generalization)**

[AnyPrimitiveType](#_72b7d3db018aa6b2ac9fd5dcdf816a12)

### <DataType> TimeMatchPattern

**Description**

A TimeMatchPattern is a pattern supporting the specification of times by partial match.

**Diagrams**

[PrimitiveDataTypes](#_1b4f676a5a33d3d5b1673552f8bd8ac1)

### <DataType> TimePrimitive

**Description**

A TimePrimitive is a primitive type specifying a point in time.

**Diagrams**

[PrimitiveDataTypes](#_1b4f676a5a33d3d5b1673552f8bd8ac1)

**Direct Known Superclasses (Generalization)**

[AnyPrimitiveType](#_72b7d3db018aa6b2ac9fd5dcdf816a12)

### <Class> Interval

**Diagrams**

[PrimitiveDataTypes](#_1b4f676a5a33d3d5b1673552f8bd8ac1)

**Attributes**

• public lower : [T](#_e36b5e306e1664a3e0bc600785109705) [0..1]

• public upper : [T](#_e36b5e306e1664a3e0bc600785109705) [0..1]

• lowerIncluded : Boolean

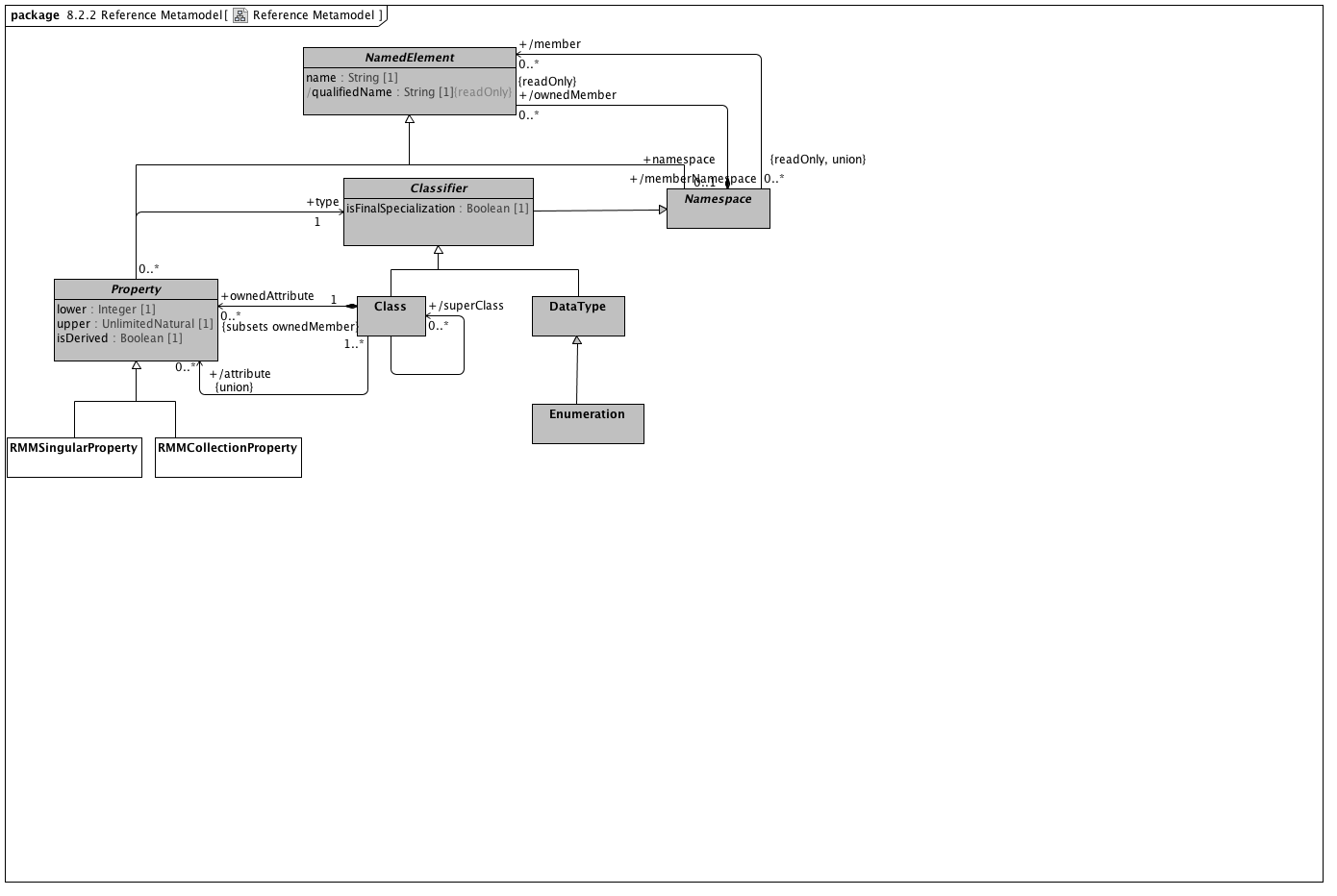
• public upperIncluded : Boolean

**Operations**

• public subsumes (e) : Boolean

• public includes (e) : Boolean

### <Package> Reference Metamodel



**Reference Metamodel**

### <Class> DataType

**Description**

DataTypes model Types whose instances are distinguished only by their value.

**Diagrams**

[Reference Metamodel](#_4067d3d86b09d7e8c4b542bda7773054)

**Direct Known Superclasses (Generalization)**

[Classifier](#_31f3bed9860f1a34043799bd12ffe873)

**Direct Known Subclasses (Specialization)**

[Enumeration](#_190e24bd48f094ad9ad981ac0b4eb47e)

### <Class> Enumeration

**Description**

Enumeration is a kind of DataType wherein each value of the Enumeration corresponds to one of its user-defined EnumerationLiterals.

**Diagrams**

[Reference Metamodel](#_4067d3d86b09d7e8c4b542bda7773054)

**Direct Known Superclasses (Generalization)**

[DataType](#_d5914eb0da42172989bbe57f23fc4310)

**Direct Known Subclasses (Specialization)**

[EnumeratedValueDomain](#_ad639ee3d4cd535b2d3e55238d69cc51)

**Associations**

• public ownedLiteral : [EnumerationLiteral](#_41faf6a7041d7068bbbf4f9ff3924d22)[0..\*]

An EnumerationLiteral defines an element of the run-time extension of an Enumeration. Values corresponding to EnumerationLiterals are immutable and may be compared for equality. EnumerationLiterals may not change during their existence, so any attributes on an Enumeration shall be read-only.

An EnumerationLiteral has a name that shall be used to identify it within its Enumeration. The EnumerationLiteral name is scoped within and shall be unique within its Enumeration. EnumerationLiteral names shall be qualified for general use.

**Constraints**

* **enumerateValueDomain**

### <Class> Property

**Description**

A Reference Model Property (RMProperty) is a proper subset of a UML property. The aspects of a property that can be addressed by Archetype include:

The property name. Only named properties may be constrained within a reference model. Referring to section 7.4.3 in the ptc/2013-09-05, a non-hidden *RMProperty* is always referred to in its unqualified form. If it is necessary to reference hidden elements within an Archetype, the qualified name (N::x) form should be used. Qualification should be the minimum that is sufficient to render the name unique.

The lower and upper bounds. The UML MultiplicityElement shows lower and upper as derived properties. The AML specification assumes that, if present, these properties have been computed and it is up to the implementer to correctly interpret MultiplicityElement lowerValue and upperValue properties to determine these results.

Derived properties cannot be constrained using AML -- the rationale being that (a) derived properties may or may not be present in object instances and (b) the primary constraints need to be applied to the parameters of the derivation rather than the result.

UML::Associations are not used in AML. The model is traversed via the *ownedAttribute* association. AssociationClasses are not differentiated from any other RMMClass in the model, and may be referenced and traversed via whatever *ownedAttribute*/type links are available.

While default values can be specified in the Reference Model, they are ignored in AML. Note, however, that AML can specify default values (with tighter semantics) in an Archetype.

All other UML::Property links, including aggregation, isComposite, isID, association, qualifier, opposite, defaultVelue, redefines, subsettedProperty, and interface are ignored within the AML profile.

**Diagrams**

[Reference Metamodel](#_4067d3d86b09d7e8c4b542bda7773054)

**Direct Known Superclasses (Generalization)**

[NamedElement](#_527fd9eb1e787c36a3748854a9431816)

**Direct Known Subclasses (Specialization)**

[RMMCollectionProperty](#_702e30cd0381b6e9726bcc6fe779a70f), [RMMSingularProperty](#_5917d2795b1a9ae4f33929e6edb8af81)

**Attributes**

• private lower : Integer [1]

• private upper : UnlimitedNatural [1]

• public isDerived : Boolean [1]

**Associations**

• public type : [Classifier](#_31f3bed9860f1a34043799bd12ffe873)[1]

A Classifier has a set of Features, some of which are Properties called the attributes of the Classifier.

**Constraints**

* **nonDerived**

[OCL]

isDerived=false

### <Class> RMMCollectionProperty

**Description**

RMMCollectionProperty represents the subset of RMMProperty instances that can occur more than one time. An RMMCollectionProperty instance is viewed by AML as a collection of objects of a given type that possess two separate characteristics:

1. The collection as a whole may be required, optional or prohibited.
2. The cardinality of the collection may be constrained.

This combination allows a number of useful constructs, including:

* requiring that a list be present but that it has no members. This construct can be used to assert a relationship between an object and an empty set of objects.
* making an attribute optional, but, if present, requiring that it have a minimum number of members

**Diagrams**

[Reference Metamodel](#_4067d3d86b09d7e8c4b542bda7773054)

**Direct Known Superclasses (Generalization)**

[Property](#_652433ba6af347b5ab3d4b0b4b2931c9)

### <Class> RMMSingularProperty

**Description**

*RMMSingularProperty* represents the subset of *RMMProperty* instances having an upper bound of 1. The AML treats RMMSingularProperties as single values (vs. collections) of attributes that can be required, optional or prohibited.

**Diagrams**

[Reference Metamodel](#_4067d3d86b09d7e8c4b542bda7773054)

**Direct Known Superclasses (Generalization)**

[Property](#_652433ba6af347b5ab3d4b0b4b2931c9)

**Constraints**

* **singular**

[OCL]

upper = 1

* **collection**

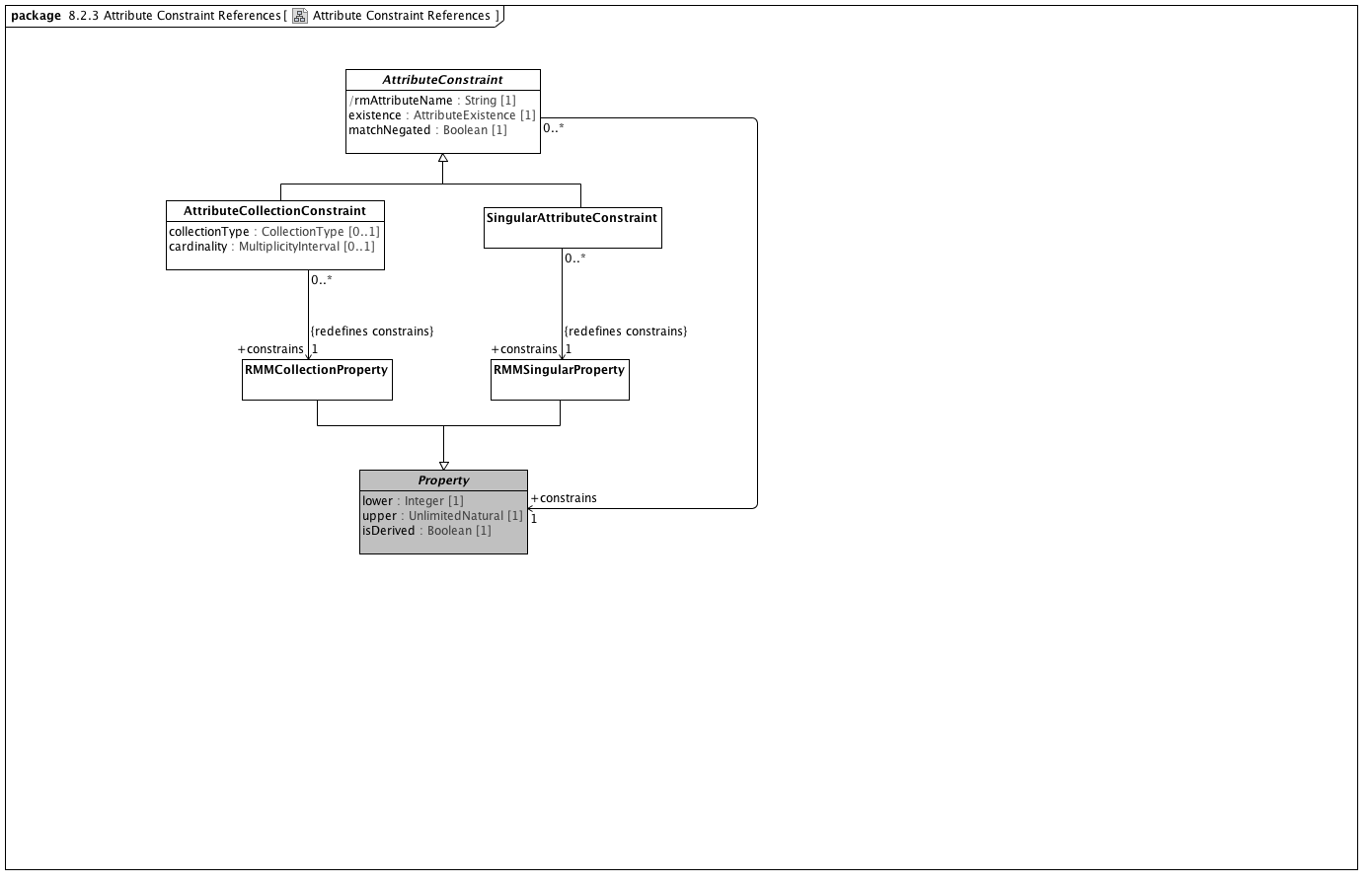
[OCL]

upper > 1 or upper.isUnlimited()

**Known other classes**

[Class](#_a75c06fc93e516ccf92a1e38e18c46f3), [Classifier](#_31f3bed9860f1a34043799bd12ffe873), [NamedElement](#_527fd9eb1e787c36a3748854a9431816), [Namespace](#_f762e4ef59f1948849a49d421126c16b)

### <Package> Attribute Constraint References



**Attribute Constraint References**

### <Class> AttributeCollectionConstraint

**Description**

An AttributeCollectionConstraint is a kind of AttributeConstraint that applies to a set of objects contained in an attribute.

**Diagrams**

[Attribute Constraint References](#_39027eaec61a2eaccc1fccb451cdda98)

**Direct Known Superclasses (Generalization)**

[AttributeConstraint](#_11f887fb6f19248bf7193bca31772c05)

**Attributes**

• public collectionType : [CollectionType](#_ac8c7771bf7f68d6747022e6924749ca) [0..1]

A classification of collection indicating whether its members must be unique or ordered.

.

• public cardinality : [MultiplicityInterval](#_c810ec7fa381fa249b7a7d9fecae85b6) [0..1]

The cardinality is the range in the number of members that can be included in a Property constrained by the AttributeCollectionConstraint.

**Associations**

• public member : [AttributeCollectionMember](#_177e37623ae3f5642980fd445bf78af1)[1..\*]

An association that matches members of a collection of attributes with specific ObjectConstraints.

• public constrains : [RMMCollectionProperty](#_702e30cd0381b6e9726bcc6fe779a70f)[1]

RMMCollectionProperty represents the subset of RMMProperty instances that can occur more than one time. An RMMCollectionProperty instance is viewed by AML as a collection of objects of a given type that possess two separate characteristics:

The collection as a whole may be required, optional or prohibited.

The cardinality of the collection may be constrained.

This combination allows a number of useful constructs, including:

requiring that a list be present but that it have no members, which can be used to assert a relationship between an object and an empty set of objects

making an attribute optional, but, if present, requiring that it have a minimum number of members

**Constraints**

* **collectionType**

[OCL]

parentProperty.isOrdered implies collectionType = CollectionType::LIST and parentProperty.isUnique implies collectionType = CollectionType.SET

* **cardinality**

[OCL]

cardinality.minimum >= parent.lower and (cardinality.maximum = unlimitedValue() or parent.upper = unlimitedValue() or cardinality.maximum <= parent.upper)

### <Class> AttributeConstraint

**Description**

A constraint on a reference model attribute

**Diagrams**

[Attribute Constraint References](#_39027eaec61a2eaccc1fccb451cdda98)

**Direct Known Subclasses (Specialization)**

[AttributeCollectionConstraint](#_5eefba8eca7402f09bd5619804038771), [SingularAttributeConstraint](#_48ee2586ffa14e5bb1cf8ad893969da7)

**Attributes**

• public rmAttributeName : String [1]

Name of attribute within the reference model that is constrained by this node

• public existence : [AttributeExistence](#_4f99fbfcf9617d7ad55eca111d84fb67) [1]

Strength of requirement that the attribute instance be present

• public matchNegated : Boolean [1]

Whether the match operator is to be inverted so that the constraint specifies anything except what is represented

**Associations**

• public attribute : [ComplexObjectConstraint](#_abfab8c8e983a73b4981f6fcfdd16134)

A constraint on a complex object, which will typically consist of other constraints

• public constrains : [Property](#_652433ba6af347b5ab3d4b0b4b2931c9)[1]

A Reference Model Property (RMProperty) is a proper subset of a UML property. The aspects of a property that can be addressed by archetype includes:

The property name. Only named properties may be constrained within a reference model. Referring to section 7.4.3 in the ptc/2013-09-05, a non-hidden *RMProperty* is always referred to in its unqualified form. If it is necessary to reference hidden elements within an archetype, the qualified name (N::x) form should be used. Qualification should be the minimum that sufficient to render the name unique.

The lower and upper bounds. The UML MultiplicityElement shows lower and upper as derived properties. The AML specification assumes that, if present, these properties have been computed and it is up to the implementer to correctly interpret MultiplicityElement lowerValue and upperValue properties to determine these results.

Derived properties cannot be constrained using AML -- the rationale being that (a) derived properties may or may not be present in object instances and (b) the primary constraints need to be applied to the parameters of the derivation rather than the result.

UML::Associations are not used in AML -- the model is traversed via the *ownedAttribute* association. AssociationClasses are not differentiated from any other RMMClass in the model, and may be referenced and traversed via. whatever *ownedAttribute*/type links that are available.

While default values can be specified in the Reference Model, they are ignored in AML. Note, however, that AML can specify default values (with tighter semantics) in an archetype.

All other UML::Property links, including aggregation, isComposite, isID, association, qualifier, opposite, defaultVelue, redefines, subsettedProperty, and interface are ignored within the AML profile.

• public parent : [Property](#_652433ba6af347b5ab3d4b0b4b2931c9)[1]

A Reference Model Property (RMProperty) is a proper subset of a UML property. The aspects of a property that can be addressed by archetype includes:

The property name. Only named properties may be constrained within a reference model. Referring to section 7.4.3 in the ptc/2013-09-05, a non-hidden *RMProperty* is always referred to in its unqualified form. If it is necessary to reference hidden elements within an archetype, the qualified name (N::x) form should be used. Qualification should be the minimum that sufficient to render the name unique.

The lower and upper bounds. The UML MultiplicityElement shows lower and upper as derived properties. The AML specification assumes that, if present, these properties have been computed and it is up to the implementer to correctly interpret MultiplicityElement lowerValue and upperValue properties to determine these results.

Derived properties cannot be constrained using AML -- the rationale being that (a) derived properties may or may not be present in object instances and (b) the primary constraints need to be applied to the parameters of the derivation rather than the result.

UML::Associations are not used in AML -- the model is traversed via the *ownedAttribute* association. AssociationClasses are not differentiated from any other RMMClass in the model, and may be referenced and traversed via. whatever *ownedAttribute*/type links that are available.

While default values can be specified in the Reference Model, they are ignored in AML. Note, however, that AML can specify default values (with tighter semantics) in an archetype.

All other UML::Property links, including aggregation, isComposite, isID, association, qualifier, opposite, defaultVelue, redefines, subsettedProperty, and interface are ignored within the AML profile.

• public object : [ObjectConstraint](#_aa52f11e5760ad2f47030803962bb855)[0..\*]

ObjectConstraint represents the properties and associations that are common to all types of object constraints:

Every object constraint may directly specialize at most one parent ObjectConstraint

Every AttributeConstraint is owned by exactly one ObjectConstraint

Every ObjectConstraint is referenced by exactly one AttributeConstraint, with the exception of the root Archetype definition ComplexObjectConstraint that is not owned by any referencing attribute.

The subtypes of ObjectConstraint include:

ObjectConstraintProxy - a reference to an existing NamedObjectConstraint. ObjectConstraintProxys only exist in SourceArchetypes and are replaced by a copy of their targetObject during the flattening process.

NamedObjectConstraint - the set of ObjectConstraints that reference a Reference Model Class and have node identifiers

EnumerationConstraint - constraints on the Reference Model Enumeration class

ArchetypeSlot - identifies a (constrained) slot to be filled by a separate archetype

ArchetypeRootProxy - references an archetype that constraints the type and/or attributes of a Reference Model Class and optionally fills an ArchetypeSlot defined in a parent Archetype

ComplexObjectConstraint - a constraint on the type and/or attributes of a Reference Model Class

TerminologyConstraint - constraints on the TerminologyCodeReference type

PrimitiveObjectConstraint - constraints on the set of primitive data types supplied in the reference model

**Constraints**

* **name**

[OCL]

rmAttributeName = parentProperty.name

### <Class> SingularAttributeConstraint

**Description**

An AttributeConstraint that identifies valid values for a single value instance

**Diagrams**

[Attribute Constraint References](#_39027eaec61a2eaccc1fccb451cdda98)

**Direct Known Superclasses (Generalization)**

[AttributeConstraint](#_11f887fb6f19248bf7193bca31772c05)

**Associations**

• public alternative : [ObjectConstraint](#_aa52f11e5760ad2f47030803962bb855)[0..\*]

ObjectConstraint represents the properties and associations that are common to all types of object constraints:

Every object constraint may directly specialize at most one parent ObjectConstraint

Every AttributeConstraint is owned by exactly one ObjectConstraint

Every ObjectConstraint is referenced by exactly one AttributeConstraint, with the exception of the root Archetype definition ComplexObjectConstraint that is not owned by any referencing attribute.

The subtypes of ObjectConstraint include:

ObjectConstraintProxy - a reference to an existing NamedObjectConstraint. ObjectConstraintProxys only exist in SourceArchetypes and are replaced by a copy of their targetObject during the flattening process.

NamedObjectConstraint - the set of ObjectConstraints that reference a Reference Model Class and have node identifiers

EnumerationConstraint - constraints on the Reference Model Enumeration class

ArchetypeSlot - identifies a (constrained) slot to be filled by a separate archetype

ArchetypeRootProxy - references an archetype that constraints the type and/or attributes of a Reference Model Class and optionally fills an ArchetypeSlot defined in a parent Archetype

ComplexObjectConstraint - a constraint on the type and/or attributes of a Reference Model Class

TerminologyConstraint - constraints on the TerminologyCodeReference type

PrimitiveObjectConstraint - constraints on the set of primitive data types supplied in the reference model

• public excludes : [ObjectConstraint](#_aa52f11e5760ad2f47030803962bb855)[0..\*]

ObjectConstraint represents the properties and associations that are common to all types of object constraints:

Every object constraint may directly specialize at most one parent ObjectConstraint

Every AttributeConstraint is owned by exactly one ObjectConstraint

Every ObjectConstraint is referenced by exactly one AttributeConstraint, with the exception of the root Archetype definition ComplexObjectConstraint that is not owned by any referencing attribute.

The subtypes of ObjectConstraint include:

ObjectConstraintProxy - a reference to an existing NamedObjectConstraint. ObjectConstraintProxys only exist in SourceArchetypes and are replaced by a copy of their targetObject during the flattening process.

NamedObjectConstraint - the set of ObjectConstraints that reference a Reference Model Class and have node identifiers

EnumerationConstraint - constraints on the Reference Model Enumeration class

ArchetypeSlot - identifies a (constrained) slot to be filled by a separate archetype

ArchetypeRootProxy - references an archetype that constraints the type and/or attributes of a Reference Model Class and optionally fills an ArchetypeSlot defined in a parent Archetype

ComplexObjectConstraint - a constraint on the type and/or attributes of a Reference Model Class

TerminologyConstraint - constraints on the TerminologyCodeReference type

PrimitiveObjectConstraint - constraints on the set of primitive data types supplied in the reference model

• public constrains : [RMMSingularProperty](#_5917d2795b1a9ae4f33929e6edb8af81)[1]

*RMMSingularProperty* represents the subset of *RMMProperty* instances having an upper bound of 1. The AML treats RMMSingular properties as single values (vs. collections) of attributes that can be required, optional or prohibited.

**Constraints**

* **existence**

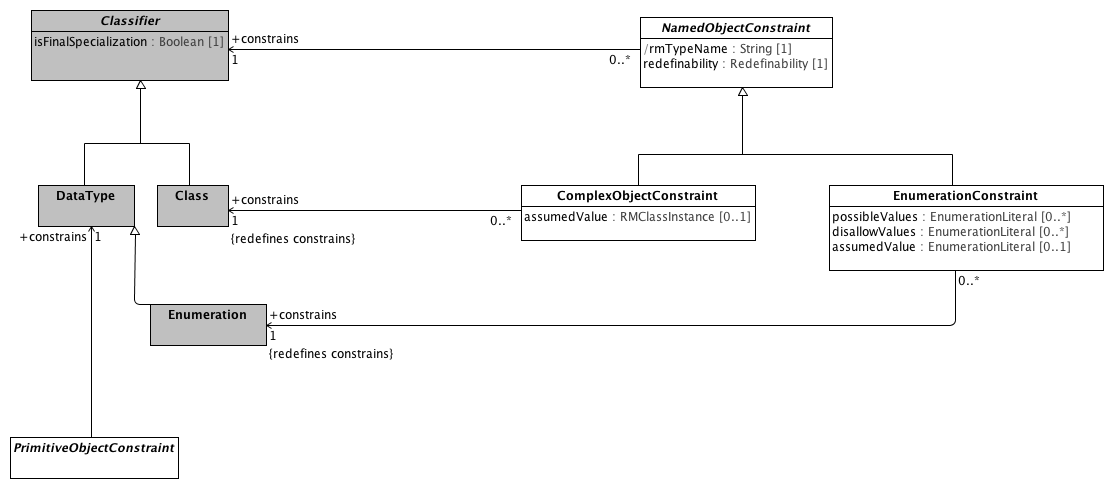
[OCL]

parentProperty.lower = 1 implies existence = AttributeExistence::REQUIRED

**Known other classes**

[Property](#_652433ba6af347b5ab3d4b0b4b2931c9), [RMMCollectionProperty](#_702e30cd0381b6e9726bcc6fe779a70f), [RMMSingularProperty](#_5917d2795b1a9ae4f33929e6edb8af81)

### <Package> Object Constraint References



**Object Constraint References**

### <Class> EnumerationConstraint

**Diagrams**

[Object Constraint References](#_960b4dbbd4fb12cd1ac74b16e654b984)

**Direct Known Superclasses (Generalization)**

[NamedObjectConstraint](#_ab5b3b01964560abb1047edd9efa4eb9), [ObjectConstraint](#_aa52f11e5760ad2f47030803962bb855)

**Direct Known Subclasses (Specialization)**

[LocalEnumerationConstraint](#_fac1fd23e79b0fc3d709d92006b38e40)

**Attributes**

• public possibleValues : [EnumerationLiteral](#_41faf6a7041d7068bbbf4f9ff3924d22) [0..\*]

The set of possible enumeration literals that are valid in the constrained instance. If *possibleValues* is empty, all literals not referenced in *disallowValues* are valid.

• public disallowValues : [EnumerationLiteral](#_41faf6a7041d7068bbbf4f9ff3924d22) [0..\*]

The set of enumeration literals that can't appear in this constrained instance.

• public assumedValue : [EnumerationLiteral](#_41faf6a7041d7068bbbf4f9ff3924d22) [0..1]

**Associations**

• public constrains : [Enumeration](#_190e24bd48f094ad9ad981ac0b4eb47e)[1]

Enumeration is a kind of DataType. Each value of an Enumeration corresponds to one of its user-defined EnumerationLiterals.

**Constraints**

* **pvValues**

The list of possible values must be a subset of the set of possible enumeration literals.

[OCL]

self.possibleValues->asSet()->forAll(d | self.parentClass.ownedLiteral->exists(p | p=d))

* **dvValues**

The list of disallowed values must be in the set of enumeration literals.

[OCL]

self.disallowValues->asSet()->forAll(d | self.parentClass.ownedLiteral->exists(p | p=d))

* **pORd**

An constraint may either specify possible values or disallow values but not both.

[OCL]

possibleValues->size() = 0 or disallowValues->size() = 0

### <Class> NamedObjectConstraint

**Description**

Abstract model of constraint on any kind of object node

**Diagrams**

[Object Constraint References](#_960b4dbbd4fb12cd1ac74b16e654b984)

**Direct Known Superclasses (Generalization)**

[ObjectConstraint](#_aa52f11e5760ad2f47030803962bb855)

**Direct Known Subclasses (Specialization)**

[ArchetypeRootProxy](#_12e58855caae51d65fb43e2837534f63), [ArchetypeSlot](#_e518b2b75b6f66417345772b8440e6f2), [ComplexObjectConstraint](#_abfab8c8e983a73b4981f6fcfdd16134), [EnumerationConstraint](#_42c2e4f902eddd2a1629a431a96cd94f)

**Attributes**

• public rmTypeName : String [1]

Reference model type that this node constrains

• public redefinability : [Redefinability](#_45bc3b03e253b26272fb450b2c34f5f2) [1]

Whether this node can be further constrained or elaborated in specializations

**Associations**

• public constrains : [Classifier](#_31f3bed9860f1a34043799bd12ffe873)[1]

A Classifier has a set of Features, some of which are Properties called the attributes of the Classifier.

### <Class> PrimitiveObjectConstraint

**Description**

A constraint on an instance of a primitive data type (see: [Primitive Data Types package](platform:/resource/metamodel/am.emx#_-pgJIByrEeONZZvjZFK4_A)) a Terminology Code Reference (See: [Core package of the Terminology Services module](platform:/resource/metamodel/am.emx#_FkmfQJgMEeOEysZ5-LoitA)) or an RMMEnumeration as defined in the [Enumeration Metamodel](platform:/resource/metamodel/am.emx#_BgPdsJW2EeOEysZ5-LoitA).

**Diagrams**

[Object Constraint References](#_960b4dbbd4fb12cd1ac74b16e654b984)

**Direct Known Superclasses (Generalization)**

[ObjectConstraint](#_aa52f11e5760ad2f47030803962bb855)

**Direct Known Subclasses (Specialization)**

[BooleanConstraint](#_72963a51b64d10f1a0ab72b99a7d95f7), [DateConstraint](#_05406539ea2335c7e43c699988cdb385), [DateTimeConstraint](#_2e2f2a2affb8ef26918289abc068cad2), [DurationConstraint](#_ac453484c1a5116843147a38bd4020c0), [IntegerConstraint](#_05ad521706377b116680c20824646d90), [RealConstraint](#_c4e09895097a057d7e6ce4b0d10c8967), [StringConstraint](#_e78c0feb207cbea2ca9911ec94e2a83e), [TimeConstraint](#_20b17cf4d1a1f7228a809f6ed68b3a0c)

**Associations**

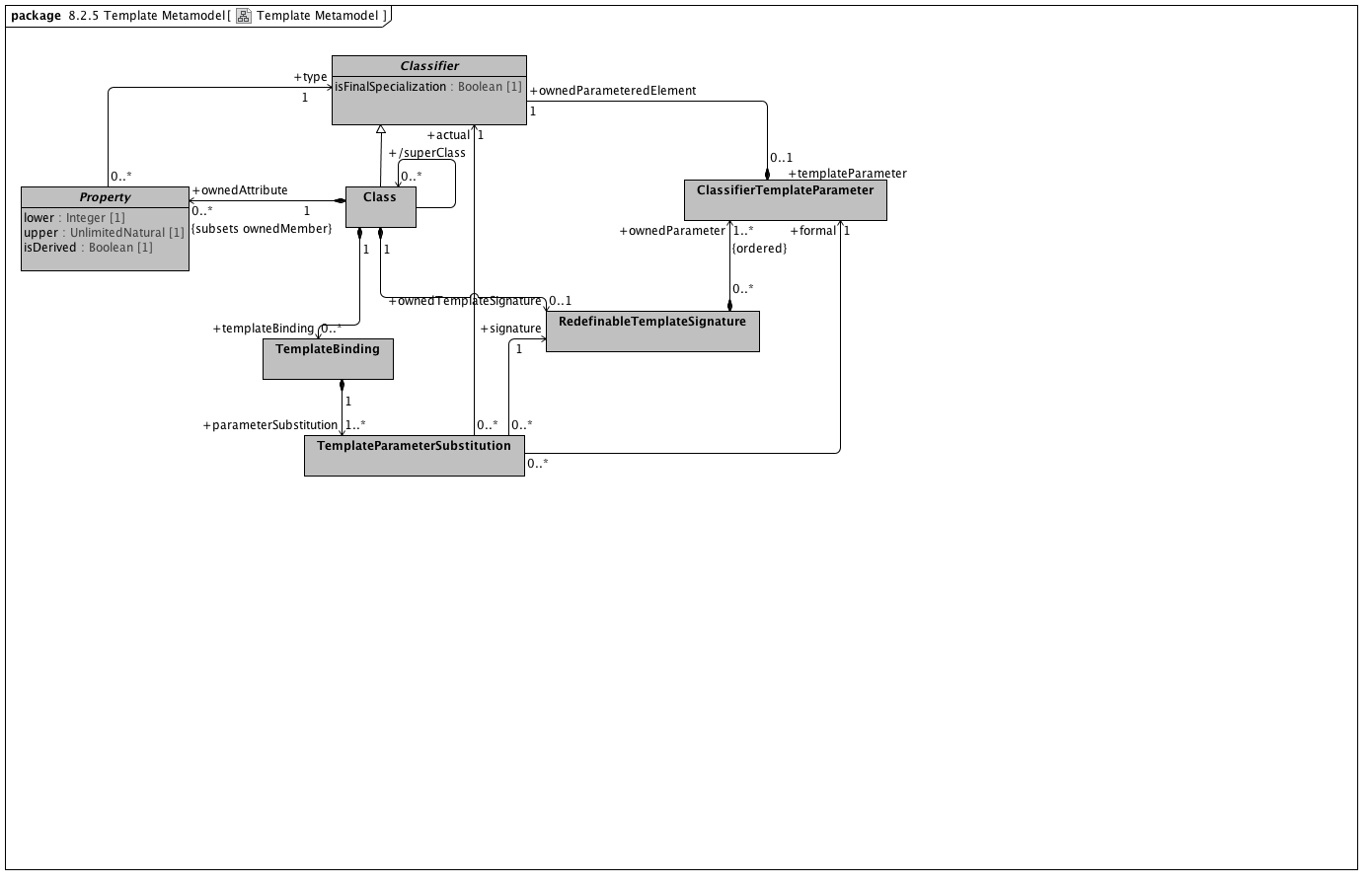
• public constrains : [DataType](#_d5914eb0da42172989bbe57f23fc4310)[1]

DataTypes model Types whose instances are distinguished only by their value.

**Known other classes**

[Class](#_a75c06fc93e516ccf92a1e38e18c46f3), [Classifier](#_31f3bed9860f1a34043799bd12ffe873), [ComplexObjectConstraint](#_abfab8c8e983a73b4981f6fcfdd16134), [DataType](#_d5914eb0da42172989bbe57f23fc4310), [Enumeration](#_190e24bd48f094ad9ad981ac0b4eb47e)

### <Package> Template Metamodel



**Template Metamodel**

### <Class> ClassifierTemplateParameter

**Diagrams**

[Template Metamodel](#_d132633918c4cbb0c3c69e16de3df9c2)

**Associations**

• public ownedParameteredElement : [Classifier](#_31f3bed9860f1a34043799bd12ffe873)[1]

A Classifier has a set of Features, some of which are Properties called the attributes of the Classifier.

### <Class> RedefinableTemplateSignature

**Description**

A *RMRedefinableTemplateSignature* associates an ordered list of *RMClassifierTemplateParameters* with an owning *RMClass.* The owning *RMClass* typically, but not always has one or more ownedAttributes that reference one of the ownedParameters of the *RMRedefinableTemplateSignature.*

**Diagrams**

[Template Metamodel](#_d132633918c4cbb0c3c69e16de3df9c2)

**Associations**

• public ownedParameter : [ClassifierTemplateParameter](#_3d9b09fe9052c8305d90ab92bc37d26b)[1..\*]

### <Class> TemplateBinding

**Description**

*RMTemplateBinding* is a subtype of the UML::TemplateBinding class. It represents a set of parameter substitutions that are to be applied to a *RMRedefinableTemplateSignature* defined by a parent or ancestor *superClass.* A *RMTemplateBinding* contains one or more parameter substitutions to be applied to one or more types referenced by an *ownedAttribute* of the parent or ancestor class.

**Diagrams**

[Template Metamodel](#_d132633918c4cbb0c3c69e16de3df9c2)

**Associations**

• public parameterSubstitution : [TemplateParameterSubstitution](#_c2a122fef357367888fcb3768852586c)[1..\*]

A *RMTemplateParameterSubstitution* indicates that the *actual RMClass* or *RMPrimitiveType* is to be substituted as the *type* for the *ownedParameteredElement* owned by the *formal* *RMClassifierTemplateParameter.*

### <Class> TemplateParameterSubstitution

**Description**

A *RMTemplateParameterSubstitution* indicates that the *actual RMClass* or *RMPrimitiveType* is to be substituted as the *type* for the *ownedParameteredElement* owned by the *formal* *RMClassifierTemplateParameter.*

**Diagrams**

[Template Metamodel](#_d132633918c4cbb0c3c69e16de3df9c2)

**Associations**

• public formal : [ClassifierTemplateParameter](#_3d9b09fe9052c8305d90ab92bc37d26b)[1]

• public signature : [RedefinableTemplateSignature](#_6afdd25f5589999ef5ae78a4eab8563d)[1]

A *RMRedefinableTemplateSignature* associates an ordered list of *RMClassifierTemplateParameters* with an owning *RMClass.* The owning *RMClass* typically, but not always has one or more ownedAttributes that reference one of the ownedParameters of the *RMRedefinableTemplateSignature.*

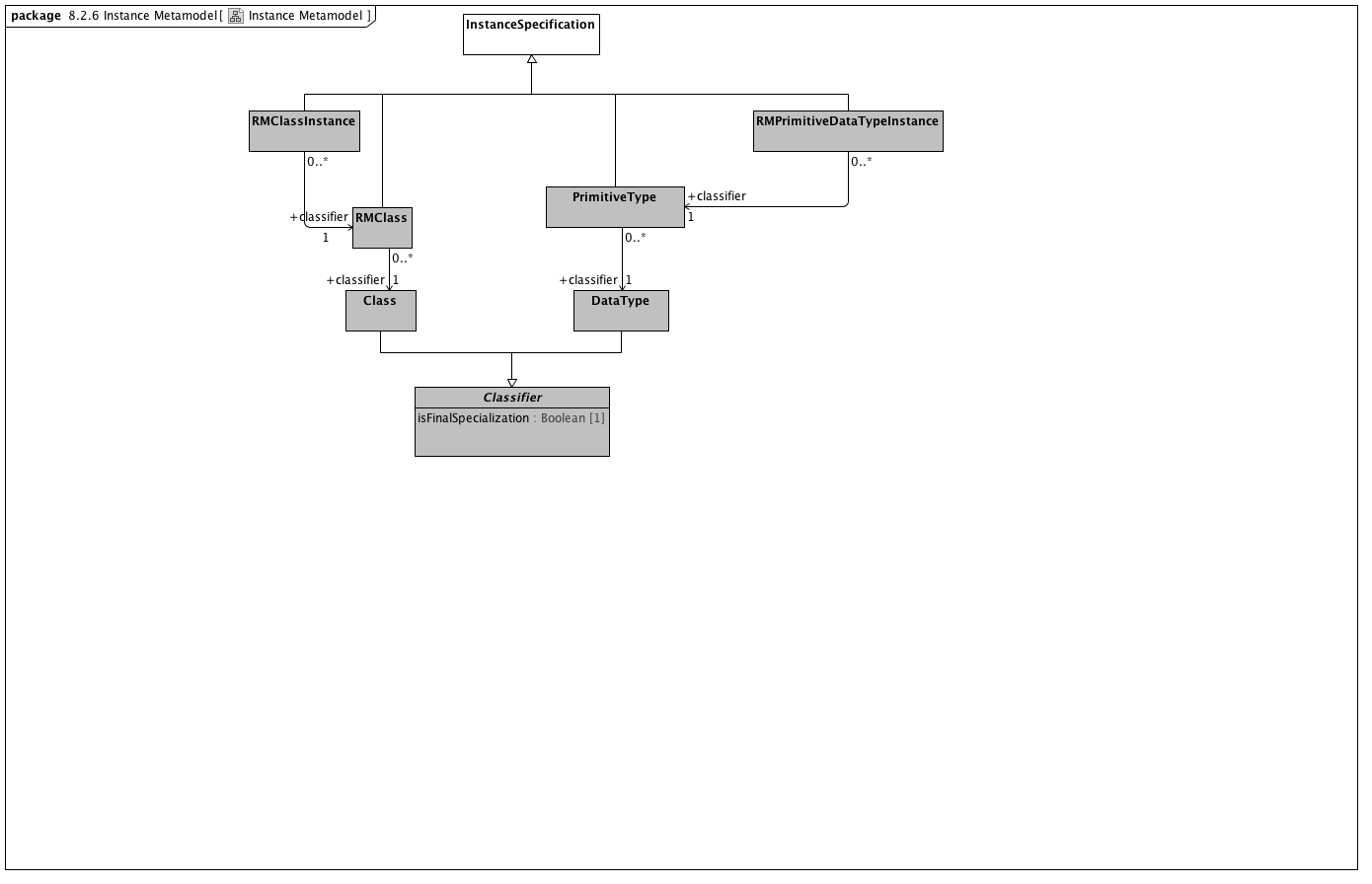
• public actual : [Classifier](#_31f3bed9860f1a34043799bd12ffe873)[1]

A Classifier has a set of Features, some of which are Properties called the attributes of the Classifier.

**Known other classes**

[Class](#_a75c06fc93e516ccf92a1e38e18c46f3), [Classifier](#_31f3bed9860f1a34043799bd12ffe873), [Property](#_652433ba6af347b5ab3d4b0b4b2931c9)

### <Package> Instance Metamodel



**Instance Metamodel**

### <Class> InstanceSpecification

**Diagrams**

[Instance Metamodel](#_4e8cc50476c42b847d941a87dd3d1d98)

**Direct Known Subclasses (Specialization)**

[PrimitiveType](#_2fd719f1d4a4fafd390224b10cd96510), [RMClass](#_977fccbc34231ad0bea8fc5ff3c8addc), [RMClassInstance](#_935cd7de4b22d47dd2c6aef93bed5c7a), [RMPrimitiveDataTypeInstance](#_44fd267adf8202d91aafd96398da0a13)

### <Class> PrimitiveType

**Description**

A PrimitiveType defines a predefined DataType, without any substructure

**Diagrams**

[Instance Metamodel](#_4e8cc50476c42b847d941a87dd3d1d98)

**Direct Known Superclasses (Generalization)**

[InstanceSpecification](#_14aa28a97b7753c26b32de25fe333e41)

**Associations**

• public classifier : [DataType](#_d5914eb0da42172989bbe57f23fc4310)[1]

DataTypes model Types whose instances are distinguished only by their value.

### <Class> RMClass

**Description**

An instance of *an instance* of an *RMClass.* As an example, if the RMClass "Automobile" were an instance of an RMClass, with the ownedAttributes "model" and "year", an RMClassInstance might be named "Ford", with the model attribute set to "Fairlane" and the year to "1965".

**Diagrams**

[Instance Metamodel](#_4e8cc50476c42b847d941a87dd3d1d98)

**Direct Known Superclasses (Generalization)**

[InstanceSpecification](#_14aa28a97b7753c26b32de25fe333e41)

**Associations**

• public classifier : [Class](#_a75c06fc93e516ccf92a1e38e18c46f3)[1]

The purpose of a Class is to specify a classification of objects and to specify the Features that characterize the structure and behavior of those objects.

### <Class> RMClassInstance

**Diagrams**

[Instance Metamodel](#_4e8cc50476c42b847d941a87dd3d1d98)

**Direct Known Superclasses (Generalization)**

[InstanceSpecification](#_14aa28a97b7753c26b32de25fe333e41)

**Associations**

• public classifier : [RMClass](#_977fccbc34231ad0bea8fc5ff3c8addc)[1]

An instance of *an instance* of an *RMClass.* As an example, if the RMClass "Automobile" were an instance of an RMClass, with the ownedAttributes "model" and "year", an RMClassInstance might be named "Ford", with the model attribute set to "Fairlane" and the year to "1965".

### <Class> RMPrimitiveDataTypeInstance

**Diagrams**

[Instance Metamodel](#_4e8cc50476c42b847d941a87dd3d1d98)

**Direct Known Superclasses (Generalization)**

[InstanceSpecification](#_14aa28a97b7753c26b32de25fe333e41)

**Associations**

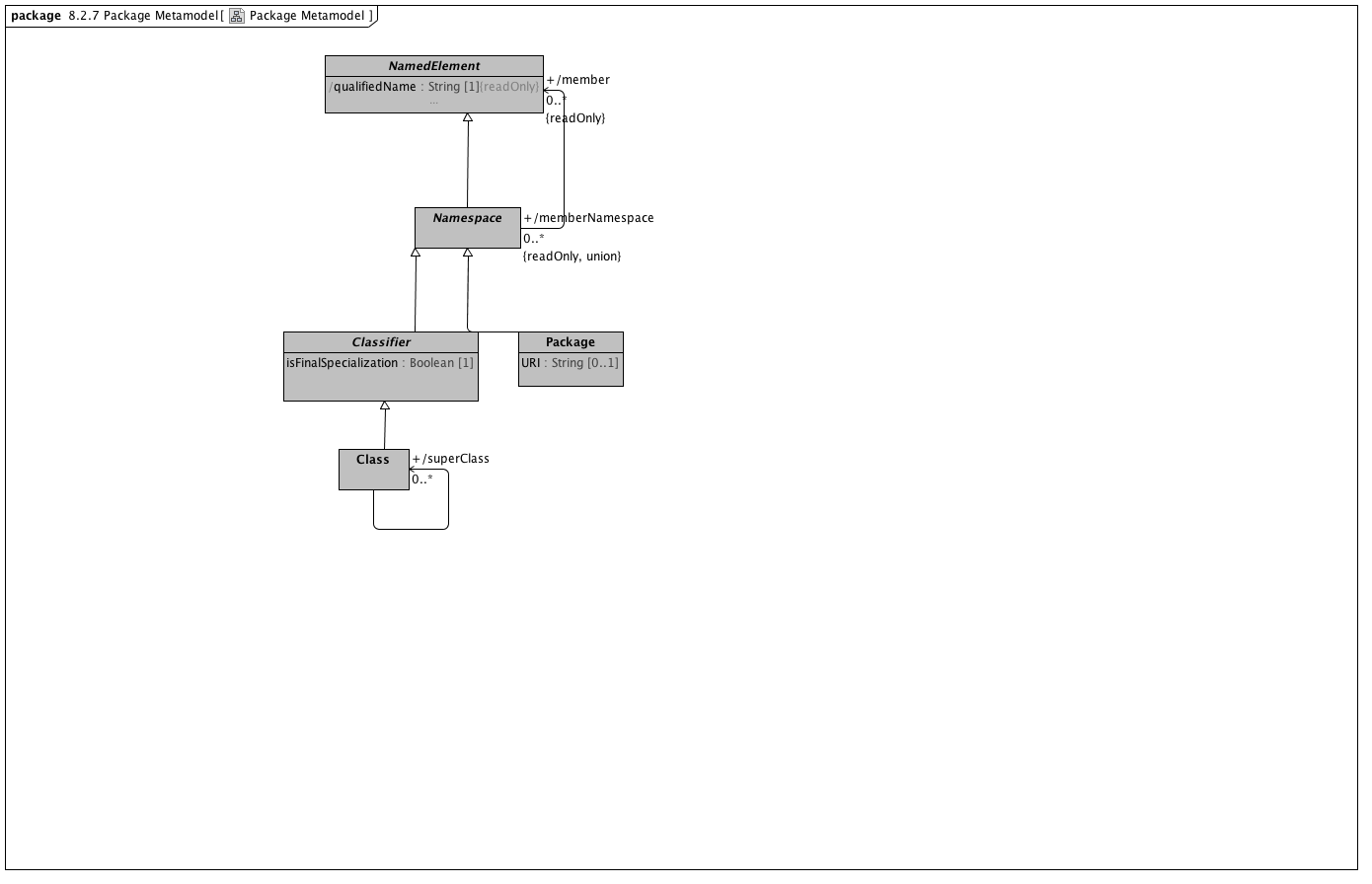
• public classifier : [PrimitiveType](#_2fd719f1d4a4fafd390224b10cd96510)[1]

A PrimitiveType defines a predefined DataType, without any substructure

**Known other classes**

[Class](#_a75c06fc93e516ccf92a1e38e18c46f3), [Classifier](#_31f3bed9860f1a34043799bd12ffe873), [DataType](#_d5914eb0da42172989bbe57f23fc4310)

### <Package> Package Metamodel

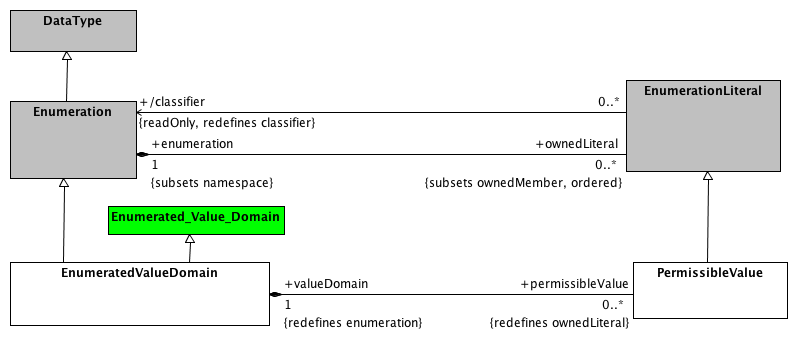


**Package Metamodel**

**Known other classes**

[Class](#_a75c06fc93e516ccf92a1e38e18c46f3), [Classifier](#_31f3bed9860f1a34043799bd12ffe873), [NamedElement](#_527fd9eb1e787c36a3748854a9431816), [Namespace](#_f762e4ef59f1948849a49d421126c16b), [Package](#_a0a843d7d41881592e31e887cebd6da4)

### <Package> Enumeration Metamodel



**Enumeration Metamodel**

### <Class> EnumeratedValueDomain

**Description**

An EnumeratedValueDomain represents a discrete set of possible values for a particular field or data element. Each permissible value represents an intended meaning that, while sometimes determinable from the string itself or its accompanying documentation, can only be fully fixed by connecting it to an official "value meaning" reference in an external terminological resource.

**Diagrams**

[AML DescribedItems](#_e01fdbb84f109fa2531577050a9e4ad8), [Enumeration Metamodel](#_c323459faa5aa97d8abc3c64dcd86661)

**Direct Known Superclasses (Generalization)**

[DescribedItem](#_4cfed31aa26e11dff4000228bcd222ae), [Enumerated\_Value\_Domain](#_f7ddc2c93ef0e7318e8f00f08a71d10c), [Enumeration](#_190e24bd48f094ad9ad981ac0b4eb47e)

**Direct Known Subclasses (Specialization)**

[ValueSet](#_73d0fab5bddf198ab14a77c3fed1636a)

**Attributes**

• public p1

**Associations**

• public permissibleValue : [PermissibleValue](#_66976d5fcaf3eff9df49b6e5dab4ad12)[0..\*]

A permissible value within the context of a value domain. While permissible values may be represented as integers, strings or simply as named data type instances (as is the case in UML), all permissible values need to have a mechanism for providing a String representation of the represented value. The String returned by the value function must be unique within the context of the containing domain.

• public valueDomainMeaning : [ValueSetAndDefinitionReference](#_25ca98283400275c47128a3c11af1824)[1]

A reference to a value set and optional definition.

### <Class> Enumerated\_Value\_Domain

**Description**

a class each instance of which models an enumerated value domain, a value domain that is specified by a list of all its permissible values.

**Diagrams**

[Enumeration Metamodel](#_c323459faa5aa97d8abc3c64dcd86661)

**Direct Known Superclasses (Generalization)**

[Value\_Domain](#_9b81960c26ee442be19393f5a5fc3ef2)

**Direct Known Subclasses (Specialization)**

[EnumeratedValueDomain](#_ad639ee3d4cd535b2d3e55238d69cc51)

**Associations**

• public member : [Permissible\_value](#_2fd4e293390c50189eca5b7ab34d63cd)[0..\*]

a class each instance of which models a permissible value.

**Note:** The AML model views Enumerated\_Value\_Domains as a extension to the UML Enumeration DataType and Permissible\_values as Enumeration\_Literals. The notion of begin\_date becomes an aspect of the model (version) itself rather than the date when the specific member "became valid". As a result, the model will assume that begin\_date is assigned a date that is sufficiently far in the past that we do not have to concern ourselves with it. In addition, permissible values will be removed from the accompanying domain, so end\_date will not be populated.

### <Class> EnumerationLiteral

**Description**

An EnumerationLiteral defines an element of the run-time extension of an Enumeration. Values corresponding to EnumerationLiterals are immutable and may be compared for equality. EnumerationLiterals may not change during their existence, so any attributes on an Enumeration shall be read-only.

An EnumerationLiteral has a name that shall be used to identify it within its Enumeration. The EnumerationLiteral name is scoped within and shall be unique within its Enumeration. EnumerationLiteral names shall be qualified for general use.

**Diagrams**

[Enumeration Metamodel](#_c323459faa5aa97d8abc3c64dcd86661)

**Direct Known Superclasses (Generalization)**

[NamedElement](#_527fd9eb1e787c36a3748854a9431816)

**Direct Known Subclasses (Specialization)**

[PermissibleValue](#_66976d5fcaf3eff9df49b6e5dab4ad12)

**Associations**

• public enumeration : [Enumeration](#_190e24bd48f094ad9ad981ac0b4eb47e)[1]

Enumeration is a kind of DataType. Each value of an Enumeration corresponds to one of its user-defined EnumerationLiterals.

• public classifier : [Enumeration](#_190e24bd48f094ad9ad981ac0b4eb47e)

Enumeration is a kind of DataType. Each value of an Enumeration corresponds to one of its user-defined EnumerationLiterals.

### <Class> PermissibleValue

**Description**

A permissible value within the context of a value domain. While permissible values may be represented as integers, strings or simply as named data type instances (as is the case in UML), all permissible values need to have a mechanism for providing a String representation of the represented value. The String returned by the value function must be unique within the context of the containing domain.

**Diagrams**

[AML DescribedItems](#_e01fdbb84f109fa2531577050a9e4ad8), [Enumeration Metamodel](#_c323459faa5aa97d8abc3c64dcd86661)

**Direct Known Superclasses (Generalization)**

[DescribedItem](#_4cfed31aa26e11dff4000228bcd222ae), [EnumerationLiteral](#_41faf6a7041d7068bbbf4f9ff3924d22)

**Associations**

• public valueDomain : [EnumeratedValueDomain](#_ad639ee3d4cd535b2d3e55238d69cc51)[1]

An EnumeratedValueDomain represents a discrete set of possible values for a particular field or data element. Each permissible value represents an intended meaning that, while sometimes determinable from the string itself or its accompanying documentation, can only be fully fixed by connecting it to an official "value meaning" reference in an external terminological resource.

• public classifier : [EnumeratedValueDomain](#_ad639ee3d4cd535b2d3e55238d69cc51)[1]

An EnumeratedValueDomain represents a discrete set of possible values for a particular field or data element. Each permissible value represents an intended meaning that, while sometimes determinable from the string itself or its accompanying documentation, can only be fully fixed by connecting it to an official "value meaning" reference in an external terminological resource.

• valueMeaning : [ConceptReference](#_4082bf99060eea349dab8f548d297f3d)[0..1]

A URI that uniquely identifies a "concept" (aka. class, entity, individual or, in some contexts "term"), accompanied by additional information that conveys the intended meaning, code and source of the information used to determine the intent of the URI.

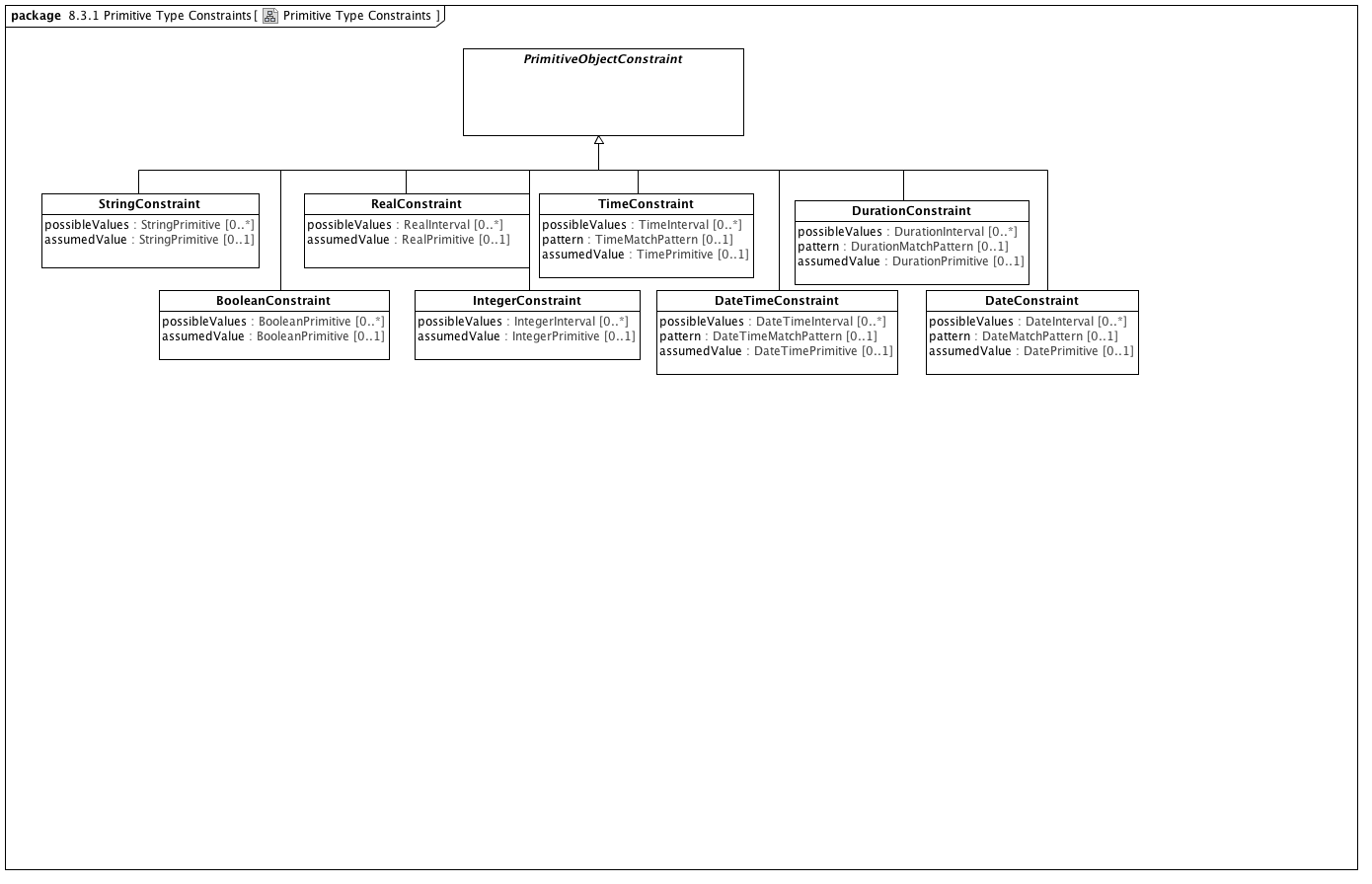
**Known other classes**

[DataType](#_d5914eb0da42172989bbe57f23fc4310), [Enumeration](#_190e24bd48f094ad9ad981ac0b4eb47e)

## <Package> Constraint Object Model

Constraint Model Package Overview The constraint model is the core of the archetype design. It illustrates how constraints are defined, showing the object-attribute-object pattern characteristic of object constraints. ComplexObjectConstraint. Because objects are composed of properties (attributes and relationships), and properties consist of objects, the archetype definition consists of alternate layers of ArchetypeRootConstraint, but rather than a single archetype, it defines a set of archetypes. It can be thought of like a keyhole, into which few or many keys might fit, depending on how specific its shape is. Logically it has the same semantics as a ComplexObjectConstraint, except that the constraints are expressed in another archetype, not the current one. ComplexObjectConstraints PrimitiveObjectConstraints AttributeConstraints SingularAttributeConstraint class. Where a SingularAttributeConstraint is associated with more than one ObjectConstraint, the ObjectConstraints are alternatives. AttributeCollectionConstraint, which differentiates between unique and repeatable and between ordered and unordered collections. In addition, while the AttributeConstraint determines whether a property may exist, the quantity of a repeating element is defined in the AttributeCollectionConstraint’s cardinality property. AttributeCollectionConstraint, with its defined cardinality, there may be different sets of sibling members with different constraints, and the number of each of these subsets is specified as the AttributeCollectionMember’s occurrences property. In an organization, for instance, the cardinality for the member property may be “two or more,” but within that set of members, we may have two constraints. One type of member, the leader (indicated by an ObjectConstraint on the person type or role), may be required to occur exactly once, whereas other types may have multiple occurrences. AttributeTupleConstraints AttributeTuple would be defined for each pair of the two values (unit code and numeric ceiling), and these tuples would be grouped into an AttributeTupleConstraint, defining an array of acceptable sets of values.

### <Package> Primitive Type Constraints



**Primitive Type Constraints**

### <Class> BooleanConstraint

**Description**

A *BooleanConstraint* restricts the possible values of a target *BooleanPrimitive* data type.

**Diagrams**

[Primitive Type Constraints](#_2ca64b6794a95c8188c5478872196c54)

**Direct Known Superclasses (Generalization)**

[PrimitiveObjectConstraint](#_db9df3b10e304d809393da4afc9a91da)

**Attributes**

• public possibleValues : [BooleanPrimitive](#_19737cdaaaee2179c50553b52361808b) [0..\*]

List of valid Boolean values for this constraint

• public assumedValue : [BooleanPrimitive](#_19737cdaaaee2179c50553b52361808b) [0..1]

### <Class> DateConstraint

**Description**

A *DateConstraint* restricts the possible values of a target *DatePrimitive* data type.

**Diagrams**

[Primitive Type Constraints](#_2ca64b6794a95c8188c5478872196c54)

**Direct Known Superclasses (Generalization)**

[PrimitiveObjectConstraint](#_db9df3b10e304d809393da4afc9a91da)

**Attributes**

• public possibleValues : [DateInterval](#_69ee53d558bd7c80a52df2f5ae7c8ab0) [0..\*]

• public pattern : [DateMatchPattern](#_3cae9b111208dfafe95b548c45720f8e) [0..1]

• public assumedValue : [DatePrimitive](#_db5d020506d0af7a330f0b4fe1cb870a) [0..1]

### <Class> DateTimeConstraint

**Description**

A *DateTimeConstraint* restricts the possible values of a target *DateTimePrimitive* data type.

**Diagrams**

[Primitive Type Constraints](#_2ca64b6794a95c8188c5478872196c54)

**Direct Known Superclasses (Generalization)**

[PrimitiveObjectConstraint](#_db9df3b10e304d809393da4afc9a91da)

**Attributes**

• public possibleValues : [DateTimeInterval](#_3145967a9ac97f161c2d4ff9e296537e) [0..\*]

Range of valid DateTime values

• public pattern : [DateTimeMatchPattern](#_b501b0aa7bbcd9ac89d6ddbf9f0bf3d9) [0..1]

A DateTimeMatchPattern to constrain valid instances of DateTime

• assumedValue : [DateTimePrimitive](#_8347879bb381db17040637cc3ba0a25c) [0..1]

### <Class> DurationConstraint

**Description**

A *DurationConstraint* restricts the possible values of a target *DurationPrimitive* data type.

**Diagrams**

[Primitive Type Constraints](#_2ca64b6794a95c8188c5478872196c54)

**Direct Known Superclasses (Generalization)**

[PrimitiveObjectConstraint](#_db9df3b10e304d809393da4afc9a91da)

**Attributes**

• public possibleValues : [DurationInterval](#_8370a7823f617661d9ddb083556dfa3f) [0..\*]

Range of valid Duration values

• private pattern : [DurationMatchPattern](#_baaa358403d3312c0779bc256e1050bd) [0..1]

A DurationMatchPattern to constrain valid instances of Duration

• public assumedValue : [DurationPrimitive](#_64b0026498682fb721ccdb38c186bba2) [0..1]

### <Class> IntegerConstraint

**Description**

An *IntegerConstraint* restricts the possible values of a target *IntegerPrimitive* data type.

**Diagrams**

[Primitive Type Constraints](#_2ca64b6794a95c8188c5478872196c54)

**Direct Known Superclasses (Generalization)**

[PrimitiveObjectConstraint](#_db9df3b10e304d809393da4afc9a91da)

**Attributes**

• possibleValues : [IntegerInterval](#_5218d9b9782bc6f925ed66b3e2bc529b) [0..\*]

• public assumedValue : [IntegerPrimitive](#_89e2b2b9de405e6d05c4c5259fc8ffd6) [0..1]

### <Class> RealConstraint

**Description**

A *RealConstraint* restricts the possible values of a target *RealPrimitive* data type.

**Diagrams**

[Primitive Type Constraints](#_2ca64b6794a95c8188c5478872196c54)

**Direct Known Superclasses (Generalization)**

[PrimitiveObjectConstraint](#_db9df3b10e304d809393da4afc9a91da)

**Attributes**

• public possibleValues : [RealInterval](#_27fcdc4a73f595dca8b4e4fa28fd59a6) [0..\*]

• public assumedValue : [RealPrimitive](#_c596f10fb93cb4f697f2f1b0b64b43ce) [0..1]

### <Class> StringConstraint

**Description**

A *StringConstraint* restricts the possible values of a target *StringPrimitive* data type.

**Diagrams**

[Primitive Type Constraints](#_2ca64b6794a95c8188c5478872196c54)

**Direct Known Superclasses (Generalization)**

[PrimitiveObjectConstraint](#_db9df3b10e304d809393da4afc9a91da)

**Attributes**

• public possibleValues : [StringPrimitive](#_6a90be7cfa784ea4b4e8cad8f4a47e82) [0..\*]

A list of valid String instances

• public assumedValue : [StringPrimitive](#_6a90be7cfa784ea4b4e8cad8f4a47e82) [0..1]

### <Class> TimeConstraint

**Description**

A *TimeConstraint* restricts the possible values of a target *TimePrimitive* data type.

**Diagrams**

[Primitive Type Constraints](#_2ca64b6794a95c8188c5478872196c54)

**Direct Known Superclasses (Generalization)**

[PrimitiveObjectConstraint](#_db9df3b10e304d809393da4afc9a91da)

**Attributes**

• public possibleValues : [TimeInterval](#_31d7d28b74d99d489c8e65e4a57e9ca9) [0..\*]

Range of valid Time values

• public pattern : [TimeMatchPattern](#_a28995c5bf2253d86854a7a97e55ba1d) [0..1]

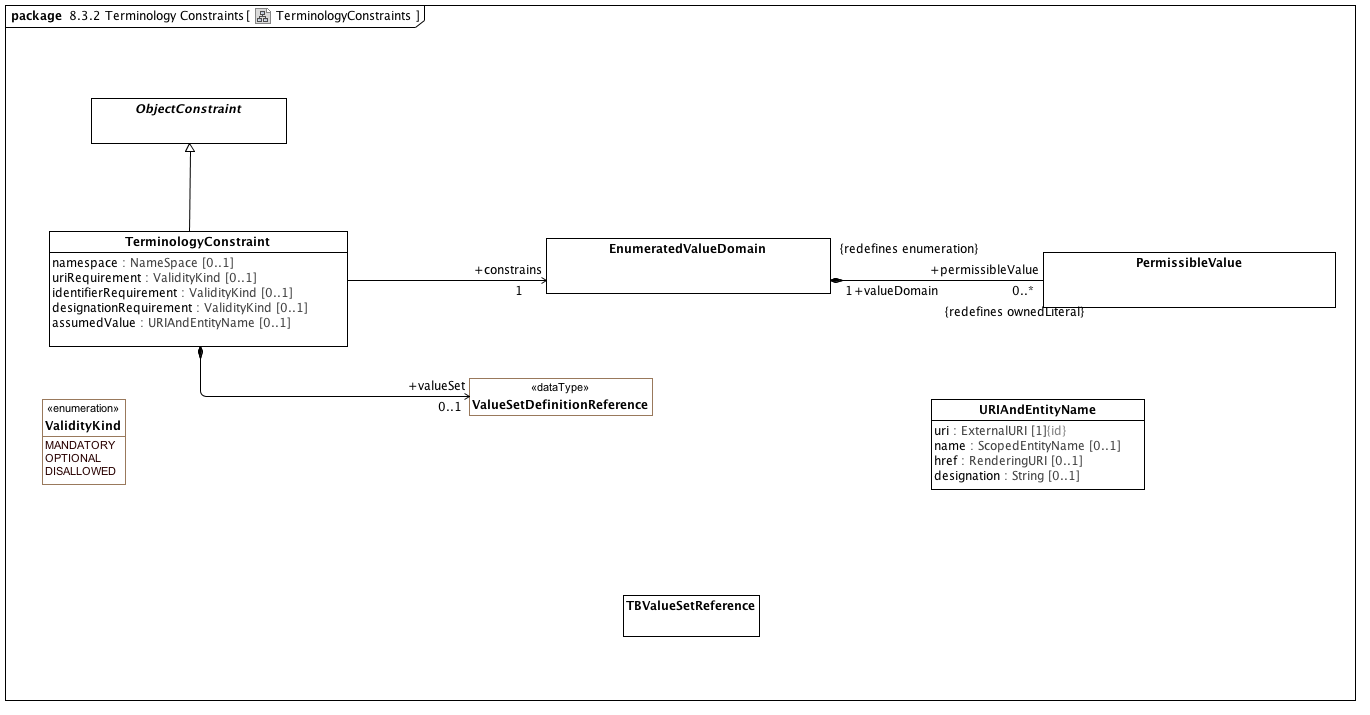
A TimeMatchPattern to constrain valid instances of Time

• assumedValue : [TimePrimitive](#_05e0f2a221f0733d693058e9253b0017) [0..1]

**Known other classes**

[PrimitiveObjectConstraint](#_db9df3b10e304d809393da4afc9a91da)

### <Package> Terminology Constraints



**TerminologyConstraints**

### <DataType> ValueSetDefinitionReference

**Description**

A reference to a set of rules for constructing a value set along with a reference to the corresponding value set if known.

**Diagrams**

[TerminologyConstraints](#_2c065eee9fec4768da07422243566f39)

**Direct Known Superclasses (Generalization)**

[NameAndMeaningReference](#_7677b94abdbbdc2c76d49314388386a7)

**Attributes**

• public valueSet : [ValueSetReference](#_5e70a8ee4206e2396a68ca679af92933) [0..1]

### <Class> ObjectConstraint

**Description**

ObjectConstraint represents the properties and associations that are common to all types of object constraints:

Every object constraint may directly specialize at most one parent ObjectConstraint

Every AttributeConstraint is owned by exactly one ObjectConstraint

Every ObjectConstraint is referenced by exactly one AttributeConstraint, with the exception of the root Archetype definition ComplexObjectConstraint that is not owned by any referencing attribute.

The subtypes of ObjectConstraint include:

ObjectConstraintProxy - a reference to an existing NamedObjectConstraint. ObjectConstraintProxys only exist in SourceArchetypes and are replaced by a copy of their targetObject during the flattening process.

NamedObjectConstraint - the set of ObjectConstraints that reference a Reference Model Class and have node identifiers

EnumerationConstraint - constraints on the Reference Model Enumeration class

ArchetypeSlot - identifies a (constrained) slot to be filled by a separate archetype

ArchetypeRootProxy - references an archetype that constraints the type and/or attributes of a Reference Model Class and optionally fills an ArchetypeSlot defined in a parent Archetype

ComplexObjectConstraint - a constraint on the type and/or attributes of a Reference Model Class

TerminologyConstraint - constraints on the TerminologyCodeReference type

PrimitiveObjectConstraint - constraints on the set of primitive data types supplied in the reference model

**Diagrams**

[AML DescribedItems](#_e01fdbb84f109fa2531577050a9e4ad8), [TerminologyConstraints](#_2c065eee9fec4768da07422243566f39)

**Direct Known Superclasses (Generalization)**

[DescribedItem](#_4cfed31aa26e11dff4000228bcd222ae)

**Direct Known Subclasses (Specialization)**

[EnumerationConstraint](#_42c2e4f902eddd2a1629a431a96cd94f), [NamedObjectConstraint](#_ab5b3b01964560abb1047edd9efa4eb9), [ObjectConstraintProxy](#_6da4a9bc7db41a2b89064f79f0c4ed36), [PrimitiveObjectConstraint](#_db9df3b10e304d809393da4afc9a91da), [TerminologyConstraint](#_b2d4edbc24f651e5a3d756933fff1326)

**Associations**

• parent : [ConceptReference](#_4082bf99060eea349dab8f548d297f3d)[0..1]

A URI that uniquely identifies a "concept" (aka. class, entity, individual or, in some contexts "term"), accompanied by additional information that conveys the intended meaning, code and source of the information used to determine the intent of the URI.

• public about : [ConceptReference](#_4082bf99060eea349dab8f548d297f3d)[0..1]

A URI that uniquely identifies a "concept" (aka. class, entity, individual or, in some contexts "term"), accompanied by additional information that conveys the intended meaning, code and source of the information used to determine the intent of the URI.

### <Class> TBValueSetReference

**Description**

The URI, identifier and name of a collection of TerminologyCodeReferences

**Diagrams**

[TerminologyConstraints](#_2c065eee9fec4768da07422243566f39)

**Associations**

• public binding : [ValueSetBinding](#_fdb7a97d4c43d7f09387cdc69dd2f65a)[0..1]

An externally specified set of coded values

### <Class> TerminologyConstraint

**Description**

A constraint on instances of the reference model Terminology type

**Diagrams**

[TerminologyConstraints](#_2c065eee9fec4768da07422243566f39)

**Direct Known Superclasses (Generalization)**

[ObjectConstraint](#_aa52f11e5760ad2f47030803962bb855)

**Attributes**

• public namespace : [NameSpace](#_b448a755fcce2af9da8221a55b6b53ff) [0..1]

• public uriRequirement : [ValidityKind](#_411de44f0fb7bb89a16c1f6c35fcd7d9) [0..1]

Whether a URI is required

• public identifierRequirement : [ValidityKind](#_411de44f0fb7bb89a16c1f6c35fcd7d9) [0..1]

Whether a concept identifier is required

• public designationRequirement : [ValidityKind](#_411de44f0fb7bb89a16c1f6c35fcd7d9) [0..1]

Whether a human-readable designation is required

• public assumedValue : [URIAndEntityName](#_4d1f571ab5e9384786ffe39444e822b4) [0..1]

**Associations**

• public valueSet : [ValueSetDefinitionReference](#_bf76d842365737931f787efdb4a36f12)[0..1]

A reference to a set of rules for constructing a value set along with a reference to the corresponding value set if known.

• public constrains : [EnumeratedValueDomain](#_ad639ee3d4cd535b2d3e55238d69cc51)[1]

An EnumeratedValueDomain represents a discrete set of possible values for a particular field or data element. Each permissible value represents an intended meaning that, while sometimes determinable from the string itself or its accompanying documentation, can only be fully fixed by connecting it to an official "value meaning" reference in an external terminological resource.

### <Class> URIAndEntityName

**Description**

A TerminologyCodeReference </i>(alias: URIAndEntityName) consists of a local identifier that references a unique meaning within the context of a given domain in a terminology service instance and a globally unique <i>URI </i>that identifies the intended meaning of the identifier.</p>

**Diagrams**

[TerminologyConstraints](#_2c065eee9fec4768da07422243566f39)

**Direct Known Subclasses (Specialization)**

[ConceptReference](#_4082bf99060eea349dab8f548d297f3d)

**Attributes**

• public uri : [ExternalURI](#_de932b9629138c166e8cfb00efa65177) [1]

The URI that represents this particular referenced entity. Note: We have added the UML {id} attribute to this attribute for clarity.

• public name : [ScopedEntityName](#_bf3eeb4d95f5d93bbd59440cca5ed9d6) [0..1]

A namespace/name combination that uniquely represents the entity. This can be the primary entityID, as determined by the service or any valid alternateId. Service implementers are encouraged to develop mechanisms that will allow clients to choose an appropriate namespace for rendering URIAndEntityName instances. As an example, it should be possible to view SNOMED-CT entity references by either the SctId, the “fully specified name” or, where appropriate, the CTV3ID or SNOMED-3 identifier. Similar mechanisms would apply to ontologies that have both id and label fields.

• public href : [RenderingURI](#_821273fdc1c3295a17225200782229ea) [0..1]

A URI that, when resolved, will return a CTS2 compliant description of the resource referenced by the uri attribute.

• public designation : String [0..1]

A string that represents the target of the reference in a given language and context. The designation attribute is strictly informative and does not participate in or contribute to the identity of the target resource.

**Constraints**

* **identifierOrURI**

A TerminologyCodeReference must either have a *uri*, an *identifier*, or both.

[OCL]

uri->notEmpty() or identifier->notEmpty()

**Known other classes**

[EnumeratedValueDomain](#_ad639ee3d4cd535b2d3e55238d69cc51), [PermissibleValue](#_66976d5fcaf3eff9df49b6e5dab4ad12)

### <Enumeration> ValidityKind

**Description**

Enumeration of classes of permissibility

**Diagrams**

[TerminologyConstraints](#_2c065eee9fec4768da07422243566f39)

**Enumeration Literals**

* **DISALLOWED**

The property must not be present in a conformant instance

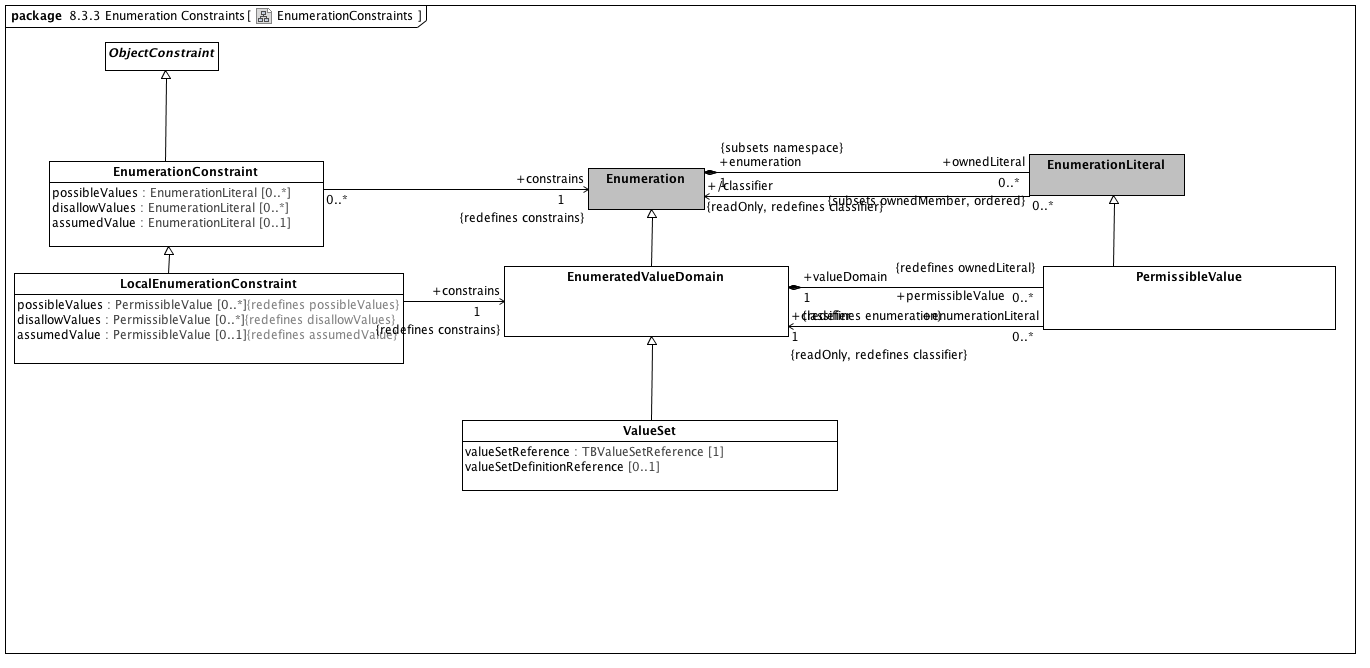
* **MANDATORY**

The property must be present in a conformant instance

* **OPTIONAL**

The property may be present in a conformant instance

### <Package> Enumeration Constraints



**EnumerationConstraints**

### <Class> LocalEnumerationConstraint

**Diagrams**

[EnumerationConstraints](#_9a0e8e1c8e8cdbf68da35592986a39eb)

**Direct Known Superclasses (Generalization)**

[EnumerationConstraint](#_42c2e4f902eddd2a1629a431a96cd94f)

**Attributes**

• public possibleValues : [PermissibleValue](#_66976d5fcaf3eff9df49b6e5dab4ad12) [0..\*]

• public disallowValues : [PermissibleValue](#_66976d5fcaf3eff9df49b6e5dab4ad12) [0..\*]

• public assumedValue : [PermissibleValue](#_66976d5fcaf3eff9df49b6e5dab4ad12) [0..1]

**Associations**

• public constrains : [EnumeratedValueDomain](#_ad639ee3d4cd535b2d3e55238d69cc51)[1]

An EnumeratedValueDomain represents a discrete set of possible values for a particular field or data element. Each permissible value represents an intended meaning that, while sometimes determinable from the string itself or its accompanying documentation, can only be fully fixed by connecting it to an official "value meaning" reference in an external terminological resource.

### <Class> ValueSet

**Diagrams**

[EnumerationConstraints](#_9a0e8e1c8e8cdbf68da35592986a39eb)

**Direct Known Superclasses (Generalization)**

[EnumeratedValueDomain](#_ad639ee3d4cd535b2d3e55238d69cc51)

**Attributes**

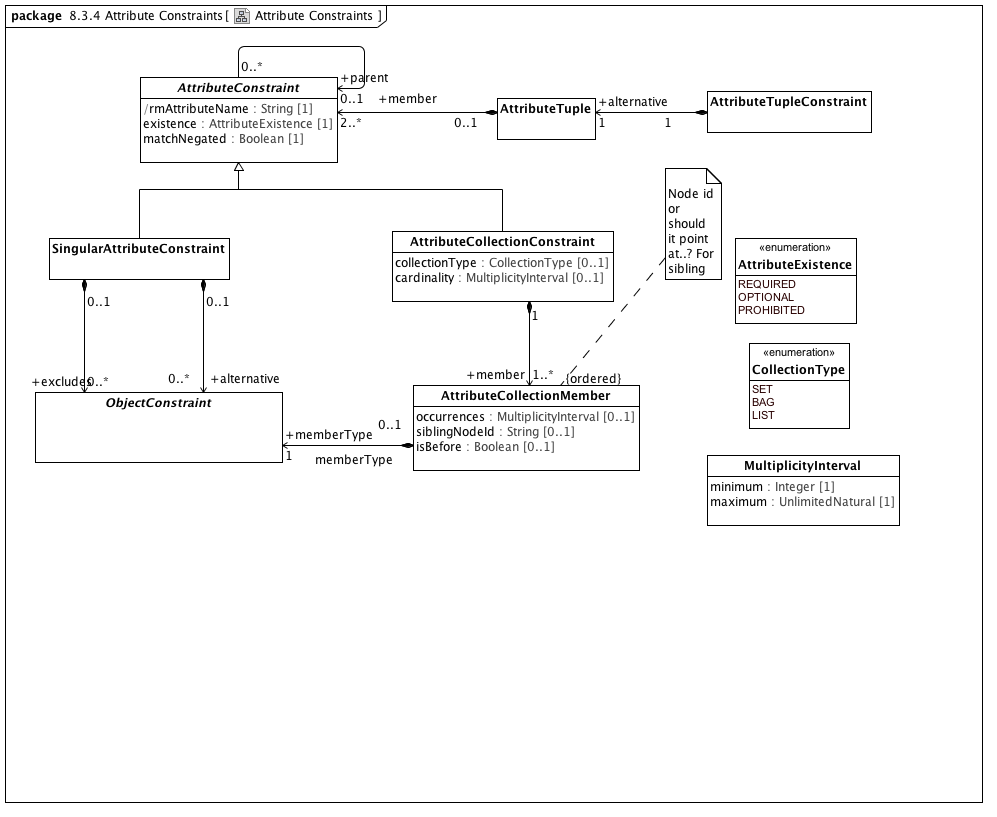
• public valueSetReference : [TBValueSetReference](#_53376ea1584b6547b15f0e1392fc93e7) [1]

• public valueSetDefinitionReference [0..1]

**Known other classes**

[EnumeratedValueDomain](#_ad639ee3d4cd535b2d3e55238d69cc51), [Enumeration](#_190e24bd48f094ad9ad981ac0b4eb47e), [EnumerationConstraint](#_42c2e4f902eddd2a1629a431a96cd94f), [EnumerationLiteral](#_41faf6a7041d7068bbbf4f9ff3924d22), [ObjectConstraint](#_aa52f11e5760ad2f47030803962bb855), [PermissibleValue](#_66976d5fcaf3eff9df49b6e5dab4ad12)

### <Package> Attribute Constraints



**Attribute Constraints**

### <Class> AttributeCollectionMember

**Description**

An association that matches members of a collection of attributes with specific ObjectConstraints

**Diagrams**

[Attribute Constraints](#_a74fdcc000d4318dcb5580a5fa8fbfce)

**Attributes**

• public occurrences : [MultiplicityInterval](#_c810ec7fa381fa249b7a7d9fecae85b6) [0..1]

Number of times the element described by this constraint can repeat

• public siblingNodeId : String [0..1]

Identification of another node under this AttributeConstraint from which *isBefore* is evaluted

• public isBefore : Boolean [0..1]

Whether an object [immediately] precedes the object named in siblingNodeId

**Associations**

• public memberType : [ObjectConstraint](#_aa52f11e5760ad2f47030803962bb855)[1]

ObjectConstraint represents the properties and associations that are common to all types of object constraints:

Every object constraint may directly specialize at most one parent ObjectConstraint

Every AttributeConstraint is owned by exactly one ObjectConstraint

Every ObjectConstraint is referenced by exactly one AttributeConstraint, with the exception of the root Archetype definition ComplexObjectConstraint that is not owned by any referencing attribute.

The subtypes of ObjectConstraint include:

ObjectConstraintProxy - a reference to an existing NamedObjectConstraint. ObjectConstraintProxys only exist in SourceArchetypes and are replaced by a copy of their targetObject during the flattening process.

NamedObjectConstraint - the set of ObjectConstraints that reference a Reference Model Class and have node identifiers

EnumerationConstraint - constraints on the Reference Model Enumeration class

ArchetypeSlot - identifies a (constrained) slot to be filled by a separate archetype

ArchetypeRootProxy - references an archetype that constraints the type and/or attributes of a Reference Model Class and optionally fills an ArchetypeSlot defined in a parent Archetype

ComplexObjectConstraint - a constraint on the type and/or attributes of a Reference Model Class

TerminologyConstraint - constraints on the TerminologyCodeReference type

PrimitiveObjectConstraint - constraints on the set of primitive data types supplied in the reference model

### <Class> AttributeTuple

**Description**

A set of constraints on related attributes to be used to differentiate scenarios where the value of one attribute affects the valid values of another

**Diagrams**

[Attribute Constraints](#_a74fdcc000d4318dcb5580a5fa8fbfce)

**Associations**

• public member : [AttributeConstraint](#_11f887fb6f19248bf7193bca31772c05)[2..\*]

A constraint on a reference model attribute

### <Class> AttributeTupleConstraint

**Description**

An AttributeTupleConstraint presents a set of two or more alternative tuples, each of which consists of two or more attributes. The containing ComplexObjectConstraint is satisfied when all of the constraints in one of the AttributeTuples are satisfied.

**Diagrams**

[Attribute Constraints](#_a74fdcc000d4318dcb5580a5fa8fbfce)

**Associations**

• public alternative : [AttributeTuple](#_6d5bfb351e19f61e0327587b0ff5fd4f)[1]

A set of constraints on related attributes to be used to differentiate scenarios where the value of one attribute affects the valid values of another

### <Class> MultiplicityInterval

**Description**

Range of quantities

**Diagrams**

[Attribute Constraints](#_a74fdcc000d4318dcb5580a5fa8fbfce)

**Attributes**

• public minimum : Integer [1]

The smallest value allowed

• public maximum : UnlimitedNatural [1]

The largest value allowed

**Known other classes**

[AttributeCollectionConstraint](#_5eefba8eca7402f09bd5619804038771), [AttributeConstraint](#_11f887fb6f19248bf7193bca31772c05), [ObjectConstraint](#_aa52f11e5760ad2f47030803962bb855), [SingularAttributeConstraint](#_48ee2586ffa14e5bb1cf8ad893969da7)

### <Enumeration> AttributeExistence

**Description**

Strengths of requirement for the existence of an attribute

**Diagrams**

[Attribute Constraints](#_a74fdcc000d4318dcb5580a5fa8fbfce)

**Enumeration Literals**

* **OPTIONAL**

The attribute may be present in a conforming instance

* **PROHIBITED**

The attribute must not be present in a conforming instance

* **REQUIRED**

The attribute must be present in a conforming instance

### <Enumeration> CollectionType

**Description**

Classification of collections

**Diagrams**

[Attribute Constraints](#_a74fdcc000d4318dcb5580a5fa8fbfce)

**Enumeration Literals**

* **BAG**

An collection of elements, neither ordered nor necessarily unique

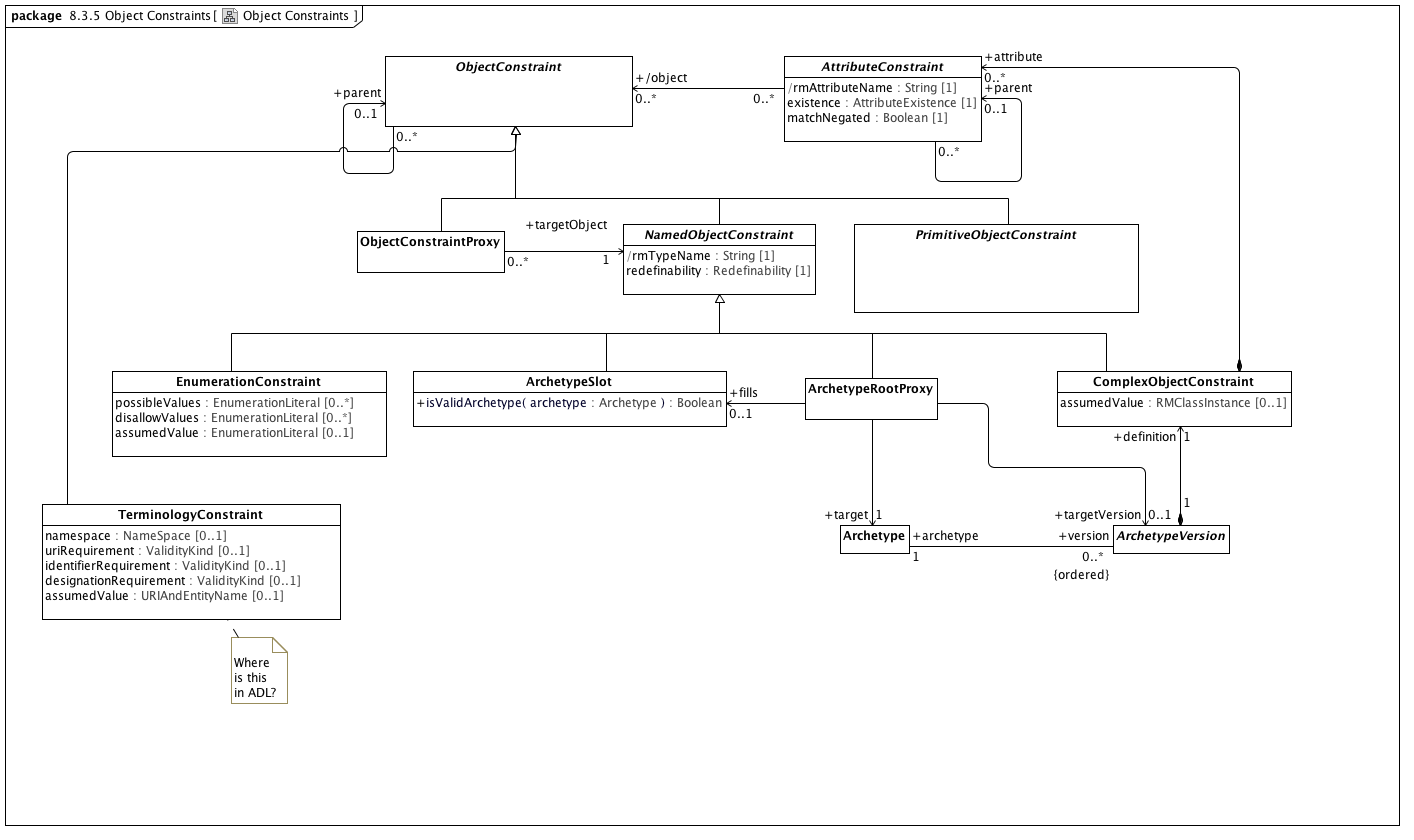
* **LIST**

An ordered collection of elements, not necessarily unique

* **SET**

An collection of unique elements, not necessarily ordered

### <Package> Object Constraints



**Object Constraints**

### <Class> ArchetypeRootProxy

**Description**

A specialization of ComplexObjectConstraint whose node\_id attribute is an archetype identifier rather than an internal node code. Used to reference external archetypes to be included in a composite archetype.

**Diagrams**

[Object Constraints](#_778c20eb327fafc2160b6a98a6014e5d)

**Direct Known Superclasses (Generalization)**

[NamedObjectConstraint](#_ab5b3b01964560abb1047edd9efa4eb9)

**Associations**

• public target : [Archetype](#_f45a7b68ecac449e953ff8a65d6eff75)[1]

A versioned constraint on a target UML class or other Archetype.

• public fills : [ArchetypeSlot](#_e518b2b75b6f66417345772b8440e6f2)[0..1]

A classifier that describes the set of archetypes that may be used to validate instances

• public targetVersion : [ArchetypeVersion](#_1de96fa71501cf96b27b14f3f9f1bb99)[0..1]

An Archetype is a set of constraints that can be applied as a predicate against instances of the constrains Reference Model class as referenced by the Archetype.

**Constraints**

* **validSlot**

[Analysis]

If there is a target fills ArchetypeSlot, the ArchetypeRootProxy archetypeId must be a member of one or more includes value sets and not be a member of one or more excludes value sets as identified in the ArchetypeSlot or any of its parents or ancestors

* **validTarget**

[OCL]

target.archetypeId = archetypeId and target.archetypeName = archetypeName -- Also

### <Class> ArchetypeSlot

**Description**

A classifier that describes the set of archetypes that may be used to validate instances

**Diagrams**

[Object Constraints](#_778c20eb327fafc2160b6a98a6014e5d)

**Direct Known Superclasses (Generalization)**

[NamedObjectConstraint](#_ab5b3b01964560abb1047edd9efa4eb9)

**Operations**

• public isValidArchetype (archetype : [Archetype](#_f45a7b68ecac449e953ff8a65d6eff75)) : Boolean

### <Class> ObjectConstraintProxy

**Description**

A constraint defined by reference to a node defined elsewhere in the same archetype

**Diagrams**

[Object Constraints](#_778c20eb327fafc2160b6a98a6014e5d)

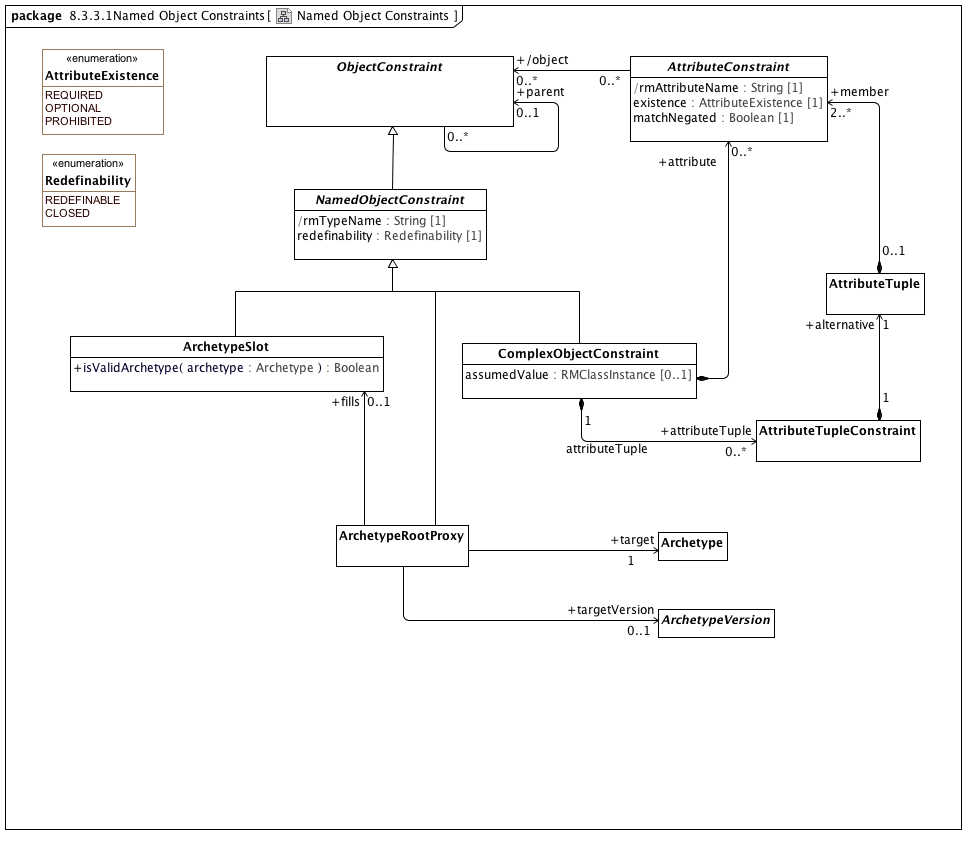
**Direct Known Superclasses (Generalization)**

[ObjectConstraint](#_aa52f11e5760ad2f47030803962bb855)

**Known other classes**

[Archetype](#_f45a7b68ecac449e953ff8a65d6eff75), [ArchetypeVersion](#_1de96fa71501cf96b27b14f3f9f1bb99), [AttributeConstraint](#_11f887fb6f19248bf7193bca31772c05), [ComplexObjectConstraint](#_abfab8c8e983a73b4981f6fcfdd16134), [EnumerationConstraint](#_42c2e4f902eddd2a1629a431a96cd94f), [NamedObjectConstraint](#_ab5b3b01964560abb1047edd9efa4eb9), [ObjectConstraint](#_aa52f11e5760ad2f47030803962bb855), [PrimitiveObjectConstraint](#_db9df3b10e304d809393da4afc9a91da), [TerminologyConstraint](#_b2d4edbc24f651e5a3d756933fff1326)

#### <Package> Named Object Constraints



**Named Object Constraints**

**Known other classes**

[Archetype](#_f45a7b68ecac449e953ff8a65d6eff75), [ArchetypeRootProxy](#_12e58855caae51d65fb43e2837534f63), [ArchetypeSlot](#_e518b2b75b6f66417345772b8440e6f2), [ArchetypeVersion](#_1de96fa71501cf96b27b14f3f9f1bb99), [AttributeConstraint](#_11f887fb6f19248bf7193bca31772c05), [AttributeTuple](#_6d5bfb351e19f61e0327587b0ff5fd4f), [AttributeTupleConstraint](#_f6da15c71717330ae1b56f8b41e3dd51), [ComplexObjectConstraint](#_abfab8c8e983a73b4981f6fcfdd16134), [NamedObjectConstraint](#_ab5b3b01964560abb1047edd9efa4eb9), [ObjectConstraint](#_aa52f11e5760ad2f47030803962bb855)

### <Enumeration> Redefinability

**Description**

Whether a node can be further constrained or elaborated in specializations

**Diagrams**

[Named Object Constraints](#_9247c31608028a4b9ce5cbb4664b4baa)

**Enumeration Literals**

* **CLOSED**

* **REDEFINABLE**

**Known other enumerations**

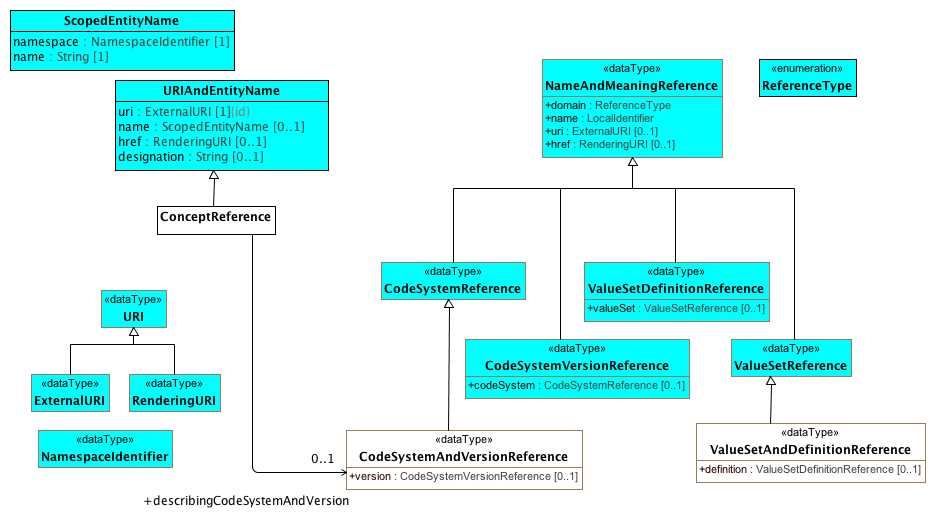
[AttributeExistence](#_4f99fbfcf9617d7ad55eca111d84fb67)

## <Package> Terminology Object Model

This section describes how a combination of the applicable elements of the Common Terminology Services (CTS2) and the ISO 11179-3 Metadata registries (MDR) - Part 3: Registry metamodel and basic attributes 3rd Edition can be combined to implement the ADL/AOM "id", "at" and "ac" identifier schemes and the term\_definitions, term\_bindings and value set sections of the Archetype Modeling Language.

We begin by describing the subsetting the core set of elements that are used from the CTS2 Core Model and some minor extensions that are needed for the AML Object Model. We then repeat this process with the ISO 11179-3 elements, integrating them with the CTS2 core elements. We finally identify the AML model components that have identities and corresponding entries in the terminology section.

### <Package> Common Terminology Services Components



**CTSCore Components**

This diagram contains the subset of the [CTS2 Core Model Elements](http://www.omg.org/cgi-bin/doc?formal/2013-12-04) that are used by the AML Object Model. The URI types are derived from Section 2.1.3, ScopedEntityName and URIAndEntityName from Section 2.2.3 Entity References and the NameAndMeaning references from Section

The AML specification adds the following restrictions to these types:

1. The href attribute is not used in URIAndENtityName or NameAndMeaningReference
2. NameAndMeaningReference.domain is determined by the type of the specialization and is omitted.
3. URI is mandatory for NameAndMeaning reference.

The ConceptReference class is specific to AML. The CTS2 model used the term "EntityReference", but the AML and 11179 communities are both used to using the notion of "Concept", although sometimes it is used to reference an identifier, sometimes a description and sometimes an abstract category in the viewer's mind. In AML, we are only interested in the first case, where ConceptReference is an identifier that can be described (or "included" in 11179 parliance) in one or more CodeSystems (Concept\_System in 11179 terms).

The ADL and AOM require that a Terminology\_Code (concept reference) include the identifier of the code system or version of a code system version that was used to determine the intended meaning of the concept at the point in time it was used. This requires one additional data type, "CodeSystemVersionReference", which in the AML context always includes the identifier of a code system but may or may not include a version.

### <DataType> CodeSystemAndVersionReference

**Description**

A reference to a code system and optional version.

**Diagrams**

[CTSCore Components](#_dcc5ad33945195c14fa1ef7b2e5b33ff)

**Direct Known Superclasses (Generalization)**

[CodeSystemReference](#_ef6a61554a10734f76a4adfa457998fd)

**Attributes**

• public version : [CodeSystemVersionReference](#_15d03d8cc99c5a7bf9bc7d48df26c292) [0..1]

### <DataType> CodeSystemReference

**Description**

A URI *reference* to a code system (aka. "concept system, "terminology" or "ontology")

**Diagrams**

[CTSCore Components](#_dcc5ad33945195c14fa1ef7b2e5b33ff)

**Direct Known Superclasses (Generalization)**

[NameAndMeaningReference](#_7677b94abdbbdc2c76d49314388386a7)

**Direct Known Subclasses (Specialization)**

[CodeSystemAndVersionReference](#_2a8e174187d7187f0716f48b9f8c271e)

### <DataType> CodeSystemVersionReference

**Description**

A reference to a specific version of a code system.

**Diagrams**

[CTSCore Components](#_dcc5ad33945195c14fa1ef7b2e5b33ff)

**Direct Known Superclasses (Generalization)**

[NameAndMeaningReference](#_7677b94abdbbdc2c76d49314388386a7)

**Attributes**

• public codeSystem : [CodeSystemReference](#_ef6a61554a10734f76a4adfa457998fd) [0..1]

A reference to the code system that the code system version reference is a version of.

### <DataType> ExternalURI

**Description**

A URI that references an "real world" (vs. digital) entity. Examples include individual people, locations, organizations as well as abstract concepts or classes. "Well behaved" External URI's do not directly reference a digital resource, although they may result in a redirection to a *RenderingURI* that resolves to a description of the actual target.

**Diagrams**

[CTSCore Components](#_dcc5ad33945195c14fa1ef7b2e5b33ff)

**Direct Known Superclasses (Generalization)**

[URI](#_887928f30f99c8a1ca89ed7a082356aa)

### <DataType> NameAndMeaningReference

**Description**

"NameAndMeaningReference consists of a local identifier that references a unique meaning within the context of a given domain in a CTS2 service instance and a globally unique URI that identifies the intended meaning of the identifier. " (p11 CTS2 1.1 Core Model Elements)

We have tweaked the NameAndMeaningReference model slightly to meet the AML

**Diagrams**

[CTSCore Components](#_dcc5ad33945195c14fa1ef7b2e5b33ff)

**Direct Known Subclasses (Specialization)**

[CodeSystemReference](#_ef6a61554a10734f76a4adfa457998fd), [CodeSystemVersionReference](#_15d03d8cc99c5a7bf9bc7d48df26c292), [ValueSetDefinitionReference](#_bf76d842365737931f787efdb4a36f12), [ValueSetReference](#_5e70a8ee4206e2396a68ca679af92933)

**Attributes**

• public domain : [ReferenceType](#_428e9af279e2df7756c01dffd2ccc1d4)

The "domain" or scope of the reference. This attribute is not used in the AML specification.

• public name : [LocalIdentifier](#_a7bba5b5da9db2f4faa58b64fb2e1eca)

The locally unique identifier string within the context of the domain. As an example, the local name of the SNOMED CT International Edition coding system might be "SNOMEDCT" or "SCT"

• public uri : [ExternalURI](#_de932b9629138c166e8cfb00efa65177) [0..1]

A URI that identifies the specific resource within a global context. As an example, the URI of the SNOMED CT International Code System would be "http://snomed.info/sct/900000000000207008".

• public href : [RenderingURI](#_821273fdc1c3295a17225200782229ea) [0..1]

A URI that, when resolved, results in a description of the referenced resource. The href attribute is not used in the AML Specification.

### <DataType> NamespaceIdentifier

**Description**

An identifier that uniquely references the scoping namespace of an Entity (class, role or individual) within a the context of a service. NamespaceIdentifier syntax must match the [PNAME\_NS](http://www.w3.org/TR/rdf-sparql-query/#rPNAME_NS) production as defined in the [SPARQL Query Specification](http://www.w3.org/TR/rdf-sparql-query/)

**Diagrams**

[CTSCore Components](#_dcc5ad33945195c14fa1ef7b2e5b33ff)

### <DataType> RenderingURI

**Description**

A URI that represents a digital resource, such as a page in a REST service, an online document or other digital artifact.

**Diagrams**

[CTSCore Components](#_dcc5ad33945195c14fa1ef7b2e5b33ff)

**Direct Known Superclasses (Generalization)**

[URI](#_887928f30f99c8a1ca89ed7a082356aa)

### <DataType> URI

**Description**

A Universal Resource Identifier (URI) as defined in [IETF 3986](http://www.ietf.org/rfc/rfc3986.txt) . Implementations are encouraged to consider implementing this data type using the IRI ([RFC3987](http://www.ietf.org/rfc/rfc3987.txt)) specification.

**Diagrams**

[CTSCore Components](#_dcc5ad33945195c14fa1ef7b2e5b33ff)

**Direct Known Subclasses (Specialization)**

[ExternalURI](#_de932b9629138c166e8cfb00efa65177), [RenderingURI](#_821273fdc1c3295a17225200782229ea)

### <DataType> ValueSetAndDefinitionReference

**Description**

A reference to a value set and optional definition.

**Diagrams**

[CTSCore Components](#_dcc5ad33945195c14fa1ef7b2e5b33ff)

**Direct Known Superclasses (Generalization)**

[ValueSetReference](#_5e70a8ee4206e2396a68ca679af92933)

**Attributes**

• public definition : [ValueSetDefinitionReference](#_bf76d842365737931f787efdb4a36f12) [0..1]

### <DataType> ValueSetReference

**Description**

A reference to a name set of entity references.

**Diagrams**

[CTSCore Components](#_dcc5ad33945195c14fa1ef7b2e5b33ff)

**Direct Known Superclasses (Generalization)**

[NameAndMeaningReference](#_7677b94abdbbdc2c76d49314388386a7)

**Direct Known Subclasses (Specialization)**

[ValueSetAndDefinitionReference](#_25ca98283400275c47128a3c11af1824)

**Known other Data Types**

[ValueSetDefinitionReference](#_bf76d842365737931f787efdb4a36f12)

### <Class> ConceptReference

**Description**

A URI that uniquely identifies a "concept" (aka. class, entity, individual or, in some contexts "term"), accompanied by additional information that conveys the intended meaning, code and source of the information used to determine the intent of the URI.

**Diagrams**

[CTSCore Components](#_dcc5ad33945195c14fa1ef7b2e5b33ff)

**Direct Known Superclasses (Generalization)**

[URIAndEntityName](#_4d1f571ab5e9384786ffe39444e822b4)

**Associations**

• public describingCodeSystemAndVersion : [CodeSystemAndVersionReference](#_2a8e174187d7187f0716f48b9f8c271e)[0..1]

A reference to a code system and optional version.

### <Class> ScopedEntityName

**Description**

The combination of a local identifier for a namespace and a name that is unique within the context of the scoping namespace. Note that ScopedEntityNames cannot be exchanged between implementations as different implementations may chose different scoping namespaces. As an example, one implementation may choose to represent the SNOMED CT namespace as "SCTID" while another may choose "SCT".

**Diagrams**

[CTSCore Components](#_dcc5ad33945195c14fa1ef7b2e5b33ff)

**Attributes**

• public namespace : [NamespaceIdentifier](#_94cb7eefb9b55dbc722d53bf1ec0f163) [1]

An identifier that references a unique namespace URI within the context of a service. The type of this identifier is NamespaceIdentifier rather than string to allow conversion to and from XML, RDF, etc.

• public name : String [1]

A unique identifier in the context of the scoping namespace.

**Known other classes**

[URIAndEntityName](#_4d1f571ab5e9384786ffe39444e822b4)

### <Enumeration> ReferenceType

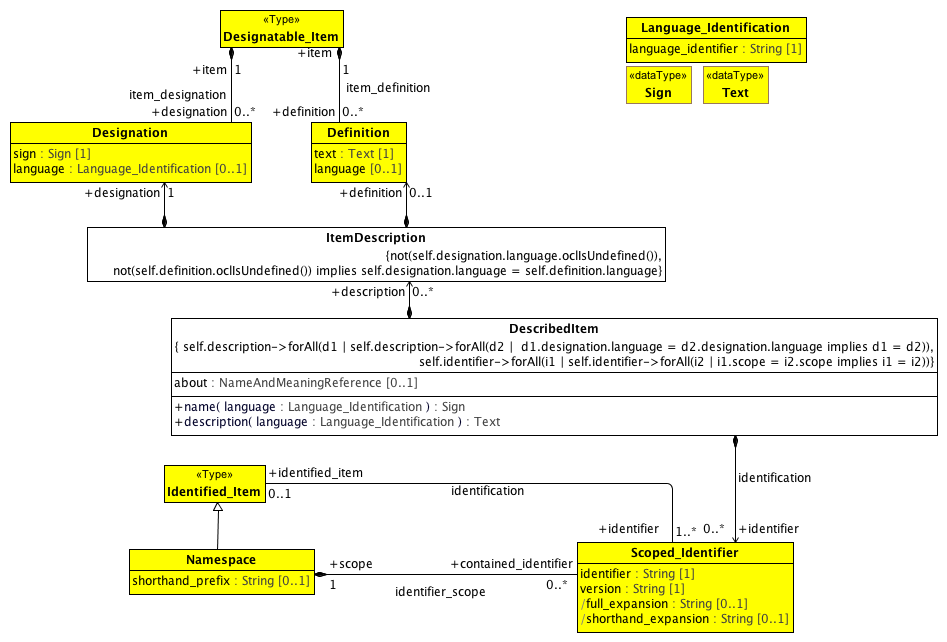
**Diagrams**

[CTSCore Components](#_dcc5ad33945195c14fa1ef7b2e5b33ff)

### <Package> ISO Model Components

This section describes the subset of the *ISO 11179-3 Information technology - Metadata registries (MDR) - Part 3: Registry metamodel and basic attributes 3rd Edition* that are used in the construction of the AML Object Model. Model elements borrowed from ISO 11179 are identified with a yellow background to distinguish them from AML, CTS2 and UML Reference Model components. The first section describes the relationship between the designation, definition and identification sections and AML while the second addresses the meaning/representation links and their AML equivalent.

#### <Package> Designation, Definition and Identification



**Designation and Definition metamodel region**

This diagram contains the subset of the ISO 11179-3:2012(E) model elements in sections 7.2.2 Identification metamodel region and 7.3.2 Designation and Definition metamodel region that are used in the AML Object Model, accompanied by supporting elements drawn from several other sections.

The AML object model is able to use the Designation, Definition and Scoped\_Identifier classes directly with the restrictions mentioned below. It is not able to use Designatable\_Item, which supports an unlimited number of definitions and/or designations while the AML specification includes the following restrictions:

1. There must be at most one ItemDescription per language
2. Every description must consist of exactly one Designation and zero or one Definitions
3. The language attribute is required
4. The language attributes of the Definition and Designation pairs must match.

It is also not able to take advantage of the Identified\_Item class because ISO 11179 requires that all Identified\_Items have at least one Scoped\_Identifier. While this is strictly true in the case of UML, the UML NamedEntity name serves this role and does not need to be exposed to the users unless they intend to export or import from an external system such as ADL. Because of this, we need to create a new relationship between DescribedItem and Scoped\_Identifier with a cardinality of 0..\*.

### <DataType> Sign

**Description**

A sign may be a character string, graphic image, sound clip or other symbol that can be used to denote or designate a concept. The Sign datatype may be implemented using the Reflit(of\_type) data structure (see ISO/IEC 19773:2011 10.4.2), where the list of supported types is implementation defined. At a minimum, datatype String must be supported.

**Note:** The AML specification uses the minimum form of sign -- String.

**Diagrams**

[Designation and Definition metamodel region](#_39265938304977d9288c6eb410872362)

### <DataType> Text

**Description**

data in the form of characters, symbols, words, phrases, paragraphs, sentences, tables, or other character arrangements, intended to convey a meaning, and whose interpretation is essentially based upon the reader's knowledge of some natural language or artificial language [ISO/IEC 2382-23:1994]

Note: AML treats the Text data type as a simple String.

**Diagrams**

[Designation and Definition metamodel region](#_39265938304977d9288c6eb410872362)

### <Class> Definition

**Description**

The Definition class records the binding of a definition text and its language to a Designatable\_Item.

**Diagrams**

[Designation and Definition metamodel region](#_39265938304977d9288c6eb410872362)

**Attributes**

• public text : [Text](#_b7253cf7959d2481cd50a89513cd395b) [1]

text of the Definition

• public language [0..1]

language (3.2.68) used to write the definition text

**Associations**

• public item : [Designatable\_Item](#_b6cf75a1c58d0a2c045bd8f3929ee2bd)[1]

the class of objects which can have designations and definitions.

### <Class> DescribedItem

**Description**

A DescribedItem is an ISO 11179 Designatable\_Item and Identifiable\_item with the following exceptionns

1. Identifiers are optional.
2. ItemDescriptions must come in designation and optional definition pairs
3. The language of a designation and definition pair is required and must match.
4. There can only be one ItemDescription for a given language.
5. There can only be one Scoped\_Identifier for any given namespace.

**Diagrams**

[AML DescribedItems](#_e01fdbb84f109fa2531577050a9e4ad8), [Designation and Definition metamodel region](#_39265938304977d9288c6eb410872362)

**Direct Known Subclasses (Specialization)**

[EnumeratedValueDomain](#_ad639ee3d4cd535b2d3e55238d69cc51), [ObjectConstraint](#_aa52f11e5760ad2f47030803962bb855), [PermissibleValue](#_66976d5fcaf3eff9df49b6e5dab4ad12)

**Attributes**

• public about : [NameAndMeaningReference](#_7677b94abdbbdc2c76d49314388386a7) [0..1]

The about attribute supports the data\_element\_meaning, value\_domain\_meaning and permissible\_value\_meaning associations in the ISO 11179 model. See

**Operations**

• public name (language : [Language\_Identification](#_290b504024c5117158636715bff68ef8)) : [Sign](#_3e3bff45f2bcef12ecdff2975e0f26aa)

Return the name of the artifact in the supplied language.

• public description (language : [Language\_Identification](#_290b504024c5117158636715bff68ef8)) : [Text](#_b7253cf7959d2481cd50a89513cd395b)

Return the description / definition of the artifact in the supplied language

**Associations**

• public identifier : [Scoped\_Identifier](#_e9986a35a9b1fdfe3b6504bff891a4a7)[0..\*]

an identifier with a particular scope provided by a Namespace.

• public description : [ItemDescription](#_70b51d918df4b49dcc323ca2d82f1258)[0..\*]

**Constraints**

* **sameLanguage**

[OCL]

not(self.definition.oclIsUndefined()) implies self.definition.language = self.designation.language

* **zeroOrOneDefinition**
* **mandatoryLanguage**

[OCL]

not(self.designation.language.oclIsUndefined())

* **oneIdPerNamespace**

[OCL]

self.identifier->forAll(i1 | self.identifier->forAll(i2 | i1.scope = i2.scope implies i1 = i2))

* **oneDescriptionPerLanguage**

[OCL]

self.description->forAll(d1 | self.description->forAll(d2 | d1.designation.language = d2.designation.language implies d1 = d2))

### <Class> Designatable\_Item

**Description**

the class of objects which can have designations and definitions.

**Diagrams**

[Designation and Definition metamodel region](#_39265938304977d9288c6eb410872362)

**Associations**

• public designation : [Designation](#_f46de41197f2753a2a6dc47a2e5c89fd)[0..\*]

Records the binding of a pair comprised of a sign and its language to a Designatable\_Item. Each Designation ... may be paired with a Definition.

• public definition : [Definition](#_de3b93f77a3823bde9b40110899bbf83)[0..\*]

The Definition class records the binding of a definition text and its language to a Designatable\_Item.

### <Class> Designation

**Description**

Records the binding of a pair comprised of a sign and its language to a Designatable\_Item. Each Designation ... may be paired with a Definition.

**Diagrams**

[Designation and Definition metamodel region](#_39265938304977d9288c6eb410872362)

**Attributes**

• public sign : [Sign](#_3e3bff45f2bcef12ecdff2975e0f26aa) [1]

used to designate a Designatable\_Item, e.g., a name of an object or concept. The sign may be a word or phrase in a natural language (as specified by the language), or it may be an icon or other symbol.

• public language : [Language\_Identification](#_290b504024c5117158636715bff68ef8) [0..1]

used to record the language or dialect in which the sign (usually a name) is used, when the sign has an associated language. Usually the language will refer to a natural human language.

**Associations**

• public item : [Designatable\_Item](#_b6cf75a1c58d0a2c045bd8f3929ee2bd)[1]

the class of objects which can have designations and definitions.

### <Class> Identified\_Item

**Description**

Identified\_Item is a class each instance of which models an identified item, a metadata item that is identified in a metadata registry.

**Diagrams**

[Designation and Definition metamodel region](#_39265938304977d9288c6eb410872362)

**Direct Known Subclasses (Specialization)**

[Namespace](#_540375a26e1a8a2f020e1b8d908aad0c)

**Associations**

• public identifier : [Scoped\_Identifier](#_e9986a35a9b1fdfe3b6504bff891a4a7)[1..\*]

an identifier with a particular scope provided by a Namespace.

### <Class> ItemDescription

**Diagrams**

[Designation and Definition metamodel region](#_39265938304977d9288c6eb410872362)

**Associations**

• public definition : [Definition](#_de3b93f77a3823bde9b40110899bbf83)[0..1]

The Definition class records the binding of a definition text and its language to a Designatable\_Item.

• public designation : [Designation](#_f46de41197f2753a2a6dc47a2e5c89fd)[1]

Records the binding of a pair comprised of a sign and its language to a Designatable\_Item. Each Designation ... may be paired with a Definition.

**Constraints**

* **mandatoryLanguage**

All designations must have a language

[OCL]

not(self.designation.language.oclIsUndefined())

* **matchingDescriptionLanguage**

[OCL]

not(self.definition.oclIsUndefined()) implies self.designation.language = self.definition.language

### <Class> Language\_Identification

**Description**

Language\_Identification serves as an identifier for a language. Language\_Identification always defines a language as spoken (or written, signed or otherwise signaled) by human beings for communication of information to other human beings. Computer languages such as programming languages are explicitly excluded.

**Note 1:** ISO 11179-3 defines several additional attributes for the Language\_Identification class. As the AML specification simply references languages, details such as script, geopolitical, variant identifiers, etc. are not in scope.

**Note 2**: The ISO 11179-3 references several language identificatin documents, including the W3C IETF, IANA and RFC5646. The AML specification is agnostic when it comes to language tags and will treat language as another terminology / value set.

**Diagrams**

[Designation and Definition metamodel region](#_39265938304977d9288c6eb410872362)

**Attributes**

• public language\_identifier : String [1]

identifies the primary language.

### <Class> Namespace

**Description**

Namespace is a class each instance of which represents a namespace. Namespace is a scoping construct used to group sets of Designations and/or Scoped\_Identifiers used in a metadata registry. Distinct Namespaces permit independent development of metadata collections and/or ontologies. They permit enforcement of uniqueness constraints on identifiers or designation\_signs within a specific Namespace without central coordination.

**Note:** The namespace Class has several additional properties and relationships in the 11179 specification. For the purposes of the AML specification, we need only to know the namespace, its official identifier and its shorthand prefix. In particular, AML will assume that (a) shorthand\_prefix is always present (multiplicity: [1..1]).

**Diagrams**

[Designation and Definition metamodel region](#_39265938304977d9288c6eb410872362)

**Direct Known Superclasses (Generalization)**

[Identified\_Item](#_1248f279d4892b4c57cebdb831316891)

**Attributes**

• public shorthand\_prefix : String [0..1]

prefix conventionally used as shorthand for a namespace, for greater

readability, in text for human consumption.

**Associations**

• public contained\_identifier : [Scoped\_Identifier](#_e9986a35a9b1fdfe3b6504bff891a4a7)[0..\*]

an identifier with a particular scope provided by a Namespace.

### <Class> Scoped\_Identifier

**Description**

an identifier with a particular scope provided by a Namespace.

**Diagrams**

[Designation and Definition metamodel region](#_39265938304977d9288c6eb410872362)

**Attributes**

• public identifier : String [1]

An identifier may be used as an unambiguous identifier for an Identified\_Item within a particular Namespace.

• public version : String [1]

version allows more than one version of an Identified\_Item to be identified within a particular Namespace.

**Note:** the AML specification does not include the notion of identified item "versioning", so the version attribute is not used in the AML specification. For the purposes of compatibility with ISO 11179, one can assume that the version attribute has a single, global default value.

• public full\_expansion : String [0..1]

full\_expansion is formed by prefixing the unique identifier of Namespace to the identifier of this Scoped\_Identifier.

• public shorthand\_expansion : String [0..1]

short\_expansion (sic) is formed by prefixing the short\_prefix of Namespace to the identifier of this Scoped\_Identifier. short\_expansion will exist if and only if the corresponding short\_prefix exists.

**Associations**

• public scope : [Namespace](#_540375a26e1a8a2f020e1b8d908aad0c)[1]

Namespace is a class each instance of which represents a namespace. Namespace is a scoping construct used to group sets of Designations and/or Scoped\_Identifiers used in a metadata registry. Distinct Namespaces permit independent development of metadata collections and/or ontologies. They permit enforcement of uniqueness constraints on identifiers or designation\_signs within a specific Namespace without central coordination.

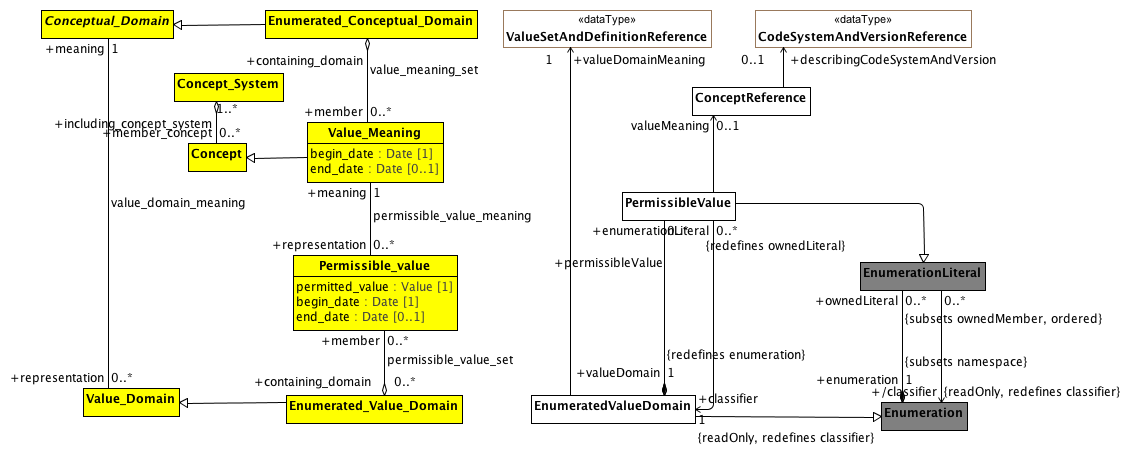
**Note:** The namespace Class has several additional properties and relationships in the 11179 specification. For the purposes of the AML specification, we need only to know the namespace, its official identifier and its shorthand prefix. In particular, AML will assume that (a) shorthand\_prefix is always present (multiplicity: [1..1]).

• public identified\_item : [Identified\_Item](#_1248f279d4892b4c57cebdb831316891)[0..1]

Identified\_Item is a class each instance of which models an identified item, a metadata item that is identified in a metadata registry.

#### <Package> Describable Items in AML

1. **Conceptual and Value Domains**



**Conceptual and value domain metamodel region**

A Conceptual\_Domain sometimes contains a finite allowed inventory of notions that can be enumerated. Such a Conceptual\_Domain is referred to as an Enumerated\_Conceptual\_Domain.

**Known other Data Types**

[CodeSystemAndVersionReference](#_2a8e174187d7187f0716f48b9f8c271e), [ValueSetAndDefinitionReference](#_25ca98283400275c47128a3c11af1824)

### <Class> Concept

**Description**

a unit of knowledge created by a unique combination of characteristics. A concept is independent of representation.

**Diagrams**

[Conceptual and value domain metamodel region](#_c940c3b06c77c12f4e6ddf81cb3280bc)

**Direct Known Subclasses (Specialization)**

[Data\_Element\_Concept](#_957a61f61bd5a7299aaa33e4cc6ae00a), [Value\_Meaning](#_2a66de942c64db0df9c0839ab3abc288)

**Associations**

• public including\_concept\_system : [Concept\_System](#_3c62b4de493817e9b573867c82d00f2a)[1..\*]

a set of concepts structured according to the relations among them.

### <Class> Conceptual\_Domain

**Description**

a set of value meanings which may either be enumerated or expressed via a description.

**Diagrams**

[Conceptual and value domain metamodel region](#_c940c3b06c77c12f4e6ddf81cb3280bc)

**Direct Known Subclasses (Specialization)**

[Enumerated\_Conceptual\_Domain](#_12c689c6a7d403bb204b68d02b775064)

**Associations**

• public representation : [Value\_Domain](#_9b81960c26ee442be19393f5a5fc3ef2)[0..\*]

### <Class> Concept\_System

**Description**

a set of concepts structured according to the relations among them.

**Diagrams**

[Conceptual and value domain metamodel region](#_c940c3b06c77c12f4e6ddf81cb3280bc)

**Associations**

• public member\_concept : [Concept](#_2f48f391605b5ae297a56304194c2a30)[0..\*]

a unit of knowledge created by a unique combination of characteristics. A concept is independent of representation.

### <Class> Enumerated\_Conceptual\_Domain

**Diagrams**

[Conceptual and value domain metamodel region](#_c940c3b06c77c12f4e6ddf81cb3280bc)

**Direct Known Superclasses (Generalization)**

[Conceptual\_Domain](#_7e63d205ca8838d42d16c4aae3fbbf3c)

**Associations**

• public member : [Value\_Meaning](#_2a66de942c64db0df9c0839ab3abc288)[0..\*]

Value\_Meaning is a class each instance of which models a value meaning, which provides *semantic content* of a possible value.

Each member of an Enumerated\_Conceptual\_Domain has a Value\_Meaning that provides its distinction from other members. In the example of ISO 3166, the notion of each country as specified would be the Value\_Meanings. The *representation of Value\_Meanings in a registry shall be independent of (and shall not constrain) their representation in any corresponding Value\_Domain.* A particular Value\_Meaning may have more than one means of representation by Permissible\_Values — each from a distinct Enumerated\_Value\_Domain.

The italics in the above definition are ours. The key aspects of Value\_Meaning are:

1. It represents a concept that is a member ("contained") of one or more Concept\_Systems
2. It provides semantic content of a permissible value.
3. The representation of the Value\_Meaning is independent of how the information is represented in an Enumerated\_Value\_Domain.

### <Class> Permissible\_value

**Description**

a class each instance of which models a permissible value.

**Note:** The AML model views Enumerated\_Value\_Domains as a extension to the UML Enumeration DataType and Permissible\_values as Enumeration\_Literals. The notion of begin\_date becomes an aspect of the model (version) itself rather than the date when the specific member "became valid". As a result, the model will assume that begin\_date is assigned a date that is sufficiently far in the past that we do not have to concern ourselves with it. In addition, permissible values will be removed from the accompanying domain, so end\_date will not be populated.

**Diagrams**

[Conceptual and value domain metamodel region](#_c940c3b06c77c12f4e6ddf81cb3280bc)

**Attributes**

• public permitted\_value : [Value](#_52e70dc394d1abd3621f32ef313b1424) [1]

the actual value of the Permissible\_Value

• public begin\_date : [Date](#_b5130c0b7557264e040caf21b3b5ef6d) [1]

date at which the Permissible\_Value became valid

• public end\_date : [Date](#_b5130c0b7557264e040caf21b3b5ef6d) [0..1]

date at which the Permissible\_Value ceased to be valid

**Associations**

• public meaning : [Value\_Meaning](#_2a66de942c64db0df9c0839ab3abc288)[1]

Value\_Meaning is a class each instance of which models a value meaning, which provides *semantic content* of a possible value.

Each member of an Enumerated\_Conceptual\_Domain has a Value\_Meaning that provides its distinction from other members. In the example of ISO 3166, the notion of each country as specified would be the Value\_Meanings. The *representation of Value\_Meanings in a registry shall be independent of (and shall not constrain) their representation in any corresponding Value\_Domain.* A particular Value\_Meaning may have more than one means of representation by Permissible\_Values — each from a distinct Enumerated\_Value\_Domain.

The italics in the above definition are ours. The key aspects of Value\_Meaning are:

1. It represents a concept that is a member ("contained") of one or more Concept\_Systems
2. It provides semantic content of a permissible value.
3. The representation of the Value\_Meaning is independent of how the information is represented in an Enumerated\_Value\_Domain.

• public containing\_domain : [Enumerated\_Value\_Domain](#_f7ddc2c93ef0e7318e8f00f08a71d10c)[0..\*]

a class each instance of which models an enumerated value domain, a value domain that is specified by a list of all its permissible values.

### <Class> Value\_Domain

**Diagrams**

[Conceptual and value domain metamodel region](#_c940c3b06c77c12f4e6ddf81cb3280bc)

**Direct Known Subclasses (Specialization)**

[Enumerated\_Value\_Domain](#_f7ddc2c93ef0e7318e8f00f08a71d10c)

**Associations**

• public meaning : [Conceptual\_Domain](#_7e63d205ca8838d42d16c4aae3fbbf3c)[1]

a set of value meanings which may either be enumerated or expressed via a description.

• public usage : [Data\_Element](#_742378c87010c3a79ca426ed64113f23)[0..\*]

Data\_Element is a class each instance of which models a data element, a unit of data that is *considered in context to be indivisible*. A data element is a basic unit of data of interest to an organization, for which the definition, identification, representation, and permissible values are specified by means of a set of attributes. Examples of data element include: a column in a table of a relational database, a field in a record or form, an XML element, the attribute of a Java class, or a variable in a program. The description of data elements is a major purpose of ISO/IEC 11179 Metadata Registries.

(Italics added)

### <Class> Value\_Meaning

**Description**

Value\_Meaning is a class each instance of which models a value meaning, which provides *semantic content* of a possible value.

Each member of an Enumerated\_Conceptual\_Domain has a Value\_Meaning that provides its distinction from other members. In the example of ISO 3166, the notion of each country as specified would be the Value\_Meanings. The *representation of Value\_Meanings in a registry shall be independent of (and shall not constrain) their representation in any corresponding Value\_Domain.* A particular Value\_Meaning may have more than one means of representation by Permissible\_Values — each from a distinct Enumerated\_Value\_Domain.

The italics in the above definition are ours. The key aspects of Value\_Meaning are:

1. It represents a concept that is a member ("contained") of one or more Concept\_Systems
2. It provides semantic content of a permissible value.
3. The representation of the Value\_Meaning is independent of how the information is represented in an Enumerated\_Value\_Domain.

**Diagrams**

[Conceptual and value domain metamodel region](#_c940c3b06c77c12f4e6ddf81cb3280bc)

**Direct Known Superclasses (Generalization)**

[Concept](#_2f48f391605b5ae297a56304194c2a30)

**Attributes**

• public begin\_date : [Date](#_b5130c0b7557264e040caf21b3b5ef6d) [1]

date at which this Value\_Meaning became, or will become, a valid Value\_Meaning.

**Note:** from the AML perspective, the Value\_Meaning membership in an Enumerated\_Conceptual\_Domain is determined by a ValueSetDefinition, which is a set of rules for determining the members of a ValueSet (Enumerated\_Conceptual\_Domain) at a given point in time. As such, the begin\_date attribute will be ignored, although it could be derived from one of the date/time attributes of the CTS2 ValueSetDefinition.

• public end\_date : [Date](#_b5130c0b7557264e040caf21b3b5ef6d) [0..1]

date on which the Value\_Meaning ceased, or will cease, to be valid.

**Note:** as with begin\_date, end\_date is not used in the AML specification.

**Associations**

• public containing\_domain : [Enumerated\_Conceptual\_Domain](#_12c689c6a7d403bb204b68d02b775064)

• public representation : [Permissible\_value](#_2fd4e293390c50189eca5b7ab34d63cd)[0..\*]

a class each instance of which models a permissible value.

**Note:** The AML model views Enumerated\_Value\_Domains as a extension to the UML Enumeration DataType and Permissible\_values as Enumeration\_Literals. The notion of begin\_date becomes an aspect of the model (version) itself rather than the date when the specific member "became valid". As a result, the model will assume that begin\_date is assigned a date that is sufficiently far in the past that we do not have to concern ourselves with it. In addition, permissible values will be removed from the accompanying domain, so end\_date will not be populated.

**Known other classes**

[ConceptReference](#_4082bf99060eea349dab8f548d297f3d), [EnumeratedValueDomain](#_ad639ee3d4cd535b2d3e55238d69cc51), [Enumerated\_Value\_Domain](#_f7ddc2c93ef0e7318e8f00f08a71d10c), [Enumeration](#_190e24bd48f094ad9ad981ac0b4eb47e), [EnumerationLiteral](#_41faf6a7041d7068bbbf4f9ff3924d22), [PermissibleValue](#_66976d5fcaf3eff9df49b6e5dab4ad12)

1. **Data Element and Data Element Concept**



**Data Element and Data Element Concept**

This diagram shows the AML equivalent for the ISO 11179 Data\_Element and Data\_Element\_Concept element. The first thing to note is that, according to ISO 11179, a Data\_Element is "a unit of data that is *considered in context to be indivisible*". While this covers certain ObjectConstraints, the AML requirements need to assign meaning to both indivisible *and* divisible information artifacts, meaning that the data\_element\_meaning relationship is a subset of ObjectConstraint.about.

The second thing to note is that, from the perspective of ISO 11179, every Data\_Element must be associated with exactly one meaning. From the AML perspective, while every ObjectConstraint possibly *should* be associated with a corresponding (Data Element) concept that provides meaning, it will frequently be the case that such an association will not be available.

**Known other Data Types**

[CodeSystemAndVersionReference](#_2a8e174187d7187f0716f48b9f8c271e)

### <Class> Data\_Element

**Description**

Data\_Element is a class each instance of which models a data element, a unit of data that is *considered in context to be indivisible*. A data element is a basic unit of data of interest to an organization, for which the definition, identification, representation, and permissible values are specified by means of a set of attributes. Examples of data element include: a column in a table of a relational database, a field in a record or form, an XML element, the attribute of a Java class, or a variable in a program. The description of data elements is a major purpose of ISO/IEC 11179 Metadata Registries.

(Italics added)

**Diagrams**

[Data Element and Data Element Concept](#_ea1c0f7548bd076f87be813f2a93f734)

**Associations**

• public meaning : [Data\_Element\_Concept](#_957a61f61bd5a7299aaa33e4cc6ae00a)[1]

a concept that is an association of a property with an object class. A data element concept is a concept that can be represented in the form of a data element described independently of any particular representation.

• public domain : [Value\_Domain](#_9b81960c26ee442be19393f5a5fc3ef2)[1]

### <Class> Data\_Element\_Concept

**Description**

a concept that is an association of a property with an object class. A data element concept is a concept that can be represented in the form of a data element described independently of any particular representation.

**Diagrams**

[Data Element and Data Element Concept](#_ea1c0f7548bd076f87be813f2a93f734)

**Direct Known Superclasses (Generalization)**

[Concept](#_2f48f391605b5ae297a56304194c2a30)

**Associations**

• public representation : [Data\_Element](#_742378c87010c3a79ca426ed64113f23)[0..\*]

Data\_Element is a class each instance of which models a data element, a unit of data that is *considered in context to be indivisible*. A data element is a basic unit of data of interest to an organization, for which the definition, identification, representation, and permissible values are specified by means of a set of attributes. Examples of data element include: a column in a table of a relational database, a field in a record or form, an XML element, the attribute of a Java class, or a variable in a program. The description of data elements is a major purpose of ISO/IEC 11179 Metadata Registries.

(Italics added)

**Known other classes**

[Concept](#_2f48f391605b5ae297a56304194c2a30), [ConceptReference](#_4082bf99060eea349dab8f548d297f3d), [Concept\_System](#_3c62b4de493817e9b573867c82d00f2a), [ObjectConstraint](#_aa52f11e5760ad2f47030803962bb855)

### <Package> AML Described Items

This section describes how the term\_definitions subsection of the ARCHETYPE\_TERMINOLOGY class is represented in the AML Object Model. The term\_definitions attribute is defined as "Hash <Hash <ARCHETYPE\_TERM, String>, String> [0..1]", where the outer hash key is a language code, e.g. "en", "de" while the inner hash codes are term codes, e.g. "id17", "at4".

ARCHETYPE\_TERM, in turn, is defined as a combination of a *code* with accompanying *text*, *description* and *other\_items*, an arbitrary collection of tag/value pairs.



**AML DescribedItems**

This document shows the AML artifacts that can have entries in the ADL terminology\_definitions section. The ADL equivalent of an ObjectConstraint is identified by an "id" code, an EnumeratedValueDomain as an "ac" code and a PermissibleValue as an "at" code.

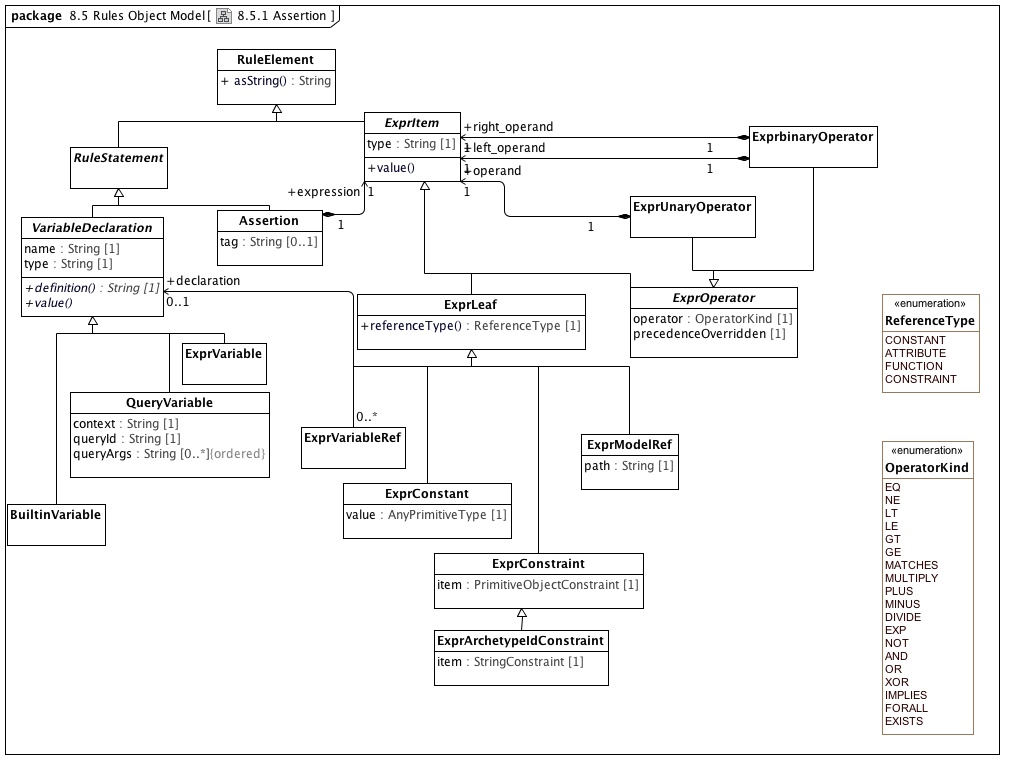
Note that the ADL language imposes an additional constraint that there must be an entry for the archetype originalLanguage as well as an entry for each translation language and that there cannot be an entry for any additional languages. The AML specification leaves enforcement of this type of constraint to language specific validation and export tools.

Also note that the ADL specification requires that every entry have an Scoped\_Identifier. Again, this requirement is not necessary if one is modeling strictly with UML and will need to be enforced in a language specific validation/export tool.

**Known other classes**

[DescribedItem](#_4cfed31aa26e11dff4000228bcd222ae), [EnumeratedValueDomain](#_ad639ee3d4cd535b2d3e55238d69cc51), [ObjectConstraint](#_aa52f11e5760ad2f47030803962bb855), [PermissibleValue](#_66976d5fcaf3eff9df49b6e5dab4ad12)

## <Package> Rules Object Model



**Assertion**

### <Class> Assertion

**Description**

A first order predicate logic assertion in the form of an expression tree

**Diagrams**

[Assertion](#_b4a88d623e0d342d0cf0c8bff23e04b9)

**Direct Known Superclasses (Generalization)**

[RuleStatement](#_f8740e8d27529166da46265bd8521c94)

**Attributes**

• public tag : String [0..1]

Expression tag, used for differentiating multiple assertions.

**Associations**

• public expression : [ExprItem](#_6584fad2aa663f8951c117400955df67)[1]

Abstract parent of all expression items

### <Class> BuiltinVariable

**Description**

A variable with a name and definition from a small set of assumed environmental variables

**Diagrams**

[Assertion](#_b4a88d623e0d342d0cf0c8bff23e04b9)

**Direct Known Superclasses (Generalization)**

[VariableDeclaration](#_8a634b04f92ff4c449cdcaaae16ba015)

### <Class> ExprArchetypeIdConstraint

**Description**

Expression tree leaf item representing a constraint on an archetype identifier

**Diagrams**

[Assertion](#_b4a88d623e0d342d0cf0c8bff23e04b9)

**Direct Known Superclasses (Generalization)**

[ExprConstraint](#_85bd43994bf2970e6d28314019bb2090)

**Attributes**

• public item : [StringConstraint](#_e78c0feb207cbea2ca9911ec94e2a83e) [1]

The constraint on archetype identifiers

### <Class> ExprbinaryOperator

**Description**

Binary operator expression node

**Diagrams**

[Assertion](#_b4a88d623e0d342d0cf0c8bff23e04b9)

**Direct Known Superclasses (Generalization)**

[ExprOperator](#_4f284e94fe833667061a7b6b72ef6466)

**Associations**

• public left\_operand : [ExprItem](#_6584fad2aa663f8951c117400955df67)[1]

Abstract parent of all expression items

• public right\_operand : [ExprItem](#_6584fad2aa663f8951c117400955df67)[1]

Abstract parent of all expression items

### <Class> ExprConstant

**Description**

Constant expression tree leaf item

**Diagrams**

[Assertion](#_b4a88d623e0d342d0cf0c8bff23e04b9)

**Direct Known Superclasses (Generalization)**

[ExprLeaf](#_9560e7eb734489fe38110da26dd95cc6)

**Attributes**

• public value : [AnyPrimitiveType](#_72b7d3db018aa6b2ac9fd5dcdf816a12) [1]

The constant value

### <Class> ExprConstraint

**Description**

Expression tree leaf item representing a constraint on a primitive type

**Diagrams**

[Assertion](#_b4a88d623e0d342d0cf0c8bff23e04b9)

**Direct Known Superclasses (Generalization)**

[ExprLeaf](#_9560e7eb734489fe38110da26dd95cc6)

**Direct Known Subclasses (Specialization)**

[ExprArchetypeIdConstraint](#_820ea321dcb1dd82f36cbb9c8fb64676)

**Attributes**

• public item : [PrimitiveObjectConstraint](#_db9df3b10e304d809393da4afc9a91da) [1]

A constraint on a primitive type

### <Class> ExprItem

**Description**

Abstract parent of all expression items

**Diagrams**

[Assertion](#_b4a88d623e0d342d0cf0c8bff23e04b9)

**Direct Known Superclasses (Generalization)**

[RuleElement](#_aca03c36f52ec3dbae79ae94604d6df9)

**Direct Known Subclasses (Specialization)**

[ExprLeaf](#_9560e7eb734489fe38110da26dd95cc6), [ExprOperator](#_4f284e94fe833667061a7b6b72ef6466)

**Attributes**

• public type : String [1]

Logical type of this item

**Operations**

• public value ()

### <Class> ExprLeaf

**Description**

Non-compositional item representing a manifest constant of any primitive type; a path referring to a value in the archetype; a constraint; or a variable reference

**Diagrams**

[Assertion](#_b4a88d623e0d342d0cf0c8bff23e04b9)

**Direct Known Superclasses (Generalization)**

[ExprItem](#_6584fad2aa663f8951c117400955df67)

**Direct Known Subclasses (Specialization)**

[ExprConstant](#_a0128bb6e758476a358e05c37712848d), [ExprConstraint](#_85bd43994bf2970e6d28314019bb2090), [ExprModelRef](#_7d47fc8980dcc79b04b0f75cd8e3f12e), [ExprVariableRef](#_9129057e4e52fc3f3b74690cdfbfbd78)

**Operations**

• public referenceType () : [ReferenceType](#_428e9af279e2df7756c01dffd2ccc1d4)

The way the leaf item value is defined

### <Class> ExprModelRef

**Description**

Expression tree leaf item representing a reference to a value found at a location specified by a path in the archetype definition

**Diagrams**

[Assertion](#_b4a88d623e0d342d0cf0c8bff23e04b9)

**Direct Known Superclasses (Generalization)**

[ExprLeaf](#_9560e7eb734489fe38110da26dd95cc6)

**Attributes**

• public path : String [1]

The path

### <Class> ExprOperator

**Description**

Abstract parent of operator types

**Diagrams**

[Assertion](#_b4a88d623e0d342d0cf0c8bff23e04b9)

**Direct Known Superclasses (Generalization)**

[ExprItem](#_6584fad2aa663f8951c117400955df67)

**Direct Known Subclasses (Specialization)**

[ExprbinaryOperator](#_35c04239c15182fcb41266740097f888), [ExprUnaryOperator](#_2ea545d67d0574bfd64976dd80367d68)

**Attributes**

• public operator : [OperatorKind](#_ecd27b09f62f1796f70d44a561b90e90) [1]

Operator kind

• public precedenceOverridden [1]

True if the natural precedence of operators is

overridden in the expression represented by

this node of the expression tree. If True,

parentheses should be introduced around the

totality of the syntax expression corresponding

to this operator node and its operands.

### <Class> ExprUnaryOperator

**Description**

Unary operator expression node

**Diagrams**

[Assertion](#_b4a88d623e0d342d0cf0c8bff23e04b9)

**Direct Known Superclasses (Generalization)**

[ExprOperator](#_4f284e94fe833667061a7b6b72ef6466)

**Associations**

• public operand : [ExprItem](#_6584fad2aa663f8951c117400955df67)[1]

Abstract parent of all expression items

### <Class> ExprVariable

**Description**

A variable whose definition is an expression

**Diagrams**

[Assertion](#_b4a88d623e0d342d0cf0c8bff23e04b9)

**Direct Known Superclasses (Generalization)**

[VariableDeclaration](#_8a634b04f92ff4c449cdcaaae16ba015)

**Associations**

• public expression : [ExprItem](#_6584fad2aa663f8951c117400955df67)[1]

Abstract parent of all expression items

### <Class> ExprVariableRef

**Description**

**Diagrams**

[Assertion](#_b4a88d623e0d342d0cf0c8bff23e04b9)

**Direct Known Superclasses (Generalization)**

[ExprLeaf](#_9560e7eb734489fe38110da26dd95cc6)

**Associations**

• public declaration : [VariableDeclaration](#_8a634b04f92ff4c449cdcaaae16ba015)[0..1]

Definition of a named variable used in an assertion expression

### <Class> QueryVariable

**Description**

A variable whose value is derived from a query run on a data context in the operational environment

**Diagrams**

[Assertion](#_b4a88d623e0d342d0cf0c8bff23e04b9)

**Direct Known Superclasses (Generalization)**

[VariableDeclaration](#_8a634b04f92ff4c449cdcaaae16ba015)

**Attributes**

• public context : String [1]

Name of context

• public queryId : String [1]

Identifier of query in the external context, e.g. �date\_of\_birth�

• public queryArgs : String [0..\*]

Arguments for query

### <Class> RuleElement

**Description**

**Diagrams**

[Assertion](#_b4a88d623e0d342d0cf0c8bff23e04b9)

**Direct Known Subclasses (Specialization)**

[ExprItem](#_6584fad2aa663f8951c117400955df67), [RuleStatement](#_f8740e8d27529166da46265bd8521c94)

**Operations**

• public asString () : String

A rule element in serialized form

### <Class> VariableDeclaration

**Description**

Definition of a named variable used in an assertion expression

**Diagrams**

[Assertion](#_b4a88d623e0d342d0cf0c8bff23e04b9)

**Direct Known Superclasses (Generalization)**

[RuleStatement](#_f8740e8d27529166da46265bd8521c94)

**Direct Known Subclasses (Specialization)**

[BuiltinVariable](#_f8424186be59dfa59367f207253d80c9), [ExprVariable](#_9a7170f73e8949fcd1203662c43a988f), [QueryVariable](#_19604fadcc5010ea7dfd6960b3ed9095)

**Attributes**

• private name : String [1]

Name of variable

• private type : String [1]

Variable type, drawn from the reference model

**Operations**

• public definition () : String

Formal definition of the variable

• public value ()

**Known other classes**

[RuleStatement](#_f8740e8d27529166da46265bd8521c94)

### <Enumeration> OperatorKind

**Description**

**Diagrams**

[Assertion](#_b4a88d623e0d342d0cf0c8bff23e04b9)

**Enumeration Literals**

* **AND**

* **DIVIDE**

* **EQ**

* **EXISTS**

* **EXP**

* **FORALL**

* **GE**

* **GT**

* **IMPLIES**

* **LE**

* **LT**

* **MATCHES**

* **MINUS**

* **MULTIPLY**

* **NE**

* **NOT**

* **OR**

* **PLUS**

* **XOR**

### <Enumeration> ReferenceType

**Description**

**Diagrams**

[Assertion](#_b4a88d623e0d342d0cf0c8bff23e04b9)

**Enumeration Literals**

* **ATTRIBUTE**

* **CONSTANT**

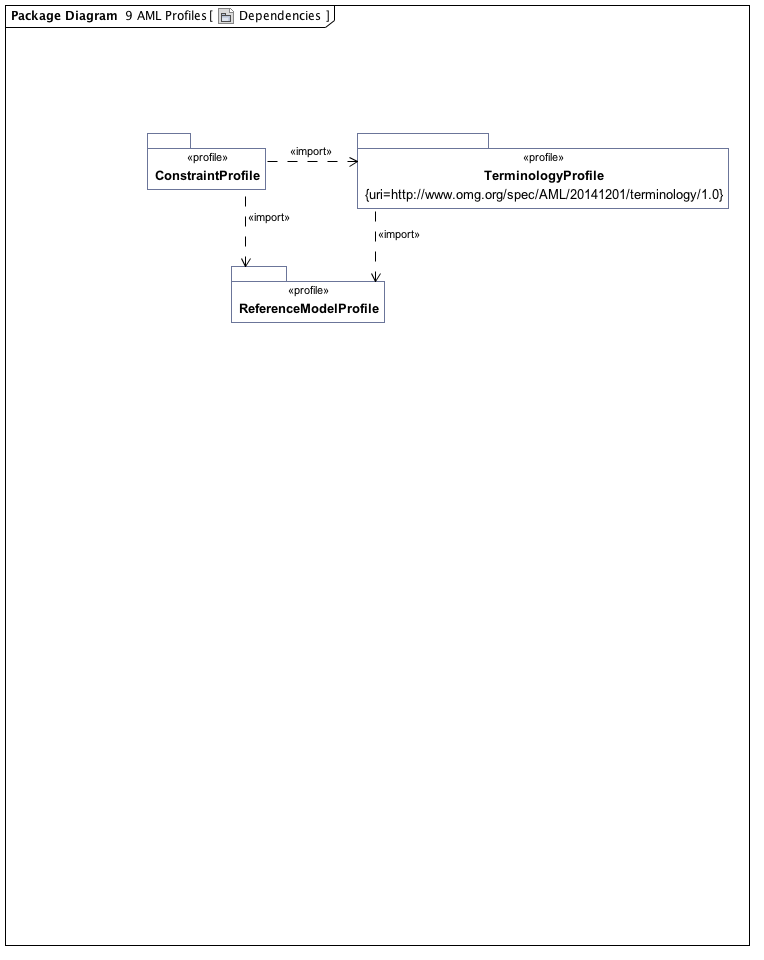
* **CONSTRAINT**

* **FUNCTION**

## <Package> Metadata Object Model

# AML Profiles

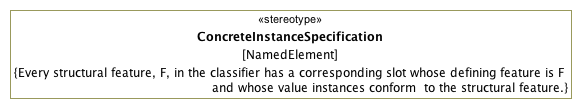
Introduction to the AML Profiles section



**Dependencies**

## <Package> Reference Model Profile

Introduction to the reference model profile section



**ConcreteInstance Specification**

### <Stereotype> ConcreteInstanceSpecification

**Description**

The ConcreteInstanceSpecification stereotype applies to an InstanceSpecification that represents an instance of a class or specialization thereof. Its purpose is to assure that the base InstanceSpecification has exactly one classifier and that the specification is a fully conformant instance of that classifier.

**Diagrams**

[ConcreteInstance Specification](#_81173a8e62656fcf2cb9e1cecdba5e4f)

**Direct Known Subclasses (Specialization)**

[ResourceReferenceInstance](#_9d682f32f4917feea358e696d1fd146d)

**Associations**

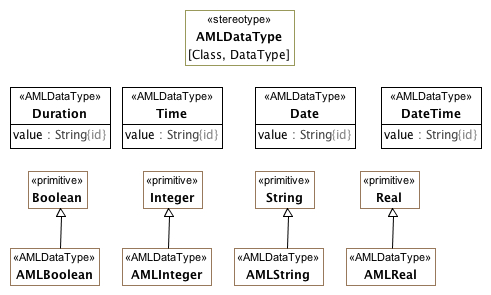
• private base\_NamedElement : NamedElement

**Constraints**

* **slotConformance**

[English]

Every structural feature, F, in the classifier has a corresponding slot whose defining feature is F and whose value instances conform to the structural feature.



**Implementable Data Types**

The AMLDataType stereotype serves two purposes:

1. To identify the set of atomic types whose possible values can be constrained in the AML Constraint Profile
2. To identify the set of types whose value will be treated as "data types" from the AML perspective.

The AMLDataType stereotype can extend both Class and DataType elements. The target reference model may choose to represent some or all of the AML DataTypes in a different fashion. A reference model may define its own String DataType rather than using the UML Native String type directly. Similarly, it may choose to represent a Date as a complex object consisting of year, month, day, granularity, zone, etc.

One of the tasks for a reference model implementor is to create maps from the appropriate AML Data Types and the corresponding reference model types.

### <Class> Date

**Diagrams**

[Implementable Data Types](#_c6849d5d118a9832b3df7e75a9d1a83d)

**Attributes**

• public value : String

### <Class> DateTime

**Diagrams**

[Implementable Data Types](#_c6849d5d118a9832b3df7e75a9d1a83d)

**Attributes**

• public value : String

### <Class> Duration

**Diagrams**

[Implementable Data Types](#_c6849d5d118a9832b3df7e75a9d1a83d)

**Attributes**

• public value : String

### <Class> Time

**Diagrams**

[Implementable Data Types](#_c6849d5d118a9832b3df7e75a9d1a83d)

**Attributes**

• public value : String

### <Stereotype> AMLDataType

**Description**

The AMLDataType stereotype represents a built-in AML data type. Instances of AMLDataType primitive types have the same identity semantics as the UML PrimitiveType. Instances AMLDataType classes are with the identical "id" properties (property.isID = True} are considered to be identical

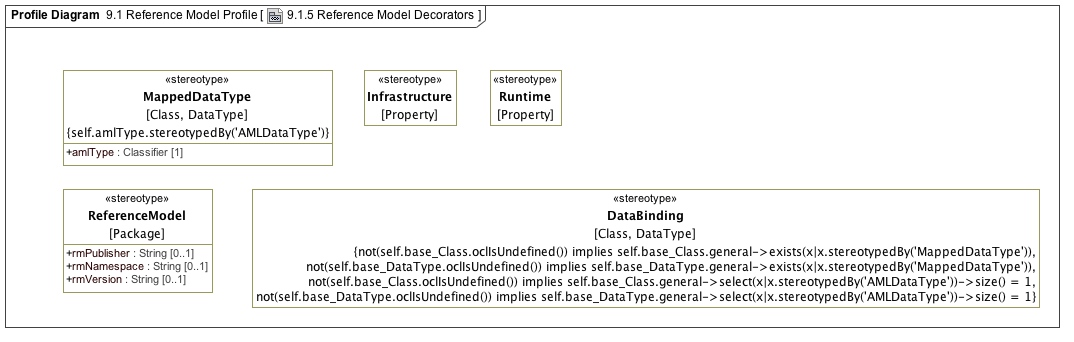
**Diagrams**

[Implementable Data Types](#_c6849d5d118a9832b3df7e75a9d1a83d)

**Associations**

• private base\_Class : Class

• private base\_DataType : DataType



**Reference Model Decorators**

### <Stereotype> DataBinding

**Diagrams**

[Reference Model Decorators](#_549d0bc2f9c4ac0312fc619b71bf4d08)

**Associations**

• private base\_Class : Class

• private base\_DataType : DataType

**Constraints**

* **oneAMLDataType1**

This Class must specialized an >

[OCL]

not(self.base\_Class.oclIsUndefined()) implies self.base\_Class.general->select(x|x.stereotypedBy('AMLDataType'))->size() = 1

* **oneAMLDataType2**

This DataType must specialize an >

[OCL]

not(self.base\_DataType.oclIsUndefined()) implies self.base\_DataType.general->select(x|x.stereotypedBy('AMLDataType'))->size() = 1

* **atLeastOneMappedDataType1**

This Class must be stereotyped by >

[OCL]

not(self.base\_Class.oclIsUndefined()) implies self.base\_Class.general->exists(x|x.stereotypedBy('MappedDataType'))

* **atLeastOneMappedDataType2**

This DataType must be specialized from >

[OCL]

not(self.base\_DataType.oclIsUndefined()) implies self.base\_DataType.general->exists(x|x.stereotypedBy('MappedDataType'))

### <Stereotype> Infrastructure

**Diagrams**

[Reference Model Decorators](#_549d0bc2f9c4ac0312fc619b71bf4d08)

**Associations**

• private base\_Property : Property

### <Stereotype> MappedDataType

**Diagrams**

[Reference Model Decorators](#_549d0bc2f9c4ac0312fc619b71bf4d08)

**Attributes**

• public amlType : Classifier [1]

**Associations**

• private base\_Class : Class

• private base\_DataType : DataType

**Constraints**

* **isAMLDataType**

The amlType must reference a classifier (Class or DataType) that has a AMLDataType stereotype.

[OCL]

self.amlType.stereotypedBy('AMLDataType')

### <Stereotype> ReferenceModel

**Description**

This stereotype identifies a package as a "reference model". An archetype library contains a collection of archetypes that constrain classes that are members of a target reference model. The stereotype also allows additional metadata such as the publisher, namespace, etc. to be added to the model itself.

**Diagrams**

[Reference Model Decorators](#_549d0bc2f9c4ac0312fc619b71bf4d08)

**Attributes**

• public rmPublisher : String [0..1]

The name of the Reference Model publisher.

• public rmNamespace : String [0..1]

The owning domain name of the archetype. Corresponds to the **namespace** attribute in AOM1.5.

• public rmVersion : String [0..1]

The reference model version identifier.

**Associations**

• private base\_Package : Package

### <Stereotype> Runtime

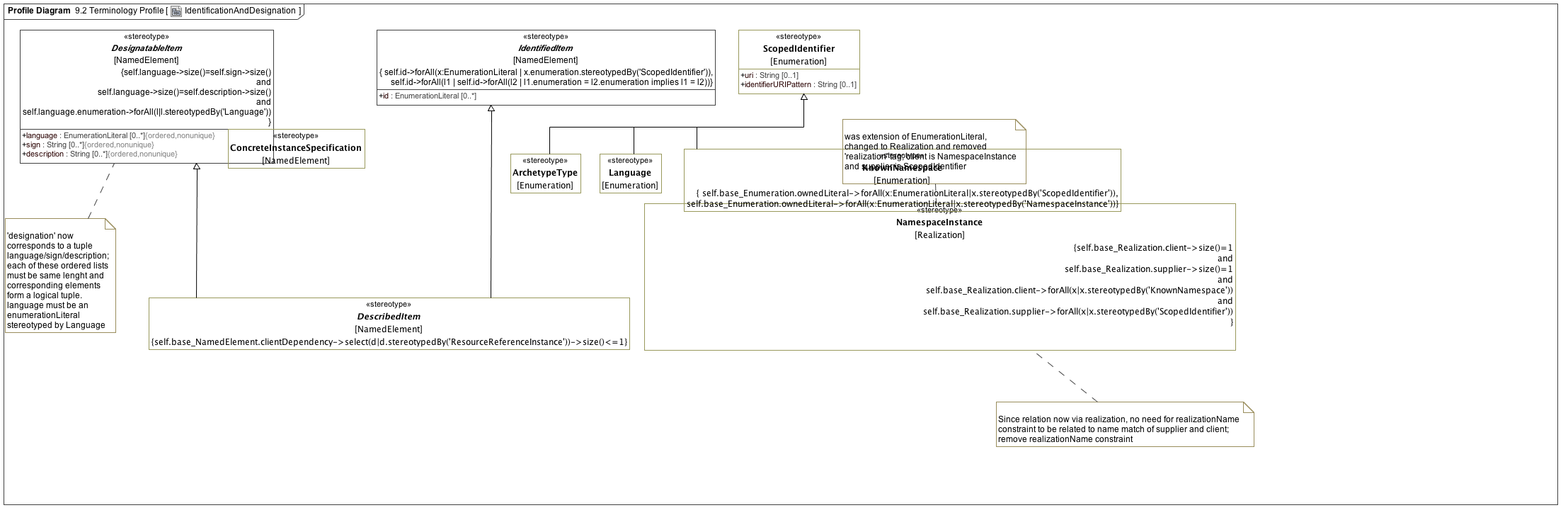
**Diagrams**

[Reference Model Decorators](#_549d0bc2f9c4ac0312fc619b71bf4d08)

**Associations**

• private base\_Property : Property

## <Package> Terminology Profile



**IdentificationAndDesignation**

### <Stereotype> ArchetypeType

**Description**

An archetype is a re-usable, formal model of a concept. An archetype is expressed as a computable set of constraint statements, on an underlying reference model (URM). Concepts that can be modeled using archetypes include weight measurement, blood pressure, microbiology results, discharge referral, prescription, diagnosis. CIMI archetypes will be represented as an instance of the ‘Archetype Object Model’.

**Diagrams**

[IdentificationAndDesignation](#_0628ce7e5381e7543ba417bb320e03fb)

**Direct Known Superclasses (Generalization)**

[ScopedIdentifier](#_59faf6918f4c546323d6df67392c366b)

**Associations**

• private base\_Enumeration : Enumeration

### <Stereotype> DescribedItem

**Description**

The class of objects which can have designations and definitions.

**Diagrams**

[IdentificationAndDesignation](#_0628ce7e5381e7543ba417bb320e03fb)

**Direct Known Superclasses (Generalization)**

[DesignatableItem](#_80448b03d480bba05b1e156796878f77), [IdentifiedItem](#_4b28f60cd7e8328f1d31dbcfa39d2ff3)

**Direct Known Subclasses (Specialization)**

[EnumeratedValueDomain](#_c7f411daaf64f83e013bec437cb8f30a), [ObjectConstraint](#_ad75af95f635bdf35f69d9db9b17aae2), [PermissibleValue](#_5bb7ce8128b60ee5eb2ca275444e9692)

**Associations**

• private base\_NamedElement : NamedElement

**Constraints**

* **validAbout**

There must be a maximum of one > Abstraction

[OCL]

self.base\_NamedElement.clientDependency->select(d|d.stereotypedBy('ResourceReferenceInstance'))->size()<=1

### <Stereotype> DesignatableItem

**Diagrams**

[IdentificationAndDesignation](#_0628ce7e5381e7543ba417bb320e03fb)

**Direct Known Subclasses (Specialization)**

[DescribedItem](#_d45578f848d02aad83980903e5bde7d1), [ResourceReferenceInstance](#_9d682f32f4917feea358e696d1fd146d)

**Attributes**

• public language : EnumerationLiteral [0..\*]

• public sign : String [0..\*]

• public description : String [0..\*]

**Associations**

• private base\_NamedElement : NamedElement

**Constraints**

* **language sign description tuples are designations**

The ordered sequences language, sign, description must be same length and language must be in a >.

[OCL]

self.language->size()=self.sign->size() and self.language->size()=self.description->size() and self.language.enumeration->forAll(l|l.stereotypedBy('Language'))

### <Stereotype> IdentifiedItem

**Description**

Every identifier must come from a different namespace

**Diagrams**

[IdentificationAndDesignation](#_0628ce7e5381e7543ba417bb320e03fb)

**Direct Known Subclasses (Specialization)**

[DescribedItem](#_d45578f848d02aad83980903e5bde7d1), [ResourceReferenceInstance](#_9d682f32f4917feea358e696d1fd146d)

**Attributes**

• public id : EnumerationLiteral [0..\*]

**Associations**

• private base\_NamedElement : NamedElement

**Constraints**

* **scopedIdentifierLiteral**

Every *id* property is an instance of an ScopedIdentifier.

[OCL]

self.id->forAll(x:EnumerationLiteral | x.enumeration.stereotypedBy('ScopedIdentifier'))

* **uniqueScopes**

Every id must belong to a unique instance specification classifier. An identified Item cannot have two or more identifiers drawn from the same ScopedIdentifier enumeration.

[OCL]

self.id->forAll(l1 | self.id->forAll(l2 | l1.enumeration = l2.enumeration implies l1 = l2))

### <Stereotype> KnownNamespace

**Description**

KnownNamespace represents the set of NameSpaceInstances that are known to an AML implementation. Each NamespaceInstance uniquely names a specific namespace and an optional ScopedIdentifier that is the actual instances of that namespace.

**Diagrams**

[IdentificationAndDesignation](#_0628ce7e5381e7543ba417bb320e03fb)

**Direct Known Superclasses (Generalization)**

[ScopedIdentifier](#_59faf6918f4c546323d6df67392c366b)

**Associations**

• private base\_Enumeration : Enumeration

**Constraints**

* **hasScopedIdentifiers**

[OCL]

self.base\_Enumeration.ownedLiteral->forAll(x:EnumerationLiteral|x.stereotypedBy('ScopedIdentifier'))

* **namespaceInstances**

All owned EnumerationLiterals must be >

[OCL]

self.base\_Enumeration.ownedLiteral->forAll(x:EnumerationLiteral|x.stereotypedBy('NamespaceInstance'))

### <Stereotype> Language

**Diagrams**

[IdentificationAndDesignation](#_0628ce7e5381e7543ba417bb320e03fb)

**Direct Known Superclasses (Generalization)**

[ScopedIdentifier](#_59faf6918f4c546323d6df67392c366b)

**Associations**

• private base\_Enumeration : Enumeration

### <Stereotype> NamespaceInstance

**Description**

**Diagrams**

[IdentificationAndDesignation](#_0628ce7e5381e7543ba417bb320e03fb)

**Associations**

• private base\_Realization : Realization

**Constraints**

* **mustBeScopedIdentifier**

There must be a single > client and a single > supplier.

[OCL]

self.base\_Realization.client->size()=1 and self.base\_Realization.supplier->size()=1 and self.base\_Realization.client->forAll(x|x.stereotypedBy('KnownNamespace')) and self.base\_Realization.supplier->forAll(x|x.stereotypedBy('ScopedIdentifier'))

### <Stereotype> ScopedIdentifier

**Diagrams**

[IdentificationAndDesignation](#_0628ce7e5381e7543ba417bb320e03fb)

**Direct Known Subclasses (Specialization)**

[ArchetypeType](#_7dc1530ae1ef855ecc3eb9bd5b555a14), [KnownNamespace](#_b9f78b93edc24bb3301ba69a57e4afc3), [Language](#_446e0591a4f825b22cd9e573c1239a72)

**Attributes**

• public uri : String [0..1]

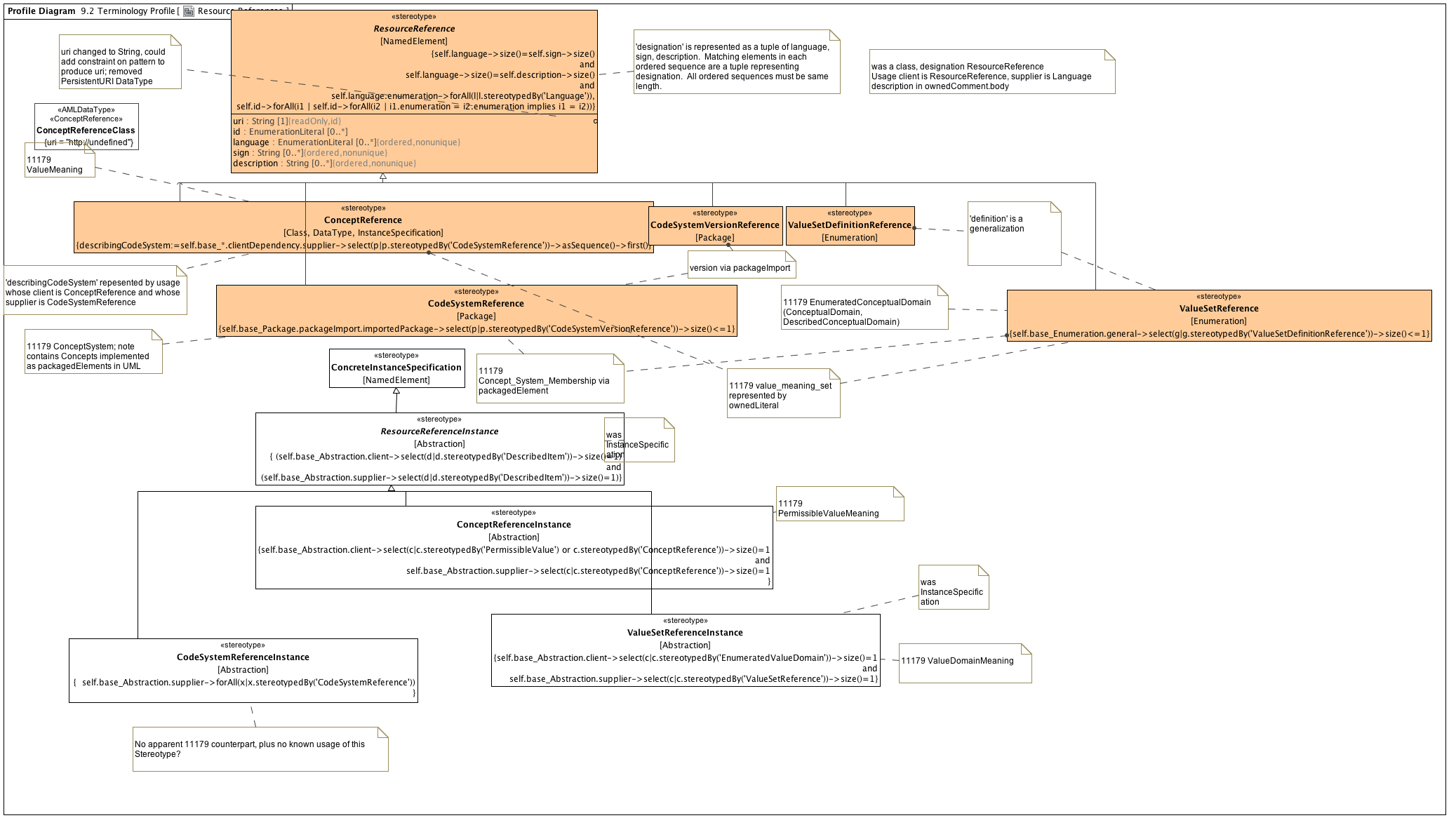
• public identifierURIPattern : String [0..1]

**Associations**

• private base\_Enumeration : Enumeration

**Known other Stereotypes**

[ConcreteInstanceSpecification](#_27a8967f003efe37629fea8e80ca6c23)



**Resource References**

### <Class> ConceptReferenceClass

**Diagrams**

[Resource References](#_aa2597b87275ed4e1724a94a9c31d90b)

### <Stereotype> CodeSystemReference

**Diagrams**

[Resource References](#_aa2597b87275ed4e1724a94a9c31d90b)

**Direct Known Superclasses (Generalization)**

[ResourceReference](#_1b2eec63ad4ef6c72d57b9985e0346ff)

**Associations**

• private base\_Package : Package

**Constraints**

* **version**

There must be at most 1 imported > Package.

[Binary]

self.base\_Package.packageImport.importedPackage->select(p|p.stereotypedBy('CodeSystemVersionReference'))->size()<=1

### <Stereotype> CodeSystemReferenceInstance

**Diagrams**

[Resource References](#_aa2597b87275ed4e1724a94a9c31d90b)

**Direct Known Superclasses (Generalization)**

[ResourceReferenceInstance](#_9d682f32f4917feea358e696d1fd146d)

**Associations**

• private base\_Abstraction : Abstraction

**Constraints**

* **isCodeSystemReference**

Must be a valid instance of a code system reference

[OCL]

self.base\_Abstraction.supplier->forAll(x|x.stereotypedBy('CodeSystemReference'))

### <Stereotype> CodeSystemVersionReference

**Diagrams**

[Resource References](#_aa2597b87275ed4e1724a94a9c31d90b)

**Direct Known Superclasses (Generalization)**

[ResourceReference](#_1b2eec63ad4ef6c72d57b9985e0346ff)

**Associations**

• private base\_Package : Package

### <Stereotype> ConceptReference

**Diagrams**

[Resource References](#_aa2597b87275ed4e1724a94a9c31d90b)

**Direct Known Superclasses (Generalization)**

[ResourceReference](#_1b2eec63ad4ef6c72d57b9985e0346ff)

**Associations**

• private base\_DataType : DataType

• private base\_Class : Class

• private base\_InstanceSpecification : InstanceSpecification

**Constraints**

* **describingCodeSystem**

Definition of describingCodeSystem describingCodeSystem:=self.base\_\*.clientDependency.supplier->select(p|p.stereotypedBy('CodeSystemReference'))->asSequence()->first()

[English]

describingCodeSystem:=self.base\_\*.clientDependency.supplier->select(p|p.stereotypedBy('CodeSystemReference'))->asSequence()->first()

### <Stereotype> ConceptReferenceInstance

**Diagrams**

[Resource References](#_aa2597b87275ed4e1724a94a9c31d90b)

**Direct Known Superclasses (Generalization)**

[ResourceReferenceInstance](#_9d682f32f4917feea358e696d1fd146d)

**Associations**

• private base\_Abstraction : Abstraction

**Constraints**

* **isConceptReference**

There must be one > or > client and one > supplier.

[OCL]

self.base\_Abstraction.client->select(c|c.stereotypedBy('PermissibleValue') or c.stereotypedBy('ConceptReference'))->size()=1 and self.base\_Abstraction.supplier->select(c|c.stereotypedBy('ConceptReference'))->size()=1

### <Stereotype> ResourceReference

**Diagrams**

[Resource References](#_aa2597b87275ed4e1724a94a9c31d90b)

**Direct Known Subclasses (Specialization)**

[CodeSystemReference](#_7de70b4fbd9e6a164f7f00cde47dfd5a), [CodeSystemVersionReference](#_9a8de95c38ebe2d6ce506cbc9bef7b7a), [ConceptReference](#_57ae94153b82f28889d42ad4aa8fe1e0), [ValueSetDefinitionReference](#_a4fedb7858ead8d2272640d51b53719a), [ValueSetReference](#_1a1ca20b54028ee5e2eb20af35411f6e)

**Attributes**

• public uri : String [1]

• public id : EnumerationLiteral [0..\*]

• public language : EnumerationLiteral [0..\*]

• public sign : String [0..\*]

• public description : String [0..\*]

**Associations**

• private base\_NamedElement : NamedElement

**Constraints**

* **uniqueId**

Every identifier must come from a different namespace

[OCL]

self.id->forAll(i1 | self.id->forAll(i2 | i1.enumeration = i2.enumeration implies i1 = i2))

* **language sign description represent tuple**

The ordered sequences language, sign, description must be the same length and language must be part of a > Enumeration.

[OCL]

self.language->size()=self.sign->size() and self.language->size()=self.description->size() and self.language.enumeration->forAll(l|l.stereotypedBy('Language'))

### <Stereotype> ResourceReferenceInstance

**Diagrams**

[Resource References](#_aa2597b87275ed4e1724a94a9c31d90b)

**Direct Known Superclasses (Generalization)**

[ConcreteInstanceSpecification](#_27a8967f003efe37629fea8e80ca6c23), [DesignatableItem](#_80448b03d480bba05b1e156796878f77), [IdentifiedItem](#_4b28f60cd7e8328f1d31dbcfa39d2ff3)

**Direct Known Subclasses (Specialization)**

[CodeSystemReferenceInstance](#_e055a6cce06d0838055b62dbfbf235f2), [ConceptReferenceInstance](#_6a2f733d1d3ea9bc232f96caadb113e1), [ValueSetReferenceInstance](#_f3184cb0f8e704f5122c5e97fb9f130c)

**Associations**

• private base\_Abstraction : Abstraction

**Constraints**

* **isResourceReference**

Must be an instance of a resource reference class

[OCL]

(self.base\_Abstraction.client->select(d|d.stereotypedBy('DescribedItem'))->size()=1) and (self.base\_Abstraction.supplier->select(d|d.stereotypedBy('DescribedItem'))->size()=1)

### <Stereotype> ValueSetDefinitionReference

**Diagrams**

[Resource References](#_aa2597b87275ed4e1724a94a9c31d90b)

**Direct Known Superclasses (Generalization)**

[ResourceReference](#_1b2eec63ad4ef6c72d57b9985e0346ff)

**Associations**

• private base\_Enumeration : Enumeration

### <Stereotype> ValueSetReference

**Diagrams**

[Resource References](#_aa2597b87275ed4e1724a94a9c31d90b)

**Direct Known Superclasses (Generalization)**

[ResourceReference](#_1b2eec63ad4ef6c72d57b9985e0346ff)

**Associations**

• private base\_Enumeration : Enumeration

**Constraints**

* **definition**

This Enumeration must have exactly one > Generalization.

[OCL]

self.base\_Enumeration.general->select(g|g.stereotypedBy('ValueSetDefinitionReference'))->size()<=1

### <Stereotype> ValueSetReferenceInstance

**Diagrams**

[Resource References](#_aa2597b87275ed4e1724a94a9c31d90b)

**Direct Known Superclasses (Generalization)**

[ResourceReferenceInstance](#_9d682f32f4917feea358e696d1fd146d)

**Associations**

• private base\_Abstraction : Abstraction

**Constraints**

* **isValueSetReference**

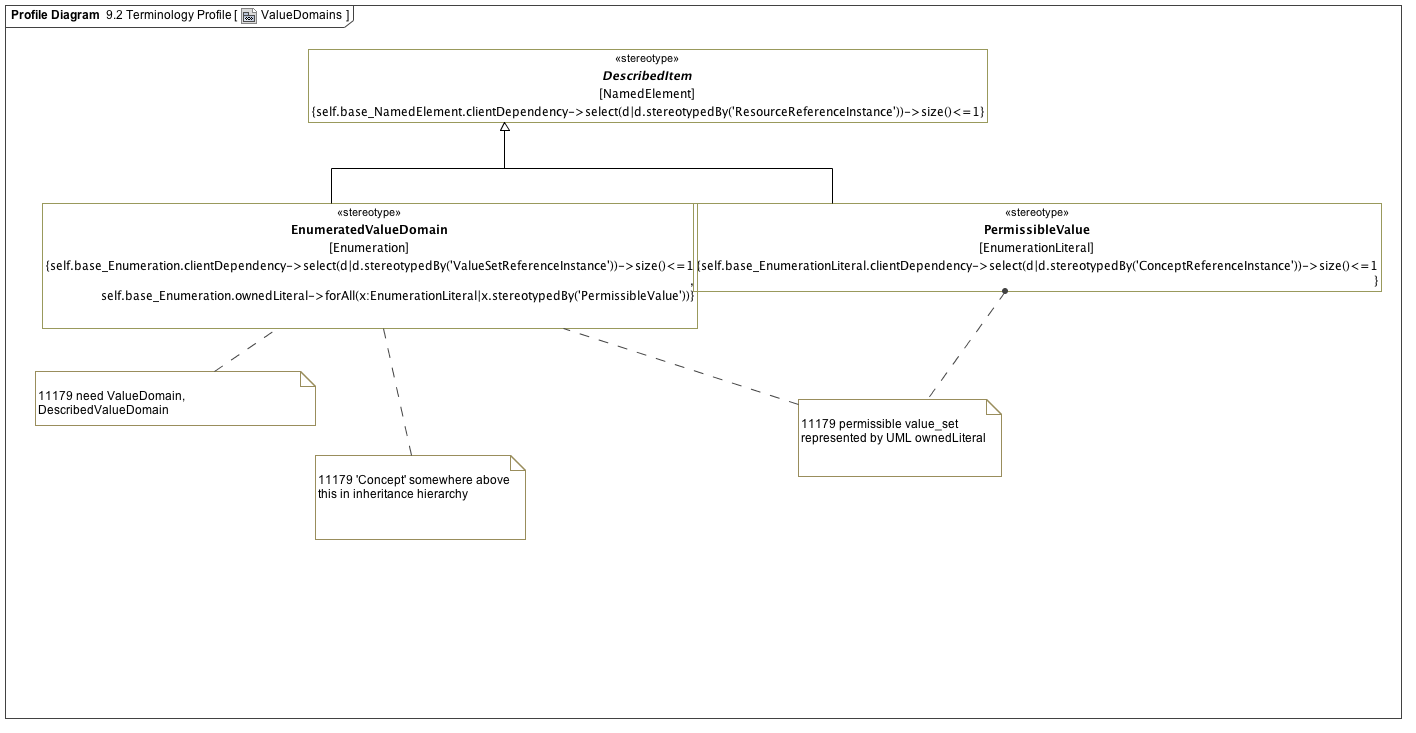
Must have one > client and one > supplier.

[OCL]

self.base\_Abstraction.client->select(c|c.stereotypedBy('EnumeratedValueDomain'))->size()=1 and self.base\_Abstraction.supplier->select(c|c.stereotypedBy('ValueSetReference'))->size()=1

**Known other Stereotypes**

[ConcreteInstanceSpecification](#_27a8967f003efe37629fea8e80ca6c23)



**ValueDomains**

### <Stereotype> EnumeratedValueDomain

**Diagrams**

[ValueDomains](#_bbceab59bf026d3b0616400766b2619e)

**Direct Known Superclasses (Generalization)**

[DescribedItem](#_d45578f848d02aad83980903e5bde7d1)

**Associations**

• private base\_Enumeration : Enumeration

**Constraints**

* **meaningIsConceptReference**

There is a maximum of one > Abstractions.

[OCL]

self.base\_Enumeration.clientDependency->select(d|d.stereotypedBy('ValueSetReferenceInstance'))->size()<=1

* **permissibleValues**

All instances must be type permissibleValue

[OCL]

self.base\_Enumeration.ownedLiteral->forAll(x:EnumerationLiteral|x.stereotypedBy('PermissibleValue'))

### <Stereotype> PermissibleValue

**Diagrams**

[ValueDomains](#_bbceab59bf026d3b0616400766b2619e)

**Direct Known Superclasses (Generalization)**

[DescribedItem](#_d45578f848d02aad83980903e5bde7d1)

**Associations**

• private base\_EnumerationLiteral : EnumerationLiteral

**Constraints**

* **valueMeaningIsConceptReference**

PermissibleValue.about must be a concept reference

[OCL]

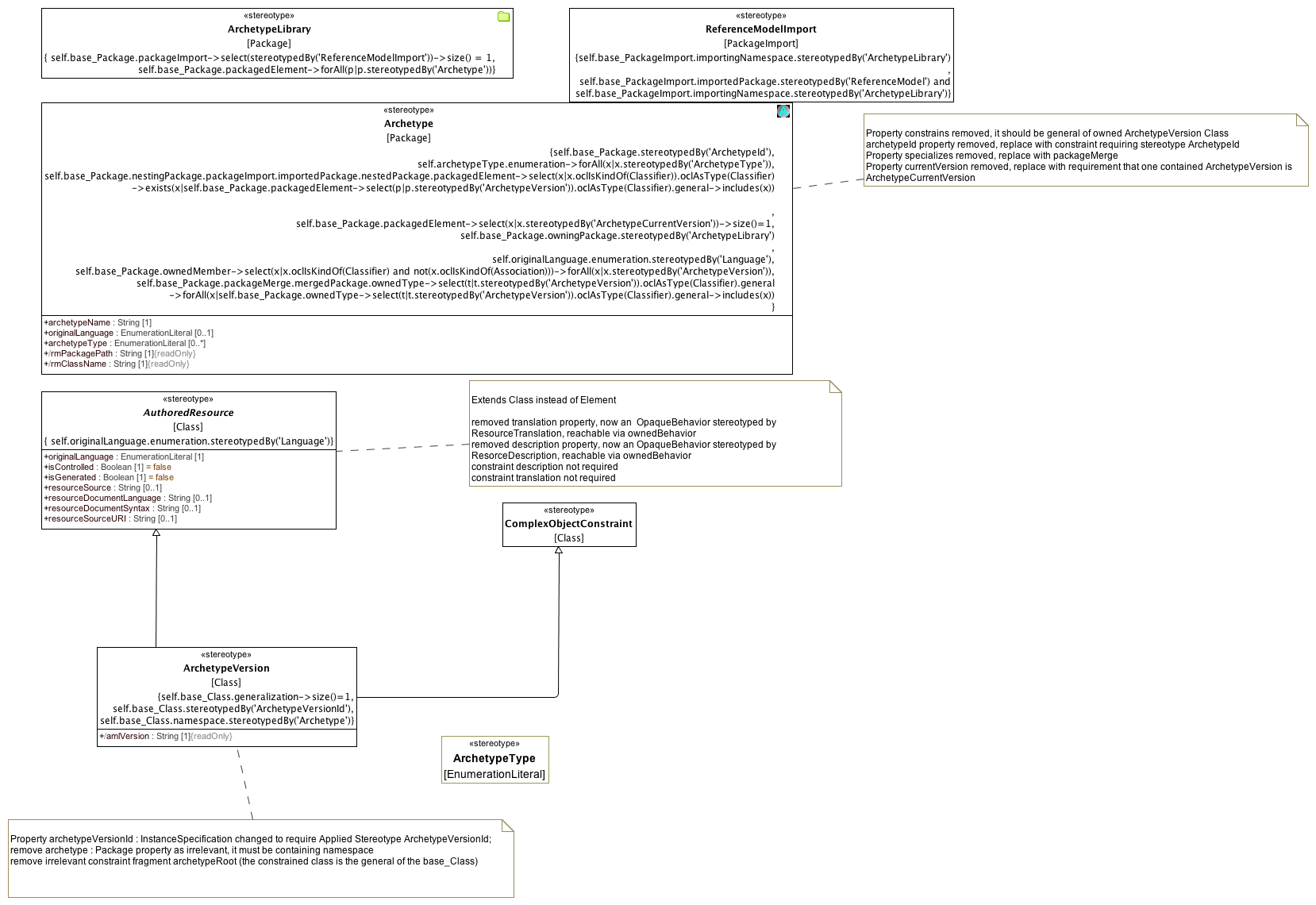
self.base\_EnumerationLiteral.clientDependency->select(d|d.stereotypedBy('ConceptReferenceInstance'))->size()<=1

**Known other Stereotypes**

[DescribedItem](#_d45578f848d02aad83980903e5bde7d1)

## <Package> Constraint Profile

### <Package> Archetypes



**Archetypes**

### <Stereotype> Archetype

**Description**

An Archetype is a package that contains a versioned set of constraints on a Class that is a member of the Reference Model that is owned by the containing Archetype Library.

**Diagrams**

[Archetypes](#_1c8c4aa3906888e5c2c8895f5ed19903)

**Attributes**

• public archetypeName : String [1]

• public originalLanguage : EnumerationLiteral [0..1]

• public archetypeType : EnumerationLiteral [0..\*]

• public rmPackagePath : String [1]

• public rmClassName : String [1]

**Associations**

• private base\_Package : Package

**Constraints**

* **mustBeOwned**

The containing Package must be an >

[OCL]

self.base\_Package.owningPackage.stereotypedBy('ArchetypeLibrary')

* **ownsVersions**

All members of a package must be >.

[OCL]

self.base\_Package.ownedMember->select(x|x.oclIsKindOf(Classifier) and not(x.oclIsKindOf(Association)))->forAll(x|x.stereotypedBy('ArchetypeVersion'))

* **originalLanguage**

originalLanguage must be contained by a > Enumeration.

[OCL]

self.originalLanguage.enumeration.stereotypedBy('Language')

* **archetypeType**

Each archetypeType must be an >

[OCL]

self.archetypeType.enumeration->forAll(x|x.stereotypedBy('ArchetypeType'))

* **constrainsRMElement**

The Class or one of its generalization ancestors is a member of the ArchetypeLibrary Reference Model

[OCL]

self.base\_Package.nestingPackage.packageImport.importedPackage.nestedPackage.packagedElement->select(x|x.oclIsKindOf(Classifier)).oclAsType(Classifier) ->exists(x|self.base\_Package.packagedElement->select(p|p.stereotypedBy('ArchetypeVersion')).oclAsType(Classifier).general->includes(x))

* **currentVersion**

The package must contain one >.

[OCL]

self.base\_Package.packagedElement->select(x|x.stereotypedBy('ArchetypeCurrentVersion'))->size()=1

* **specializesArchetype**

If an Archetype specializes another Archetype, they both constrain the same class.

[OCL]

self.base\_Package.packageMerge.mergedPackage.ownedType->select(t|t.stereotypedBy('ArchetypeVersion')).oclAsType(Classifier).general ->forAll(x|self.base\_Package.ownedType->select(t|t.stereotypedBy('ArchetypeVersion')).oclAsType(Classifier).general->includes(x))

* **archetypeIdType**

The Package must be stereotyped by ArchetypeId.

[OCL]

self.base\_Package.stereotypedBy('ArchetypeId')

### <Stereotype> ArchetypeLibrary

**Description**

An ArchetypeLibrary is a Package that contains a collection of archetypes that constrain classes within the same Reference Model. An ArchetypeLibrary must import exactly one reference model.

**Diagrams**

[Archetypes](#_1c8c4aa3906888e5c2c8895f5ed19903)

**Associations**

• private base\_Package : Package

**Constraints**

* **oneReferenceModel**

The must be one > PackageImport.

[OCL]

self.base\_Package.packageImport->select(stereotypedBy('ReferenceModelImport'))->size() = 1

* **onlyArchetypes**

All packaged elements must be >.

[OCL]

self.base\_Package.packagedElement->forAll(p|p.stereotypedBy('Archetype'))

### <Stereotype> ArchetypeType

**Diagrams**

[Archetypes](#_1c8c4aa3906888e5c2c8895f5ed19903)

**Associations**

• private base\_EnumerationLiteral : EnumerationLiteral

### <Stereotype> ArchetypeVersion

**Diagrams**

[Archetypes](#_1c8c4aa3906888e5c2c8895f5ed19903)

**Direct Known Superclasses (Generalization)**

[AuthoredResource](#_13ad6987e15b787d385f0b30ff25d6c9), [ComplexObjectConstraint](#_bd9b14c4d7198d36c5a9dec9c2836b62)

**Attributes**

• public amlVersion : String [1]

**Associations**

• private base\_Class : Class

**Constraints**

* **archetypeRoot**

This Class must have a generalization.

[OCL]

self.base\_Class.generalization->size()=1

* **ownedByArchetype**

The namespace owner must be >

[OCL]

self.base\_Class.namespace.stereotypedBy('Archetype')

* **archetypeVersionIdType**

This Class must also be stereotyped by >.

[OCL]

self.base\_Class.stereotypedBy('ArchetypeVersionId')

### <Stereotype> AuthoredResource

**Diagrams**

[Archetypes](#_1c8c4aa3906888e5c2c8895f5ed19903)

**Direct Known Subclasses (Specialization)**

[ArchetypeVersion](#_61831f1c446a753e3069251f603bfa37)

**Attributes**

• public originalLanguage : EnumerationLiteral [1]

• public isControlled : Boolean [1] = false

• public isGenerated : Boolean [1] = false

• public resourceSource : String [0..1]

• public resourceDocumentLanguage : String [0..1]

• public resourceDocumentSyntax : String [0..1]

• public resourceSourceURI : String [0..1]

**Associations**

• private base\_Class : Class

**Constraints**

* **languageEnumeration**

originalLanguage must be in a > Enumeration

[OCL]

self.originalLanguage.enumeration.stereotypedBy('Language')

### <Stereotype> ComplexObjectConstraint

**Description**

A ComplexObjectConstraint is a specialization of a single UML Class. A ComplexObjectConstraint may restrict the cardinality or possible values for one or more of the general class properties.

**Diagrams**

[Archetypes](#_1c8c4aa3906888e5c2c8895f5ed19903)

**Direct Known Superclasses (Generalization)**

[ObjectConstraint](#_ad75af95f635bdf35f69d9db9b17aae2)

**Direct Known Subclasses (Specialization)**

[ArchetypeVersion](#_61831f1c446a753e3069251f603bfa37)

**Associations**

• private base\_Class : Class

**Constraints**

* **singleParent**

Every constraint must specialize exactly one Class

[OCL]

self.base\_Class.generalization->size() = 1

* **allAttributeConstraints**

All owned attributes must be >

[OCL]

self.base\_Class.ownedAttribute->forAll(x|x.stereotypedBy('AttributeConstraint'))

### <Stereotype> ReferenceModelImport

**Description**

ReferenceModelImport is a PackageImport where the importingNamespace is an instance of an ArchetypeLibrary and the importedPackage is an instance of a ReferenceModel.

**Diagrams**

[Archetypes](#_1c8c4aa3906888e5c2c8895f5ed19903)

**Associations**

• private base\_PackageImport : PackageImport

**Constraints**

* **libraryOnly**

importing namespace must be an >

[OCL]

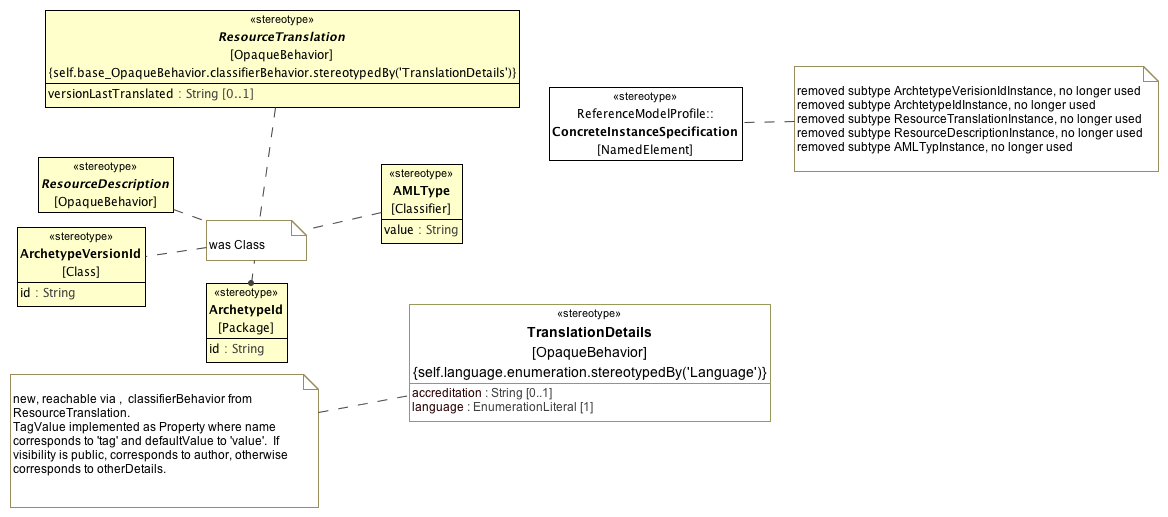
self.base\_PackageImport.importingNamespace.stereotypedBy('ArchetypeLibrary')

* **libraryReferenceModel**

importedPackage must be > and importNamespace must be >

[OCL]

self.base\_PackageImport.importedPackage.stereotypedBy('ReferenceModel') and self.base\_PackageImport.importingNamespace.stereotypedBy('ArchetypeLibrary')



**Archetype Metadata**

### <Stereotype> AMLType

**Description**

An AMLType represents a "primitive" or "atomic" type in the AML specification. An instance of an AMLType is identified solely by its value. AMLType is not a specialization of the UML DataType because it may need to be represented as a specialization of UML Class in a Reference Model binding.

**Diagrams**

[Archetype Metadata](#_c8719d73828cfab6778459fc65cfee21)

**Attributes**

• public value : String

**Associations**

• private base\_Classifier : Classifier

### <Stereotype> ArchetypeId

**Diagrams**

[Archetype Metadata](#_c8719d73828cfab6778459fc65cfee21)

**Attributes**

• public id : String

**Associations**

• private base\_Package : Package

### <Stereotype> ArchetypeVersionId

**Diagrams**

[Archetype Metadata](#_c8719d73828cfab6778459fc65cfee21)

**Attributes**

• public id : String

**Associations**

• private base\_Class : Class

### <Stereotype> ResourceDescription

**Diagrams**

[Archetype Metadata](#_c8719d73828cfab6778459fc65cfee21)

**Associations**

• private base\_OpaqueBehavior : OpaqueBehavior

### <Stereotype> ResourceTranslation

**Diagrams**

[Archetype Metadata](#_c8719d73828cfab6778459fc65cfee21)

**Attributes**

• public versionLastTranslated : String [0..1]

**Associations**

• private base\_OpaqueBehavior : OpaqueBehavior

**Constraints**

* **hasOneTranslationDetails**

classifierBehavior must be >

[OCL]

self.base\_OpaqueBehavior.classifierBehavior.stereotypedBy('TranslationDetails')

### <Stereotype> TranslationDetails

**Diagrams**

[Archetype Metadata](#_c8719d73828cfab6778459fc65cfee21)

**Attributes**

• public accreditation : String [0..1]

• public language : EnumerationLiteral [1]

**Associations**

• private base\_OpaqueBehavior : OpaqueBehavior

**Constraints**

* **languageEnumeration**

language must be contained by a > Enumeration

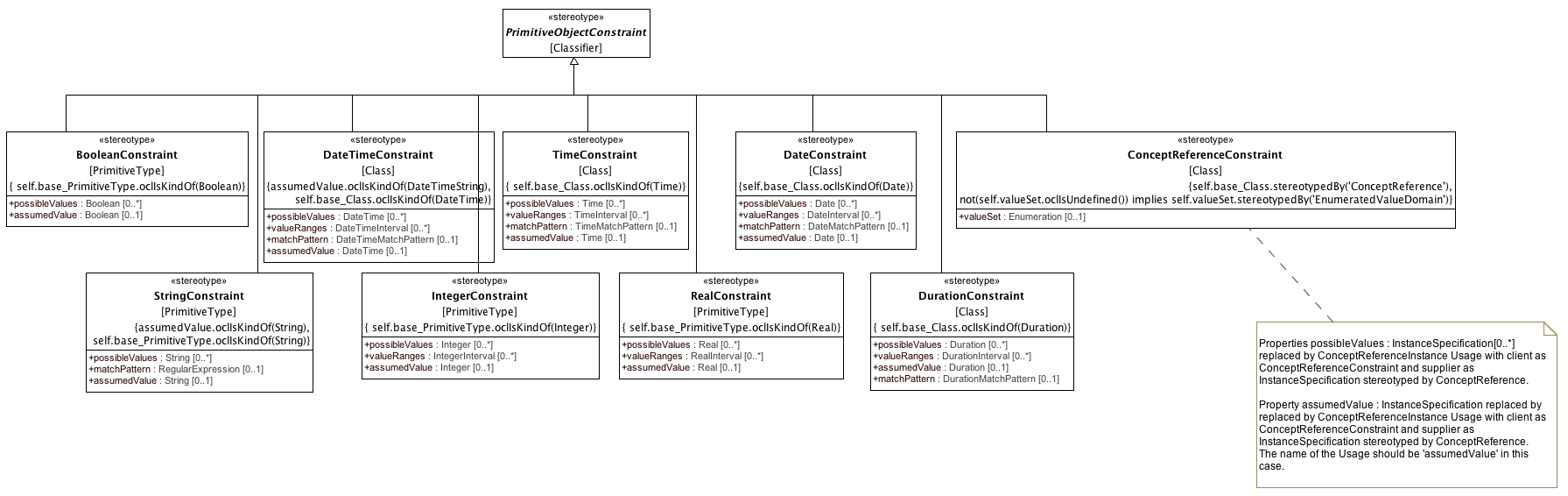
[OCL]

self.language.enumeration.stereotypedBy('Language')

**Known other Stereotypes**

[ConcreteInstanceSpecification](#_27a8967f003efe37629fea8e80ca6c23)

### <Package> Data Type Constraints



**Primitive Type Constraints**

### <Stereotype> BooleanConstraint

**Diagrams**

[Primitive Type Constraints](#_ae5d1776f07899a7af4725351a338b35)

**Direct Known Superclasses (Generalization)**

[PrimitiveObjectConstraint](#_c72b6d9c8a46b96f02fdfefe3b8b0568)

**Attributes**

• public possibleValues : Boolean [0..\*]

• public assumedValue : Boolean [0..1]

**Associations**

• private base\_PrimitiveType : PrimitiveType

**Constraints**

* **constrainsBoolean**

[English]

self.base\_PrimitiveType.oclIsKindOf(Boolean)

### <Stereotype> ConceptReferenceConstraint

**Diagrams**

[Primitive Type Constraints](#_ae5d1776f07899a7af4725351a338b35)

**Direct Known Superclasses (Generalization)**

[PrimitiveObjectConstraint](#_c72b6d9c8a46b96f02fdfefe3b8b0568)

**Attributes**

• public valueSet : Enumeration [0..1]

**Associations**

• private base\_Class : Class

**Constraints**

* **mustBeEnumeratedValueDomain**

valueSet, if defined, must be an >

[OCL]

not(self.valueSet.oclIsUndefined()) implies self.valueSet.stereotypedBy('EnumeratedValueDomain')

* **constrainsConceptReference**

This Class must be a >

[OCL]

self.base\_Class.stereotypedBy('ConceptReference')

### <Stereotype> DateConstraint

**Diagrams**

[Primitive Type Constraints](#_ae5d1776f07899a7af4725351a338b35)

**Direct Known Superclasses (Generalization)**

[PrimitiveObjectConstraint](#_c72b6d9c8a46b96f02fdfefe3b8b0568)

**Attributes**

• public possibleValues : [Date](#_78ee642abf9938398776ce11b2ae5595) [0..\*]

• public valueRanges : [DateInterval](#_eb65cb2938a6220d8f4a10f0d8aba136) [0..\*]

• public matchPattern : [DateMatchPattern](#_d029702cd1184b62db7fbbb690187cd5) [0..1]

• public assumedValue : [Date](#_78ee642abf9938398776ce11b2ae5595) [0..1]

**Associations**

• private base\_Class : Class

**Constraints**

* **constrainsDate**

[English]

self.base\_Class.oclIsKindOf(Date)

### <Stereotype> DateTimeConstraint

**Diagrams**

[Primitive Type Constraints](#_ae5d1776f07899a7af4725351a338b35)

**Direct Known Superclasses (Generalization)**

[PrimitiveObjectConstraint](#_c72b6d9c8a46b96f02fdfefe3b8b0568)

**Attributes**

• public possibleValues : [DateTime](#_7ba7e85df09d292033e869c3e8664062) [0..\*]

• public valueRanges : [DateTimeInterval](#_956e6c028c830c0453b74cbd2204109e) [0..\*]

• public matchPattern : [DateTimeMatchPattern](#_0fe09862b0893b7f5636966c344bc986) [0..1]

• public assumedValue : [DateTime](#_7ba7e85df09d292033e869c3e8664062) [0..1]

**Associations**

• private base\_Class : Class

**Constraints**

* **assumedDateTime**

[English]

assumedValue.oclIsKindOf(DateTimeString)

* **constrainsDateTime**

[English]

self.base\_Class.oclIsKindOf(DateTime)

### <Stereotype> DurationConstraint

**Diagrams**

[Primitive Type Constraints](#_ae5d1776f07899a7af4725351a338b35)

**Direct Known Superclasses (Generalization)**

[PrimitiveObjectConstraint](#_c72b6d9c8a46b96f02fdfefe3b8b0568)

**Attributes**

• public possibleValues : [Duration](#_6f1e8a2b40ce6a6203e07d9c5daded71) [0..\*]

• public valueRanges : [DurationInterval](#_3d4fd0ce80d2a2e88d2f42b3cb7dbec5) [0..\*]

• public assumedValue : [Duration](#_6f1e8a2b40ce6a6203e07d9c5daded71) [0..1]

• public matchPattern : [DurationMatchPattern](#_f432daf955c8241a03bff56a5a7d8e66) [0..1]

**Associations**

• private base\_Class : Class

**Constraints**

* **constrainsDuration**

[English]

self.base\_Class.oclIsKindOf(Duration)

### <Stereotype> IntegerConstraint

**Diagrams**

[Primitive Type Constraints](#_ae5d1776f07899a7af4725351a338b35)

**Direct Known Superclasses (Generalization)**

[PrimitiveObjectConstraint](#_c72b6d9c8a46b96f02fdfefe3b8b0568)

**Attributes**

• public possibleValues : Integer [0..\*]

• public valueRanges : [IntegerInterval](#_86618450de28d822bd6b57b67a32ab2b) [0..\*]

• public assumedValue : Integer [0..1]

**Associations**

• private base\_PrimitiveType : PrimitiveType

**Constraints**

* **assumedInteger**

[English]

assumedValue.oclIsKindOf(Integer)

* **constrainsInteger**

[English]

self.base\_PrimitiveType.oclIsKindOf(Integer)

### <Stereotype> PrimitiveObjectConstraint

**Diagrams**

[Primitive Type Constraints](#_ae5d1776f07899a7af4725351a338b35)

**Direct Known Superclasses (Generalization)**

[ObjectConstraint](#_ad75af95f635bdf35f69d9db9b17aae2)

**Direct Known Subclasses (Specialization)**

[BooleanConstraint](#_40ee863e6fd02692437dae1d81ba12de), [ConceptReferenceConstraint](#_ef76317db67a290898f39af3c51eee9c), [DateConstraint](#_ff8930d68c378c02c221704764a5c9d4), [DateTimeConstraint](#_7dde1322feeec9c32a95df44c39d8e48), [DurationConstraint](#_384c080719f5bd1b45eae1293215b466), [IntegerConstraint](#_2219fb1dcaf5f26a0ed07de77d69cd5e), [RealConstraint](#_b921bc493035fb4e067213114372e254), [StringConstraint](#_982033c222702fafb1d4d3ed7b399317), [TimeConstraint](#_d8c772ca77efc45bee8711f1de17afc0)

**Associations**

• private base\_Classifier : Classifier

### <Stereotype> RealConstraint

**Diagrams**

[Primitive Type Constraints](#_ae5d1776f07899a7af4725351a338b35)

**Direct Known Superclasses (Generalization)**

[PrimitiveObjectConstraint](#_c72b6d9c8a46b96f02fdfefe3b8b0568)

**Attributes**

• public possibleValues : Real [0..\*]

• public valueRanges : [RealInterval](#_d4f7314ff920dd15ee0e834cfbd4e6f2) [0..\*]

• public assumedValue : Real [0..1]

**Associations**

• private base\_PrimitiveType : PrimitiveType

**Constraints**

* **constrainsReal**

[English]

self.base\_PrimitiveType.oclIsKindOf(Real)

### <Stereotype> StringConstraint

**Diagrams**

[Primitive Type Constraints](#_ae5d1776f07899a7af4725351a338b35)

**Direct Known Superclasses (Generalization)**

[PrimitiveObjectConstraint](#_c72b6d9c8a46b96f02fdfefe3b8b0568)

**Attributes**

• public possibleValues : String [0..\*]

• public matchPattern : [RegularExpression](#_7dba62a18c5b1da31d9d2e3df675a297) [0..1]

• public assumedValue : String [0..1]

**Associations**

• private base\_PrimitiveType : PrimitiveType

**Constraints**

* **assumedString**

The assumed value must be a String

[English]

assumedValue.oclIsKindOf(String)

* **constrainsString**

[English]

self.base\_PrimitiveType.oclIsKindOf(String)

### <Stereotype> TimeConstraint

**Diagrams**

[Primitive Type Constraints](#_ae5d1776f07899a7af4725351a338b35)

**Direct Known Superclasses (Generalization)**

[PrimitiveObjectConstraint](#_c72b6d9c8a46b96f02fdfefe3b8b0568)

**Attributes**

• public possibleValues : [Time](#_cba83b2c77167c96697f3caaa1886f5c) [0..\*]

• public valueRanges : [TimeInterval](#_2db4f3574d756c0312a2a6559efd3ad9) [0..\*]

• public matchPattern : [TimeMatchPattern](#_4b701856b8ef798f1dd89caa4d9efe11) [0..1]

• public assumedValue : [Time](#_cba83b2c77167c96697f3caaa1886f5c) [0..1]

**Associations**

• private base\_Class : Class

**Constraints**

* **constrainsTime**

[English]

self.base\_Class.oclIsKindOf(Time)



**Date and Time Match Types**



**Intervals**

### <Class> DateInterval

**Diagrams**

[Intervals](#_8566828d89ca01e8919a51d19aa6cd8b)

**Direct Known Superclasses (Generalization)**

[Interval](#_3ae971b2839139d9692e47ec472148b6)

**Attributes**

• public lower : [Date](#_78ee642abf9938398776ce11b2ae5595) [0..1]

• public upper : [Date](#_78ee642abf9938398776ce11b2ae5595) [0..1]

### <Class> DateTimeInterval

**Diagrams**

[Intervals](#_8566828d89ca01e8919a51d19aa6cd8b)

**Direct Known Superclasses (Generalization)**

[Interval](#_3ae971b2839139d9692e47ec472148b6)

**Attributes**

• public lower : [DateTime](#_7ba7e85df09d292033e869c3e8664062) [0..1]

• public upper : [DateTime](#_7ba7e85df09d292033e869c3e8664062) [0..1]

### <Class> DurationInterval

**Diagrams**

[Intervals](#_8566828d89ca01e8919a51d19aa6cd8b)

**Direct Known Superclasses (Generalization)**

[Interval](#_3ae971b2839139d9692e47ec472148b6)

**Attributes**

• public lower : [Duration](#_6f1e8a2b40ce6a6203e07d9c5daded71) [0..1]

• public upper : Duration [0..1]

### <Class> IntegerInterval

**Diagrams**

[Intervals](#_8566828d89ca01e8919a51d19aa6cd8b)

**Direct Known Superclasses (Generalization)**

[Interval](#_3ae971b2839139d9692e47ec472148b6)

**Attributes**

• public lower : Integer [0..1]

• public upper : Integer [0..1]

### <Class> Interval

**Diagrams**

[Intervals](#_8566828d89ca01e8919a51d19aa6cd8b)

**Direct Known Subclasses (Specialization)**

[DateInterval](#_eb65cb2938a6220d8f4a10f0d8aba136), [DateTimeInterval](#_956e6c028c830c0453b74cbd2204109e), [DurationInterval](#_3d4fd0ce80d2a2e88d2f42b3cb7dbec5), [IntegerInterval](#_86618450de28d822bd6b57b67a32ab2b), [RealInterval](#_d4f7314ff920dd15ee0e834cfbd4e6f2), [TimeInterval](#_2db4f3574d756c0312a2a6559efd3ad9)

**Attributes**

• public lowerIncluded : Boolean [1]

• public upperIncluded : Boolean [1]

### <Class> RealInterval

**Diagrams**

[Intervals](#_8566828d89ca01e8919a51d19aa6cd8b)

**Direct Known Superclasses (Generalization)**

[Interval](#_3ae971b2839139d9692e47ec472148b6)

**Attributes**

• public lower : Real [0..1]

• public upper : Real [0..1]

### <Class> TimeInterval

**Diagrams**

[Intervals](#_8566828d89ca01e8919a51d19aa6cd8b)

**Direct Known Superclasses (Generalization)**

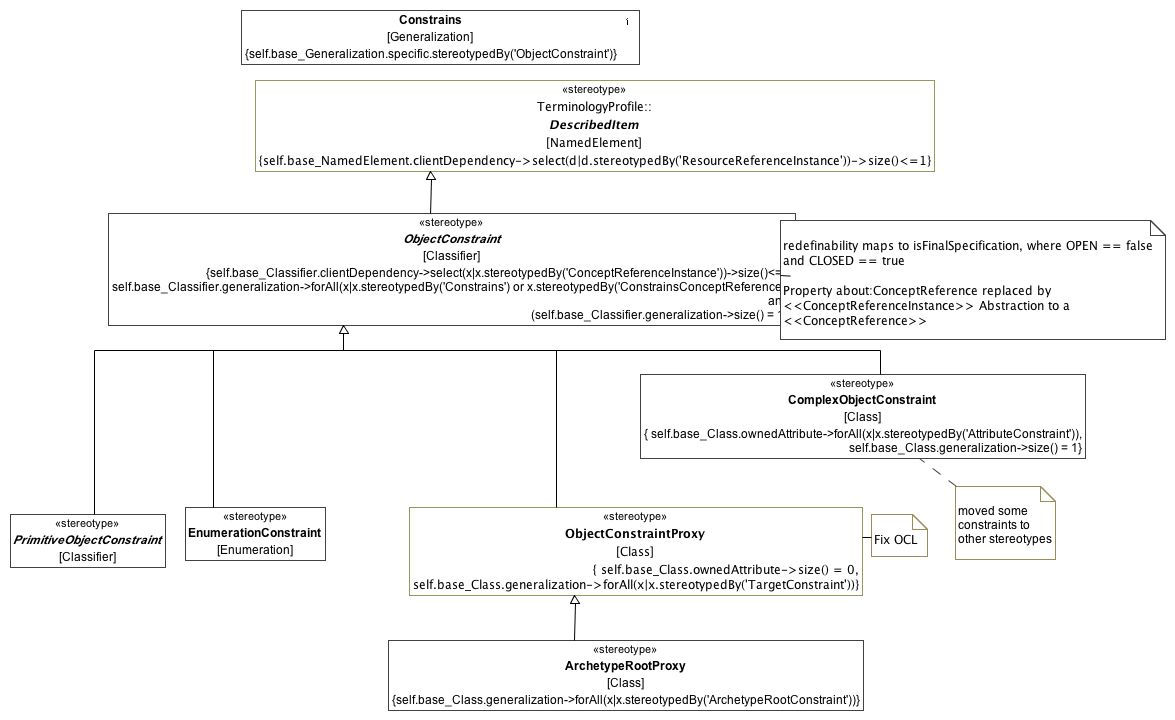
[Interval](#_3ae971b2839139d9692e47ec472148b6)

**Attributes**

• public lower : [Time](#_cba83b2c77167c96697f3caaa1886f5c) [0..1]

• public upper : [Time](#_cba83b2c77167c96697f3caaa1886f5c) [0..1]

### <Package> Object and Property Constraints



**Constraints**

### <Stereotype> ArchetypeRootProxy

**Diagrams**

[Constraints](#_f139650e10793c0005c93b604992495c)

**Direct Known Superclasses (Generalization)**

[ObjectConstraintProxy](#_c8ae60f7f44b70cf5dce7db03aa6ac1e)

**Associations**

• private base\_Class : Class

**Constraints**

* **redefinesConstrains**

Any generalization must be >

[OCL]

self.base\_Class.generalization->forAll(x|x.stereotypedBy('ArchetypeRootConstraint'))

### <Stereotype> Constrains

**Diagrams**

[Constraints](#_f139650e10793c0005c93b604992495c)

**Direct Known Subclasses (Specialization)**

[TargetConstraint](#_6de94cd3c6736f017766fe61020a5a13)

**Associations**

• private base\_Generalization : Generalization

**Constraints**

* **specificObjectConstraint**

Specific must be an >

[OCL]

self.base\_Generalization.specific.stereotypedBy('ObjectConstraint')

### <Stereotype> EnumerationConstraint

**Description**

**Diagrams**

[Constraints](#_f139650e10793c0005c93b604992495c)

**Direct Known Superclasses (Generalization)**

[ObjectConstraint](#_ad75af95f635bdf35f69d9db9b17aae2)

**Direct Known Subclasses (Specialization)**

[EnumeratedValueDomainConstraint](#_7b4688dbd3826f33c726c87847ae4a72)

**Associations**

• private base\_Enumeration : Enumeration

### <Stereotype> ObjectConstraint

**Description**

The Classifier stereotyped by an ObjectConstraint must participate in exactly one Generalization relationship, which must be stereotyped by the Constrains stereotype.

**Diagrams**

[Constraints](#_f139650e10793c0005c93b604992495c)

**Direct Known Superclasses (Generalization)**

[DescribedItem](#_d45578f848d02aad83980903e5bde7d1)

**Direct Known Subclasses (Specialization)**

[ComplexObjectConstraint](#_bd9b14c4d7198d36c5a9dec9c2836b62), [EnumerationConstraint](#_1bc74c3698f61990aff3aec96088f0a9), [ObjectConstraintProxy](#_c8ae60f7f44b70cf5dce7db03aa6ac1e), [PrimitiveObjectConstraint](#_c72b6d9c8a46b96f02fdfefe3b8b0568)

**Associations**

• private base\_Classifier : Classifier

**Constraints**

* **redefinesGeneralization**

This Classifier must have exactly one generalization, and that is a > or > generalization.

[OCL]

self.base\_Classifier.generalization->forAll(x|x.stereotypedBy('Constrains') or x.stereotypedBy('ConstrainsConceptReference')) and (self.base\_Classifier.generalization->size() = 1)

* **about**

at most one "about" reference via > Abstraction

[OCL]

self.base\_Classifier.clientDependency->select(x|x.stereotypedBy('ConceptReferenceInstance'))->size()<=1

### <Stereotype> ObjectConstraintProxy

**Diagrams**

[Constraints](#_f139650e10793c0005c93b604992495c)

**Direct Known Superclasses (Generalization)**

[ObjectConstraint](#_ad75af95f635bdf35f69d9db9b17aae2)

**Direct Known Subclasses (Specialization)**

[ArchetypeRootProxy](#_f5f73ce565f73d8b4808997e54e8e698)

**Associations**

• private base\_Class : Class

**Constraints**

* **redefinesConstrains**

All generals are >

[OCL]

self.base\_Class.generalization->forAll(x|x.stereotypedBy('TargetConstraint'))

* **noNewAttributes**

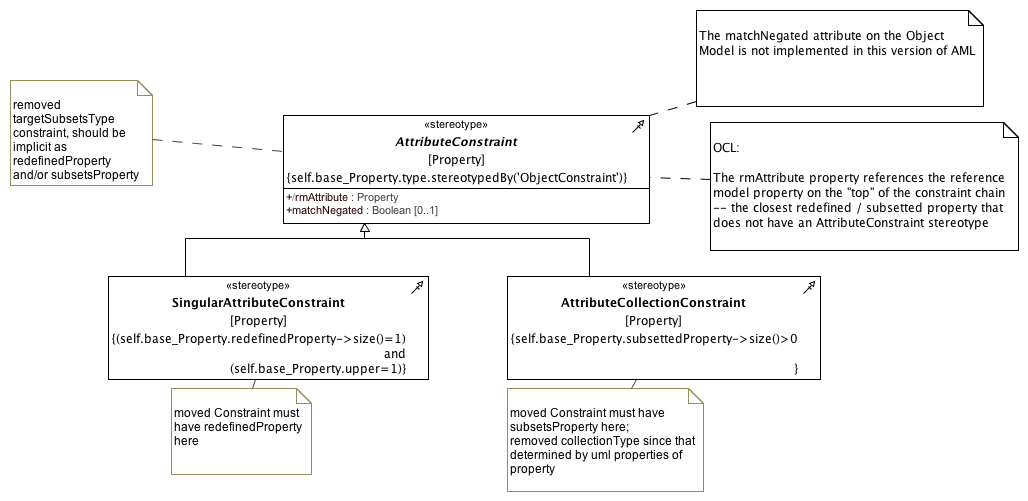
No attributes allowed on this Class

[OCL]

self.base\_Class.ownedAttribute->size() = 0

**Known other Stereotypes**

[ComplexObjectConstraint](#_bd9b14c4d7198d36c5a9dec9c2836b62), [DescribedItem](#_d45578f848d02aad83980903e5bde7d1), [PrimitiveObjectConstraint](#_c72b6d9c8a46b96f02fdfefe3b8b0568)



**Attribute Constraints**

### <Stereotype> AttributeCollectionConstraint

**Diagrams**

[Attribute Constraints](#_88dce20413dd8833c4e90da9fe432855)

**Direct Known Superclasses (Generalization)**

[AttributeConstraint](#_1bf8a3231ae21af2dec84426b5618c38)

**Associations**

• private base\_Property : Property

**Constraints**

* **mustHaveSubsettedProperty**

Must have at least one subsettedProperty.

[OCL]

self.base\_Property.subsettedProperty->size()>0

### <Stereotype> AttributeConstraint

**Diagrams**

[Attribute Constraints](#_88dce20413dd8833c4e90da9fe432855)

**Direct Known Subclasses (Specialization)**

[AttributeCollectionConstraint](#_4bc615eb2707782fc8254702b7e0b435), [SingularAttributeConstraint](#_2d1a6d8b2806092b50ec3fd4cd2db35b)

**Attributes**

• public rmAttribute : Property

• public matchNegated : Boolean [0..1]

**Associations**

• private base\_Property : Property

**Constraints**

* **objectConstraintTarget**

The Property type must be an >

[OCL]

self.base\_Property.type.stereotypedBy('ObjectConstraint')

### <Stereotype> SingularAttributeConstraint

**Diagrams**

[Attribute Constraints](#_88dce20413dd8833c4e90da9fe432855)

**Direct Known Superclasses (Generalization)**

[AttributeConstraint](#_1bf8a3231ae21af2dec84426b5618c38)

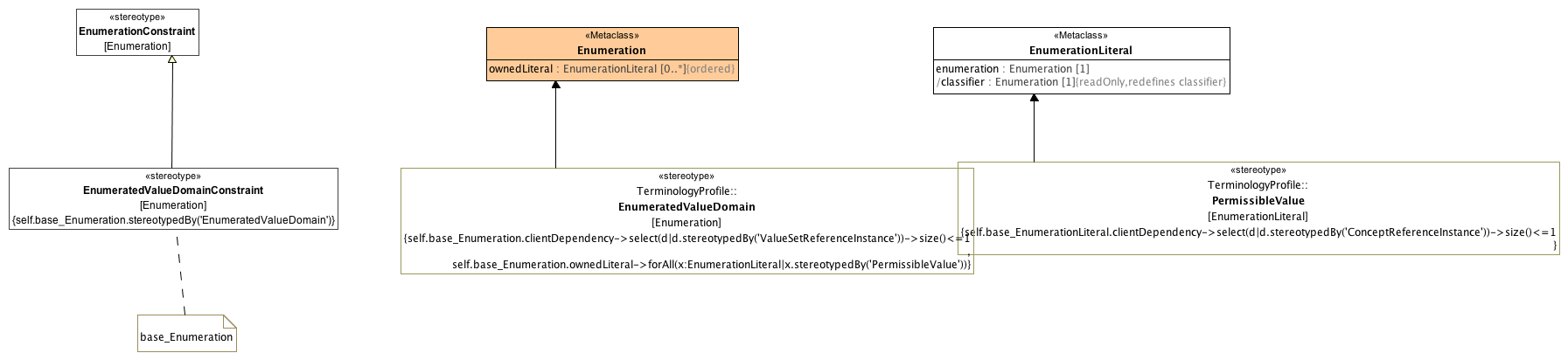
**Constraints**

* **mustHaveRedefinedProperty**

Property must have exactly one redefined Property and an upper bound of 1

[OCL]

(self.base\_Property.redefinedProperty->size()=1) and (self.base\_Property.upper=1)



**Enumeration Constraints**

### <Class> Enumeration

**Diagrams**

[Enumeration Constraints](#_4049ef2e39f1ca7b7abb65f10409d85f)

**Direct Known Superclasses (Generalization)**

[DataType](#_687a198ec4e5e49dcd9a605729dc1c24)

**Attributes**

• public ownedLiteral : EnumerationLiteral [0..\*]

The ordered set of literals owned by this Enumeration.

### <Class> EnumerationLiteral

**Diagrams**

[Enumeration Constraints](#_4049ef2e39f1ca7b7abb65f10409d85f)

**Direct Known Superclasses (Generalization)**

[InstanceSpecification](#_a0d9dad3458f9785e2dd80931202582d)

**Attributes**

• public enumeration : Enumeration [1]

The Enumeration that this EnumerationLiteral is a member of.

• public classifier : Enumeration [1]

### <Stereotype> EnumeratedValueDomainConstraint

**Diagrams**

[Enumeration Constraints](#_4049ef2e39f1ca7b7abb65f10409d85f)

**Direct Known Superclasses (Generalization)**

[EnumerationConstraint](#_1bc74c3698f61990aff3aec96088f0a9)

**Associations**

• private base\_Enumeration : Enumeration

**Constraints**

* **constrainsEVD**

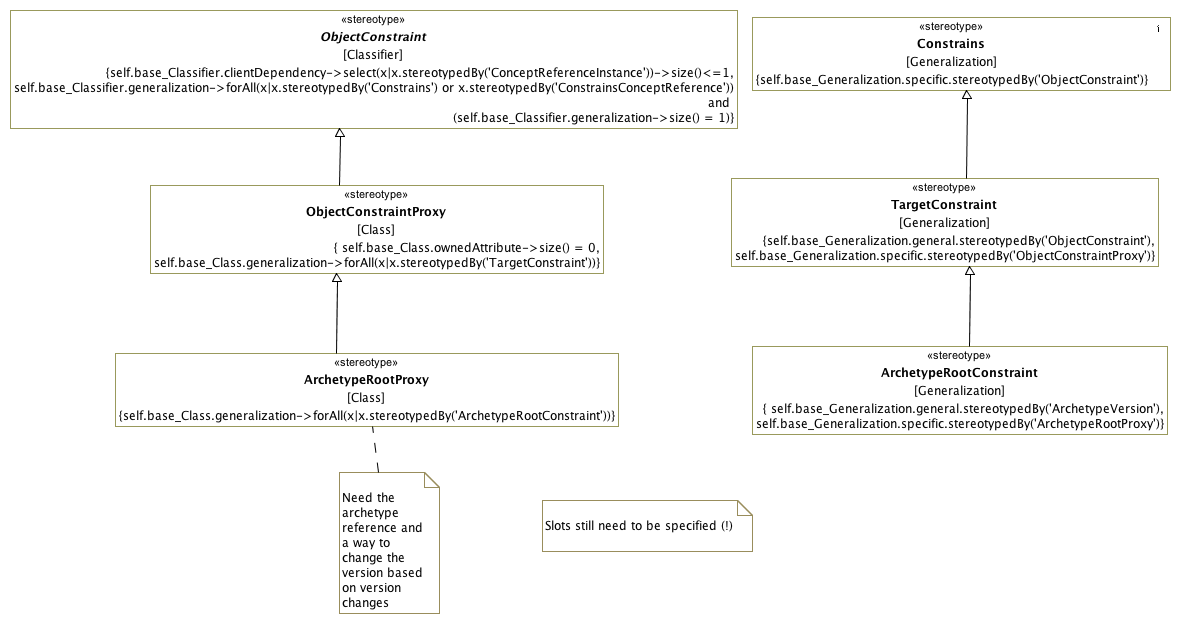
This Enumeration is an >

[OCL]

self.base\_Enumeration.stereotypedBy('EnumeratedValueDomain')

**Known other Stereotypes**

[EnumeratedValueDomain](#_c7f411daaf64f83e013bec437cb8f30a), [EnumerationConstraint](#_1bc74c3698f61990aff3aec96088f0a9), [PermissibleValue](#_5bb7ce8128b60ee5eb2ca275444e9692)



**Constraint Proxies**

### <Stereotype> ArchetypeRootConstraint

**Diagrams**

[Constraint Proxies](#_17547bff44c4353bd3a454a0c3c7e577)

**Direct Known Superclasses (Generalization)**

[TargetConstraint](#_6de94cd3c6736f017766fe61020a5a13)

**Associations**

• private base\_Generalization : Generalization

**Constraints**

* **specificIsRootProxy**

Specific must be an >

[OCL]

self.base\_Generalization.specific.stereotypedBy('ArchetypeRootProxy')

* **generalIsArchetypeVersion**

The general must be an >

[OCL]

self.base\_Generalization.general.stereotypedBy('ArchetypeVersion')

### <Stereotype> TargetConstraint

**Diagrams**

[Constraint Proxies](#_17547bff44c4353bd3a454a0c3c7e577)

**Direct Known Superclasses (Generalization)**

[Constrains](#_f91b532413834ad1de94d0b0af526f5b)

**Direct Known Subclasses (Specialization)**

[ArchetypeRootConstraint](#_6eafe370f24f01390e7ab79d6568ea94)

**Associations**

• private base\_Generalization : Generalization

**Constraints**

* **specificIsProxy**

Specific must be an >

[OCL]

self.base\_Generalization.specific.stereotypedBy('ObjectConstraintProxy')

* **generalIsObjectConstraint**

General must be >

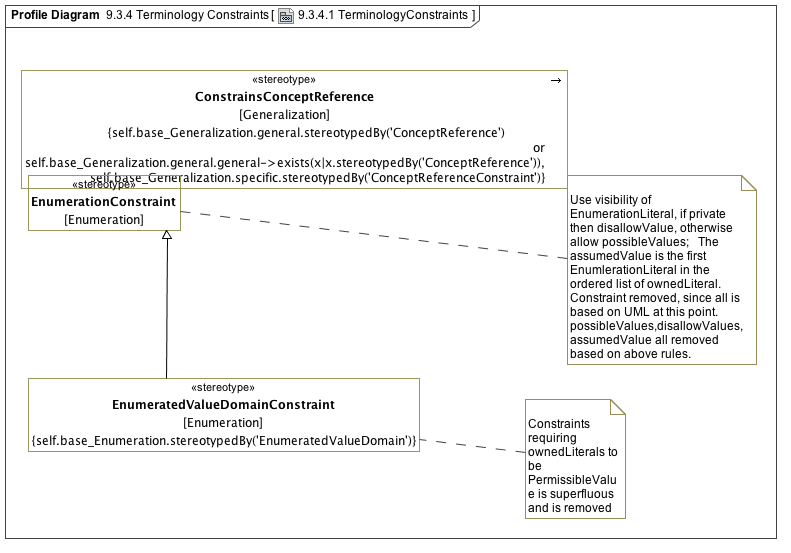
[OCL]

self.base\_Generalization.general.stereotypedBy('ObjectConstraint')

**Known other Stereotypes**

[ArchetypeRootProxy](#_f5f73ce565f73d8b4808997e54e8e698), [Constrains](#_f91b532413834ad1de94d0b0af526f5b), [ObjectConstraint](#_ad75af95f635bdf35f69d9db9b17aae2), [ObjectConstraintProxy](#_c8ae60f7f44b70cf5dce7db03aa6ac1e)

### <Package> Terminology Constraints



**TerminologyConstraints**

### <Stereotype> ConstrainsConceptReference

**Diagrams**

[TerminologyConstraints](#_759880e6f359cf189858504f68956934)

**Associations**

• private base\_Generalization : Generalization

**Constraints**

* **specificCRC**

Specific must be a >

[OCL]

self.base\_Generalization.specific.stereotypedBy('ConceptReferenceConstraint')

* **generalIsConceptReference**

General must be a > or perhaps derived from such

[OCL]

self.base\_Generalization.general.stereotypedBy('ConceptReference') or self.base\_Generalization.general.general->exists(x|x.stereotypedBy('ConceptReference'))

**Known other Stereotypes**

[EnumeratedValueDomainConstraint](#_7b4688dbd3826f33c726c87847ae4a72), [EnumerationConstraint](#_1bc74c3698f61990aff3aec96088f0a9)