DataType model report

DataType Model

Version 1.0 • Proposed

Date/Time Generated: 6/18/2016 9:05:30 AM

Author: Compiled by S. M. Richard

EA Repository : C:\Workspace\Projects\RDA ResearchDataAlliance\DataTypes\DataTypeRegistryModel.eap

Table of Contents

1 Introduction

The scope of this model is the formal representation of information objects that are the basic units of data representation in computer information systems. The model specifies the concept of a DataObject ('type', 'entity', 'object', etc.) that has a collection of attributes, with value domains, data type, and cardinalities for those attributes, constituting the representation of instances of that type/entity. The model distinguishes the conceptual level definition of objects and properties from the implementation of those concepts with a particular representation. Description and documentation of the conceptual level (ObjectClass and Property) is important for interfaces through which domain practitioners interact with data. Description and documentation of the implementation level (DataObject and Attribute) is important for software systems that automate operations on the data. Data types that represent the conceptual objects might be implemented as JSON objects, XML elements, rows in a relation, RDF graphs etc.
This model is a synthesis of a variety of existing models for documenting schema and vocabulary used to define representations of information about entities of interest in the world. Inputs include ISO19110, ISO19115, ISO11179, OGC10-090r3 (NetCDF common data model) and the RDA data Type registry prototype (WG output, March 2015).

1.1 Data Type

For the purposes of this model, the term data type is used to mean "A specification of the representation of a single value in an information system" (http://earthlexicon.sdsc.edu/wiki/Data_type, http://en.wikipedia.org/wiki/Data_type). The use of this term often leads to confusion because it is applied to representations at a conceptual, logical, and physical implementation level, as well as a wide spectrum of granularity, ranging from primitive types like 'integer' or 'character' to complex structured data types like 'ISO19139 metadata record'. The data type concept mighty also be used to denote an information item representing some 'thing' in the domain of interest, or to denote an information item representing a value for a property of some thing in the domain of interest. At the conceptual level, the 'data type' concept is labeled 'ObjectClass', at the logical level the concept is labeled 'DataObject', and in this model, 'Data Type' is reserved for a class that subsumes all the kinds of data structures that may be used to assign values to Attributes of a DataObject (including other DataObjects).

1.2 Target applications

- Reference for communities to document the meaning of entities and attributes in data that they share.
- Discover existing data type and attribute definitions for use in constructing data models, to foster interoperability.
- Machine-assisted data integration, based on identification of matching or ‘integratable’ attribute content.
- Validation of data instances against a type definition.
- Tools that spin up a UI for a particular data type.

2 Diagrams

2.1 Overview diagram

This section presents a proposed model for representing schema for structured data. The Overview presents the major aspects of the model in one summary figure intended to serve as a quick reference to the entire model. The following sections present views focused on particular elements to facilitate understanding the model. It is recommended that one study the detail diagrams first and then return to this summary diagram after studying the different simplified views. The following section describes each class in the model, listed in alphabetic order.

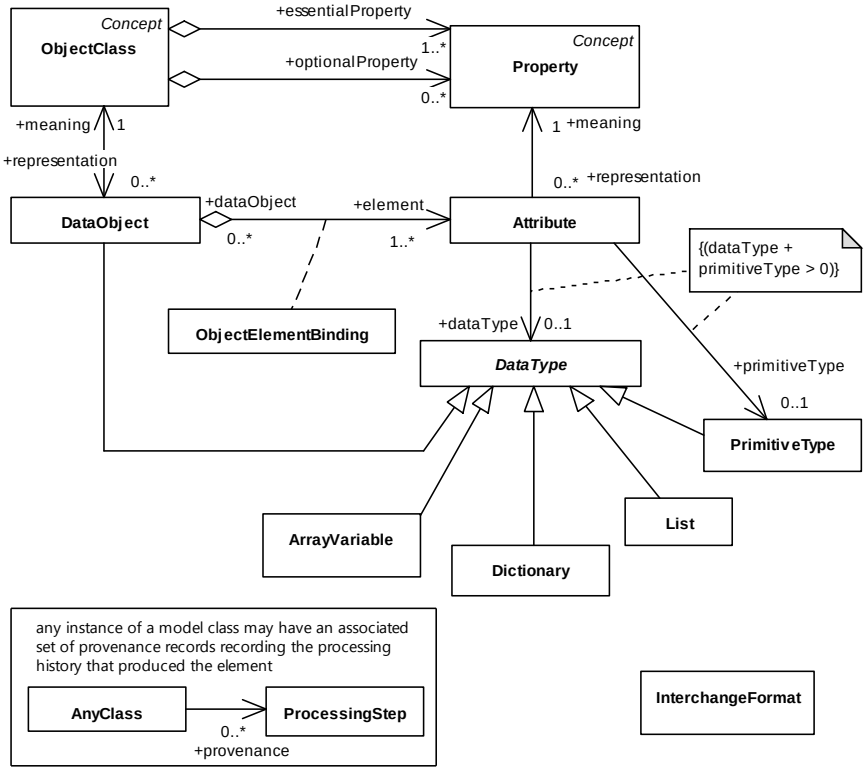

Figure 1: Overview

2.2 Conceptual representation diagram

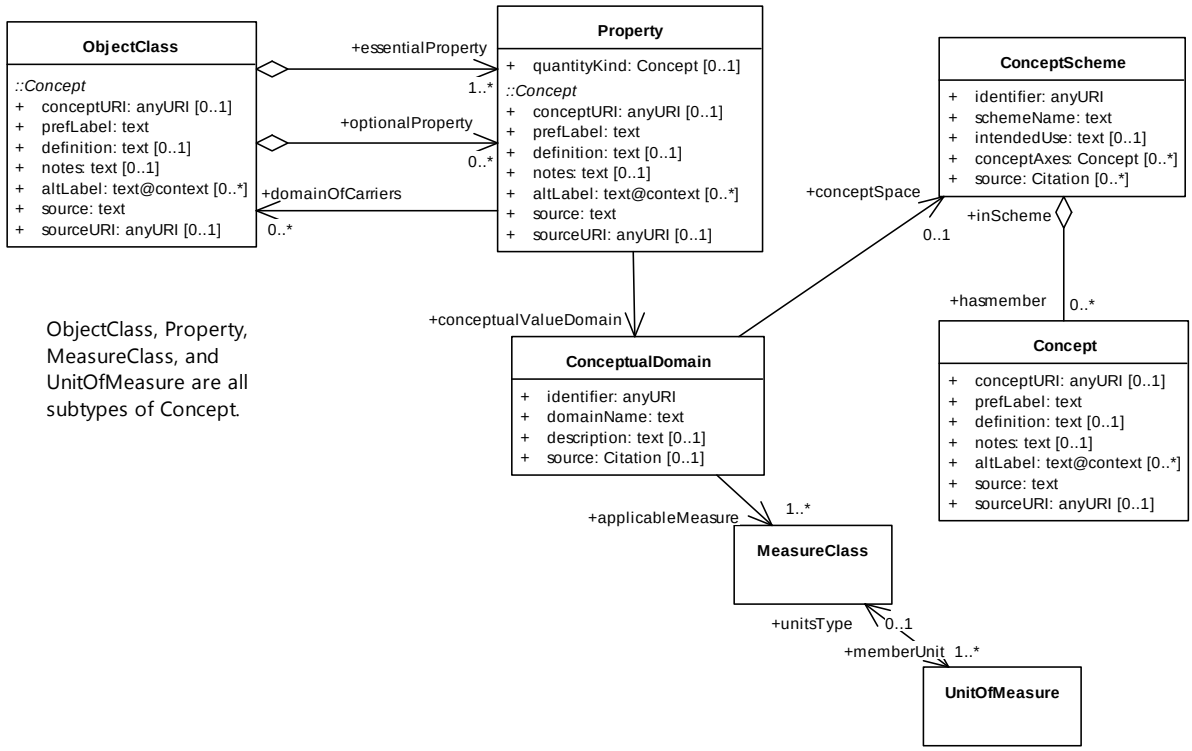

Figure 2: Conceptual representation

2.3 Concepts diagram

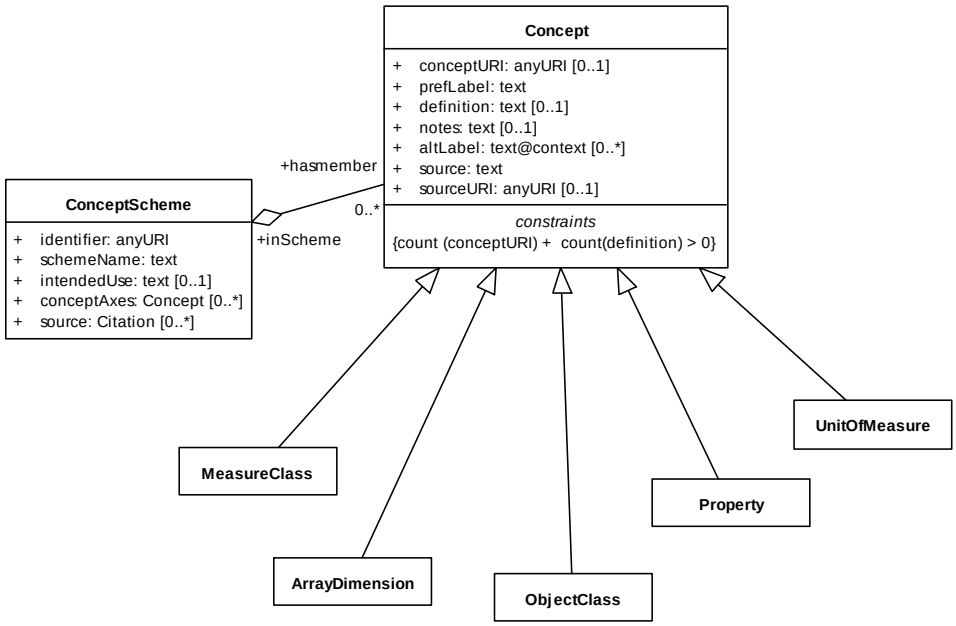

Figure 3: Concepts

2.4 Attribute diagram

In de-normalized implementations, the elements of a DataObject may be distributed into another DataObject. Thus in the table (or simple feature, or csv) a collection of elements that have primitive types (string, boolean, number) may together represent a dataType that is also a DataObject. For instance a US Cadastral location consists of a Tuple {meridian names, Township, Range, Section, SectionPart} that is a DataObject representing a geospatial location property. In some implementations this object may be represented by a single string "GSR T27N, R12W, sec. 12, NWSE", but a common implementation would include separate fields for each part of the location description each as a string data type. In both these cases, the string fields would have a primitiveType = 'string', and a dataType='US Cadastral Location'.

A primitiveType might also be used to implement a LogicalType, for instance a string primitive might implement a list DataType. The List type specifies the data type of the list elements and the delimiter character(s).

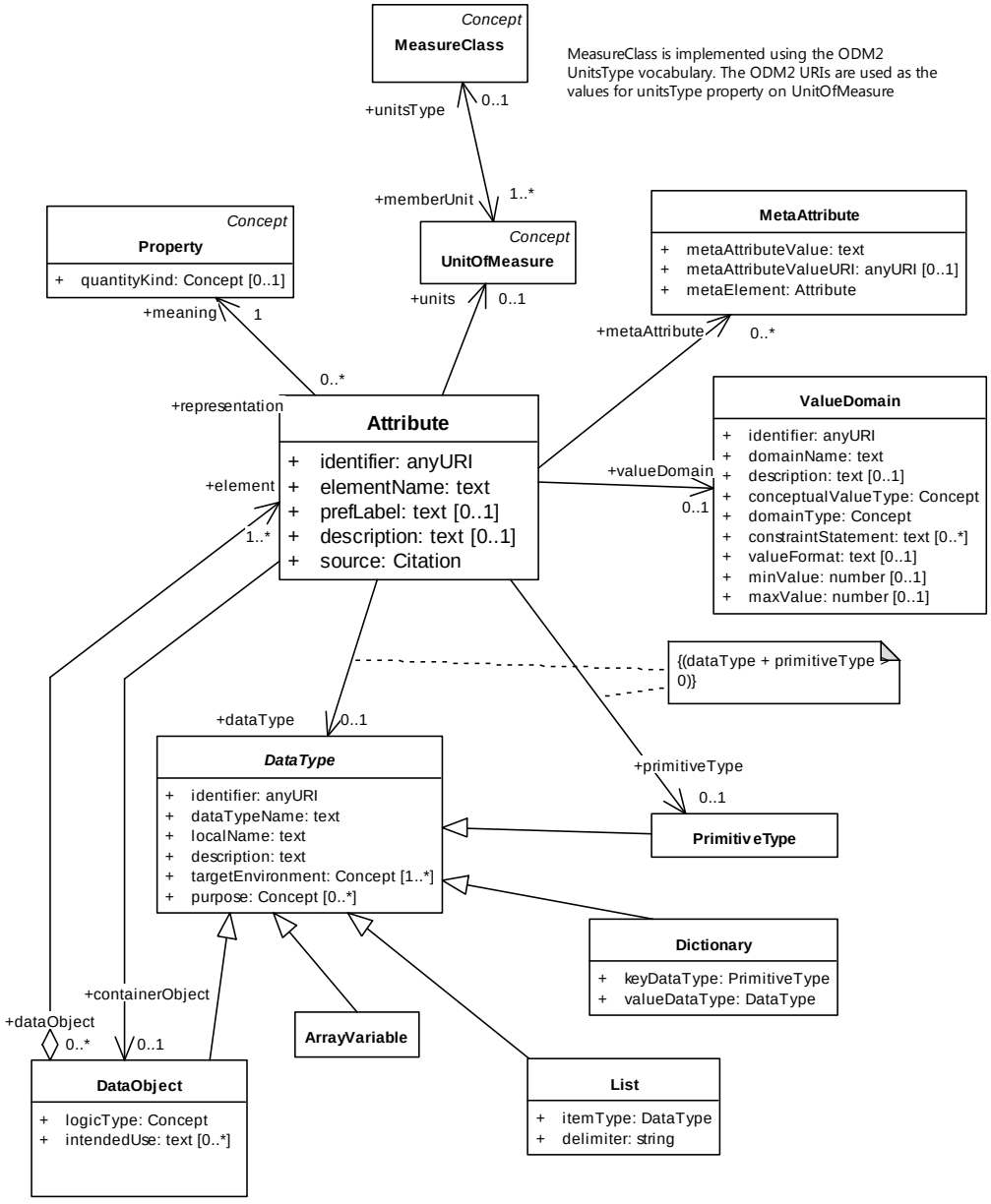

Figure 4: Attribute

2.5 DataObject diagram

A DataObject is a DataType that provides an implementable representation of an ObjectClass. The ObjectClass represents the concept of some entity in a domain of interest that is to be represented in an information system.

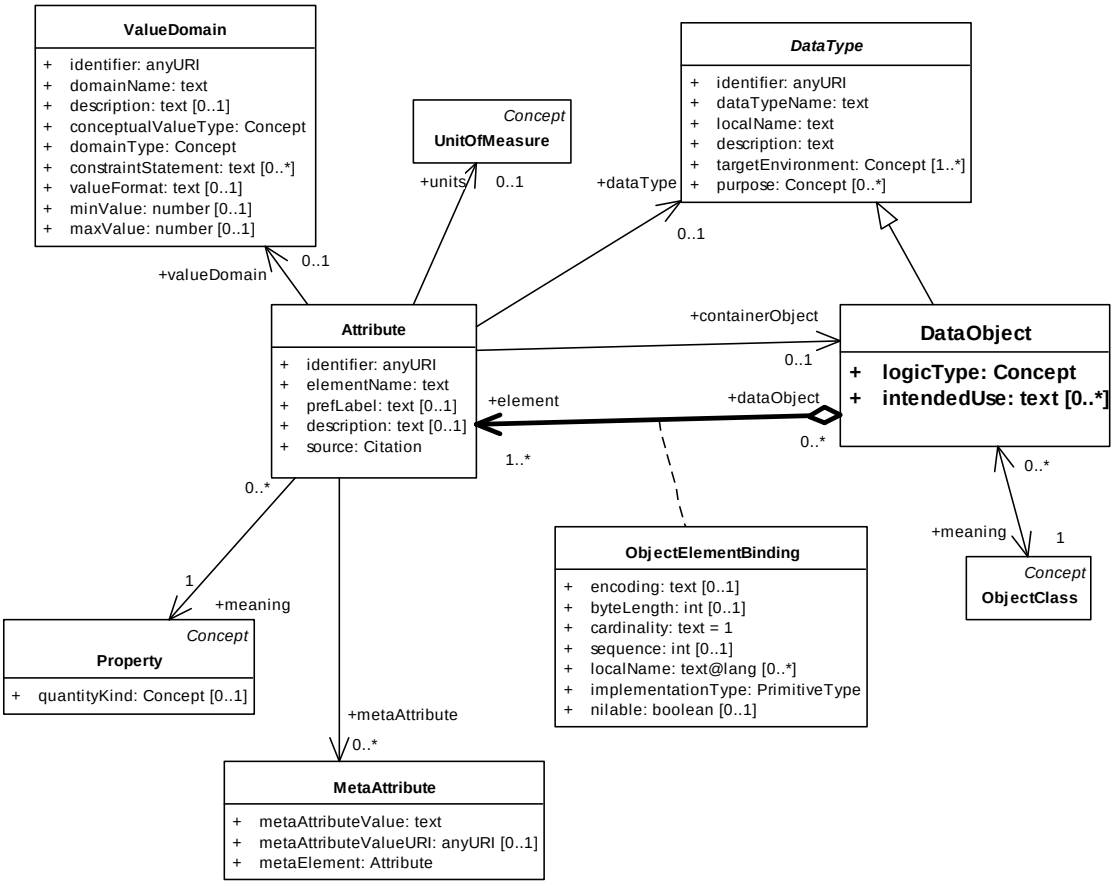

Figure 5: DataObject

2.6 ArrayVariable diagram

An ArrayVariable assigns a value for each combination of the dimension indexes [0...dimLength]. This model element represents NetCDF common Data Model 'variable' (OGC 10-090r3). The Type of the values assigned to each dimension index is determined by ArrayDimension.valueType.Attribute.dataType, which may be primitive, another ArrayVariable, a List, Dictionary, or a DataObject.
For example an ArrayVariable may contain a 100 by 100 array of air temperature values measured at lat, long locations. The Array dimension length is 100, there are 2 array dimensions that represent lists of the latitude and longitude coordinates. The metaAttributes of the Attribute define the grid geometry.

The valueType.Attribute for each dimension index (given by the ArrayDimension.sequence) could be specified by a 1 dimensional ArrayVariable of length 100 that contains the actual coordinate values for the measurement points. The ArrayDimension.valueType.Attribute for these coordinate arrays would represent individual lat and long coordinates with an appropriate numeric data type domain (e.g. decimal number between -180 and 180).

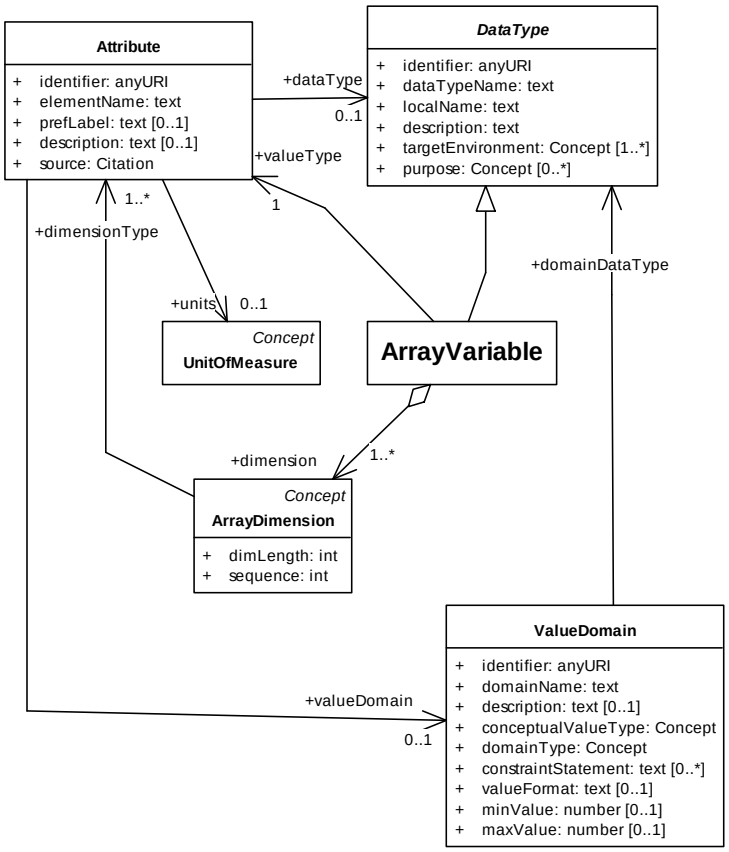

Figure 6: Context:ArrayVariable

2.7 DataType diagram

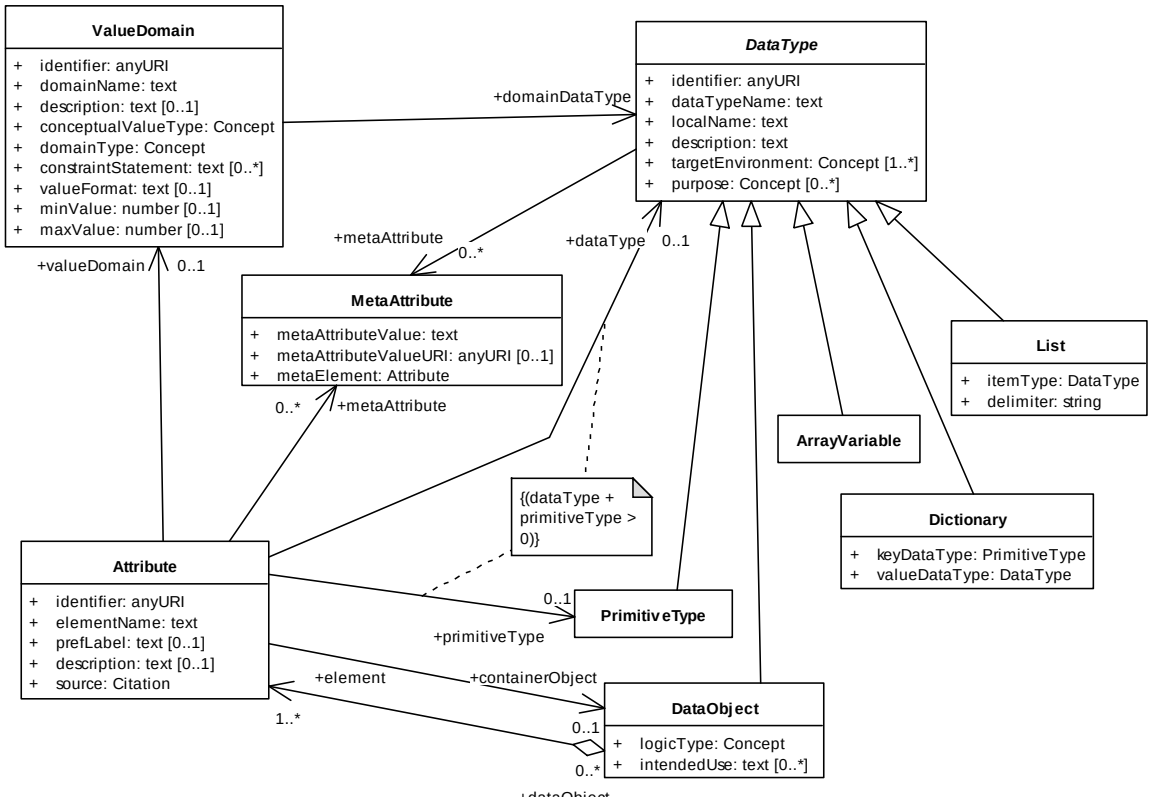

Figure 7: Context:DataType

2.8 ValueDomain diagram

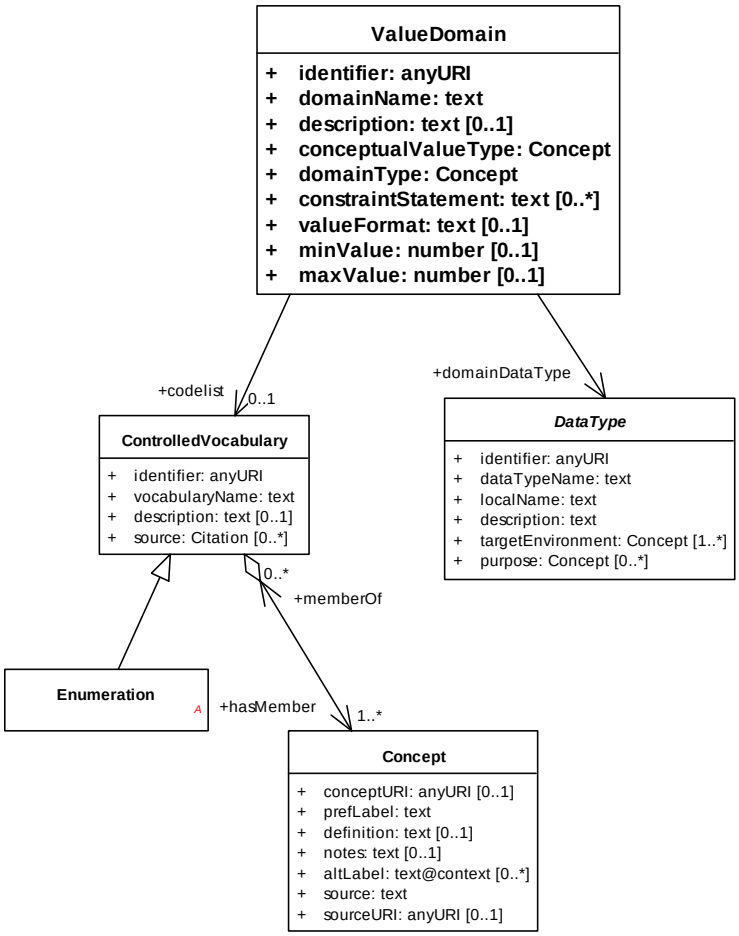

Figure 8: Context:ValueDomain

3 Entities

3.1 ArrayDimension

Extends Concept

NetCDF common data model 'Dimension'. Represents a dimension of an array, with an associated dimensiontType that assigns meaning for values on this dimension.

OUTGOING STRUCTURAL RELATIONSHIPS

- Generalization from ArrayDimension to Concept
- Aggregation from ArrayDimension to ArrayVariable

ATTRIBUTES

- dimLength : int the number of values allowed for this dimension in the array.
- sequence : int non negative integer that orders the dimensions in the array coordinate scheme

ASSOCIATIONS

Association (direction: Source -> Destination)

Source: (Class) ArrayDimension

Target: dimensionType (Class) Attribute
Cardinality: [1..*]

the associated attribute defines the meaning of each dimension on the array. For instance if the array variable is a geospatial grid, the dimensions might be UTM Easting and Northing.

3.2 ArrayVariable

Extends DataType

A dataType that represents a multidimensional array of values of the same type (OGC 10-090r3). The dimension properties associated with the variable define the axes of the array. ArrayVariable.metaAttribute properties describe the gridding scheme used to assign values to the dimension coordinates for the array cells; this part of the model is not detailed here and should be treated as an extension to the metaAttribute class. Array variables are used to represent a coverage (see ISO19123).

CONSTRAINTS

Invariant. dimension.ArrayDimension.characterizedBy is NOT self
an ArrayVariable dimenstion SHALL not be characterized by the same ArrayVariable

[Approved, Weight is 0.]

OUTGOING STRUCTURAL RELATIONSHIPS

- Generalization from ArrayVariable to DataType

INCOMING STRUCTURAL RELATIONSHIPS

- Aggregation from ArrayDimension to ArrayVariable

Association (direction: Source -> Destination)

Source: (Class) ArrayVariable

Target: valueType (Class) Attribute Cardinality: [1]

The valueType.Attribute defines the semantics of the values in each element of the array. metaAttribute properties on the Attribute associated with an ArrayVariable define the sampling scheme (if any) that maps array dimension coordinates to some domain coordinate space (location, temperature, pressure...)

3.3 Attribute

ATTRIBUTES

identifier : anyURI URI that identifies this DataElement

elementName : text full name to designate this DataElement in the context of the containing DataType.

prefLabel : text Multiplicity: ([0..1]). Optional label suggested for use to identify the DataElement in tables for computer use; generally a shorter version of the full elementName.

description : text Multiplicity: ([0..1]). documentation of any special considerations for the logical representation of the dataElementConcept by this element.

source : Citation

ASSOCIATIONS

Association (direction: Source -> Destination)
Source: (Class) Attribute
Target: primitiveType (Class) PrimitiveType Cardinality: [0..1]

Association (direction: Source -> Destination)
Source: (Class) Attribute
Target: valueDomain (Class) ValueDomain Cardinality: [0..1]

Domain restriction on the value of the physical attribute in the context of an ImplementionObject. If not specified, the attribute value is only restricted to be consistent with the physical data type.

Association (direction: Source -> Destination)
Source: (Class) Attribute
Target: dataType (Class) DataType Cardinality: [0..1]

Association (direction: Source -> Destination)
Source: (Class) Attribute
Target: containerObject (Class) DataObject Cardinality: [0..1]

In flattened schema, the properties of an object may be implemented as properties of a higher level object. The container object provides a means to specify the immediated dataObject that the properties apply to. For instance, 'coordinate geographic location' is represented by a data object with lat, long, and srs, but a simple feature implementation of a 'sample' might be the actual data object in which lat,long and srs occur; identify the container object allows all feature types with 'coordinate geographic location' to be found.

Association (direction: Source -> Destination)
Source: (Class) Attribute
Target: metaAttribute (Class) MetaAttribute Cardinality: [0..*]

Association (direction: Source -> Destination)
Source: representation (Class) Attribute Cardinality: [0..*]
Target: meaning (Class) Property Cardinality: [1]

Association (direction: Source -> Destination)
Source: (Class) Attribute
Target: units (Class) UnitOfMeasure Cardinality: [0..1]

ATTRIBUTES

Association (direction: Source -> Destination)
Source: (Class) ArrayDimension
Target: dimensionType (Class) Attribute Cardinality: [1..*]

the associated attribute defines the meaning of each dimension on the array. For instance if the array variable is a geospatial grid, the dimensions might be UTM Easting and Northing.

AssociationClass (direction: Source -> Destination)
Source: dataObject (Class) DataObject Cardinality: [0..*]
Target: element (Class) Attribute Cardinality: [1..*]

Association (direction: Source -> Destination)
Source: (Class) ArrayVariable
Target: valueType (Class) Attribute Cardinality: [1]

The valueType.Attribute defines the semantics of the values in each element of the array. metaAttribute properties on the Attribute associated with an ArrayVariable define the sampling scheme (if any) that maps array dimension coordinates to some domain coordinate space (location, temperature, pressure...)

3.4 Concept

CONSTRAINTS

Invariant. count (conceptURI) + count(definition) > 0
either a conceptURI or a definition SHALL be provided for each concept.

[Approved, Weight is 0.]

OUTGOING STRUCTURAL RELATIONSHIPS

Aggregation from Concept to ControlledVocabulary
Aggregation from Concept to ConceptScheme

INCOMING STRUCTURAL RELATIONSHIPS

Generalization from Property to Concept
Generalization from ObjectClass to Concept
Generalization from ArrayDimension to Concept
Generalization from UnitOfMeasure to Concept
Generalization from MeasureClass to Concept

ATTRIBUTES

conceptURI : anyURI Multiplicity: ([0..1]). a URI that identifies the concept.

prefLabel : text preferred name for humans to use when talking about this concept.

definition : text Multiplicity: ([0..1]). skos:definition

notes : text Multiplicity: ([0..1]). non normative information about the derivation of the concept. skos:note.

altLabel : text@context Multiplicity: ([0..*]). other text strings by which the concept may be know in other contexts.
Use altLabels for designations used in other contexts.
The context may specify a language, community, or particular application environment. skos:altLabel

ATTRIBUTES
source : text Text description of provenance of concept definition.
sourceURI : anyURI Multiplicity: ([0..1]). URI identifying source resource

3.5 ConceptScheme

INCOMING STRUCTURAL RELATIONSHIPS
Aggregation from Concept to ConceptScheme

ATTRIBUTES
identifier : anyURI
schemeName : text
intendedUse : text Multiplicity: ([0..1]).
conceptAxes : Concept Multiplicity: ([0..*]). if axes are specified, implication is that every concept that is a member of the scheme denotes some value or range of values for each axis.
source : Citation Multiplicity: ([0..*]).
Association (direction: Source -> Destination) Source: (Class) ConceptualDomain Target: conceptSpace (Class) ConceptScheme Cardinality: [0..1]

3.6 ConceptualDomain

ATTRIBUTES
identifier : anyURI
domainName : text
description : text Multiplicity: ([0..1]).
source : Citation Multiplicity: ([0..1]).
ASSOCIATIONS
Association (direction: Source -> Destination) Source: (Class) ConceptualDomain Target: applicableMeasure (Class) MeasureClass Cardinality: [1..*]
Association (direction: Source -> Destination) Source: (Class) ConceptualDomain Target: conceptSpace (Class) ConceptScheme Cardinality: [0..1]
Association (direction: Source -> Destination) Source: (Class) Property Target: conceptualValueDomain (Class) ConceptualDomain association to a concept for the range of values that are valid to quantify a property.

3.7 ControlledVocabulary

INCOMING STRUCTURAL RELATIONSHIPS

- Generalization from Enumeration to ControlledVocabulary
- Aggregation from Concept to ControlledVocabulary

ATTRIBUTES

- identifier : anyURI
- vocabularyName : text
- description : text Multiplicity: ([0..1]).
- source : Citation Multiplicity: ([0..*]).
- Association (direction: Source -> Destination)
  logically, multiple controlled vocabularies might be available that represent a particular enumerated domain. At the implementation level, a specific vocabulary must be specified. The implementation vocabulary SHALL be logically compatible with the ConceptScheme associated with the ConceptualDomain, if there is one.
  Source: (Class) ValueDomain
  Target: codelist (Class) ControlledVocabulary Cardinality: [0..1]
  role name from ISO19115 used here

3.8 DataObject

Extends DataType

An information object that represents an entity of interest (ObjectClass in this model, based on ISO11179) in some domain; the representation consists of a collection of Attributes that are used to quantify properties of instances of the entity. Corresponds to 'dataType' in ISO11179, Entity in Entity-Relationship models, Object in object models, 'document' in document type noSQL databases (e.g. CouchDb, MongoDb), 'Variable' in the netCDF common data model (OGC 10-090r3).

An information object that has internal structure in which the parts can be operated on independently; a data structure

CONSTRAINTS

- Invariant. A DataObject SHALL have attribute associations that correspond to the element.Property associations for the meaning.ObjectClass associated with the DataObject.
  Basically, a DataObject must have attribute.DataElement association that bind at least one DataElement whose meaning.Property is also an element.Property of the meaning.ObjectClass associated with the DataObject

  [Approved, Weight is 0.]

OUTGOING STRUCTURAL RELATIONSHIPS

- Generalization from DataObject to DataType

ATTRIBUTES

- logicType : Concept categorize the logical paradigm for the representation-- e.g. relational, object-oriented, graph, tabular text. Logical type for primitive types is 'atomic'
- intendedUse : text Multiplicity: ([0..*]). an explanation of the intention for the dataObject

  Properties:
  source = RDA DataType Model 2015
- AssociationClass (direction: Source -> Destination)
  Source: dataObject (Class) DataObject Cardinality: [0..*]
  Target: element (Class) Attribute Cardinality: [1..*]
- Association (direction: Bi-Directional)
  Source: meaning (Class) ObjectClass Cardinality: [1]
  Target: representation (Class) DataObject Cardinality: [0..*]
- Association (direction: Source -> Destination)
  Source: (Class) Attribute
  Target: containerObject (Class) DataObject

ATTRIBUTES	
	Cardinality: [0..1] In flattened schema, the properties of an object may be implemented as properties of a higher level object. The container object provides a means to specify the immediated dataObject that the properties apply to. For instance, 'coordinate geographic location' is represented by a data object with lat, long, and srs, but a simple feature implementation of a 'sample' might be the actual data object in which lat,long and srs occur; identify the container object allows all feature types with 'coordinate geographic location' to be found.

3.9 DataType

CONSTRAINTS
Invariant. implemenationDataType + implementedObjectAttribute = 1 [Approved, Weight is 0.]

INCOMING STRUCTURAL RELATIONSHIPS
Generalization from DataObject to DataType
Generalization from ArrayVariable to DataType
Generalization from Dictionary to DataType
Generalization from List to DataType
Generalization from PrimitiveType to DataType

ATTRIBUTES
identifier : anyURI
dataTypeName : text
localName : text designation for this information object in its native environment
description : text
targetEnvironment : Concept Multiplicity: ([1..*]). identification of the specific software environment for which this implementation is designed. e.g. Oracle 10 relational db, XML v1.0, GML 3.2 application schema
purpose : Concept Multiplicity: ([0..*]). categorize the intention of this implementation, e.g. interchange format, database table, data acquisition tool, data archive, object oriented software, semantic application

ASSOCIATIONS	
Association (direction: Source -> Destination) Source: (Class) DataType	Target: metaAttribute (Class) MetaAttribute Cardinality: [0..*]
Association (direction: Source -> Destination) Source: (Class) Attribute	Target: dataType (Class) DataType Cardinality: [0..1]
Association (direction: Source -> Destination) Source: (Class) ValueDomain	Target: domainDataType (Class) DataType

3.10 Dictionary

Extends DataType

A collection of key-value pairs. Also known as Hash, Associative Array, Map. This dataType represents values for which the values of the keys are not known in advance or defined as part of the data model, otherwise this would be represented as a DataObject.

OUTGOING STRUCTURAL RELATIONSHIPS
Generalization from Dictionary to DataType

ATTRIBUTES
keyDataType : PrimitiveType identifier for key data type. Must be string or whole number.
valueDataType : DataType identifier for value data type

3.11 Enumeration

Extends ControlledVocabulary

Another name for a controlled vocabulary. Usually indicates a vocabulary that is defined as part of a schema, as opposed to vocabularies that are user or application-defined.

OUTGOING STRUCTURAL RELATIONSHIPS
Generalization from Enumeration to ControlledVocabulary

3.12 InterchangeFormat

ATTRIBUTES
identifier : anyURI
formatName : text
label : text Multiplicity: ([0..1]). short text version of format name. In data that is identifying this interchange format using a text string (not a URI), this string should be used (if specified).
description : text Multiplicity: ([0..1]).
fileType : Concept mime type for the file
source : Citation should identify an accessible document that defines the interchange format.
schemaDocument : Link Multiplicity: ([0..1]). a link to a schema document (xsd, schematron, RuleML) that can be used to validate instance documents.
schemaType : Concept Multiplicity: ([0..1]). category that identifies the kind of schema document available to validate interchange document instances.

3.13 List

Extends DataType

A DataType that represents a sequence of values separated by a character string (bit sequence) that can unambiguously be distinguished from the content of the values. A special array for which there is no semantics associated with the position in the list. Also physically implemented to required different parsing algorithms.

OUTGOING STRUCTURAL RELATIONSHIPS
Generalization from List to DataType

ATTRIBUTES
itemType : DataType
delimiter : string

3.14 MeasureClass

Extends Concept

a set of equivalent **units of measure** that may be shared across multiple **dimensionalities**. *Measure_Class* allows a grouping of units of measure to be specified once, and reused by multiple dimensionalities.
EXAMPLE: We could define the *Measure_Classes*: Metric Linear Distance, Imperial Linear Distance, each associated with the appropriate *Units_of_Measure*; and associate them with *Dimensionalities*: Height, Width, and Depth to model the three spatial dimensions. (From ISO11179)
Also allow dimensionless, and categorical

UOM under metric linear distance would include cm, m, km. It would appear that the members of a measure class would all belong to a single system of units.

OUTGOING STRUCTURAL RELATIONSHIPS	
Generalization from MeasureClass to Concept	
Association (direction: Bi-Directional) Source: memberUnit (Class) UnitOfMeasure Cardinality: [1..*]	Target: unitsType (Class) MeasureClass Cardinality: [0..1]
Association (direction: Source -> Destination) Source: (Class) ConceptualDomain	Target: applicableMeasure (Class) MeasureClass Cardinality: [1..*]

3.15 MetaAttribute

ATTRIBUTES
metaAttributeValue : text Text that specifies the value. If the Attribute.dataType is not a primitive type, the string should be a text representation of the complex data type value
metaAttributeValueURI : anyURI Multiplicity: ([0..1]). If the Attribute.dataType is not a primitive type, the value must be an identifier for a data instance with the specified type that provides the value.
metaElement : Attribute

ASSOCIATIONS	
Association (direction: Source -> Destination) Source: (Class) Attribute	Target: metaAttribute (Class) MetaAttribute Cardinality: [0..*]
Association (direction: Source -> Destination) Source: (Class) DataType	Target: metaAttribute (Class) MetaAttribute Cardinality: [0..*]

3.16 ObjectClass

Extends Concept

object class is a **concept** (3.2.18) that represents a set of ideas, abstractions, or things in the real world that can be identified with explicit boundaries and meaning and whose properties and behavior follow the same rules.

OUTGOING STRUCTURAL RELATIONSHIPS
Generalization from ObjectClass to Concept

INCOMING STRUCTURAL RELATIONSHIPS
Aggregation from Property to ObjectClass
Aggregation from Property to ObjectClass

ASSOCIATIONS	
Association (direction: Bi-Directional) Source: meaning (Class) ObjectClass Cardinality: [1]	Target: representation (Class) DataObject Cardinality: [0..*]
Association (direction: Source -> Destination) Source: (Class) Property	Target: domainOfCarriers (Class) ObjectClass Cardinality: [0..*] association to an ObjectClass that specifies the kinds of things that may carry a given property. Object classes that have the property as an essentialProperty or optionalProperty must be subsumed by an ObjectClasst specified as a domainof Carriers. This association corresponds to the 'object' defined by Scott Peckham in his categorization of 'variables'

3.17 ObjectElementBinding

ATTRIBUTES
encoding : text Multiplicity: ([0..1]). encoding scheme for representing values, e.g. UTF-8, big-endian, different specs for floating point and double values.
byteLength : int Multiplicity: ([0..1]). number of 8 bit bytes used to represent values
cardinality : text
sequence : int Multiplicity: ([0..1]). if the order of the attributes in the implementation instance is fixed, sequence numbers SHALL be provided to define the order.
localName : text@lang Multiplicity: ([0..*]). other names that may be used to identify the dataElement; should be language or context-localized.
implementationType : PrimitiveType Physical implementation data type; the type specified on the attribute might be a logical type like 'integer', but in specific object-element binding for implementation might be more specific 'long integer' or 'short integer'. DataTypes here are specific to software environments.
nilable : boolean Multiplicity: ([0..1]). true for mandatory properties that may be declared nil for a reason

3.18 PrimitiveType

Extends DataType

PrimitiveType represents a machine-level, physical implementation of a low level data type, corresponding to the Apache Avro concept of primitive Type, which is enumerated as {null, int, long, float, double, boolean, bytes, and string}, or XML primitive types. A registry of these primitive types will be required, with mapping to the various existing schemes. Note that some hierarchy in the scheme for the primitive types would be useful, for instance defining an 'integer' as any whole number (no range restriction), with subtypes 'int' (short integer), and 'long' (long integer) which have different domains of values that can be represented.

CONSTRAINTS

Invariant. logicType="atomic"

[Approved, Weight is 0.]

OUTGOING STRUCTURAL RELATIONSHIPS

Generalization from PrimitiveType to DataType

Association (direction: Source -> Destination)

Source: (Class) Attribute

Target: primitiveType (Class) PrimitiveType
Cardinality: [0..1]

3.19 ProcessingStep

ATTRIBUTES

event : Event

source : Citation Multiplicity: ([0..*]).

contributor : QualifiedAttribution Multiplicity: ([1..*]).

ASSOCIATIONS

Association (direction: Source -> Destination)

Source: (Class) AnyClass

Target: provenance (Class) ProcessingStep
Cardinality: [0..*]
record of the processing that was done to create an instance of one of the model elements.

3.20 Property

Extends Concept

A conceptual property. A quality that characterizes some aspect of instances of an **object class**. A property may be any feature that humans naturally use to distinguish one individual object from another. It is the human perception of a single quality of an object class in the real world. It is conceptual and thus has no particular associated means of representation by which the property can be communicated. A quality that inheres in an entity either permanently or over some time interval..

This is derived from ISO11179 **data element concept:** a **concept** that is an **association** of a **property** with an **object class**. A data element concept can be represented in the form of a **data element**, described independently of any particular representation. Since elementProperty is mandatory and single valued, there doesn't seem to be much gained by separating property and dataElementConcept

OUTGOING STRUCTURAL RELATIONSHIPS

Generalization from Property to Concept

Aggregation from Property to ObjectClass

Aggregation from Property to ObjectClass

ATTRIBUTES

quantityKind : Concept Multiplicity: ([0..1]). quantityKind: aspect common to mutually comparable quantities. categorizes a property according to the quantifiable thing that it represents. e.g. time, distance, velocity, mass, temperature, energy, and weight, area, volume, independent of the measurement procedure of quantification scheme (e.g. categorical,

ATTRIBUTES

ordered, interval, and ratio measures.). same as Kind of quantity JCGM_200:2008 (http://www.bipm.org/utils/common/documents/jcgm/JCGM_pack_2012-10.zip)
A quantity kind will be associated with a quantity dimension.

Comparability and transformability are the equivalence properties for quantityKind as used here, measured values that quantify the same quantityKind can be compared using
a quantity-preserving one-to-one correspondence between values measured using different units of measure.

Appears to correspond (exactMatch?) to NetCDF common data model 'dimension' concept: "represents a real physical dimension, for example, time, latitude, longitude, or height. A dimension might also be used to index other quantities, for example station or model-run-number." (NetCDF User Guide, Version 4.1.3, 2011-06).

When a quantityKind is specified, then the Unit_of_Measure specified for any Value_Domain that is based on this Conceptual_Domain SHALL be consistent with this dimensionality.

EXAMPLES from Note 1 on 1.1 in JCGM_200:2008e length, >radius, >wavelength, >diameter, >circumference; energy, >kinetic energy, >heat, >potential energy; electric charge, electric resistance, concentration of entity (mass/volume), number concentration (count/volume), Rockwell hardness. '>' indicates quantities of the same kind.

Notes:
--The division of the concept of ‘quantity’ according to ‘kind of quantity’ is to some extent arbitrary. (JCGM 200:2008)
-- This concept is in contrast to 'dimensionality' as defined in ISO11179, which adds the requirement that mutually comparable quantities have the same dimensionality if they have common characterizing operations. Thus with respect to temperature, absolute temperature coordinates (e.g. Kelvins) are considered to be a different dimensionality than "offset" temperature coordinates (e.g. degrees Celsius or Fahrenheit). It is meaningful to take the ratio of absolute temperature coordinates, but not of "offset" temperature coordinates, wherein the arbitrary translation of zero renders ratios meaningless. The notion of characterizing operations used here has been adapted from the statistics literature where distinctions are commonly made among categorical, ordered, interval, and ratio measures. (ISO11179). This distinction is considered more closely related to the concept of MeasureClass in this model.

Association (direction: Source -> Destination)
Source: (Class) Property

Target: conceptualValueDomain (Class) ConceptualDomain
association to a concept for the range of values that are valid to quantify a property.

Association (direction: Source -> Destination)
Source: (Class) Property

Target: domainOfCarriers (Class) ObjectClass Cardinality: [0..*]

association to an ObjectClass that specifies the kinds of things that may carry a given property. Object classes that have the property as an essentialProperty or optionalProperty must be subsumed by an ObjectClasst specified as a domainof Carriers. This association corresponds to the 'object' defined by Scott Peckham in his categorization of 'variables'

Association (direction: Source -> Destination)
Source: representation (Class) Attribute Cardinality: [0..*]

Target: meaning (Class) Property Cardinality: [1]

3.21 UnitOfMeasure

Extends Concept
a convention for how the magnitude of a quantifiable thing is specified. "real scalar quantity, defined and adopted by convention, with which any other quantity of the same kind can be compared to express the ratio of the two quantities as a number" (http://www.iso.org/sites/JCGM/VIM/JCGM_200e_FILES/MAIN_JCGM_200e/01_e.html#L_1_9)

Units of measure are not limited to physical categories. Examples of physical categories are: linear measure, area, volume, mass, velocity, time duration. Examples of non-physical categories are: currency, quality indicator, color intensity.

OUTGOING STRUCTURAL RELATIONSHIPS
Generalization from UnitOfMeasure to Concept
ASSOCIATIONS
Association (direction: Bi-Directional) Source: memberUnit (Class) UnitOfMeasure Cardinality: [1..*] Target: unitsType (Class) MeasureClass Cardinality: [0..1]
Association (direction: Source -> Destination) Source: (Class) Attribute Target: units (Class) UnitOfMeasure Cardinality: [0..1]

3.22 ValueDomain

ATTRIBUTES
identifier : anyURI
domainName : text
description : text Multiplicity: ([0..1]). description of intention of domain
conceptualValueType : Concept high level categorization of the kind of values in this domain: e.g. narrative text, count, coordinate measurement, ratio measurement, interval measurement, concept, truth value, DateTime, Date, Time, vector, continuous field, sequence, name, rate (see 19103, maybe ISO80000?)
domainType : Concept type specifies the kind of domain; places logical restrictions on properties that the domain may have. E.g. enumeration, controlled vocabulary, free text, name, number, count... (have to figure out...)
constraintStatement : text Multiplicity: ([0..*]). statement of constraints on values within the domain. Ideally text should use some kind of formal constraint language like OCL.
valueFormat : text Multiplicity: ([0..1]). template for the structure of the presentation of the value(s) EXAMPLE – YYYY-MM-DD for a date., limitations on character string length. Typically some sort of regular expressions specifying syntax for the alphanumeric string that specifies data values. If the data type allows lists, format text should specify list boundary and delimiter characters.
minValue : number Multiplicity: ([0..1]). minimum allowed value if the domain has ordering, and values are limited to a subset of the possible range.
maxValue : number Multiplicity: ([0..1]). maximum allowed value if the domain has ordering, and values are limited to a subset of the possible range.
ASSOCIATIONS
Association (direction: Source -> Destination) Source: (Class) ValueDomain Target: domainDataType (Class) DataType
Association (direction: Source -> Destination) logically, multiple controlled vocabularies might be available that represent a particular enumerated domain. At the implementation level, a specific vocabulary must be specified. The implementation vocabulary SHALL be logically compatible with the ConceptScheme associated with the ConceptualDomain, if there is one. Source: (Class) ValueDomain Target: codelist (Class) ControlledVocabulary Cardinality: [0..1] role name from ISO19115 used here
Association (direction: Source -> Destination) Source: (Class) Attribute Target: valueDomain (Class) ValueDomain Cardinality: [0..1] Domain restriction on the value of the physical attribute in the context of an ImplementionObject. If not specified, the attribute value is only restricted to be consistent with the physical data type.