Committee Draft ISO/IEC CD 11179-3	
Date: **2007-08-19**	Reference number: ISO/JTC 1/SC 32**N1667**
Supersedes document SC 32N1168	

THIS DOCUMENT IS STILL UNDER STUDY AND SUBJECT TO CHANGE. IT SHOULD NOT BE USED FOR REFERENCE PURPOSES.

ISO/IEC JTC 1/SC 32 Data Management and Interchange Secretariat: USA (ANSI)	Circulated to P- and O-members, and to technical committees and organizations in liaison for voting (P-members only) by: **2007-11-19** Please return all votes and comments in electronic form directly to the SC 32 Secretariat by the due date indicated.

ISO/IEC CD 11179-3: 2007(E)

Title: Information technology -- Metadata Registries (MDR) - Part 3: Registry Metamodel and basic attributes, 3rd Edition

Project: 1.32.15.03.03.00

Introductory note:

text of CD 11179-3; this document is sent to NBs for 3-month letter ballot. The ballot starts 2007-08-19 and closes 2007-11-19.

Medium: E

No. of pages: 160

Dr. Timothy Schoechle, Secretary, ISO/IEC JTC 1/SC 32
Farance Inc *, 3066 Sixth Street, Boulder, CO, United States of America
Telephone: +1 303-443-5490; E-mail: Timothy@Schoechle.org
available from the JTC 1/SC 32 WebSite http://www.jtc1sc32.org/
*Farance Inc. administers the ISO/IEC JTC 1/SC 32 Secretariat on behalf of ANSI

ISO/IEC JTC 1/SC 32 N 1667

Date: 2007-08-15

ISO/IEC CD 11179-3

ISO/IEC JTC 1/SC 32/WG 2

Secretariat: ANSI

Information technology — Metadata registries (MDR) — Part 3: Registry metamodel and basic attributes

Technologies de l'information — Registres de métadonnées (RM) — Partie 3: Métamodèle de registre et attributs de base

Document type: International Standard
Document subtype:
Document stage: (30) Committee
Document language: E

Macintosh HD:RecoveredDocs:SC32:Ballots-out:CD 11179-3-Gates 8-07:32N1667-CD1-11179-3_3rd-ed_2007-08-15.doc STD Version 2.1c2

Contents

Table of Figures

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 11179-3 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information Technology*, Subcommittee SC 32, *Data Management and Interchange*.

This third edition cancels and replaces the second edition (ISO/IEC 11179-3:2003), which has been technically revised.

ISO/IEC 11179 consists of the following parts, under the general title *Information technology — Metadata registries (MDR)*:

EDITOR'S NOTE #1. (Action Required by FCD) For the 3rd edition of ISO/IEC 11179, it is expected that part 2 will be re named, and terms and definitions consolidated into a new part 7. WG2 needs to confirm these other projects.

— *Part 1: Framework*

— *Part 2: Conceptual Schemes (To be confirmed)*

— *Part 3: Registry metamodel and basic attributes*

— *Part 4: Formulation of data definitions*

— *Part 5: Naming and identification principles*

— *Part 6: Registration*

— *Part 7: Terminology (To be confirmed)*

Introduction

EDITOR'S NOTE #2. (Informational) Throughout this Committee Draft, EDITOR'S NOTEs make reference to 'issues' that are either addressed or not addressed by this document. Details of these issues may be found on the WG2 Issue Management website at: http://issues.metadata-stds.org . To locate a specific issue, the generic format of the URL is: http://issues.metadata-standards.org/show_bug.cgi?id=221 where the number at the end is the issue number, without leading zeroes.

EDITOR'S NOTE #3. (Action required) There have been extensive changes from both the second edition of this standard, and from WD4 of the third edition. The whole document needs careful review and comment. It is expected that a second CD will be required, before the document moves to FCD.

Data processing and electronic data interchange rely heavily on accurate, reliable, controllable and verifiable data recorded in databases. A prerequisite for correct and proper use and interpretation of data is that both users and owners of data have a common understanding of the meaning and representation of the data. To facilitate this common understanding, a number of characteristics, or attributes, of the data have to be defined. These characteristics of data are known as “metadata”, that is, “data that describes data”. This part of ISO/IEC 11179 provides for the attributes of data elements and associated metadata to be specified and registered as metadata items in a *Metadata Registry*.

The structure of a *Metadata Registry* is specified in the form of a conceptual data model. The *Metadata Registry* is used to keep information about data elements and associated concepts, such as “data element concepts”, “conceptual domains” and “value domains”. Generically, these are all referred to as “metadata items”. Such metadata are necessary to clearly describe, record, analyse, classify and administer data.

When considering data and metadata, it is important to distinguish between types of data/metadata, and instances of these types. Clause 4 of this part of ISO/IEC 11179 specifies the types of metadata objects that form the structure of a *Metadata Registry*. A *Metadata Registry* will be populated with instances of these metadata objects (metadata items), which in turn define types of data, e.g. in an application database. In other words, instances of metadata specify types of application level data. In turn, the application database will be populated by the real world data as instances of those defined data types.

NOTE ISO/IEC 10027:1990 *Information technology — Information resource dictionary system (*IRDS) Framework and ISO/IEC TR 10032:2003 *Information technology —* Reference model for data management explain the concepts of different levels of modelling.

This part of ISO/IEC 11179 also describes the basic attributes of metadata items for purposes where a complete *Metadata Registry* is not appropriate.

This part of ISO/IEC 11179 is of interest to information developers, information managers, data administrators, standards developers and others who are responsible for making data understandable and shareable. ISO/IEC 11179 has broad applicability across subject area domains and information technologies.

Information technology — Metadata registries (MDR) — Part 3: Registry metamodel and basic attributes

1 Scope

1.1 Overview

EDITOR'S NOTE #4. (Action required for FCD) The scope statement will need to be revised to reflect whatever changes we make in the third edition. We should add a sub-clause listing what is new or modified in Edition 3.

The primary purpose of ISO/IEC 11179-3 is to specify the structure of a *Metadata Registry* (see subclause 1.2). ISO/IEC 11179-3 also specifies basic attributes which are required to describe metadata items, and which may be used in situations where a complete metadata registry is not appropriate (e.g. in the specification of other International Standards) (see subclause 1.3). Subclause 1.4 provides examples of activities where ISO/IEC 11179-3 may be applied.

1.2 Scope – Structure of a Metadata Registry

A comprehensive *Metadata Registry* management function requires a set of rules and procedures. These rules and procedures are set out in the following Clauses and Annexes and are complemented elsewhere in this document as follows:

a) the definitions of metadata objects are in Clause 3.3 of this part of ISO/IEC 11179;

b) the structure of the registry in the form of a conceptual data model is in Clauses 5 through 10 of this part of ISO/IEC 11179;

Aspects of the registry are expanded on in other parts of ISO/IEC 11179, as follows:

c) the overall framework for this family of International Standards is specified in ISO/IEC 11179-1;

d) rules and guidelines for classifying metadata are in ISO/IEC 11179-2;

e) rules and guidelines for the formulation of definitions are in ISO/IEC 11179-4;

f) naming and identifying principles for metadata are in ISO/IEC 11179-5;

g) rules and guidelines for registering metadata are in ISO/IEC 11179-6.

While the model diagrams are presented in UML notation, this part of ISO/IEC 11179 does not assume nor endorse any specific system environment, database management system, database design paradigm, system development methodology, data definition_language, command language, system interface, user interface, computing platform, or any technology required for implementation. This part of ISO/IEC 11179 does not directly apply to the actual use of data in communications and information processing systems.

1.3 Scope – Basic attributes of metadata items

This part of ISO/IEC 11179 also specifies basic attributes which are required to describe metadata items, and which may be used in situations where a complete *Metadata Registry* is not appropriate (e.g. in the specification of other International Standards). These basic attributes are described in Clause 11.

1.4 Areas of Applicability

This part of ISO/IEC 11179 applies to activities including:

a) the definition, specification and contents of metadata registries, including interchanging or referencing among various collections of data elements;

b) the design and specification of application-oriented data models, databases and message types for data interchange;

c) the actual use of data in communications and information processing systems;

d) interchange or reference among various collections of metadata.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IETF RFC 4646, *Tags for Identifying Languages*[1]

ISO 31-0, *Quantities and units — Part 0: General principles*

ISO 639-2:1998, *Codes for the representation of the names of languages — Part 2: Alpha-3 code*

ISO 1087-1:2000, *Terminology work — Vocabulary — Part 1: Theory and application*

ISO/IEC 2382-1:1993, *Information technology — Vocabulary — Part 1: Fundamental terms*

ISO/IEC 2382-17:1999, *Information technology — Vocabulary — Part 17: Databases*

ISO 3166-1:1997, *Codes for the representation of names of countries and their subdivisions — Part 1: Country codes*

ISO 5127:2001, *Information and documentation — Vocabulary*

ISO/IEC 6523-1:1998, *Information technology — Structure for the identification of organization and organization parts — Part 1: Identification of organization identification schemes*

ISO/IEC 6523-2:1998, *Information technology — Structure for the identification of organization and organization parts — Part 2: Registration of organization identification schemes*

ISO 8601:2000, *Data elements and interchange formats — Information exchange — Representation of dates and times*

ISO/IEC 11179-1, *Information technology — Metadata registries (MDR) — Part 1: Framework*

ISO/IEC 11179-2, *Information technology — Metadata registries (MDR) — Part 2: Classification*

ISO/IEC 11179-4, *Information technology — Metadata registries (MDR) — Part 4: Formulation of data definitions*

[1] IETF RFC 4646 is available at http://www.rfc-editor.org/rfc/rfc4646.txt

ISO/IEC 11179-5, *Information technology — Metadata registries (MDR) — Part 5: Naming and identification principles*

ISO/IEC 11179-6, *Information technology — Metadata registries (MDR) — Part 6: Registration*

ISO/IEC 11404:1996, *Information technology — Programming languages, their environments and system software interfaces — Language-independent datatypes*

ISO 12620:1999, *Computer applications in terminology — Data categories*

ISO/IEC 19501:2005, *Information technology — Unified Modeling Language (UML) Version 1.4.2*

ISO/IEC 19773:200n, Information technology — Metadata registries (MDR) Modules

ITU-T Recommendation E.164 (2005-02) The international public telecommunications numbering plan[2]

Universal Postal Union (UPU) S42-1:2003 International postal address components and templates[3]

3 Definitions

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

3.1 defines metamodel constructs, used in specifying the registry metamodel.

3.2 lists broader terms, and their definitions, used in this document that are not included in either 3.1 or 3.3.

3.3 defines metadata objects prescribed by the metamodel itself.

An alphabetical list of terms from all three Clauses is provided in Annex A.

3.1 Definitions of Metamodel Constructs

This subclause defines the metamodel constructs used in specifying the registry metamodel in Clauses 4 through 7.

3.1.1
association
⟨metamodel⟩ semantic **relationship** between two **classes**

NOTE An **association** is a type of **relationship**.

[Adapted from ISO/IEC 19501-1:2005, 4.5.2.3]

3.1.2
association class
⟨metamodel⟩ **association** that is also a **class**

NOTE It not only connects a set of **class**es, but also defines a set of features that belong to the **association** itself.

[Adapted from ISO/IEC 19501-1:2005, 4.5.2.4]

[2] ITU-T E.164 is available at http://www.itu.int/rec/T-REC-E.164-200502-I/en

[3] UPU is the Universal Postal Union at "http://www.upu.int". UPU S42-1 is based on EN 14142-1, Postal services – Address data bases – Part 1 – Components of Postal_Addresses.

3.1.3
attribute
⟨metamodel⟩ **characteristic** of an **object** or **entity**

3.1.4
class
⟨metamodel⟩ description of a set of **object**s that share the same **attribute**s, operations, methods, **relationship**s, and semantics

[ISO/IEC 19501-1:2005, 4.5.2.9]

3.1.5
composite attribute
⟨metamodel⟩ **attribute** whose **datatype** is non-atomic

3.1.6
composite datatype
⟨metamodel⟩ **datatype** that is also a **class**

NOTE A **composite datatype** is used as a **datatype** for a **composite attribute**.

3.1.7
datatype
set of distinct values, characterized by properties of those values and by operations on those values

[ISO/IEC 11404:1996, 4.11]

3.1.8
generalization
⟨metamodel⟩ **relationship** between a more general **class** (the parent) and a more specific **class** (the child) that is fully consistent with the first **class** (i.e. it has all of its **attributes** and **relationships**) and that adds additional information.

NOTE A **generalization** is a type of **relationship**.

[Adapted from ISO/IEC 19501-1:2005, 4.5.2.24]

3.1.9
identifier
⟨metamodel⟩ sequence of characters, capable of uniquely identifying that with which it is associated, within a specified context

NOTE A **name** should not be used as an identifier because it is not linguistically neutral.

3.1.10
primitive datatype
datatype that cannot be decomposed into other datatypes without loss of associated semantics.

[Adapted from ISO/IEC 11404:1996, 4.34]

3.1.11
relationship
⟨metamodel⟩ connection among model elements

NOTE In ISO/IEC 11179-3, a relationship is either an **association** or a **generalization**.

[ISO/IEC 19501-1:2005, 4.5.2.36]

3.2 Broad Terms used in this part of ISO/IEC 11179

3.2.1
attribute instance
specific instance of an **attribute**

NOTE Amended from ISO 2382-17:1993 (17.02.13) to distinguish an instance of an attribute from its value.

3.2.2
attribute value
the value associated with an **attribute instance**

NOTE Amended from ISO 2382-17:1993 (17.02.13) to distinguish an instance of an attribute from its value.

3.2.3
basic attribute
<metadata> **attribute** of a **metadata item** commonly needed in its specification

3.2.4
binding
mapping from one framework or specification to another

3.2.5
characteristic
abstraction of a property of an **object** or of a set of objects

NOTE Characteristics are used for describing **concepts**.

[ISO 1087-1:2000, 3.2.4]

3.2.6
common attribute
<metadata> **basic attribute** that is applicable to all types of metadata item

3.2.7
conceptual data model
data model that represents an abstract view of the real world

3.2.8
conditional
required under certain specified conditions

NOTE 1 One of three obligation statuses applied to the attributes of metadata items, indicating the conditions under which the attribute is required. See also **mandatory** (3.2.17) and **optional** (3.2.28).

NOTE 2 Obligation statuses apply to metadata items with a Registration Status of "recorded" or higher.

3.2.9
coordinate
measurement from the origin of a frame of reference

3.2.10
data
re-interpretable representation of information in a formalized manner suitable for communication, interpretation or processing

NOTE Data can be processed by human or automatic means.

[ISO/IEC 2382-1:1993, 01.01.02]

3.2.11
data model
graphical and/or lexical representation of **data**, specifying their properties, structure and inter-relationships

3.2.12
definition
representation of a **concept** by a descriptive statement which serves to differentiate it from related concepts

[ISO 1087-1:2000, 3.3.1]

NOTE See also **Definition** <designatable item> (3.3.54).

3.2.13
designation
representation of a **concept** by a sign which denotes it

[ISO 1087-1:2000, 3.4.1]

NOTE See also **Designation** <designatable item> (3.3.71) and **name** (3.2.26).

3.2.14
entity
any concrete or abstract thing that exists, did exist, or might exist, including associations among these things

EXAMPLE A person, object, event, idea, process, etc…

NOTE Please observe that an entity exists whether data about it are available or not.

[ISO/IEC 2382-17:1999, 17.02.05]

3.2.15
extension
<11179-3> a feature not defined by ISO/IEC 11179-3

<registry metamodel> a **class**, an **attribute** or a **relationship** that an implementation of a **Metadata Registry** provides that is not defined by ISO/IEC 11179-3

3.2.16
language
system of signs for communication, usually consisting of a vocabulary and rules

[ISO 5127:2001, 1.1.2.01]

3.2.17
mandatory
always required

NOTE 1 One of three obligation statuses applied to the attributes of metadata items, indicating the conditions under which the attribute is required. See also **conditional** (3.2.8) and **optional** (3.2.28).

NOTE 2 Obligation statuses apply to metadata items with a Registration Status of "recorded" or higher.

3.2.18
metadata
data that defines and describes other **data**

3.2.19
metadata item
instance of a **metadata object**

NOTE 1 In all parts of ISO/IEC 11179, this term is applied only to instances of metadata objects described by the metamodel in Clause 4 of ISO/IEC 11179-3. Examples include instances of Data_Elements, Data_Element_Concepts, Permissible_Values etc.

NOTE 2 A metadata item has associated attributes, as appropriate for the metadata object it instantiates.

3.2.20
metadata object
object type defined by a metamodel

NOTE In all parts of ISO/IEC 11179, this term is applied only to metadata objects described by the metamodel in Clause 4 of ISO/IEC 11179-3. Examples include Data_Elements, Data_Element_Concepts, Permissible_Values etc. See 3.3 for a complete list.

3.2.21
metadata register
information store or database maintained by a **Metadata Registry**

3.2.22
Metadata Registry
MDR
information system for registering **metadata**

NOTE The associated information store or database is known as a **metadata register**.

3.2.23
metadata set
any collection of **metadata**

3.2.24
metamodel
data model that specifies one or more other data models

3.2.25
metamodel construct
unit of notation for modelling

NOTE The metamodel constructs used in ISO/IEC 11179-3 are defined in 3.1.

3.2.26
name
designation of an **object** by a linguistic expression

NOTE See also **name** <Individual> (3.3.114) and **name** <Organization> (3.3.115)

3.2.27
object
anything perceivable or conceivable

NOTE Objects may also be material (e.g. an engine, a sheet of paper, a diamond), immaterial (e.g. a conversion ratio, a project plan) or imagined (e.g. a unicorn).

[Adapted from ISO 1087-1:2000, 3.1.1]

3.2.28
optional
permitted but not required

NOTE 1 One of three obligation statuses applied to the attributes of metadata items, indicating the conditions under which the attribute is required. See also **conditional** (3.2.8) and **mandatory** (3.2.17).

NOTE 2 Obligation statuses apply to metadata items with a Registration Status of "recorded" or higher.

3.2.29
organization part
any department, service or other entity within an organization which needs to be identified for information exchange

[ISO/IEC 6523-1:1998, 3.2]

3.2.30
quantity
Value with an associated unit of measure.

NOTE: 32º Fahrenheit and 0º Celsius are quantities, and they are equivalent values in different measuring systems.

3.2.31
registration
<generic>inclusion of an item in a registry

<Metadata Registry> inclusion of a **metadata item** in a **Metadata Registry**

3.2.32
registry item
metadata item recorded in a **Metadata Registry**

3.2.33
registry metamodel
metamodel specifying a **Metadata Registry**

3.2.34
related metadata reference
reference from one **metadata item** to another

NOTE A **Registration_Authority** could choose to use a **Reference_Document**, an **administrative_note** or an **explanatory_comment** to record a **related metadata reference**.

3.2.35
stewardship
<metadata> the responsibility for the maintenance of **Administration_Record**s applicable to one or more **Administered_Item**s

NOTE 1 The responsibility for the registration of metadata may be different from the responsibility for stewardship of metadata.

NOTE 2 See also **stewardship** <Administered_Item> (3.3.176).

3.2.36
text
paragraph, page or document that can be used to define or describe an entity. Text is 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]

3.3 Alphabetical list of metadata objects in the metamodel

This subclause provides definitions for terms which are the names of metadata objects in the metadata model in Clause 4. Each metadata object is modelled on one of the metamodel constructs from 3.1 (i.e. classes, attributes, composite attributes, associations or association classes). The metamodel construct applicable to each metadata object is indicated after the definition. For attributes, the associated class is also identified.

This subclause follows the capitalization convention of the model, which is to capitalize the names of classes, association classes and composite datatypes, but not attributes or associations.

EDITOR'S NOTE #5. (Action Required) It has previously been suggested that the Classes defined here should be explicitly so named: e.g. Administered_Item class, and that the corresponding term in lower case should be separately defined, though it is unclear in which sub-clause they should be listed. The Editor seeks direction as to whether or not to implement this, and if so, assistance with the additional definitions would be appreciated.
At the 2007 New York meeting, it was agreed that we should separate the definition of metadata objects from the definition of concepts, and that the concepts that the model objects represent should be separately defined. It was proposed that the definitions of metamodel objects do not belong in clause 3 at all, but rather in the clause in which the corresponding model region is described. This change will be made for CD2.

3.3.1
Administered_Item

EDITOR'S NOTE #6. (Action required) Since we use 'metadata object' to refer to types and 'metadata item' to refer to instances of metadata, should we rename 'Administered_Item' to 'Administered Object' (strictly an Administered metadata object) for consistency. See Issue 275.

Registered_Item for which administrative information is recorded in an **Administration_Record**

NOTE Metamodel construct is: *Sub-class of* ***Registered_Item***.

3.3.2
Administration_Record

EDITOR'S NOTE #7. (Action Required) Issue 249.2 suggests collapsing Administration_Record into Administered_Item. Discussion of issue 176 in NYC suggested moving it to an Administration association class.

collection of administrative information

NOTE Metamodel construct is: *Composite datatype*.

3.3.3
administrative_note
general note about the **Registration**

NOTE Metamodel construct is: *Attribute of* ***Registration***.

3.3.4
administrative_status
designation of the status in the administrative process of a **Registration_Authority**

NOTE 1 Metamodel construct is: *Attribute of* ***Administration_Record***.

NOTE 2 The values and associated meanings of “administrative_status” are determined by each **Registration_Authority**. C.f. “**registration_status**”.

3.3.5
antisymmetric_indicator
indicator that a **Binary_Relation** is antisymmetric rather than symmetric.

NOTE Metamodel construct is: *Attribute of* ***Binary_Relation***

3.3.6
Assertion
sentence or proposition in logic which is asserted (or assumed) to be true.

NOTE Metamodel construct is: *Class*

3.3.7
assertion_formula
text which expresses the **Assertion**

NOTE Metamodel construct is: *Attribute of* ***Assertion***

3.3.8
assertion_term
association which records the inclusion of an **Ontology_Entry** as a term in the **assertion_formula** of the **Assertion**

NOTE Metamodel construct is: *Association*

3.3.9
Asymmetric_Relation
Binary_Relation, R, such that for all x,y: R(x,y) does not imply R(y,x).

NOTE *1* Metamodel construct is: *Sub-class of* ***Binary_Relation***

NOTE *2* In this metamodel, **Asymmetric_Relations** have two distinguishable (non-identical) roles, one for each **Link_End** of the **Link**. Examples of asymmetric relations include: *less than*, *likes*, *father of*, etc.

3.3.10
Attached_Item
Registered_Item for which administrative information is recorded in an **Administration_Record** of another **Registered_Item**.

NOTE *1* This is often a member of a group of **Registered_Items** that is managed as a whole.

NOTE *2* Metamodel construct is: *Sub-class of* ***Registered_Item***

3.3.11
attachment
association denoting that the **Attached_Item** shares all of the administration characteristics of the **Administered_Item**.

NOTE Metamodel construct is: *Association*

3.3.12
authority_rule
rule identifying the authority that assigns **names** and/or enforces **naming conventions**.

[Derived from ISO/IEC 11179-5]

NOTE Metamodel construct is: *Attribute of* ***Naming_Convention***

3.3.13
base_form

EDITOR'S NOTE #8. (Action required) A definition is required for this attribute.

NOTE Metamodel construct is: *Attribute of* ***Identified_Ontology_Entry***

3.3.14
Binary_Relation
Relation of arity 2 (having two Link_Ends).

NOTE *1* Metamodel construct is: *Sub-class of* ***Relation***

NOTE *2* Most common semantic relations are binary, e.g. *equals*, *less than*, *greater than*, *is part of*, etc. An example of a relation which is not binary is *betweenness*.

NOTE *3* Binary relations are commonly represented as edges (or directed edges for asymmetric binary relations) in graphs. C.f. the Resource Description Framework (RDF) of the W3C.

3.3.15
Boolean
mathematical datatype associated with two-valued logic

[ISO/IEC 11404:1996, 8.1.1]

NOTE Metamodel construct is: *Primitive datatype*

3.3.16
change_description
description of what has changed since the prior version of the **Administered_Item**

NOTE Metamodel construct is: *Attribute of* ***Administration_Record***.

3.3.17
Characteristic
abstraction of a property of an **object** or set of objects.

[ISO 1087-1:2000, 3.2.4]

EDITOR'S NOTE #9. (Action required) Does it make sense to say that a Characteristic is itself a Concept, and that it is used to describe (presumably other) Concepts?

NOTE 1 Metamodel construct is: *Sub-class of* ***Concept***

NOTE 2 Characteristics are used for describing concepts.

3.3.18
Classifiable_Item

EDITOR'S NOTE #10. (Action required) US NB had changed the definition to 'object which might be classified', but we should be able to classify more than just objects. Editor has changed 'object' to 'metadata item'. Both changes need confirmation.

EDITOR'S NOTE #11. (Action required) Is this definition sufficient? One of justifications for not making Classifiable_Item a sub-class of Identified_Item is that we may want to classify things which are not identified items (such as Organizations or Contacts), but do they fall within the definition of metadata items? (If they do, we need to ensure that that is appropriate everywhere that the term is used.

metadata item which might be classified

NOTE Metamodel construct is: *Class*

3.3.19
Classification
association denoting that a **Hierarchy_Node** classifies a **Classifiable_Item** within a **Hierarchical_Classification_Scheme.**

NOTE Metamodel construct is: *Association Class*.

3.3.20
Concept
unit of knowledge created by a unique combination of **characteristics**

NOTE Metamodel construct is: *Class*.

[ISO 1087-1:2000, 3.2.1]

3.3.21
Concept_System
set of **concepts** structured according to the **relations** among them. [ISO 1087-1]

NOTE Metamodel construct is: *Class*.

3.3.22
concept_system_inclusion
association denoting that a **Concept_System** is a subsystem in another Concept_System.

NOTE Metamodel construct is: *Association*.

3.3.23
concept_system_notation
notation used to describe the **Concept_System**

NOTE Metamodel construct is: *Attribute of **Concept_System***.

3.3.24
Conceptual_Domain
CD

EDITOR'S NOTE #12.	(Action required) Definition changed by comment from US NB. To be agreed.
EDITOR'S NOTE #13.	(Action required) Do we need the word 'valid'?

Concept that expresses its valid instance meanings or description

NOTE 1 Metamodel construct is: *Sub-class of **Concept***.

3.3.25
conceptual_domain_dimensionality
expression of measurement without units

NOTE 1 Metamodel construct is: *Attribute of Conceptual_Domain*

NOTE 2 ISO 31-0 specifies physical dimensions (e.g. length, mass, velocity). ISO/IEC 11179-3 also allows non-physical dimensions (e.g. value dimensions such as: currency, quality indicator)

NOTE 3 See also **Unit_of_Measure** (3.3.188).

3.3.26
Contact
instance of a role of an individual or an organization (or organization part or organization person) to whom an information item(s), a material object(s) and/or person(s) can be sent to or from in a specified context

NOTE Metamodel construct is: *Class*.

3.3.27
contact_individual
Individual that is the **Contact**

NOTE Metamodel construct is: *Attribute of* ***Contact***.

3.3.28
contact_mail_address
postal address for the **Contact**

NOTE Metamodel construct is: *Attribute of* ***Contact***.

3.3.29
contact_organization
Organization for which the **Contact** acts as a representative

NOTE Metamodel construct is: *Attribute of* ***Contact***.

3.3.30
contact_phone
phone number for the **Contact**

NOTE Metamodel construct is: *Attribute of* ***Contact***.

3.3.31
contact_title
name of the position held by the **Contact**

NOTE Metamodel construct is: *Attribute of* ***Contact***.

3.3.32
contained_concept
attribute recording that a **Concept** is contained in a **Concept_System**

NOTE Metamodel construct is: *Attribute* of ***Concept_System***.

3.3.33
contained_link
attribute recording that a **Link** is contained in a **Concept_System**

NOTE Metamodel construct is: *Attribute* of ***Concept_System***.

3.3.34
contained_relation
attribute recording that a **Relation** is contained in a **Concept_System**

NOTE Metamodel construct is: *Attribute* of ***Concept_System***.

3.3.35
Context
<designation and definition> universe of discourse in which a **Designation** or **Definition** is used

NOTE Metamodel construct is: *Class*.

3.3.36
coordinate_indicator
predicate on the **Dimensionality** whose value is true if the Dimensionality is a **coordinate**

NOTE Metamodel construct is: *Attribute of* ***Dimensionality***.

3.3.37
creation_date
date the **Administered_Item** was created

NOTE Metamodel construct is: *Attribute of* ***Administration_Record***.

3.3.38
Data_Element
DE

EDITOR'S NOTE #14. (Action required) There has been feedback from users of 11179 that this is not a useful definition. It should be reviewed.

unit of **data** for which the **definition**, **identification**, representation and **Value_Domain** are specified by means of a set of **attribute**s

NOTE Metamodel construct is: *Class*.

3.3.39
Data_Element_Concept
DEC
concept for which the **definition**, **identification** and **Conceptual_Domain** are specified independently of any particular representation

NOTE 1 Metamodel construct is: *Sub-class of Concept*.

NOTE 2 A Data_Element is a representation of a Data_Element_Concept.

3.3.40
data_element_concept_characteristic
association denoting the **Characteristic** for a **Data_Element_Concept**

NOTE Metamodel construct is: *Association*

3.3.41
data_element_concept_domain
association denoting the **Conceptual_Domain** that provides the domain for a **Data_Element_Concept**

NOTE Metamodel construct is: *Association*.

3.3.42
data_element_concept_object_class
association denoting the **Object_Class** for a **Data_Element_Concept**

NOTE Metamodel construct is: *Association*

3.3.43
Data_Element_Derivation
association among a **Data_Element** which is derived, the **Derivation_Rule** controlling its derivation, and the **Data_Element**(s) from which it is derived

NOTE Metamodel construct is: *Association Class*.

3.3.44
data_element_domain

EDITOR'S NOTE #15. (Action required) Since the association is with a Value_Domain, this association would be more aptly named: 'data element value domain'. Feedback on this proposed change is requested.

association denoting the **Value_Domain** that provides the domain for a **Data_Element**

NOTE Metamodel construct is: *Association*.

3.3.45
Data_Element_Example
representative illustration of the **Data_Element**

NOTE Metamodel construct is: *Class*.

3.3.46
data_element_meaning
association denoting the **Data_Element_Concept** that provides meaning for a **Data_Element**.

NOTE Metamodel construct is: *Association*.

3.3.47
data_element_precision
the degree of specificity for a **Data_Element**

NOTE 1 Metamodel construct is: *Attribute of **Data_Element***.

NOTE 2 Expressed as a number of decimal places to be used in any associated **Data_Element** values.

3.3.48
Datatype
set of distinct values, characterized by properties of those values and by operations on those values

NOTE Metamodel construct is: *Composite Datatype*.

[ISO/IEC 11404:1996, 4.11]

3.3.49
datatype_annotation
specifying information to further define the **Datatype**

NOTE Metamodel construct is: *Attribute of **Datatype***.

3.3.50
datatype_description
descriptive information to further clarify the **Datatype**

NOTE Metamodel construct is: *Attribute of **Datatype***.

3.3.51
datatype_name
designation for the **Datatype**

NOTE Metamodel construct is: *Attribute of **Datatype***.

3.3.52
datatype_scheme_reference
reference identifying the source of the **Datatype** specification

NOTE 1 In this edition of ISO/IEC 11179-3, the manner of reference is specified by the Registration_Authority.

NOTE 2 Metamodel construct is: *Attribute of **Datatype***.

3.3.53
Date
family of **datatypes** whose values are points in time to various common resolutions: year, month, day, hour, minute, second, and fractions thereof.

NOTE Metamodel construct is: *Primitive Datatype*.

[ISO/IEC 11404:1996, 8.1.6]

3.3.54
Definition
<designatable item> representation of a **Designatable_Item** by a descriptive statement which, in a given **language** and **context**(s) serves to differentiate it from related Designatable_Items

NOTE 1 Metamodel construct is: *Class*.

NOTE 2 See also **definition** (3.2.12).

3.3.55
Definition_Context
<designatable item> **association** denoting **Definitions** that are relevant in a **Context**

NOTE Metamodel construct is: *Association Class*.

3.3.56
definition_language
language used to write the **definition_text**

NOTE Metamodel construct is: *Attribute of **Definition***.

3.3.57
definition_source_reference
reference to the source from which the **Definition** is taken

NOTE Metamodel construct is: *Attribute of **Definition***.

3.3.58
definition_text
text of the **Definition**

NOTE Metamodel construct is: *Attribute of **Definition***.

3.3.59
derivation_input
association denoting the source **Data_Element**(s) for a **Data_Element_Derivation**

NOTE Metamodel construct is: *Association*.

3.3.60
derivation_output
association denoting the result of a **Data_Element_Derivation**

NOTE Metamodel construct is: *Association*.

3.3.61
Derivation_Rule
logical, mathematical, and/or other operations specifying derivation

NOTE Metamodel construct is: *Class*.

3.3.62
derivation_rule_application
association denoting the **Derivation_Rule** for a **Data_Element_Derivation**

NOTE Metamodel construct is: *Association*.

3.3.63
derivation_rule_notation
notation used to describe the **Derivation_Rule**

NOTE Metamodel construct is: *Attribute of **Derivation_Rule***.

3.3.64
derivation_rule_specification
text of a specification of **Data_Element_Derivation**

NOTE Metamodel construct is: *Attribute of **Derivation_Rule***.

3.3.65
Described Conceptual_Domain
Conceptual_Domain that is specified by a description

NOTE 1 Metamodel construct is: *Sub-class of **Conceptual_Domain***.

3.3.66
described_conceptual_domain_description
description or specification of a rule, reference, or range for a set of all **Value_Meanings** for the **Conceptual_Domain**

NOTE Metamodel construct is: *Attribute of **Described Conceptual_Domain***.

3.3.67
Described_Value_Domain
Value_Domain that is specified by a description

NOTE 1 Metamodel construct is: *Sub-class of **Value_Domain***.

3.3.68
described_value_domain_description
description or specification of a rule, reference, or range for a set of all **Permissible_Values** for the **Value_Domain**

NOTE Metamodel construct is: *Attribute of **Described_Value_Domain***.

3.3.69
described_value_domain_meaning
association denoting the meaning of a **Described_Value_Domain** provided by a **Described Conceptual_Domain**

NOTE Metamodel construct is: *Association*

3.3.70
Designatable_Item

EDITOR'S NOTE #16. (Action required) Since we use 'metadata object' to refer to types and 'metadata item' to refer to instances of metadata, should we rename 'Designatable_Item' to ' Designatable Object' (strictly a designatable metadata object) for consistency. See Issue 275.

Identified_Item which can have **Designations** and/or **Definitions.**

NOTE 1 Metamodel construct is: *Sub-class of **Identified_Item***.

3.3.71
Designation
<designatable item> representation of a **Designatable_Item** by a **Sign** which denotes it.

NOTE 1 Metamodel construct is: *Class*.

NOTE 2 See also **designation** (3.2.13).

3.3.72
Designation_Context
<designatable item> **association** denoting that a **Designation** is within a given **Context**

NOTE Metamodel construct is: *Association Class*.

3.3.73
designation_language

EDITOR'S NOTE #17. (Action required) Is 'used' the right word? Would 'specified' or 'expressed' be better?

<designatable item> **Language** or dialect in which a **Sign** (usually a **Name**) is used.

NOTE Metamodel construct is: *Attribute of **Designation***.

3.3.74
designation_sign
<designatable item> **Sign** of the **Designation**.

NOTE Metamodel construct is: *Attribute of **Designation***.

3.3.75
Designation_Space

EDITOR'S NOTE #18. (Action required) The definition needs to be reviewed.

<designatable item> **Namespace** within which a **Designation** is specified.

NOTE 1 Metamodel construct is: *Sub-class of **Namespace***.

3.3.76
designation_space_membership
association denoting a **Designation** is bound to a **Designation_Space**

NOTE Metamodel construct is: *Association*

3.3.77
Dimensionality

EDITOR'S NOTE #19. (Action required) Should we define the concept "quantity preserving one-to-one correspondence"?

set of equivalent units of measure, where equivalence between two units of measure is determined by the existence of a quantity preserving one-to-one correspondence between values measured in one unit of measure and values measured in the other unit of measure, independent of context, and where characterizing operations are the same.

NOTE 1 The equivalence defined here forms an equivalence relation on the set of all units of measure. Each equivalence class corresponds to a dimensionality. The units of measure "temperature in degrees Fahrenheit" and "temperature in degrees Celsius" have the same dimensionality, because:
a) given a value measured in degrees Fahrenheit there is a value measured in degrees Celsius with the same quantity, and vice-versa, by the well-known correspondences C° = (5/9)*(F° - 32) and F° = (9/5)*(C°) + 32.
b) the same operations can be performed on both values.

NOTE 2 The units of measure "temperature in degrees Celsius" and "temperature in degrees Kelvin" do not belong to the same dimensionality. Even though it is easy to convert quantities from one unit of measure to the other (C° = K° - 273.15 and K° = C° + 273.15), the characterizing operations in degrees Kelvin include taking ratios, whereas this is not the case for degrees Celsius. For instance, 20° K is twice as warm as 10° K, but 20° C is not twice as warm as 10° C.

NOTE 3 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, colour intensity

NOTE 4 Quantities may be grouped together into categories of quantities which are mutually comparable. Lengths, diameters, distances, heights, wavelengths and so on would constitute such a category. Mutually comparable quantities have the same dimensionality. ISO 31-0 calls these “quantities of the same kind”.

NOTE 5 ISO 31-0 specifies physical dimensions (e.g. length, mass, velocity). ISO/IEC 11179-3 also allows non-physical dimensions (e.g. value dimensions such as: currency, quality indicator). The present concept of dimensionality equates to what ISO 31 calls Dimensional Product, rather than to Dimension.

NOTE 6 Metamodel construct is: *Class*.

NOTE 7 See also **Unit_of_Measure** (3.3.188).

3.3.78
documentation_language_identifier
identifier of the **Language** used for documentation by the **Registration_Authority**

NOTE Metamodel construct is: *Attribute of **Registration_Authority***.

3.3.79
effective_date
date an **administered item** became/becomes available to registry users

NOTE Metamodel construct is: *Attribute of **Administration_Record***.

3.3.80
Enumerated_Conceptual_Domain
Conceptual_Domain that is specified by a list of all its **Value_Meaning**s

NOTE Metamodel construct is: *Sub-class of **Conceptual_Domain***.

3.3.81
Enumerated_Value_Domain

EDITOR'S NOTE #20. (Action required) Issue 100 notes that ISO/IEC 11404 distinguishes:
- state is a family of datatypes, each of which comprises a finite number of distinguished but unordered values.
- enumerated is a family of datatypes, each of which comprises a finite number of distinguished values having an intrinsic order.
Does Enumerated_Value_Domain imply an intrinsic order?

Value_Domain that is specified by a list of all its **Permissible_Value**s

NOTE Metamodel construct is: *Sub-class of **Value_Domain***.

3.3.82
example_item
actual illustrative case of the **Data_Element**

NOTE Metamodel construct is: *Attribute of **Data_Element_Example***.

3.3.83
exemplification
association denoting a **Data_Element_Example** and the **Data_Element** that is exemplified

NOTE Metamodel construct is: *Association*.

3.3.84
explanatory_comment
descriptive comments about the **Administered_Item**

NOTE Metamodel construct is: *Attribute of **Administration_Record***.

3.3.85
extension_identifier
identifies an **extension** to a **language_identifier**

NOTE 1 Metamodel construct is: *Attribute of* ***Language_Identification***.

NOTE 2 an extension_identifier provides a mechanism for extending (or qualifying) language_identifiers for purposes not supported by standardized qualifiers.

3.3.86
external_form

EDITOR'S NOTE #21. (Action required) A definition is required for this attribute.

NOTE Metamodel construct is: *Attribute of* ***Identified_Ontology_Entry***

3.3.87
Hierarchical_Classification_Scheme

EDITOR'S NOTE #22. (Action Required) The specification that classification is based on characteristics of objects seems to apply only to schemes based on concepts. If the classification scheme uses nodes which are not concepts, it would seem possible to classify metadata items whether or not they are objects, and regardless of any particular characteristics.

descriptive information for an arrangement or division of **objects** into groups based on **characteristics**, which the objects have in common

EXAMPLE Origin, composition, structure, application, function, etc.; See ISO/IEC 11179-2.

NOTE Metamodel construct is: *Class*.

3.3.88
hierarchy_link
association between **Hierarchy_Nodes** in a **Hierarchical_Classification_Scheme**

NOTE Metamodel construct is: *Association*.

3.3.89
hierarchy_membership
association denoting that a **Hierarchy_Node** is contained in a **Hierarchical_Classification_Scheme**

NOTE Metamodel construct is: *Association*.

3.3.90
Hierarchy_Node
item of content in a **Hierarchical_Classification_Scheme**

NOTE 1 Metamodel construct is: *Class*.

NOTE 2 This may be a node in a taxonomy or ontology, a term in a thesaurus, etc.

3.3.91
hierarchy_node_reference_concept
Concept that is used to partition the **Hierarchical_Classification_Scheme** at this **Hierarchy_Node.**

NOTE Metamodel construct is: *Attribute of* ***Hierarchy_Node***.

3.3.92
identification
<identified item> **association** denoting the **Identified_Item** identified by a **Scoped_Identifier**.

NOTE Metamodel construct is: *Association*

3.3.93
Identified_Item
metadata item identified in a **metadata registry**

NOTE Metamodel construct is: *Class*

3.3.94
identifier

EDITOR'S NOTE #23. (Action required) Term changed from *item_identifier* with the intent that the Identification region be applied more generally than to just Identified_Items. However, the former definition (**identifier** from a given **Identifier_Space** given to a **Identified_Item)** had to be changed to remove the use of identifier, even though it was used in the generic defined in 3.1.9. That generic sense is used elsewhere in this standard, e.g. in Language_Identification. The terms we use for metamodel elements must be different from the terms we use for generic concepts, to avoid confusion. The equivalent term in edition 2 was *data_identifier*. Perhaps we should revert to that.

EDITOR'S NOTE #24. (Informational) The reference to Identifier_Space has been removed because that is already in the definition of the class Scoped_Identifier.

<Identified_Item>**String** used to identify an **Identified_Item**

NOTE Metamodel construct is: *Attribute of **Scoped_Identifier***

3.3.95
Identifier_Space
<identified item> **Namespace** which provides the scope for **Scoped_Identifiers**

NOTE Metamodel construct is: *Sub-class of **Namespace***

3.3.96
Individual
single human being.

NOTE Metamodel construct is: *Class*

3.3.97
Integer
mathematical datatype comprising the exact integral values

[ISO/IEC 11404:1996, 8.1.7]

NOTE Metamodel construct is: *Primitive datatype*

3.3.98
international_code_designator
identifier of an organization identification scheme

NOTE 1 Metamodel construct is: *Attribute of **Registration_Authority_Identifier***.

NOTE 2 Based on ISO/IEC 6523-1:1998, 3.8.

NOTE 3 See also ISO/IEC 11179-6.

3.3.99
item_definition
association denoting that a **Definition** specifies the meaning of a specific **Designatable_Item**

NOTE Metamodel construct is: *Association*.

3.3.100
item_designation
association denoting that a **Designation** designates a specific **Designatable_Item**

NOTE Metamodel construct is: *Association*.

3.3.101
item_slot
association denoting that a **Slot** is owned by a specific **Identified_Item**

NOTE Metamodel construct is: *Association*.

3.3.102
Language_Identification
collection of **identifier**s required to identify a **language** or language variation for a particular purpose

NOTE 1 Metamodel construct is: *Composite datatype*.

NOTE 2 The identifiers specified are based on IETF RFC 4646 *Tags for Identifying Languages*

3.3.103
language_identifier
identifier for the **language**

NOTE 1 Use the three character alphabetic codes and names from ISO 639-2/Terminology, with extensions if required.

NOTE 2 Metamodel construct is: *Attribute of* ***Language_Identification***.

3.3.104
last_change_date
the date the **Administered_Item** was last changed

NOTE Metamodel construct is: *Attribute of* ***Administration_Record***.

3.3.105
lexical_rule
rule specifying the appearance of **names**: preferred and non-preferred terms, synonyms, abbreviations, part length, spelling, permissible character set, case sensitivity, etc.

[Derived from ISO/IEC 11179-5]

NOTE Metamodel construct is: *Attribute of* ***Naming_Convention***

3.3.106
Link

EDITOR'S NOTE #25. (Action required) There is some dissatisfaction with this term.

instance of a **Relation**, i.e., a individual n-tuple which comprises the corresponding relation.

NOTE Metamodel construct is: *Class*

3.3.107
Link_End

EDITOR'S NOTE #26. (Action required) If Link is renamed, then Link-End also needs to be renamed.

association that identifies a particular **Role** that a **Concept** plays in a **Link**.

NOTE Metamodel construct is: *Association Class*

3.3.108
link_end_role
the **Role** that a **Concept** plays in a **Link**.

NOTE Metamodel construct is: *Attribute of **Link_End***

3.3.109
mail_address
<organization> mailing address of an **Organization**

NOTE Metamodel construct is: *Attribute of **Organization***.

3.3.110
management
<metadata registry>**association** denoting the **Registration_Authority** that is responsible for managing and maintaining the **Register**.

NOTE Metamodel construct is: *Association*

3.3.111
mandatory_naming_convention_indicator
indicator specifying whether all **Designations** in this **Namespace** have to conform to exactly one **Naming_Convention**.

NOTE 1 Metamodel construct is: *Attribute of **Namespace***.

NOTE 2 If mandatory_naming_convention_indicator is true (a) there must be exactly one acceptable convention Naming_Convention associated with this Namespace in the naming_convention_utilization association and (b) every binding Designation must have a naming_convention_conformance association with the same Naming_Convention used in part (a) above. If mandatory_naming_convention_indicator is false, it is possible for a Namespace to be associated with zero or more acceptable conventions and/or a binding Designation to conform to more than one convention.

3.3.112
max_cardinality
maximum number of n-tuples (**Links**) in a **Relation** of arity n for for which all of the other n-1 roles have been given fixed values (i.e., the maximum taken with respect to all of the n-tuples of the projection of the Relation over the n-1 other roles)

NOTE Metamodel construct is: *Attribute of **Relation_Role***.

3.3.113
min_cardinality
minimum number of n-tuples (**Links**) in a **Relation** of arity n for for which all of the other n-1 roles have been given fixed values

NOTE Metamodel construct is: *Attribute of **Relation_Role***.

3.3.114
name
<Individual> **designation** of an **Individual**

NOTE 1 Metamodel construct is: *Attribute of **Individual***.

NOTE 2 The generic definition from (3.2.26) is applied to an **Individual**.

3.3.115
name
<Organization> **designation** of an **Organization**

NOTE 1 Metamodel construct is: *Attribute of **Organization***.

NOTE 2 The generic definition from (3.2.26) is applied to an **Organization**.

3.3.116
Namespace

EDITOR'S NOTE #27. (Action required) Changed by comment from US NB. While this is a reasonable definition for a generic term, it does not accurately reflect the class in the model, since the sub-class IdentiferSpace assigns *item identifiers* not designations. Note 2 added by the editor.

set of **designations** for a particular business need.

NOTE 1 Metamodel construct is: *Class*

NOTE 2 Namespace has sub-classes **Designation_Space** and **Identifier_Space**

3.3.117
Naming_Convention
specification of how **signs** of **Designations** are formulated.

NOTE Metamodel construct is: *Class*

3.3.118
naming_convention_conformance
association denoting that a **Designation** conforms to a **Naming_Convention**

NOTE Metamodel construct is: *Association*

3.3.119
naming_convention_utilization
association denoting that **Designations** in a **NameSpace** are specified by a **Naming_Convention**

NOTE Metamodel construct is: *Association*

3.3.120
Notation
formal notation, meant for machine processing. For example: UML, MOF, OCL, OWL/RDF, SKOS, CGIF, XCL, XTM, or ISO/IEC 11404

NOTE Metamodel construct is: *Class*.

3.3.121
Object_Class
Concept whose **extension** is the set of **objects** of interest for **data**.

NOTE Metamodel construct is: *Sub-class of **Concept***.

3.3.122
one_item_per_name_indicator
indicator that denotes whether or not each **Designation** in a **Namespace** must be unambiguous.

NOTE Metamodel construct is: *Attribute of **Namespace***.

3.3.123
one_name_per_item_indicator
indicator that denotes whether or not each **Designation** in a **Namespace** must be unique.

NOTE Metamodel construct is: *Attribute of **Namespace***.

3.3.124
Ontology
Concept_System that incorporates a declarative definition of a universe of discourse that models entities and the relationships that hold among them utilizing a formalized structure which incorporates standardized terminology

NOTE Metamodel construct is: *Sub-class of Concept_System*.

3.3.125
ontology_assertion
association denoting that an **Assertion** is asserted in one **Ontology**.

NOTE 1 Used to model the grouping (inclusion) of axioms within the ontology within which they are asserted.

NOTE 2 Metamodel construct is: *Association*

3.3.126
Ontology_Entry
entry in an **Ontology** that represents a term in an **Assertion**

NOTE Metamodel construct is: *Class*.

3.3.127
ontology_inclusion
association denoting the inclusion of one **Ontology** within another

NOTE Metamodel construct is: *Association*

3.3.128
Organization
unique framework of authority within which **Individuals** act, or are designated to act, towards some purpose

NOTE 1 Metamodel construct is: *Class*.

NOTE 2 The kinds of organizations covered by ISO/IEC 6523-1 include the following examples:

a) an organization incorporated under law;

b) an unincorporated organization or activity providing goods and/or services including:

   1) partnerships;

   2) social or other non-profit organizations or similar bodies in which ownership or control is vested in a group of individuals;

   3) sole proprietorships

   4) governmental bodies .

c) groupings of the above types of organizations where there is a need to identify these in information interchange.

[Adapted from ISO/IEC 6523-1:1998, 3.1]

3.3.129
organization_identifier

EDITOR'S NOTE #28. (Action Required) Should this be an attribute of Organization, which is then inherited by Registration_Authority?

identifier assigned to an **Organization** within an organization identification scheme, and unique within that scheme

NOTE Metamodel construct is: *Attribute of* ***Registration_Authority_Identifier***.

[ISO/IEC 6523-1:1998, 3.10]

3.3.130
organization_part_identifier
opi
an **identifier** allocated to a particular **organization part**

NOTE 1 Metamodel construct is: *Attribute of* ***Registration_Authority_Identifier***.

NOTE 2 See also ISO/IEC 11179-6.

[ISO/IEC 6523-1:1998, 3.11]

3.3.131
organization_part_identifier_source
the source for the **organization_part_identifier**

NOTE 1 Metamodel construct is: *Attribute of* ***Registration_Authority_Identifier***.

NOTE 2 See also ISO/IEC 11179-6.

[Based on ISO/IEC 6523-1:1998, 3.12]

3.3.132
origin
⟨Administered item⟩ the source (document, project, discipline or model) for the **Administered_Item**

NOTE Metamodel construct is: *Attribute of* ***Administration_Record***.

3.3.133
Permissible_Value
designation of a **Value_Meaning** within a specific **Value_Domain**

NOTE Metamodel construct is: *Class*.

3.3.134
permissible_value_begin_date
date this value became/becomes allowed in the **Value_Domain**

NOTE 1 Metamodel construct is: *Attribute of* ***Permissible_Value***.

NOTE 2 A Registration_Authority may determine whether this date is the date the value becomes valid in a registry or the date the value becomes part of the source domain or some other date.

3.3.135
permissible_value_end_date
date this value became/becomes no longer allowed in the **Value_Domain**

NOTE 1 Metamodel construct is: *Attribute of* ***Permissible_Value***.

NOTE 2 A Registration_Authority may determine whether this date is the date the value becomes no longer valid in a registry or the date the value becomes no longer part of the source domain or some other date.

3.3.136
permissible_value_meaning
association denoting that a **Permissible_Value** denotes a **Value_Meaning**

NOTE Metamodel construct is: *Association*.

3.3.137
permissible_value_set
set of **Permissible_Value**s for an **Enumerated_Value_Domain**

NOTE Metamodel construct is: *Association*.

3.3.138
permitted_value
representation of a **Value_Meaning** in a specific **Value_Domain** – the actual **Value**

NOTE Metamodel construct is: *Attribute of **Permissible_Value***.

3.3.139
Phone_Number
telephone number

NOTE 1 Metamodel construct is: *Composite datatype*.

NOTE 2 Specified by ITU-T Recommendation E.164 (2005-02), the international public telecommunications numbering plan.

3.3.140
Postal_Address
set of information which, for a postal item, allows the unambiguous determination of an actual or potential delivery point, usually combined with the specification of an addressee and/or a mailee.

[UPU S42]

NOTE Metamodel construct is: *Composite datatype*.

NOTE Metamodel construct is: *Association*

3.3.141
preferred_definition_indicator

EDITOR'S NOTE #29. (Action required) The model does not currently allow us to specify that the preference is for a particular language, as the definition suggests.

EDITOR'S NOTE #30. (Action required) It has been suggested that instead of a Boolean indicator, we use a set of acceptability ratings as defined by ISO 10241. I.e. 'preferred', 'accepted', 'deprecated', 'obsolete', 'superseded'. If this proposal is accepted, a better name might be: *definition acceptability rating*

indicator that a **Definition** is the preferred **Definition** within a **Context** in a **language**

NOTE Metamodel construct is: *Attribute of **Definition_Context***.

3.3.142
preferred_designation_indicator

EDITOR'S NOTE #31. (Action required) The model does not currently allow us to specify that the preference is for a particular language, as the definition suggests.

EDITOR'S NOTE #32. (Action required) It has been suggested that instead of a Boolean indicator, we use a set of acceptability ratings as defined by ISO 10241. I.e. 'preferred', 'accepted', 'deprecated', 'obsolete', 'superseded'. If this proposal is accepted, a better name might be: *designation acceptability rating*

indicator that a **Designation** is the preferred **Designation** within a **Context** in a **language**

NOTE 1 Metamodel construct is: *Attribute of* ***Designation_Context***.

NOTE 2 See **"main entry term"** in ISO 12620:1999.

3.3.143
ra_identifier
identifier assigned to a **Registration_Authority**

NOTE 1 Metamodel construct is: *Attribute of* ***Registration_Authority***.

NOTE 2 See ISO/IEC 11179-6 and ISO/IEC 6523-2.

3.3.144
record
<administered item> administrative information for an **Administered_Item**

NOTE Metamodel construct is: *Attribute of* ***Administered_Item***

3.3.145
Reference
association denoting that a **Reference_Document** provides additional information about an **Administered_Item**

NOTE Metamodel construct is: *Association Class*.

3.3.146
Reference_Document
document that provides pertinent details for consultation about a subject

NOTE Metamodel construct is: *Class*.

3.3.147
reference_document_identifier
identifier for the **Reference_Document**

NOTE Metamodel construct is: *Attribute of* ***Reference_Document***.

3.3.148
reference_document_language_identifier
identifier of the natural or special **language** used in the **Reference_Document**

NOTE Metamodel construct is: *Attribute of* ***Reference_Document***.

3.3.149
reference_document_title
title of the **Reference_Document**

NOTE Metamodel construct is: *Attribute of* ***Reference_Document***.

3.3.150
reference_document_type_description
description of the type of **Reference_Document**

NOTE Metamodel construct is: *Attribute of* ***Reference_Document***.

3.3.151
reference_provider

EDITOR'S NOTE #33. (Action Required) Do we need to be able to distinguish different types of reference_provider? For example, one organization might maintain the document, but another might publish it or make it available.

Organization that maintains or carries an official copy of the reference document.

NOTE Metamodel construct is: *Attribute of **Reference_Document***.

3.3.152
reference_type

EDITOR'S NOTE #34. (Action required) Attribute added by Issue 238. Definition added by the editor, but we need further explanation and examples as to what is meant.

specification of the type of **Reference**

NOTE Metamodel construct is: *Attribute of **Reference***.

3.3.153
reflexive_indicator
indicator that records whether a **Binary_Relation** is reflexive.

NOTE 1 A reflexive **Binary_Relation** is one where the presence of a **Concept** anywhere in the relation implies that an additional **Link** must exist between the Concept and itself.

NOTE 2 Metamodel construct is: *Attribute of **Binary_Relation***.

3.3.154
region_identifier

EDITOR'S NOTE #35. (Action required) Several of the changes proposed by Issue 240 are intended to support IETF RFC 4646 Tags for Identifying Languages. However, RFC 4646 recommends using ISO 3166-1 2-char alpha codes where available, and 3 digit numeric codes where no 2-char alpha code exists. 11179-3 edition 2 specified the use of 3 digit numeric codes, with extensions if necessary. In RFC 4646, extensions are supported either through the use of 2-char alpha codes reserved for private use by 3166-1, or by separate extension and private use identifiers. We need to decide which approach to use in edition 3. An application needing 2 char alpha codes, could translate from the 3 digit numeric code.

identifies a specific country, territory, or region whose linguistic variations apply.

NOTE Metamodel construct is: *Attribute of **Language_Identification***.

3.3.155
Register
data store where **Registered_Items** are recorded and managed.

NOTE Metamodel construct is: *Class*.

3.3.156
Registered_Item
metadata item that is recorded and managed in a **Metadata Registry**.

NOTE 1 A Registered_Item is either an **Administered_Item** or an **Attached_Item**.

NOTE 2 Metamodel construct is: *Sub-class of **Designatable_Item***.

3.3.157
Registrar
representative of a **Registration_Authority**

NOTE 1 Metamodel construct is: *Sub-class of **Contact***

3.3.158
registrar_identifier
identifier for the **Registrar**

NOTE Metamodel construct is: *Attribute of* ***Registrar***.

3.3.159
Registration

EDITOR'S NOTE #36. (Action Required) Now that we have introduced Registered_Item as a super-type of Administered_Item, it would seem that the 'registration' association should be with 'Registered_Item', and a separate 'administration' association with 'Administered_Item', or, if the association name is considered correct, then the classes need to be renamed to better reflect their usage. We need a statement of requirements before we can correctly model this.

EDITOR'S NOTE #37. (Action Required) Registration has been changed from a simple association to an association class to allow an Administered_Item to be registered by more than one Registration_Authority, in response to Issue 176. Some of the attributes formerly in Administration_Record have been moved to Registration. Each attribute needs to be reviewed to ensure it is appropriately positioned.

association between an **Administered_Item** and the **Registration_Authority**

NOTE 1 Metamodel construct is: *Association Class*.

NOTE 2 See also 3.2.31 **registration**.

3.3.160
Registration_Authority
RA
Organization responsible for maintaining a **Register**

NOTE Metamodel construct is: *Sub-class of* ***Organization*** *and of* ***Identifier_Space***.

3.3.161
Registration_Authority_Identifier
identifier assigned to a **Registration_Authority**

NOTE 1 Metamodel construct is: *Composite datatype*.

NOTE 2 See ISO/IEC 11179-6 and ISO/IEC 6523-2.

3.3.162
registration_authority_registrar
association between a **Registrar** and the **Registration_Authority** s/he represents

NOTE Metamodel construct is: *Association*.

3.3.163
registration_status
designation of the status in the registration life-cycle of an **Administered_Item**

NOTE 1 Metamodel construct is: *Attribute of* ***Registration***.

NOTE 2 Designation values are described in ISO/IEC 11179-6.

3.3.164
Relation

EDITOR'S NOTE #38. (Action Required) Is it possible to provide a plain English definition for this term, and move the mathematical definition to a NOTE.

n-ary relation on sets A_1, ..., A_n is a set of ordered *n*-tuples <a_1, ..., a_n> where a_i is an element of A_i for all *i*, *i between 1 and n* .

NOTE 1 Metamodel construct is: *Class*.

NOTE 2 Thus an *n*-ary relation on sets $A_1, ..., A_n$ is a subset of Cartesian product $A_1 \times ... \times A_n$. Membership of an n-tuple in the relation is specified by means of a predicate which must be true for the n-tuple to be a member of the corresponding relation. In our metamodel, relations are defined over sets of concepts.

3.3.165
relation_membership
association denoting the **Relation** of which a **Link** is a member

NOTE Metamodel construct is: *Association*.

3.3.166
Relation_Role

EDITOR'S NOTE #39. (Action required) Review impact of latest model changes.

Designatable_Item that distinguishes and (optionally) names and describes one or more of the elements of a tuple in a **Relation.**

NOTE Metamodel construct is: *Class*.

3.3.167
scope_rule
rule specifying the range within which the naming convention is in effect

[Derived from ISO/IEC 11179-5]

NOTE Metamodel construct is: *Attribute of **Naming_Convention***

3.3.168
Scoped_Identifier
identifier of an **Identified_Item** within a specified **Identifier_Space**

NOTE 1 Metamodel construct is: *Class*

NOTE 2 The **Identifier_Space** provides the scope within which the **Scoped_Identifier** uniquely identifies the **Identified_Item**.

3.3.169
script_identifier
identifies the set of graphic characters used for the written form of one or more languages

NOTE 1 Metamodel construct is: *Attribute of **Language_Identification***

NOTE 2 Use the four character codes from ISO 15924:2004 codes for the representation of the names of scripts

3.3.170
semantic_rule
rule specifying the meanings of name parts and possibly separators that delimit them in a Naming_Convention

[Derived from ISO/IEC 11179-5]

NOTE Metamodel construct is: *Attribute of **Naming_Convention***

3.3.171
Sign
textual string or symbol that can be used to denote a concept

NOTE Metamodel construct is: *Datatype*

3.3.172
Slot
container for extensions to **Identified_Items**

NOTE Metamodel construct is: *Class*

3.3.173
slot_name
name of the **Slot**

NOTE Metamodel construct is: *Attribute of **Slot***

3.3.174
slot_type
datatype of the **slot_value**

NOTE Metamodel construct is: *Attribute of **Slot***

3.3.175
slot_value
value assigned to the **Slot**

NOTE Metamodel construct is: *Attribute of **Slot***

3.3.176
stewardship
<Administered_Item> **association** of an **Administered_Item** to a **Stewardship_Record**

NOTE 1 Metamodel construct is: *Association*

NOTE 2 See also **3.2.34 stewardship** (of metadata).

3.3.177
stewardship_contact
Contact information associated with a **Stewardship**

NOTE Metamodel construct is: *Attribute of **Stewardship***.

3.3.178
Stewardship_Record
record of a **steward** (an **Organization**) and a **stewardship_contact** (a **Contact**) involved in the **stewardship** of an **Administered_Item**

NOTE Metamodel construct is: *Class*.

3.3.179
String
family of datatypes which represent strings of symbols from standard character-sets.

[ISO/IEC 11404:1996 10.1.5 Character String]

NOTE 1 The syntax and semantics of the String datatype are as defined in ISO/IEC 11404:1996 10.1.5 Character String

NOTE 2 Metamodel construct is: *Datatype*

3.3.180
submission
association of a **Registered_Item** to a **Submission_Record**

NOTE Metamodel construct is: *Association*

3.3.181
submission_contact
Contact information associated with a **Submission**

NOTE Metamodel construct is: *Attribute of* ***Submission***.

3.3.182
Submission_Record
record of a **submitter** (an **Organization**) and a **submission_contact** (a **Contact**) involved in the **submission** of a **Registered_Item**

NOTE Metamodel construct is: *Class*.

3.3.183
Symmetric_Relation
Binary_Relation where a **Link** between any two **Concepts,** Concept A and Concept B necessarily means that a **Link** of the same type also exists in the opposite direction between Concept B and Concept A

NOTE Metamodel construct is: *Sub-class of* ***Binary_Relation***

3.3.184
syntactic_rule
rule specifying the arrangement of parts within a name for a Naming_Convention.

[Derived from ISO/IEC 11179-5]

NOTE Metamodel construct is: *Attribute of* ***Naming_Convention***

3.3.185
term_definition_pairing

EDITOR'S NOTE #40. (Action required) It has been proposed that the pairing needs to be related to Context.

association binding a **Designation** to its associated **Definition**

NOTE Metamodel construct is: *Association*.

3.3.186
Text
<datatype> datatype that supports the recording of textual data

NOTE Metamodel construct is: *Datatype*

3.3.187
transitive_indicator
Boolean that determines whether a **Binary_Relation** is transitive or not.

NOTE 1 A transitive **Binary_Relation** is one where a **link** between two **concepts** - Concept A and Concept B - and a second **link** of the same type between Concept B and Concept C implies that there is also a link between Concept A and Concept C.

NOTE 2 Metamodel construct is: *Attribute of* ***Binary_Relation***

3.3.188
Unit_of_Measure
⟨Value_Domain⟩ actual units in which the associated values are measured

NOTE 1 Metamodel construct is: *Composite Datatype*.

NOTE 2 ISO 31-0:1982 specifies a system of physical measurement (the International System of Units, SI). Physical measurement is only one type of measurement. Value measurement is another type of measurement. ISO/IEC 11179-3 allows the use of any appropriate system of measurement.

NOTE 3 The **dimensionality** of the associated **Conceptual_Domain** must be appropriate for the specified **Unit_of_Measure**.

3.3.189
unit_of_measure_dimensionality
Dimensionality that specifies the equivalence relation that applies to all values representing this particular unit.

NOTE Metamodel construct is: *Attribute of **Unit_of_Measure***.

3.3.190
UnlimitedNatural
datatype comprising the "natural numbers", i.e. the positive integers, excluding zero.

EDITOR'S NOTE #41. (Action required) If the difference between the datatype defined here and that defined in 11404 as 'Natural Number' is the exclusion of zero as described here, a better name for our datatype would be 'Non-zero_Natural_Number'. If we keep the existing name, we should insert an underscore for consistency with our naming convention.

NOTE 1 ISO/IEC 11404:1996 10.1.1 defines the datatype, Natural Number, including zero.

NOTE 2 Metamodel construct is: *Datatype*

3.3.191
unresolved_issue
any problem that remains unresolved regarding proper documentation of the **Administered_Item**

NOTE Metamodel construct is: *Attribute of **Administration_Record***.

3.3.192
until_date
date the **Registration** of an **Administered_Item** by a **Registration_Authority** in a Registry is no longer effective

NOTE Metamodel construct is: *Attribute of **Registration***.

3.3.193
Value

EDITOR'S NOTE #42. (Informational) Issue 229 has collapsed the Edition 2 Value class into Permissible_Value, and has renamed value_item to permitted_value.

EDITOR'S NOTE #43. (Action required) issue 240 has changed the interpretation of Value to be equivalent to *multivalue* in 19773. When Text and Value were introduced in the model, the intent was to be able to leave the specification of the datatypes until later, and be able to do it in one place. While we need to provide functionality equivalent to that provided by *multivalue/mulittext*, it seems unnecessarily confusing to associate that functionality with the terms *Value/Text*. Should we simply use the terms 'multivalue/multitext'?

EDITOR'S NOTE #44. (Action required) Contextualized-Value is not defined.

set of values that all have the same meaning, but may have different representations and different datatypes that are dependent upon the context of use.

NOTE 1 **Value** represents a datatype that conforms to the semantics of the contextualized_value portion of the "multivalue" datatype as described in ISO/IEC 19773 module 11. A **Value** is a list of contextualized_values, each of which is a combination of "context-designation" that determines both the value space and the operation set of the actual value and an octet string and that represents the value itself.

See: [ISO/IEC 11404:1996 10.1.9 Private] for further explanation.

NOTE 2 The datatype associated with each contextualized_value is that of the **Value_Domain** of which this **Value** is a member..

NOTE 3 Metamodel construct is: *Datatype*.

3.3.194
Value_Domain
VD
set of **Permissible_Value**s

NOTE 1 Metamodel construct is: *Class*.

NOTE 2 The **Value_Domain** provides representation, but has no implication as to what **Data_Element_Concept** the **Value**s may be associated with nor what the **Value**s mean

NOTE 3 The **Permissible_Value**s may either be enumerated or expressed via a description.

3.3.195
value_domain_datatype
Datatype used in a **Value_Domain**

NOTE Metamodel construct is: *Attribute of **Value_Domain***.

3.3.196
value_domain_format
template for the structure of the presentation of the **Value**(s)

EXAMPLE – YYYY-MM-DD for a date.

NOTE Metamodel construct is: *Attribute of **Value_Domain***.

3.3.197
value_domain_maximum_character_quantity
the maximum number of characters to represent the **Data_Element** value

NOTE 1 Metamodel construct is: *Attribute of **Value_Domain***.

NOTE 2 Applicable only to character datatypes.

3.3.198
value_domain_meaning
association between a **Conceptual_Domain** and a **Value_Domain**

NOTE Metamodel construct is: *Association*.

3.3.199
value_domain_unit_of_measure
Unit_of_Measure used in a **Value_Domain**

NOTE Metamodel construct is: *Attribute of **Value_Domain***.

3.3.200
Value_Meaning
meaning or semantic content of a **Value**

NOTE 1 Metamodel construct is: *Sub-class of **Concept***.

NOTE 2 The representation of **Value_Meaning**s in a registry shall be independent of (and shall not constrain) their representation in any corresponding **Value_Domain**.

3.3.201
value_meaning_begin_date
effective_date of this **Value_Meaning** in the **Conceptual_Domain**

NOTE 1 Metamodel construct is: *Attribute of **Value_Meaning***.

NOTE 2 A Registration_Authority may determine whether this date is the date the **Value_Meaning** becomes valid in a registry or the date the **Value_Meaning** becomes part of the source domain or some other date.

3.3.202
value_meaning_end_date
date this **Value_Meaning** became/becomes invalid

NOTE 1 Metamodel construct is: *Attribute of **Value_Meaning***.

NOTE 2 A Registration_Authority may determine whether this date is the date the **Value_Meaning** becomes no longer valid in a registry or the date the **Value_Meaning** becomes no longer part of the source domain or some other date.

3.3.203
value_meaning_set
association between a **Conceptual_Domain** and a set of **Value_Meaning**s.

NOTE Metamodel construct is: *Association*.

3.3.204
variant_identifier
identifies a language variant, which indicates additional, well-recognized variations that define a language or its dialects that are not covered by other available identifiers.

NOTE 1 Variant identifiers are typically represented as dates and are used distinguish events such as spelling reforms. Variant identifiers can be order dependent. String Numeric variant_identifiers are interpreted to be Gregorian calendar year numbers. Alphanumeric variant_identifiers reference IANA variant subtags.

NOTE 2 Metamodel construct is: *Attribute of **Language_Identification***

3.3.205
version
unique version **identifier** of the **Administered_Item**

NOTE Metamodel construct is: *Attribute of **Administered_Item***.

3.4 List of Abbreviations

The following abbreviations are defined for use within the subject domain of this document.

3.4.1
CD
Conceptual Domain

3.4.2
DE
Data Element

3.4.3
DEC
Data Element Concept

3.4.4
MDR
Metadata Registry

3.4.5
opi
organization_part_identifier

3.4.6
RA
Registration Authority

3.4.7
RDF
Resource Description Framework

3.4.8
UML
Unified Modeling Language

3.4.9
VD
Value Domain

3.4.10
W3C
World Wide Web Consortium

3.4.11
XML
eXtensible Markup Language

4 Structure of a Metadata Registry

4.1 Metamodel for a Metadata Registry

A metamodel is a model that describes other models. A metamodel provides a mechanism for understanding the precise structure and components of the specified models, which are needed for the successful sharing of the models by users and/or software facilities.

This part of ISO/IEC 11179 uses a metamodel to describe the structure of a *Metadata Registry*. The registry in turn will be used to describe and model other data, for example about enterprise, public administration or business applications. The *registry metamodel* is specified as a conceptual data model, i.e. one that describes how relevant information is structured in the natural world. In other words, it is how the human mind is accustomed to thinking of the information.

As a conceptual data model, there need be no one-to-one match between the attributes in the model and fields, columns, objects, et cetera in a database. There may be more than one field per attribute and some entities and relationships may be implemented as fields. There is no intent that an implementation should have a table for each relationship or entity. The metamodel need not be physically implemented as specified.

The structure described by this metamodel may be distributed over several implementations. These implementations may be databases, data repositories, metadata registers, metadata registries, dictionaries, etc.

The model shows constraints on minimum and maximum occurrences of attributes. The constraints on maximum occurrences are to be enforced at all times. The constraints on minimum occurrences are to be enforced when the registration_status for the metadata item is "recorded" or higher. In other words, a registration_status of "recorded" indicates that all mandatory attributes have been documented.

4.2 Application of the metamodel

Some of the objectives of the metamodel for a *Metadata Registry* are to:

— provide a unified view of concepts, terms, value domains and value meanings;

— promote a common understanding of the data described;

— enable the sharing and reuse of the contents of implementations.

A metamodel is necessary for coordination of data representation between persons and/or systems that store, manipulate and exchange data. The metamodel will assist registrars in maintaining consistency among different registries. The metamodel enables systems tools and information registries to store, manipulate and exchange the metadata for data attribution, classification, definition, naming, identification, and registration. In this manner, consistency of data content supports interoperability among systems tools and information registries.

Using the metamodel, mappings to the schema of each tool set can be developed. The metamodel constructs can be translated into the language of each tool set, preserving the concepts represented in the original model.

It is assumed that an implementer will use this conceptual data model to develop a more specific logical data model of the identical sphere of interest. A logical data model describes the same data, but as structured in an information system. It is often referred to as a Model of the Information System. A logical data model can be directly used for database design.

4.3 Specification of the metamodel

4.3.1 Terminology used in specifying the metamodel

When using a model to specify another model, it is easy for the reader to become confused about which model is being referred to at any particular point. To minimize this confusion, this document deliberately uses different terms in the model being specified from those used to do the specification.

The *registry metamodel* is specified using a subset of the Unified Modelling Language (UML). This document uses the term "metamodel construct" for the model constructs it uses, but "metadata objects" for the model constructs it specifies. The metamodel constructs used are: classes, associations, association classes, attributes, composite attributes and composite datatypes. These terms are defined in 3.1, and their use is described in Annex B. The specified metadata objects are defined in 3.3, and as the main subject of Clauses 5, 6 and 7.

EDITOR'S NOTE #45. (Action required) The above text will need to be revised when we move the definitions of metadata objects out of clause 3.

However, there are certain parallels between the two models. For example, the "Object_Class" specified in the model is equivalent to the metamodel construct “class” used to specify the model, and the “Characteristic” specified in the model is equivalent to the metamodel construct “attribute” used to specify the model. The different terms are used to make it clear which model is being referred to, not because they represent different concepts. One term that this document uses at both levels is “datatype”, but the level to which it applies should be apparent from the context in which it is used.

4.3.2 Use of UML Packages

For descriptive and conformance purposes, the metamodel is organized into packages:

- **Basic package** (clause 5) – contains simple classes that are reused by other classes
- **Identification, designation and definition package** (clause 6) – contains classes that allow the contents of a registry to be identified, named or otherwise designated, and defined. This package is sub-divided into the following regions:
  - Identification region (see 6.1)
  - Designation and Definition region (see 6.2)
- **Registration package** (clause 7) – contains classes that allow metadata items to be registered
  - Registration region (see 7.1)
  - Administration_Record region (see 7.2)
- **Relations package** (clause 8) – contains classes that allow concepts to be related
- **Data Descriptions Package** (clause 9) – contains classes that allow the description of specific metadata objects:
  - Data_Element_Concepts region (see 9.2)
  - Conceptual and Value_Domains region (see 9.2.3.3)
  - Data_Elements region (see 9.4).
- **Classification Package** (clause 10) – contains classes that allow the description of classification schemes, concept systems and ontologies.
  - Hierarchical Classification (see 10.1)

- Concept_Systems (see 10.2)
- Ontologies (see 10.3)

4.3.3 Package Dependencies

Figure 1 illustrates the dependencies among the packages.

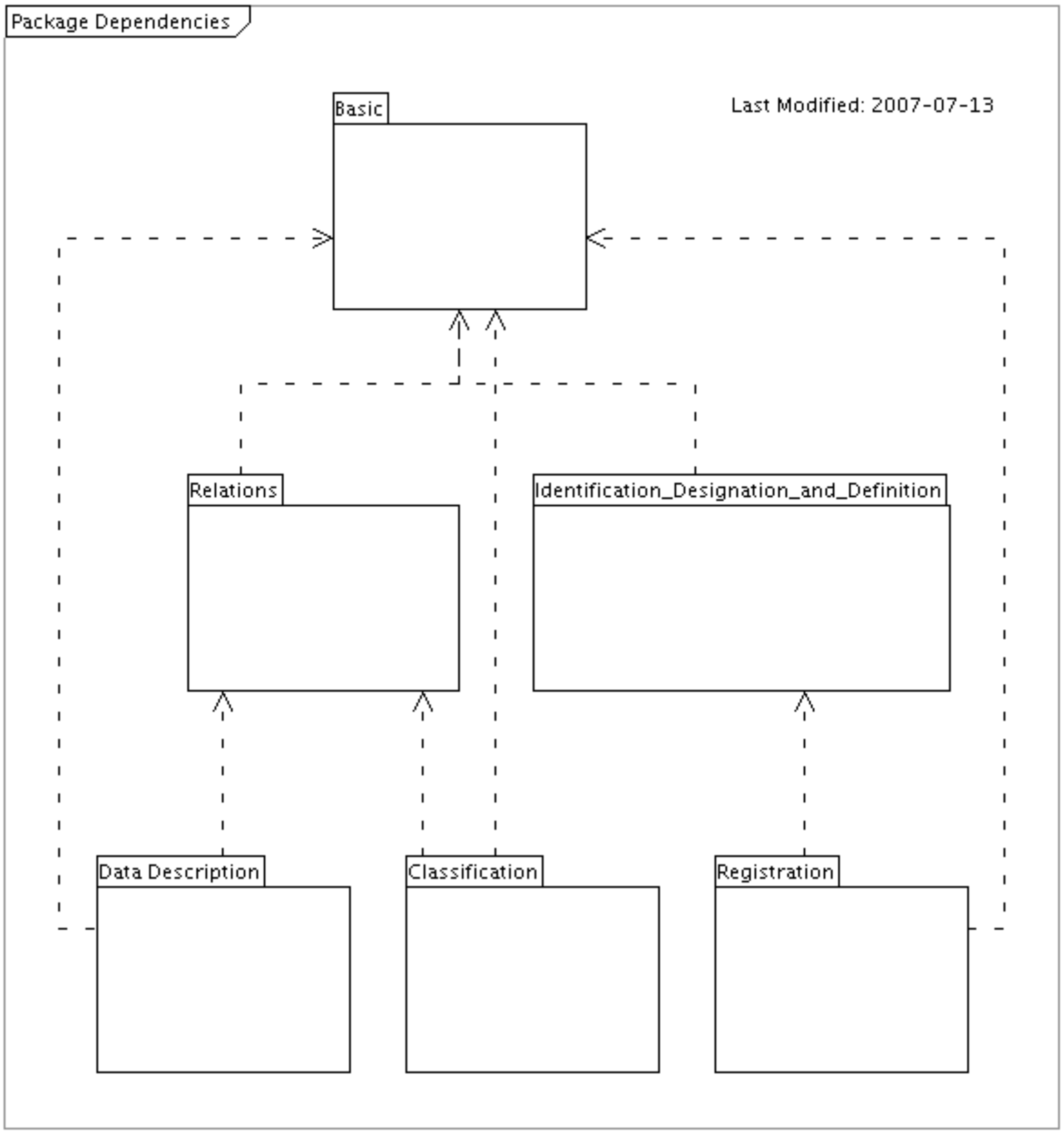


Figure 1 — Package dependencies

4.3.4 Use of UML Class diagrams and textual description

This standard uses both text and UML class diagrams to describe the metamodel. Both are normative, and are intended to be complementary. However, if a conflict exists between what is specified in UML and what is specified in text, the text takes precedence until such time as a correction is made to make them consistent.

A consolidated UML class hierarchy is included as Annex B.

EDITOR'S NOTE #46. (Action required) In clauses 5 thru 10, the text description of the metamodel uses a variety of styles, some text being a rigorous representation of the model, other text using a looser English prose. We should aim to be more consistent, and we should try to make the text as easy to read as possible, while still being accurate.

4.4 Types, Instances and Values

When considering data and metadata, it is important to distinguish between types of data/metadata, and instances of these types and their associated values. The metamodel specifies types of classes, attributes and associations. Any particular instance of one of these will be of a specific type, and at any point in time, that instance will have a specific value (possibly null). As examples, this document defines *attribute instance* and *attribute value*, but the same principle applies to classes, relationships and all other metamodel constructs defined in 3.1.

Clauses 5 through 10 of this document specify the types of metadata objects that form the structure of a metadata registry. A metadata registry will be populated with instances of these metadata objects (metadata items), which in turn define types of data, e.g. in an application database. In other words, instances of metadata specify types of application level data. In turn, the application database will be populated by the real world data as instances of those defined data types.

NOTE ISO/IEC 10027:1990 IRDS Framework explains the concepts of different levels of modelling.

4.5 Types of Items in an ISO/IEC 11179 metadata registry

Figure 2 shows the types of items specified by this Part of this International Standard. These types are explained in subsequent clauses. Any *metadata item* entered into a *metadata registry* shall be immediately at least an *Identified_Item*, so the item may be referenced, and must become a *Designatable_Item* in order to be named and/or defined. A *Registration_Authority* responsible for the registry shall determine which *Identified_Items* should become *Registered_Items* and/or *Designatable_Items*. Annex C shows an example of how the various *metadata objects* specified by this standard could be sub-typed from *Registered_Item*.

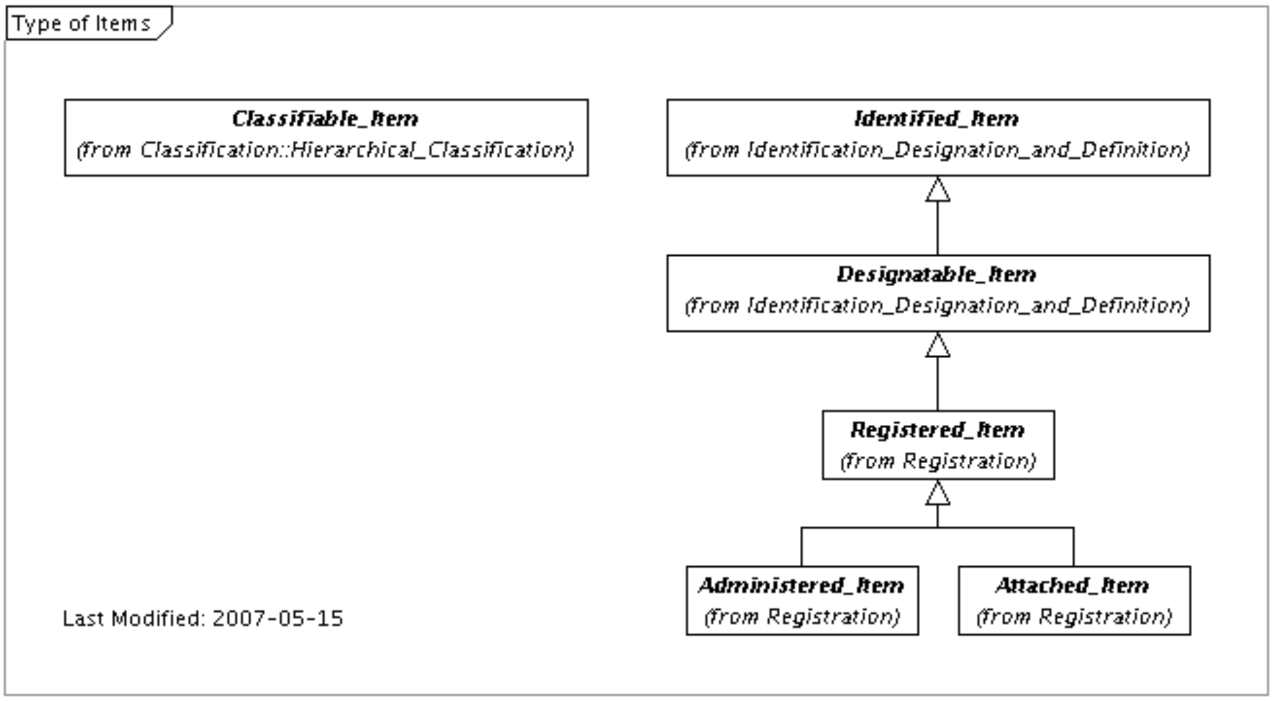

Type of Items
Classifiable_Item
(from Classification::Hierarchical_Classification)
Identified_Item
(from Identification_Designation_and_Definition)
Designatable_Item
(from Identification_Designation_and_Definition)
Registered_Item
(from Registration)
Administered_Item
(from Registration)
Attached_Item
(from Registration)
Last Modified: 2007-05-15

Figure 2 — Types of items

EDITOR'S NOTE #47. (Action required) Last minute changes at the New York meeting removed the explicit sub-typing of Registered_Item by those other classes specified in this standard. Figure C.2 in Annex C illustrates the explicit sub-typing. The intent behind this change was to allow an implementer (or possibly even a registration authority) to specify which classes should be sub-typed from Registered_item. This change is problematic from an interoperability perspective, since different implementations may make different choices. It is also problematic from the perspective of 20944, which needs a defined data model to navigate. Should we restore the explicit sub-typing we had before? See also clause 7. Metadata items must be Designatable_Items in order to be named (designated) and/or defined.

4.6 Extensibility

It is not expected that this metamodel will completely accommodate all users. Particular sectors, such as document management, scientific data, statistical data, require metadata attributes not addressed in this standard. Such extensions shall be considered conformant if they do not violate any of the rules inherent in the structure and content as specified by the metamodel in this standard. Classes, relationships, and attributes may be added to this conceptual data model.

Implementers of this standard may include extensions as part of an implementation, and/or they may provide facilities to allow a registry user to define their own extensions. This standard provide *Slots* (see 6.1.2.5) as a mechanism to extend metadata items with custom attributes.

EDITOR'S NOTE #48. (Action required) Can we provide an equivalent mechanism for adding classes and associations?

4.7 Date References

In this standard, dates are important attributes of an Administration_Record and of operations of a registry. For the purpose of this standard, “date” refers to Gregorian calendar date {see ISO 8601:2000} and the associated default representation is YYYY-MM-DD (i.e. Year-Month-Day). For example, 12 October, 2001 if referenced in numeric form should be 2001-10-12 and not, for example, as 12-10-2001 (which might be confused with 10 December, 2001).

For the present, the specification of time in addition to date should be consider a user extension to this standard.

EDITOR'S NOTE #49. (Action required) Why do we allow DateTime instead of just Date? How is a user supposed to add this extension?

5 Basic Package

5.1 Datatypes and Classes in the Basic package

5.1.1 Overview

The Basic package specifies common datatypes for use elsewhere in the metamodel. All of the other types used in the model are based on this core set of types, and any compliant implementation of a metadata registry should include an implementation of the semantics specified in these core types.

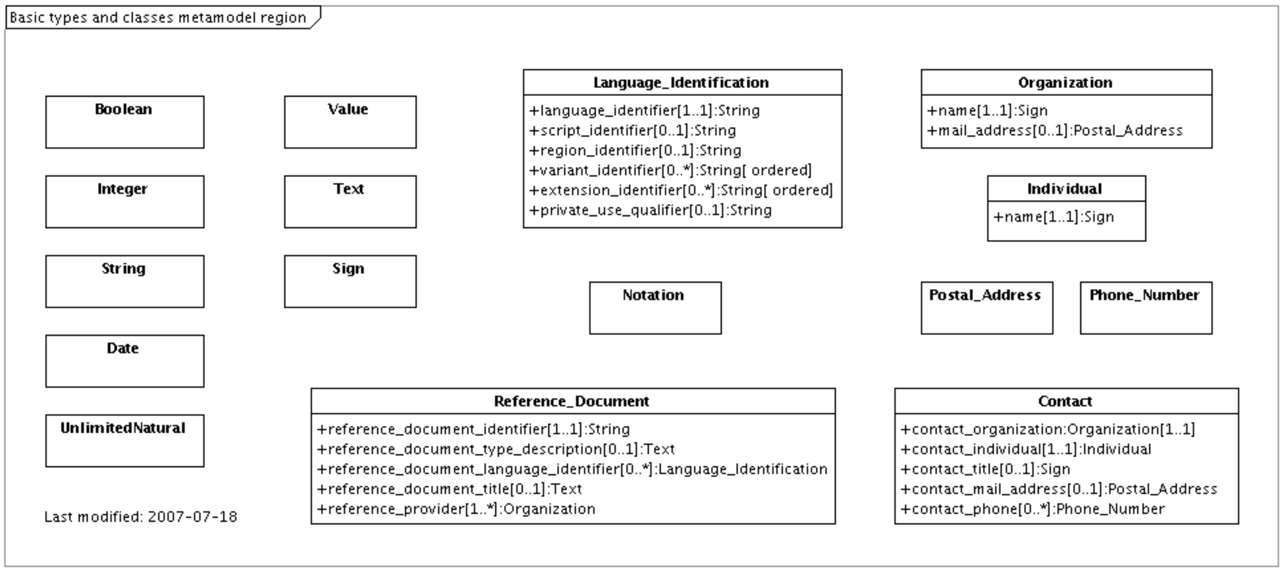

Basic types and classes metamodel region
Boolean
Integer
String
Date
UnlimitedNatural
Value
Text
Sign
Language_Identification
+language_identifier[1..1]:String
+script_identifier[0..1]:String
+region_identifier[0..1]:String
+variant_identifier[0..*]:String[ordered]
+extension_identifier[0..*]:String[ordered]
+private_use_qualifier[0..1]:String
Notation
Organization
+name[1..1]:Sign
+mail_address[0..1]:Postal_Address
Individual
+name[1..1]:Sign
Postal_Address
Phone_Number
Reference_Document
+reference_document_identifier[1..1]:String
+reference_document_type_description[0..1]:Text
+reference_document_language_identifier[0..*]:Language_Identification
+reference_document_title[0..1]:Text
+reference_provider[1..*]:Organization
Contact
+contact_organization:Organization[1..1]
+contact_individual[1..1]:Individual
+contact_title[0..1]:Sign
+contact_mail_address[0..1]:Postal_Address
+contact_phone[0..*]:Phone_Number
Last modified: 2007-07-18

Figure 3 — Datatypes and Classes in the Basic package

5.1.2 Boolean

A mathematical datatype associated with two-valued logic. [ISO/IEC 11404:1996, 8.1.1].

NOTE The notation and semantics for Boolean is as described in ISO/IEC 11404.

5.1.3 Contact

EDITOR'S NOTE #50. (Action required) Issue 106 proposes restructuring Contact based on the Contact Information module of 19773.

EDITOR'S NOTE #51. (Action required) Should more than one mail_address be permitted? If so, do we need to be able to distinguish the addresses as to purpose or usage?

Contact is the class of object to whom an information item(s), a material object(s) and/or person(s) can be sent to or from in a specified context.

Every *Contact* shall have exactly one *contact_organization* of type *Organization* for which the *Contact* is a representative.

Every *Contact* shall have exactly one *contact_individual* of type *Individual.* The *contact_individual* is the *Individual* that the *Contact* information relates to.

A *Contact* may have zero or one *contact_titles* of type *Sign*, that identifies the position held by the *contact_individual*.

A *Contact* may have zero or one *contact_mail_addresses* of type *Postal_Address*, where the *contact_individual* may be contacted by mail.

A *Contact* may have zero or more *contact_phones* of type *Phone_Number* where the *contact_individual* may be contacted by phone.

Registrar is a subclass of *Contact*.

Attribute	**Occurrences**	**Datatype**
contact_organization	One per *Contact*.	*Organization*
contact_individual	One per *Contact*.	*Individual*
contact_title	Zero or one per *Contact.*	*Sign*
contact_mail_address	Zero or one per *Contact.*	*Postal_Address*
contact_phone	Zero, one or many per Contact	*Phone_Number*

5.1.4 Date

A family of datatypes whose values are points in time to various common resolutions: year, month, day, hour, minute, second, and fractions thereof [ISO/IEC 11404:1996, 8.1.6].

NOTE Both the notation and semantics of the Date datatype is as specified in ISO/IEC 11404:1996:8.1.6. As specified, the representation of time literals are defined in ISO 8601:1988

5.1.5 Individual

Individual is a single human being. Every *Individual* shall have exactly one *name* of type *Sign*.

A *name* is a *Sign* that uniquely identifies the *Individual* within the context of a *Designation_Space*.

Attribute name	**Occurrences**	**Datatype**
name	One per *Individual*	*Sign*

5.1.6 Integer

A mathematical datatype comprising the exact integral values [ISO/IEC 11404:1996, 8.1.7].

NOTE Both the notation and semantics of the Integer datatype is as specified in ISO/IEC 11404:1996:8.1.7.

5.1.7 Language_Identification

The composite datatype *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.

EDITOR'S NOTE #52. (Action required) It has been suggested that we expand Language_Identification to include formal languages. No detailed proposal has been provided.

EDITOR'S NOTE #53. (Action required) The attributes of the Identifier come from IETF RFC 4646. This is a change from Edition 2. We need to provide backwards compatibility. The former *country_identifier* is renamed to *region_identifier* (which is more accurate since not all the codes in 3166-1 represent countries). The new attributes are all optional.

EDITOR'S NOTE #54. (Informational) The W3C has a description of the use of the IETF language tags at: http://www.w3.org/International/articles/language-tags/Overview.en.php

The identifier is comprised of the following parts, which are based on IETF RFC 4646:

- a mandatory *language_identifier* that identifies the primary language
- an optional *script_identifier* that identifies the set of graphic characters used for the written form of one or more languages
- an optional *region_identifier* that denotes the area or region in which a word, term, phrase or language variant is used.
- zero or more *variant_identifiers* that denotes a specific variant or variants of a given language. Variant identifiers are typically represented as dates and are used to distinguish events such as spelling reforms.
- zero or more *extension_identifiers* that denote extensions to a given language. Extensions consist of key-value pairs, which may be order dependent.

EDITOR'S NOTE #55. (Action required) In the table below, the NOTE specifies that the key value pairs are separated by the EQUALS SIGN (=). Presumably this means that each 'key' is separated from its corresponding value by the EQUALS SIGN (=). How are the pairs themselves to be separated?

EDITOR'S NOTE #56. (Action required) RFC 4646 requires the extension identifiers to be prefixed by a single character that identifiers the registration authority that has registered the extension. As of 2007-07-31, no extensions have been registered.

- an optional *private use qualifier* that provides additional qualification for specific non-standardized purposes and uses.

Attribute	**Occurrences**	**Datatype**
language_identifier	One per *Language identification*	String NOTE Use the three character alphabetic codes from ISO 639-2/Terminology, with extensions if required.
script_identifier	Zero or one per *Language identification*	String NOTE Use the four character codes from ISO 15924:2004 codes for the representation of the names of scripts.
region_identifier	Zero or one per *Language identification*	String NOTE Use the three digit numeric codes from ISO 3166-1, with extensions if required.

Attribute	Occurrences	Datatype
variant_identifier	Zero or more per *Language identification*	String NOTE Numeric variant_identifiers are interpreted to be Gregorian calendar year numbers. Alphanumeric tags reference IANA variant subtags
extension_identifier	Zero or more per *Language identification*	String NOTE Extension identifiers are ordered and consist of key-value pairs, separated by the EQUALS SIGN (=). The values must be alphanumeric with no embedded white-space.
private use qualifier	Zero or one per *Language identification*	String

5.1.8 Notation

Notation denotes a notation used by a concept system. Examples of such notations include XCL Common Logic (ISO 24707) or OWL-DL XML notation.

EDITOR'S NOTE #57. (Action required) Does notation denote the syntax, the semantics or both?

EDITOR'S NOTE #58. (Action required) We need some sort of standard list of notation representations.

5.1.9 Organization

Organization is a unique framework of authority within which *individuals* act, or are designated to act, towards some purpose.

Every *Organization* shall have exactly one *name* of type *Sign*.

EDITOR'S NOTE #59. (Action required) Many organizations have more than one name. For example, a legal name and one or more trade names. Do we want to allow for this?

An *Organization* may have zero or more *organization contact* associations with a *representative* of type *Contact*

An *Organization* may have zero or one *mail_address*es of type *Postal_Address,* where the *Organization* can be contacted by mail.

Attribute name	Occurrences	Datatype
name	One per *Organization*	*Sign*
mail_address	Zero or one per *Organization*	*Postal_Address*

EDITOR'S NOTE #60. (Action required) Why does an organization not have a phone number?

5.1.10 Phone_Number

EDITOR'S NOTE #61. (Action required) Do we need to reference both ITU-T E.164 and ISO/IEC 19773 module 17. What value does the reference to ISO/IEC 19773 add here?

A phone number uniquely identifies a telephone line within a telephone network. The data structure of the Phone_Number data element shall conform to ITU-T E 164 and may conform to ISO 19773 Information technology – Metadata registries (MDR) Modules – Module 17: Data structure for ITU-T E.164 phone number data.

5.1.11 Postal address

EDITOR'S NOTE #62. (Action required) Should we reference UPU S42 directly in addition to or instead of ISO/IEC 19773 module 16? Why do we say 'may conform to ISO/IEC 19773' instead of 'shall conform'? What is the value of stating optional conformance?

A postal address allows the unambiguous determination of an actual or potential delivery point, usually combined with the specification of an addressee and/or a mailee. The data structure of Postal address may conform to ISO/IEC 19773 Information technology - Metadata registries (MDR) Modules - Module 16: Data Structure for UPU postal data.

5.1.12 Reference_Document

EDITOR'S NOTE #63. (Action required) The changes to the Datatype from String to Text come from the resolution of Issue 18. We need some explanation of how the Text datatype should be used in this context.

Attribute	Occurrences	Datatype
reference_document_identifier	One per *Reference_Document*	Text
reference_document_type_description	One per *Reference_Document*	Text
reference_document_language_identifier	Zero, one or many per *Reference_Document* (absence of a language indicates use of the same language as specified by Registration_Authority documentation_language_identifier)	*Language_Identification*
reference_document_title	Zero or one per *Reference_Document*	Text
reference_provider	One or many per *Reference_Document*	Organization

5.1.13 Sign

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 represented by various expressions such as character string, sentence, code, icon, and so on.

5.1.14 String

[Character]string is a family of datatypes which represent strings of symbols from standard character-sets. The syntax and semantics of the String datatype are as defined in ISO/IEC 11404:1996 10.1.5 Character String.

5.1.15 Text

EDITOR'S NOTE #64. (Action required) CD19773 uses the term 'multitext' with the same definition as this. We should be consistent, or explain why we are inconsistent. More detail is required to explain the structure of the set of values within *Text*. Should we reference 19773 multitext?

Text is a set of textual values that all have the same meaning, but may have different representations and different datatypes that are dependent upon the context of use.

Text is 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]

EXAMPLE A business letter printed on paper or displayed on a screen.

5.1.16 UnlimitedNatural

UnlimitedNatural is a datatype comprising the “natural numbers“, i.e. the positive integers, excluding zero. It is used in this model to specify an upper bound for a cardinality.

EDITOR'S NOTE #65. (Action required) If the difference between the datatype defined here and that defined in 11404 as 'Natural Number' is the exclusion of zero as described here, a better name for our datatype would be 'Non-zero_Natural_Number'. If we keep the existing name, we should insert an underscore for consistency with our naming convention.

5.1.17 Value

EDITOR'S NOTE #66. (Action required) CD19773 uses the term 'multivalue' with the same definition as this. We should be consistent.

EDITOR'S NOTE #67. (Action required) Since the definition below references 'multidata' does that also need to be included as a defined datatype.

A *Value* is a set of values that all have the same meaning, but may have different representations and different datatypes that are dependent upon the context of use. Value represents a datatype that conforms to the semantics of the contextualized-value portion “multidata” datatype as described in 19773-03. A *Value* is a list of *Contextualized-Values*, each of which is a combination of “context-designation” that determines both the value space and the operation set of the actual value and an octet string and that represents the value itself.

See: [ISO/IEC 11404:1996 10.1.9 Private] for further explanation.

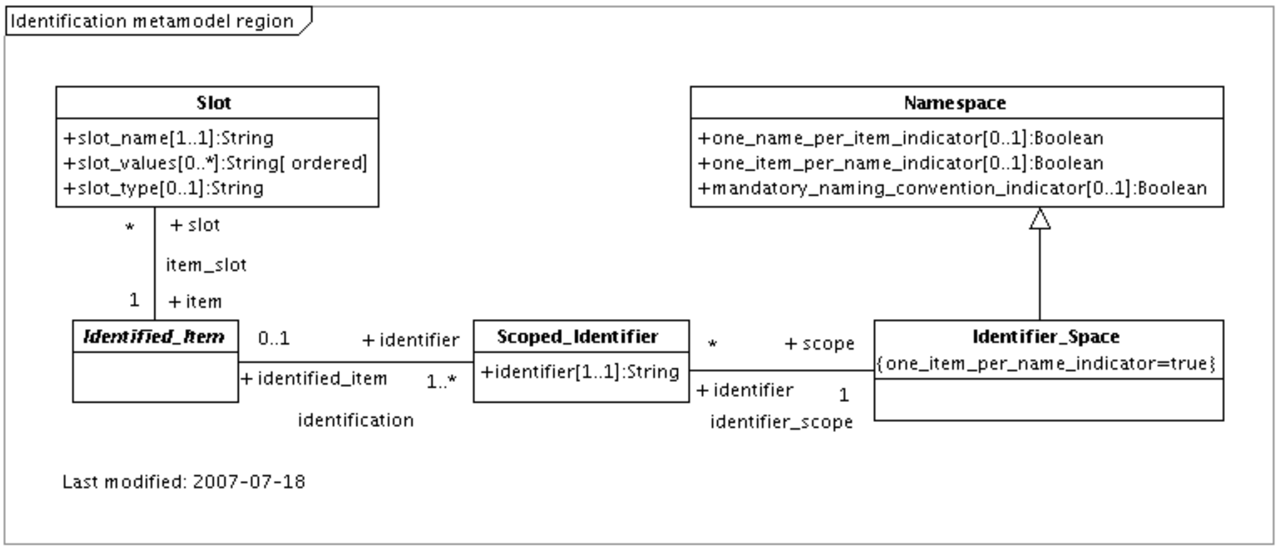
EDITOR'S NOTE #68. (Action required) 11404 and 19773 both have similar definitions for the contextualized value (Private in 11404). Do we really want to spec *sets* vs. individual entries? Do we want to specify some sort of default (e.g. a single character string?

6 Identification, Designation and Definition Package

EDITOR'S NOTE #69. (Action required) Comments on this clause from US NB noted in the attachments to Issue 248 have not yet been applied.

6.1 Identification region

6.1.1 Overview



Identification metamodel region
Slot
+slot_name[1..1]:String
+slot_values[0..*]:String[ordered]
+slot_type[0..1]:String
Namespace
+one_name_per_item_indicator[0..1]:Boolean
+one_item_per_name_indicator[0..1]:Boolean
+mandatory_naming_convention_indicator[0..1]:Boolean
*
+slot
item_slot
1
+item
Identified_Item
0..1
+identifier
+identified_item
1..*
identification
Scoped_Identifier
+identifier[1..1]:String
*
+scope
+identifier
1
identifier_scope
Identifier_Space
{one_item_per_name_indicator=true}
Last modified: 2007-07-18

Figure 4 — Identification metamodel region

EDITOR'S NOTE #70. (Action required) In WD4, Namespace was shown as a subtype of Designatable_Item. That relationship has been removed. Namespace needs to be at least a subtype of Identified_Item, so it can be identified, a subtype of Designatable_Item, if it is to be named, and preferably of Registered_Item so it can be administered. See Editor's Note at the end of sub-clause 4.5 for the more general related issues.

EDITOR'S NOTE #71. (Action required) Slot has been added as an extension mechanism similar to that used in ebXML. Slots have names, so we need to say something about the scope of those names. Since a Slot is related to exactly one Identified_Item, it would make sense that the scope of the name be restricted to that item, allowing duplicate slot names across the registry.

EDITOR'S NOTE #72. (Action required) How could a Registration_Authority add a new sub-type of Registered_Item? Slot does not seem sufficient to do this.

EDITOR'S NOTE #73. (Action required) Should we be able to associate an *Identifier_Space* or a *Namespace* with a *Context*?

6.1.2 Classes in the Identification metamodel region

EDITOR'S NOTE #74. (Action required) The classes in this region are listed in alphabetical order. Is there a more useful sequence to use?

6.1.2.1 Identified_Item

Identified_Items represent *metadata items* that are identified in a *metadata registry*.

Every *Identified_Item* shall have one or more *Identification* associations with a *Scoped_Identifier* that provides the *identifier* for the item within a specific *Identifier_Space*.

An *Identified_Item* may be associated with zero, one or many *Slots* via the association *item_slot*.

Designatable_Item is a subclass of *Identified_Item*.

EDITOR'S NOTE #75. (Informational) The composite datatype Item_identifier that existed in Edition 2 has been eliminated, and its components separated. *identifier* here within *Scoped_Identifier* is equivalent to what was *data identifier* before. See also the Editor's note in 3.3.94.

EDITOR'S NOTE #76. (Action required) Is there a better location for the table below? Is it useful?

The table below illustrates the differences between a *Scoped_Identifier* and a *Sign* (formerly name).

	Sign (formerly name)	**Scoped_Identifier**
Scope	Designation_Space and Context, independently	Identifier_Space
Occurrence	Many allowable per *Designatable_Item*	Many allowable per *Identified_Item*
Language dependent	Yes	No
Type	*Sign*	*String*
Metamodel construct	Attribute of *Designation*	Class that associates zero or one *Identified_Items* with an *Identifier_Space*.

6.1.2.2 Identifier_Space

An *Identifier_Space* is a sub-class of *Namespace* that provides the scope for the *Scoped_Identifier* of an *Identified_Item*. *Identifier_Spaces* must have the *one_item_per_name_indicator* set to 'true'.

An *Identifier_Space* may be associated with zero, one or more *Scoped_Identifiers*, via *identifier_scope* associations.

6.1.2.3 Namespace

A *Namespace* is a scoping construct used to partition the set of designations or 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 designations within a specific *Namespace* without central coordination.

Namespace is a superclass of *Designation_Space* (6.2.2.5) and *Identifier_Space* and the latter in turn is a superclass of *Registration_Authority*.

EDITOR'S NOTE #77. (Action required) Identifier_Space is shown as a super-class of Registration_Authority, but this restricts a Registration_Authority to being/having a single Identifier_Space. It would be more flexible to make this a many-to-many association.

EDITOR'S NOTE #78. Issue 119 proposes making *Namespace* as an Administered_Item. This CD has removed explicit sub-classing of any item type.

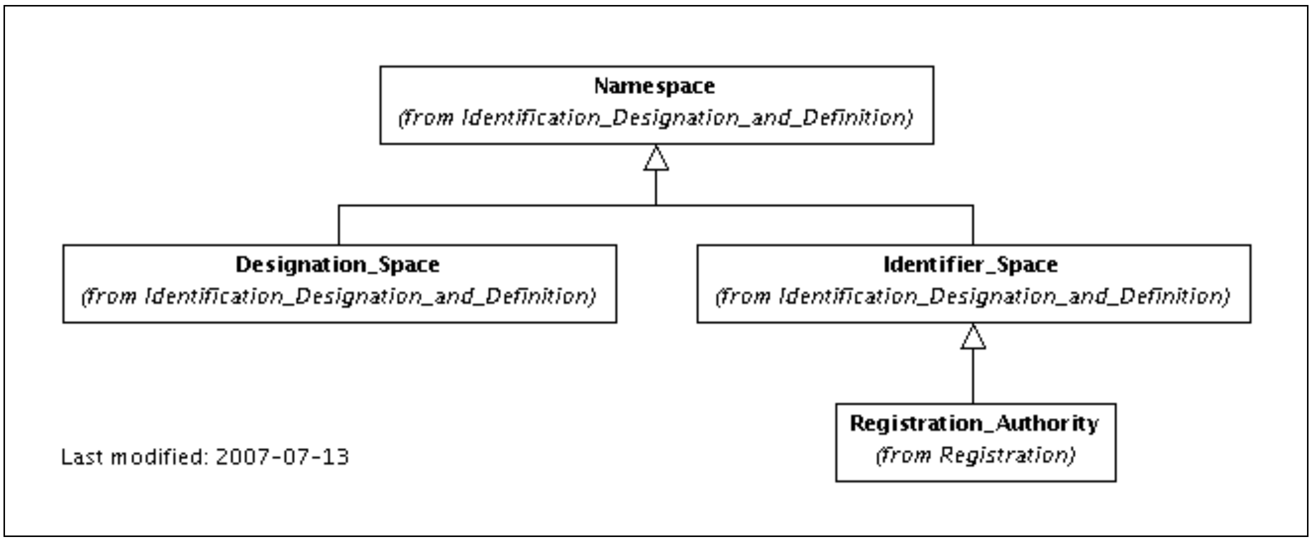

Namespace
(from Identification_Designation_and_Definition)
Designation_Space
(from Identification_Designation_and_Definition)
Identifier_Space
(from Identification_Designation_and_Definition)
Registration_Authority
(from Registration)
Last modified: 2007-07-13

Figure 5 — Types of Namespaces

Depending on the particular subclass, a *Namespace* contains either a set of *Designations* or a set of *Identifications*.

NOTE These are NOT XML Namespaces. However, it may be possible to add additional subclasses of Namespaces to model XML Namespaces.

EDITOR'S NOTE #79. (Action required) *Namespace* is no longer an explicit sub-type of *Identified_Item* and *Designatable_Item*, so the following sentence is no longer true.

A *Namespace* is a sub-type of *Identified_Item* and of *Designatable_Item*, inheriting their attributes and relationships, which allows it to be identified, and optionally named, defined and classified.

EDITOR'S NOTE #80. (Action required) The attributes *of Namespace* are shown as optional. Is this really what we want? We want the facts represented by these indicators to be optional, but we should probably always specify either a True or a False value for each. Otherwise there can be no way to enforce the constraints that these indicators imply.

EDITOR'S NOTE #81. (Action required) Further, are these indicators purely descriptive, or is there an expectation that the registry enforce the implied constraints?

The Namespace class has the following attributes:

Attribute	Occurrences	Datatype
one_name_per_item_indicator	Zero or one per *Namespace*	Boolean
one_item_per_name_indicator	Zero or one per *Namespace*	Boolean
mandatory_naming_convention_indicator	Zero or one per *Namespace*	Boolean

The *one_name_per_item_indicator* is a *Boolean* that determines:

- for a *Designation_Space,* whether or not many *Designations* from the *Designation_Space* can be bound to one *Designatable_Item, and*
- for an *Identifier_Space*, whether or not many *Scoped_Identifiers* from the *Identifier_Space* can be bound to one *Identified_Item*.

If the *one_name_per_item_indicator* is null, then the rule is unspecified.

The *one_item_per_name_indicator* is a *Boolean* that determines:

- for a *Designation_Space,* whether or not a given *Sign* may denote many *Designatable_Items*, and
- for an *Identifier_Space*, that a *Scoped_Identifier* must identify only a single *Identified_Item* (since the indicator must always be true for an *Identifier_Space)*.

If the *one_item_per_name_indicator* is null, then the rule is unspecified.

The *mandatory_naming_convention_indicator* is a *Boolean* that determines whether or not all *Designations* in a *Designation_Space* have to conform to exactly one *Naming_Convention*.

The *mandatory_naming_convention_indicator* is not applicable to *Identifier_Spaces*.

6.1.2.4 Scoped_Identifier

Scoped_Identifier provides the *identifier* of an *Identified_Item* within a specified *Identifier_Space* which in turn provides the scope in which the *identifier* unambiguously identifies this *Identified_Item*.

Scoped_Identifier has one attribute, *identifier* of type *String,* that unambiguously denotes the associated *Identified_Item* in the context of the associated *Identifier_Space*.

Attribute	**Occurrences**	**Datatype**
identifier	One per *Scoped_Identifier*	String

6.1.2.5 Slot

A *Slot* provides a way to record extensions to an *Identified_Item*. A *Slot* is associated with exactly one *Identified_Item* through the association *item_slot*.

Attribute	**Occurrences**	**Datatype**
slot_name	One per *Slot*	String
slot_value	Zero, one or many per *Slot*	String [ordered]
slot_type	Zero or one per *Slot*	String

6.1.3 Associations in the Identification region

6.1.3.1 identification

identification has two roles – identifier (verb form: identifies) and identified_item (verb form: identified_by). Every *Identified_Item* must have one or more *identification* associations with a *Scoped_identifier that* provides an *identifier* for the *Identified_Item*.

6.1.3.2 identifier_scope

EDITOR'S NOTE #82. (Action required) The role name shown in the figure as *identifier* has been renamed to *scoped_identifier* in the text below because this better matches the proposed verb form. The figure and the text need to be made consistent.

identifier_scope has two roles – scope (verb form: provides_scope) and scoped_identifier (verb form: has scope). *identifier_scope* associates zero, one or more *Scoped_Identifier* with exactly one *Identifier_Space*.

6.1.3.3 item_slot

item_slot has two roles – item and slot. *item_slot* associates an *Identified_Item* with zero, one or more *Slots*.

6.2 Designation and Definition region

6.2.1 Overview

The Designation and Definition region is used to manage the designations and definitions of *Designatable_Items* and the *Contexts* for the designations and definitions. A designatable item may have many signs that will vary depending on discipline, locality, technology, etc. This sub-clause describes the classes, associations, and association classes of this region.

Figure 6 represents the Designation and Definition region. This region of the metamodel is based on, and is consistent with, terminological models developed by ISO/TC 37.

EDITOR'S NOTE #83. (Action required) Is the above statement about TC37 still true now we have removed Terminological Entry and Language Section? Language_Identification has been moved into Designation and Definition.

EDITOR'S NOTE #84. (Action required) Issue 94 calls for support of XML tags.

EDITOR'S NOTE #85. (Action required) *Designatable_Item* was introduced as a sub-type of *Identified_Item*, and a super-type of *Registered_Item*, to avoid dependencies among packages. However, since the Identification region and the Designation and Definition region have been positioned within a single package, the distinction no longer seems useful. Is there any objection to using *Identified_Item* wherever *Designatable_Item* is currently used, and to remove *Designatable_Item* from the model?

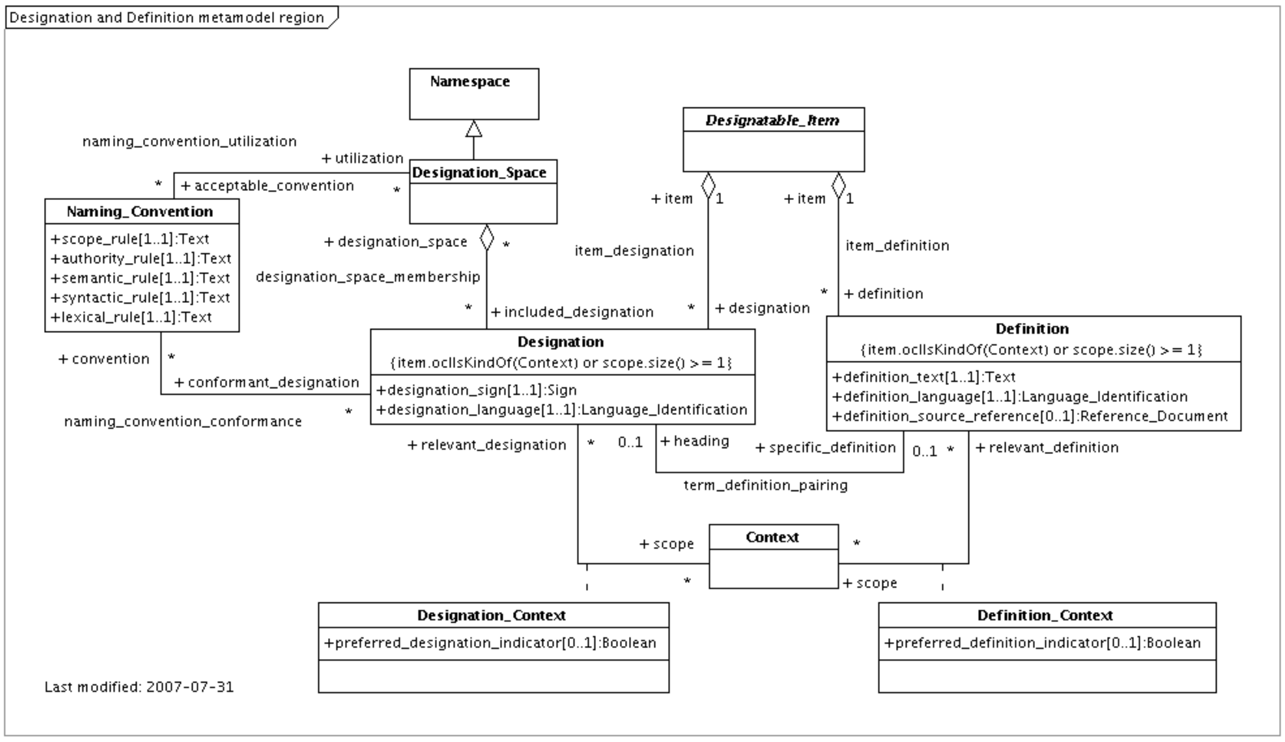

Designation and Definition metamodel region
Namespace
Designatable_Item
Designation_Space
Naming_Convention
+scope_rule[1..1]:Text
+authority_rule[1..1]:Text
+semantic_rule[1..1]:Text
+syntactic_rule[1..1]:Text
+lexical_rule[1..1]:Text
naming_convention_utilization
+ utilization
+ acceptable_convention
+ designation_space
designation_space_membership
+ included_designation
+ item
item_designation
+ designation
item_definition
+ definition
Designation
{item.oclIsKindOf(Context) or scope.size() >= 1}
+designation_sign[1..1]:Sign
+designation_language[1..1]:Language_Identification
Definition
{item.oclIsKindOf(Context) or scope.size() >= 1}
+definition_text[1..1]:Text
+definition_language[1..1]:Language_Identification
+definition_source_reference[0..1]:Reference_Document
+ convention
+ conformant_designation
naming_convention_conformance
+ relevant_designation
+ heading
+ specific_definition
+ relevant_definition
term_definition_pairing
Context
+ scope
Designation_Context
+preferred_designation_indicator[0..1]:Boolean
Definition_Context
+preferred_definition_indicator[0..1]:Boolean
Last modified: 2007-07-31

Figure 6 — Designation and Definition metamodel region

ISO/IEC 11179-4 provides rules and guidelines for the formulation of data definitions.

ISO/IEC 11179-5 provides naming and identification principles for Designatable_Items within a Context.

EDITOR'S NOTE #86. (Action required) It has been suggested that *term_definition_pairing* may be dependent on *Context*. This needs to be reflected in the model. The Editor suggests making *Term_Definition_Pairing* and association class, and associating it with *Context*. What should the cardinalities of the new association be?

EDITOR'S NOTE #87. (Action required) Should we be able to explicitly relate *Naming_Convention* and *Designation_Space* to *Context*?

EDITOR'S NOTE #88. (Action required) Designation is modelled as a Class with a containment relationship to Designation_Space and another containment relationship with Designatable_Item, while *identification* is modelled as an association between *Identified_Item* and *Scoped_Identifier*. Why the difference?

EDITOR'S NOTE #89. (Action required) In the above figure, the attribute designation_language of *Designation* is shown as mandatory, but in the text it is described as optional.

6.2.2 Classes in the Designation and Definition region

EDITOR'S NOTE #90. (Action required) This subclause starts by describing *Designatable_Item,* instead of listing the classes alphabetically. Does this make the subclause easier to understand? Would a different sequence be better?

6.2.2.1 Designatable_Item

Designatable_Item is the class of objects which can have designations and definitions. While it is not necessary for all *Designatable_Items* to have a designation and/or definition in a metadata registry, a metadata registry must be able to support the association of designations or definitions with *Designatable_Items* should they actually exist. A *Designatable_Item* may participate in the following associations: *item_designation* and *item_definition*

6.2.2.2 Context

A *Context* is a universe of discourse in which certain *Designations* or *Definitions* are used to designate or define a set of *Designatable_Items*. Each *Designatable_Item* may be designated and/or defined within zero, one or more *Contexts*.

EDITOR'S NOTE #91. (Informational) In Edition 2, Context was mandatory, but it was effectively playing the role of a Namespace. Now that we have explicitly added Namespaces, it seems reasonable that Context be optional.

A *Context* defines the scope within which the subject data has meaning. A *Context* may be a business domain, an information subject area, an information system, a database, file, data model, standard document, or any other environment determined by the owner of the registry. Each *Context* may itself be managed as a *Registered_Item*, and therefore also a *Designatable_Item* within the registry and be given a *designation* and/or a *definition*.

A *Context* may have zero or more *Definition_Context* associations with a *relevant definition* of type *Definition* where the *Context* provides the *scope* of the associated *Definition*.

A *Context* may have zero or more *Designation_Context* associations with a *relevant designation* of type *Designation* where the *Context* provides the *scope* of the associated *Designation*.

NOTE The *Context* within which a *Context* is named and defined will probably be the registry itself, but could be broader, and could simply be specified as being this International Standard, or it may be omitted. When omitted, the *Context* for a *Context* is assumed to be the registry in which the *Context* being designated is recorded.

6.2.2.3 Definition

The *Definition* class provides the *definition_text* for a *Designatable_Item* as it applies in zero, one or more *Contexts*. Each *Designatable_Item* may be associated with zero, one or more *Definitions*, each specified in a particular language. Where multiple *Definitions* are provided within the same *Context*, one of them may be specified as the *preferred definition*.

The *Definition* class records the binding of a pair of *definition_text* and its *definition_language* to a *Designatable_Item*. The *definition_text* is a statement (commonly in a natural language) which specifies the meaning of the *Designatable_Item*. It may additionally record a *definition_source_reference* for the *definition_text*.

A *Designatable_Item* may have zero or more *Definitions* with varying *definition_languages*, *definition_contexts*, and *term_definition_pairings*.

A *Definition* may participate in the associations: *item_definition*, *term_definition_pairing*, and *definition_context*.

The *Definition* class has the following attributes:

— exactly one *definition_text* attribute of datatype *Text* which contains the text which constitutes the definition.

— exactly one *language* attribute of dataype *Language Identifier*. The *language* attribute records the language in which the definition_text is written.

EDITOR'S NOTE #92. (Action required) The datatype of *definition_text* has been changed from String to Text by Issue 18. Since the Text datatype inherently supports multiple languages, do we still need the explicit *definition_language* attribute here?

— zero or one *definition_source_reference* attribute of datatype *Reference_Document*. The *definition_source_reference* attribute may be used to record the origin of the definition.

EDITOR'S NOTE #93. (Action required) The following table appears to simply duplicate the above text. Do we need both? We need a standard format for describing the attributes of all classes.

Attribute	**Occurrences**	**Datatype**
definition_text	One per *Definition*	Text
definition_language	One per *Definition*	*Language_Identification*
definition_source_reference	Zero or one per *Definition*	*Reference_Document*

6.2.2.4 Designation

The *Designation* class records the binding of a pair comprised of a *sign* and its *language* to a *Designatable_Item*. Each *Designation* is situated with respect to a *Context*, a *Naming_Convention*, a *Namespace*, and may be paired with a *Definition*. *Designatable_Items* may have many different (or identical) *signs* in various *languages*, *Contexts*, *Naming_Conventions*, and *Namespaces*.

NOTE In Edition 2, the term *name* was used for what is now called *sign*. This change in Edition 3 has been made to bring its terminology into conformity with ISO 1087 Part 1 (from ISO TC 37). The *language* of a *designation* is a written natural language, and the *sign* is a word or phrase in the language.

EDITOR'S NOTE #94. (Action required) Since a *Sign* may be an image (e.g. a traffic sign), it would seem that *designation_language* may not always be applicable, and should therefore be made optional, perhaps with an additional constraint that it be mandatory if the *Sign* is actually text. Alternatively, can we specify that a designation is either a Sign or Text, and handle the language within the capabilities of the Text datatype?

A *Designation* must participate in the association *item_designation*, and may participate in the associations: *term_definition_pairing*, *designation_context*, *naming_convention_conformance*, and *designation_space_membership*.

Where multiple *Designations* are provided within the same C*ontext*, one of them may be specified as the *preferred designation*.

The *Designation* class has two attributes:

— Exactly one *designation_sign* attribute, of type *Sign,* which is used to designate a *Designatable_Item*, e.g., a name of an object or concept.

— Zero or one *designation_language* attribute, of type *Language_Identification*, which is used to record the language or dialect in which the *designation_sign* (usually a name) is used. Usually the language will refer to a natural human language.

Attribute	**Occurrences**	**Datatype**
designation_sign	One per *Designation*	Sign
designation_language	Zero or one per *Designation*	*Language_Identification*

6.2.2.5 Designation_Space

Designation_Space is a *Namespace* that assigns *Signs* to *Designatable_Items*. A *Designation_Space* may have zero or more *designation_space_membership* associations with an *included_designation* of type *Designation* where the *Designation_Space* provides the *namespace* of the associated *Designation*.

One use of *Designation_Spaces* is to permit the *Designatable_Item* represented by a *designation_sign* to be uniquely determined for a sign within a particular *Designation_Space*.

A *Designation_Space* may participate in the associations: *naming_convention_utilization* and *designation_space_membership*.

If the *one_name_per_item_indicator* (in *Namespace)* is true for a *Designation_Space* (a.k.a. unique names), then each *Designatable_Item* within the *Designations* of the *Designation_Space* has exactly one *Designation* within this *Designation_Space*.

Unique names implies a functional mapping from *Designatable_Items* to *Signs*. In common parlance, no possibility of aliases exists. Thus two distinct signs (names) within a *Designation_Space* must refer to separate *Designatable_Items*.

If the *one_name_per_item_indicator* is false, then each *Designatable_Item* within the *Designations* of the *Designation_Space* may have more than one *Designation* within this *Designation_Space*.

If the *one_item_per_name_indicator* attribute is true (a.k.a. unambiguous names), then there exists at most one *Designatable_Item* associated with each *Designation* in the *Designation_Space*.

Unambiguous names implies a functional mapping from *designation_signs* to *Designatable_Items*, and from *identifiers* to *Identified_Items*.

6.2.2.6 Naming_Convention

EDITOR'S NOTE #95. (Action required) Although *Designations* can be associated with both Naming_Conventions and Contexts, there is currently no way to specify that a particular Naming_Convention applies to a Designation in a particular Context. If one were to associate Naming_Convention to a Context, this could imply one of two things depending on the cardinality of the association.
(1) A Context can have many Naming_Conventions
Question: How would one know if a particular name came from a particular naming convention?
(2) A Context can have only one Naming_Convention
This then requires that ALL names in this Context have this Naming_Convention. This then implies that:

1. ALL names in ALL languages use this Naming_Convention
   (i.e. French, English, Korean, etc.)
2. ALL names in a language use this Naming_Convention
   (i.e. preferred term and non-preferred terms [synonyms])

Question: Doesn’t this seem to be overly restrictive and unrealistic?

EDITOR'S NOTE #96. Issue 118 proposes making *Naming_Convention* as an Administered_Item. This CD has removed explicit sub-classing of any item type.

The *Naming_Convention* class provides the specification by which the *sign* (name) of a *Designation* is developed. Naming_Conventions may range in complexity from very simple to very complex. The semantic, syntactic, and lexical_rules may have their own complexity.

A *Naming_Convention* is a sub-type of *Identified_Item*, and *Designatable_Item*, inheriting their attributes and relationships, which allows it to be identified, and optionally named, defined classified.

The *Naming_Convention* class records a set of rules for constructing *signs* (names) to designate *Designatable_Items*.

The *Naming_Convention* class may participate in the associations: *naming_convention_utilization* and *naming_convention_conformance*.

The Naming_Convention class has the following attributes:

Attribute	**Occurrences**	**Datatype**
scope_rule	One per *Naming_Convention*	Text
authority_rule	One per *Naming_Convention*	Text
semantic_rule	One per *Naming_Convention*	Text
syntactic_rule	One per *Naming_Convention*	Text
lexical_rule	One per *Naming_Convention*	Text

The *scope_rule* attribute records the rule by which the scope of the name created by the *Naming_Convention* is determined. The scope of a naming convention specifies the range within which the Naming_Convention is in effect. In terms of the metadata registry, the scope of a naming convention may be as broad or narrow as the Registration_Authority, or other authority, determines is appropriate. The scope should document whether the naming convention is descriptive or prescriptive.

The *authority_rule* attribute identifies the authority that assigns names, specifies and/or enforces the naming convention. Examples of authorities include information technology standards committees or nomenclature standardization bodies (e.g., in biology).

The *semantic_rule* attribute records the rules for specifying the meanings of portions of a composite name. These rules record whether or not names convey meaning, and if so, how.

The *syntactic_rule* attribute specifies the syntax (arrangement of parts) within a name. The arrangement may be specified as relative or absolute, or some combination of the two. Relative arrangement specifies parts in terms of other parts, e.g., a rule within a convention might require that a qualifier term must always appear before the part being qualified appears. Absolute arrangement specifies a fixed occurrence of the part, e.g., a rule might require that the property term is always the last part of a name. The syntactic_principle might also specify the syntactic forms of the name (noun phrase or verb phrase) and the parts of speech used to construct a name.

The *lexical_rule* attribute specifies a set of rules for the lexical construction of a name according to the Naming_Convention. Lexical issues concern the appearance of names: preferred and non-preferred terms, synonyms, abbreviations, part length, spelling, permissible character set, case sensitivity, etc. The result of applying lexical_rules should be that all names governed by a specific naming convention have a consistent appearance. An example lexical principle might be the specification of the use of camelCase capitalization of words in a phrase which are concatenated together.

NOTE Part 5 of this standard has a more elaborate discussion of naming conventions.

6.2.2.7 Namespace

Namespace is described in 6.1.2.3. The following additional statements apply to this region.

If the *mandatory_naming_convention_indicator* is true:

(a) there must be exactly one acceptable *Naming_Convention* associated with this *Namespace* in the *naming_convention_utilization* association, and

(b) every *included_designation Designation* must have a *naming_convention_conformance* association with the same *Naming_Convention* used in part (a) above.

If *mandatory_naming_convention_indicator* is false, it is possible for a *Designation_Space* to be associated with zero or more acceptable conventions and/or a *included_designation Designation* to conform to more than one convention.

6.2.3 Association Classes in the Designation and Definition Region

6.2.3.1 Definition_Context

The Definition_Context association class records the Context in which a Definition occurs The definition_context. association has two roles: relevant_definition (verb form: includes_relevant_definition) and scope (verb form: occurs_in_scope). The relevant_definition (includes) role refers to a Definition class. The scope (verb form: occurs_in_scope) role refers to a Context class. A scope (Context) may include zero or more relevant_definitions (Definitions). A relevant_definition (Definition) may occur within of zero or more scopes (Contexts).

The definition_context association class has one attribute:

Attribute	**Occurrences**	**Datatype**
preferred_definition_indicator	Zero or one per *Definition_Context*	Boolean

EDITOR'S NOTE #97. (Action required) It has been suggested that instead of a Boolean indicator, we use a set of acceptability ratings as defined by ISO 10241. I.e. 'preferred', 'accepted', 'deprecated', 'obsolete', 'superseded'. If this proposal is accepted, a better name might be: *definition_acceptability_rating*

If the preferred_definition_indicator attribute is true then the associated Definition is preferred to all others in the associated Context The preferred_definition attribute is optonal, it may occur zero or one time per definition_context.

6.2.3.2 Designation_Context

The designation_context association (class) records the Context in which a Designation occursThe designation_context association has two roles: relevant_designation (verb form: includes_relevant_designation) and scope (verb form: occurs_in_scope). The relevant_designation (verb form: includes_relevant_designation) role refers to a Designation class. The scope (verb form: occurs_in_scope) role refers to a Context class. A scope (Context) may include zero or more relevant_designations (Designations). A relevant_designation (Designation) may have zero or more scopes (Contexts).

The designation_context association class has one attribute:

Attribute	**Occurrences**	**Datatype**
preferred designation	Zero or one per *Designation_Context*	Boolean

EDITOR'S NOTE #98. (Action required) It has been suggested that instead of a Boolean indicator, we use a set of acceptability ratings as defined by ISO 10241. I.e. 'preferred', 'accepted', 'deprecated', 'obsolete', 'superseded'. If this proposal is accepted, a better name might be: *designation_acceptability_rating*

If the preferred_designation attribute is true then the associated Designation is preferred overall others in the associated Context The preferred_designation attribute is optional, it may occur zero or one times per designation_context.

6.2.4 Associations in the Designation and Definition Region

6.2.4.1 designation_space_membership

Namespace membership is an association between a *Designation_Space* and a *Designation* that indicates that the *Designation* is bound to that *Designation_Space*.

The namespace_membership association is used to record the namespaces in which a designation is valid. The designation_namespace association has two roles: namespace (verb form: occurs_in_namespace) and binding (verb form: binds_to). The namespace (verb form: occurs_in_namespace) role refers to a Namespace. The binding (verb form: binds_to) role refers to a Designation. Each namespace (Namespace) may have zero or more bindings (Designations). Each binding (Designation) may occur_in zero or more namespaces (Namespaces).

The namespace_membership association is a weak containment association, thus Designations are contained in Namespace. The weakness of the containment implies that deletion of the containing Namespace does not cause cascading deletions of the contained Designations.

6.2.4.2 item_definition

The item_definition association is used to record all of the definitions for a specific Designatable_ItemThe item_definition association has two roles: item (verb form: used_for_item) and definition (verb form: has_definition). The item (verb form: used_for_item) role references the Designatable_Item class. The definition role (verb form: has_definition) references the Definition Class. Each definition (Definition) shall have exactly one item (Deisgnatable Item). Each item (Designatable_Item) may have zero or more definitions (Definitions).

The item_definition association is a strong containment relation. Hence, a definition is used for exactly one designatable item. Deletion of the designatable item implies a cascading deletion of the associated definitions. Note that definitions may not be not be reused across multiple designatable items.

6.2.4.3 item_designation

Item_designation is the binary association which records all of the Designations (sign + language pairs) of a Designatable_Item. The item_designation association has two roles: item (verb form: used_for_item) and designation (verb form: has_designation). The item (verb form: used_for_item) role references the Designatable_Item class. The designation (verb form: has_designation) role references the Designation class. Each designation (Designation) shall be used for exactly one Designatable_Item. An item(Designatable_Item) may have zero or more designations (Designations).

The item_designation association is a strong containment relation. Hence, a Designation is used for exactly one Designatable_Item. Deletion of the designatable item implies a cascading deletion of the associated Designations. Note that Designations may not be not be reused across multiple designatable items.

6.2.4.4 naming_convention_conformance

The naming_convention_conformance association records which Naming_Conventions (if any) a particular Designation conforms toThe conformant_designation association has two roles: convention (verb form: conforms_to) and conformant_designation (verb form: has_conformant_designation). The convention (verb form: conforms_to) role refers to a Naming_Convention class. The conformant_designation (verb form: has_conformant_designation) role refers to a Designation class. Each conformant_designation (Designation) may have zero or more conventions (Naming_Convention). Each convention (Naming_Convention) may have zero or more conformant_designations (Designations).

6.2.4.5 naming_convention_utilization

The naming_convention_utilization association records the Naming_Convention utilized by a Namespace (if any The naming_convention_utilization association has two roles: utilization (verb form: utilized_by) and acceptable_convention (verb form: accepted_convention). The utilization (verb form: utilized_by) role refers to a Namespace class. The utilization (verb form: utilizing) role refers to a Naming_Convention class. Each utilization (Namespace) may utilize zero or one accepted_conventions (Naming_Conventions). Each accepted_convention (Naming_Convention) has zero or more utilizations (namespaces).

6.2.4.6 term_definition_pairing

The term_definition_pairing association is used to bind together Designations (here referred to as terms) to their associated Definitions The term_definition_pairing association has two roles: heading (verb form: used_for_heading) and specific_definition (verb form: defined_as). The heading (verb form: used_for_heading) role refers to a Designation class. The specific_definition (verb form: defined_as) role refers to a Definition class. Each specific_definition (Definition) may have zero or one headings (Designations). Each heading (Designation) may have zero or one specific_definitions (Definitions)

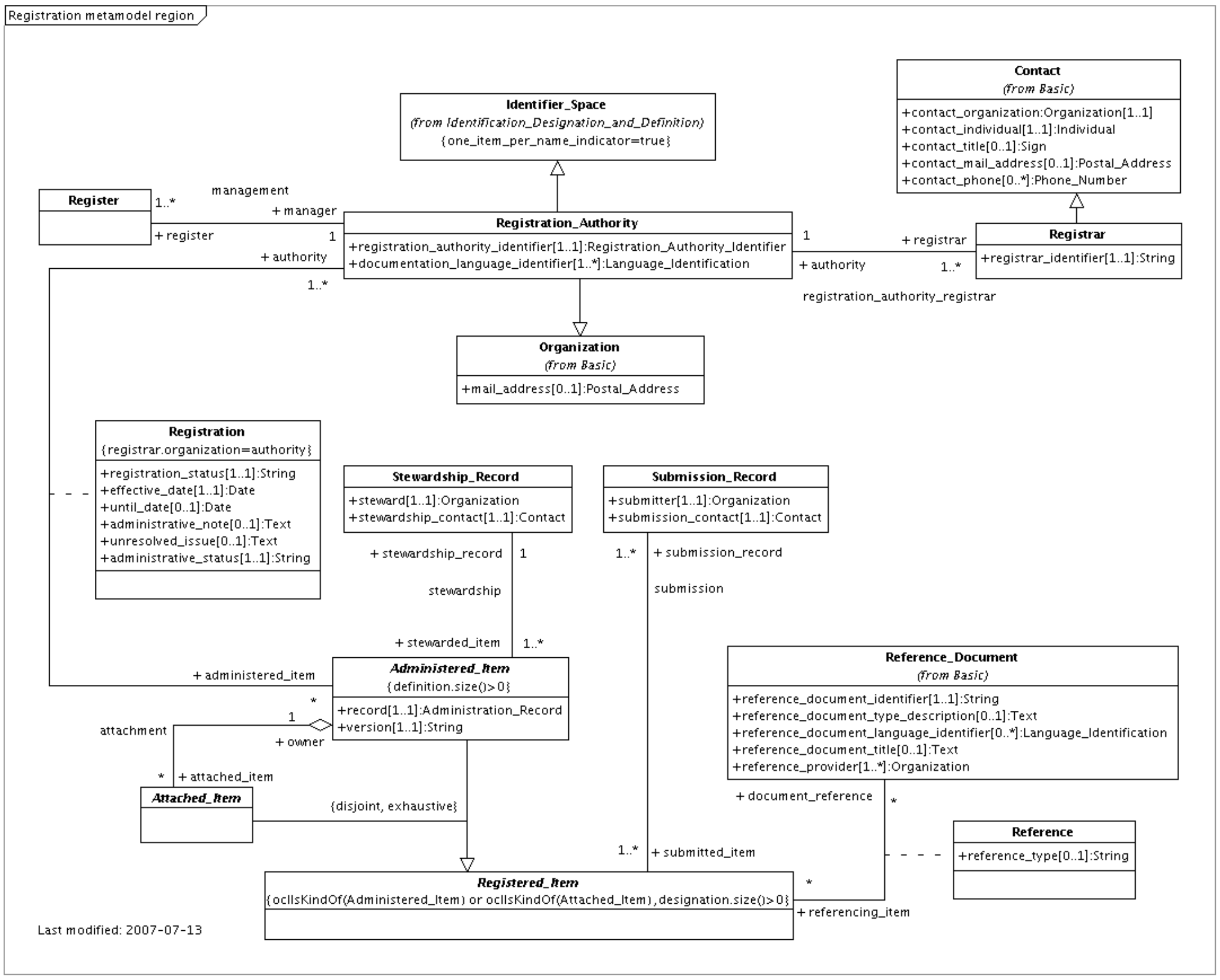
7 Registration Package

7.1 Registration region

7.1.1 Overview

The Registration region supports the registration of items in a registry. The registration of Administered_Items is described in ISO/IEC 11179-6.

Figure 7 shows the classes, relationships, attributes and composite attributes that support Administration and Identification. Figure 8 shows the composite datatypes used on composite attributes.



Registration metamodel region
Identifier_Space
(from Identification_Designation_and_Definition)
{one_item_per_name_indicator=true}
Contact
(from Basic)
+contact_organization:Organization[1..1]
+contact_individual[1..1]:Individual
+contact_title[0..1]:Sign
+contact_mail_address[0..1]:Postal_Address
+contact_phone[0..*]:Phone_Number
Register
management
+ manager
1..*
+ register
1
+ authority
1..*
Registration_Authority
+registration_authority_identifier[1..1]:Registration_Authority_Identifier
+documentation_language_identifier[1..*]:Language_Identification
1
+ registrar
+ authority
1..*
Registrar
+registrar_identifier[1..1]:String
registration_authority_registrar
Organization
(from Basic)
+mail_address[0..1]:Postal_Address
Registration
{registrar.organization=authority}
+registration_status[1..1]:String
+effective_date[1..1]:Date
+until_date[0..1]:Date
+administrative_note[0..1]:Text
+unresolved_issue[0..1]:Text
+administrative_status[1..1]:String
Stewardship_Record
+steward[1..1]:Organization
+stewardship_contact[1..1]:Contact
Submission_Record
+submitter[1..1]:Organization
+submission_contact[1..1]:Contact
+ stewardship_record
1
stewardship
+ stewarded_item
1..*
1..*
+ submission_record
submission
Reference_Document
(from Basic)
+reference_document_identifier[1..1]:String
+reference_document_type_description[0..1]:Text
+reference_document_language_identifier[0..*]:Language_Identification
+reference_document_title[0..1]:Text
+reference_provider[1..*]:Organization
+ administered_item
Administered_Item
{definition.size()>0}
+record[1..1]:Administration_Record
+version[1..1]:String
*
1
attachment
+ owner
*
+ attached_item
Attached_Item
{disjoint, exhaustive}
+ document_reference
*
Reference
+reference_type[0..1]:String
1..*
+ submitted_item
Registered_Item
{oclIsKindOf(Administered_Item) or oclIsKindOf(Attached_Item),designation.size()>0}
*
+ referencing_item
Last modified: 2007-07-13

Figure 7 — Registration metamodel region

EDITOR'S NOTE #99. (Action required) In Figure 6, the Registration association is currently with Administered_Item, rather than Registered_Item. The names would suggest that Registration should associate with Registered_Item, and a separate Administration association should associate with Administered_Item (which seems to be what the Stewardship association class is about). We need a clearer definition of the requirements we are trying to satisfy before we can determine how best to model this.

EDITOR'S NOTE #100. (Action Required) Since Administration_Record is used only by Administered_Item.record, why have we bothered to create a separate record? Why not just include the attributes in Administered_Item, or as an Administration association class, like Registration.

EDITOR'S NOTE #101. (Action required) The *stewardship* association requires at least one Administered_Item, and the *submission* association requires at least one Registered_Item, which means that the Stewardship_Record and the Submission_Record cannot be set up in advance of an associated item being registered. Is this what we want? The cardinalities of the attributes within the Stewardship_Record and Submission_Record need to be specified.

EDITOR'S NOTE #102. (Action required) 'version' has been left in Administered_Item, and 'registration_authority_identifier is related through the administration association with Administered_Item. Version control was inadequate in Edition 2, and is still inadequate in this edition. Since each *item_identifier* has to be unique on its own, how do we indicate that different versions of items are version of the same item? We need a way to relate different versions of an item, or group of items.

EDITOR'S NOTE #103. (Action Required) Making *Registration_Authority* a subclass of *Identifier_Space* restricts us to a single *Identifier_Space* per *Registration_Authority*. Is there a reason to do this? Simply associating *Registration_Authority* with *Identifier_Space* would admit the possibility that a *Register* could use more than one *Identifier_Space*. For example, when importing metadata items from another *Registry*, the *Identifier_Space* and *Scope_Identifiers* from that other *Registry* could be preserved.

7.1.2 Classes in the Registration region

7.1.2.1 Registered_Item

Registered_Item is a *Designatable_Item* that is designated and managed in a metadata registry.

As a *Designatable_Item*, a *Registered_Item* has at least one *item identifier* and may also have *Designations* and *Definitions*. In addition, a *Registered_Item* is required to have at least one *Designation*.

EDITOR'S NOTE #104. (Action required) *Registered_Item* is no longer sub-typed from *Classifiable_Item*. If this change is confirmed, the following sentence should be removed.

As a *Classifiable_Item*, a *Registered_Item* may be associated with zero or more *Hierarchy_Nodes* in one or more *Classification_Schemes*.

A *Registered_Item* must either be an *Administered* or an *Attached_Item* but not both.

A *Registered_Item* must have a *submission* association with one or more *submitter Organizations*. The *submitter Organization* is the organization that has submitted the *Registered_Item* for addition, change or cancellation/withdrawal within a metadata registry.

All *metadata items* that are registered in a *metadata registry* are implicitly subclasses of *Registered_Item*. See Figure C.1 in Annex C for the types of *Registered_Items* specified in this standard. Additional types of *Registered_Item* may be defined as extensions to this standard. The metamodel may be extended by adding other sub-types of *Registered_Item*.

EDITOR'S NOTE #105. (Action required) It has been proposed by the XMDR project that the explicit sub-typing of Registered_Item be removed, and replaced by one or more conformance profiles. The explicit sub-typing has been removed, but no replacement has been provided. A detailed proposal is required.

7.1.2.2 Administered_Item

An *Administered_Item* is a *Registered_Item* for which administrative information is recorded in an *Administration_Record*. *Administered_Item* is a subclass of *Registered_Item*, that is administered by a *Registration_Authority*.

EDITOR'S NOTE #106. (Action required) In Figure 6, the *administration* association has become the *Registration* association class, but this renaming does not make sense.

Every *Administered_Item* must have a *Registration* association with one or more *Registration Authorities*. The *Registration_Authority* is an *Organization* that is responsible for maintaining a *Register*.

Every *Administered_Item* shall have exactly one *Stewardship* association with a *steward* of type *Organization* and a *stewardship_contact* of type *Contact,* which is responsible for maintaining the item's administration record.

An *Administered_Item* **may** have an *attachment* association with zero or more *Attached_Items*. The set of *Attached_Items* that participate in this association are administered collectively under a single administration record – they all share the same *stewardship*, *registration*, *record* and *version*. Note that *Registered_Items* may share the same Administration_Record and still have different submitting organizations.

Every *Administered_Item* shall have exactly one *administration record* of type *Administration_Record*. Every *Administered_Item* shall also have exactly one *version* of type *String*. A *version* is the unique version identifier of the *Administered_Item*.

An *Administered_Item* may be *described by* zero or more *Reference_Documents* as represented by the relationship *reference* in **Error! Reference source not found.**.

Attribute name	**Occurrences**	**Datatype**
administration record	One per *Administered_Item*.	*Administration_Record*
version	One per *Administered_Item*.	*String*

7.1.2.3 Attached_Item

An *Attached_Item* is *Registered_Item* for which administrative information is recorded in an *Administration_Record* of another *Registered_Item* (an *Administered_Item)*. Every *Attached_Item* has an *attachment* association with an *owner Administered_Item*, which supplies the *Administration_Record*,

NOTE *Attached_Items* provide the means to administer a package of items as a collection.

7.1.2.4 Organization

Organization is a unique framework of authority within which *individuals* act, or are designated to act, towards some purpose.

EDITOR'S NOTE #107. (Action required) Using the sentences from the model in the text makes for very tortuous reading. Is this really the best approach?

An *Organization* can play one or more roles with respect to a Metadata Registry. All *Registration Authorities* are *Organizations*, but not all *Organizations* are necessarily *Registration Authorities*. An *Organization* may be a *submitter* within a *Submission_Record,* with zero or more *submitted item Registered_Items*, where the *Organization* acts as the *submitter.* An *Organization* may also have a *stewardship* association with zero or more *stewarded item Administered_Items* where the *Organization* acts as the *steward* for the item.

Organization is further described in 5.1.9.

7.1.2.5 Registrar

Registrar is a *Contact* that is a representative of the *Registration_Authority*. A *Registration_Authority* is represented by one or more *Registrars*. A *Registrar* has a *registration_authority_registrar* association with exactly one *authority Registration_Authority.* A *Registrar* has one mandatory attribute, *registrar_identifier,* of type *String* that identifies a *Registrar*.

EDITOR'S NOTE #108. (Action required) Since an Organization already has Representatives, and a Registration_Authority is an organization, why do we need to show Registrar separately? What is the registrar_identifier used for?

Registrars are the persons who perform the administrative steps to register *Administered_Items* in a *Metadata Registry*.

Attribute	**Occurrences**	**Datatype**
registrar_identifier	One for each *registrar* in a *registration authority*	String

Registrar is a subclass of *Contact*. Contact is described in 5.1.3.

7.1.2.6 Registration_Authority

A *Registration_Authority* is any *Organization* authorized to register *metadata*. A *Registration_Authority* is a subtype of *Organization* and inherits all of its attributes and relationships. An *Administered_Item* has a *Registration_Authority* that is its *authority*, shown by the relationship *Registration* in Figure 4. A *Registration_Authority* may register many *Administered_Items*.

Attribute	**Occurrences**	**Datatype**
registration_authority_identifier	One per *registration authority*	*Registration_Authority_Identifier*
documentation_language	From one to many per *registration authority*	*Language_Identification*

EDITOR'S NOTE #109. *(Informational) documentation_language_identifier* has been renamed to *documentation language*, with no explanation. The classword *identifier* was there in edition 2 to follow the naming guidelines of part 5.

7.1.2.7 Register

EDITOR'S NOTE #110. (Action required) Why is *Register* needed in the model?

A *Register* is a data store where *Registered_Items* are recorded and managed. The *Register* is managed by a *Registration_Authority*.

7.1.2.8 Stewardship_Record

An *Organization* shall be identified as the steward responsible for *administering* each *Administered_Item,* as represented by the s*tewardship* association with *Stewardship_Record* in Figure 5. The Stewardship_Record identifies both the *Organization* that is the *steward*, and a *stewardship_contact* at the *Organization*, for one or more *Administered_Items.*

A **Stewardship** is the relationship of an *Administered_Item*, a *Contact*, and an *Organization* involved in the stewardship of the metadata. A *stewardship_contact* is the contact information associated with a Stewardship. Every **Stewardship** shall have exactly one *stewardship_contact* of type **Contact.**

Attribute	**Occurrences**	**Datatype**
steward	One per *Stewardship_Record*	*Organization*
stewardship_contact	One per *Stewardship_Record*	*Contact*

7.1.2.9 Submission_Record

For each *Administered_Item,* an *Organization* shall be identified as the submitter as represented by the relationship *Submission* in Figure 5. This relationship identifies a *submission_contact* for the *Administered_Item*

Attribute	**Occurrences**	**Datatype**
submitter	One per *Submission_Record*	*Organization*
submission_contact	One per *Submission_Record*	*Contact*

7.1.3 Association Classes in the Registration region

7.1.3.1 Registration

Registration is an association between a *Registration_Authority* and an *Administered_Item* where the *Registration_Authority* manages the *Administered_Item* in a metadata *Register*.

Registration is also a class, and has the following attributes:

Attribute	**Occurrences**	**Datatype**
administrative_status	One per *Registration* association	String
registration_status	One per *Registration* association	String
effective_date	One per *Registration* association	Date
until_date	Zero or one per *Registration* association	Date
administrative_note	Zero or one per *Registration* association	Text
unresolved_issue	Zero or one per *Registration* association	Text

An *administrative_status* of type *String* which designates the status of an *Administered_Item* in the administrative process of a *Registration_Authority*. NOTE: The values and associated meanings of “administrative_status” are determined by each Registration_Authority. C.f. “registration_status”.

A *registration_status* of type *String* which designates the status of an *Administered_Item* in the registration live-cycle.

An *effective_date* is a *Date* that identifies the date and time that an *Administered_Item* became or will become available to registry users.

Optionally, an *until_date* is a *Date* that identifies the date and time that an *Administered_Item* is or will no longer be effective in the registry.

Optionally, an *administrative_note* is a *Text* that contains general comments and instructions about the *Administered_Item*.

Optionally, an *unresolved_issue* is a *Text* that documents any problem that remains unresolved regarding proper documentation of the *Administered_Item*.

7.1.3.2 Reference

A *Reference* is the association between a *Reference_Document* and an *Administered_Item*.

A *Reference* is also a class, and may have zero or one *reference_types* of type *String*.

EDITOR'S NOTE #111. (Action required) What are some examples of reference_type? We should provide candidate meanings. Should the datatype be Text instead of String, to allow the type to be expressed in multiple languages?

Attribute	Occurrences	Datatype
reference_type	zero or one per Reference	String

7.1.4 Associations in the Registration region

7.1.4.1 attachment

attachment is a subclass of *ownership*. Every instance of *attachment* is also an instance of *ownership*.

Attachment is an association between an *Administered_Item* and an *Attached_Item* that indicates that the *Attached_Item* shares all of the administration characteristics of the *Administered_Item*. *Attachment* allows collections of *Registered_Items* to be administered collectively as a block.

Attachment has two roles: *owner* (verb form: *has owner*) and *attached item* (verb form: *attached to*). The *owner* role references an *Administered_Item* and the *attached item* role references an *Attached_Item*. Every *Attached_Item* shall have an *attachment* association with exactly one *owner Administered_Item*. An *Administered_Item* may have an *attachment* association with zero or more *Attached_Items*.

7.1.4.2 management

EDITOR'S NOTE #112. (Action required) This association name is too generic. It needs to be qualified in some way. E.g. register management

Management is an association between a *Registration_Authority* and a *Register* that identifies the *Registration_Authority* that is responsible for managing and maintaining the *Register*.

Every *Registration_Authority* shall have one or more *management* associations with a *managed Register*.

Every *Register* shall have exactly one *management* association with a *manager Registration_Authority*.

7.1.4.3 registration_authority_registrar

registration_authority_registrar is an association between a *Registration_Authority* and a *Registrar* that indicates that the *Registrar* is a representative of the *Registration_Authority*.

Every *Registration_Authority* shall have one or more *registration_authority_registrar* associations with a *Registrar* where the *Registration_Authority* provides the *authority* of the associated *Registrar.*

7.1.4.4 stewardship

stewardship is the association of an *Administered_Item* to a *Stewardship_Record*, which records the *steward Organization* and *stewardship_contact* involved in the stewardship of the *Administered_Item*.

7.1.4.5 submission

submission is the association of a *Registered_Item* with a *Submission_Record*, which records the *submitter Organization* involved in the *submission* of the *Registered_Item.*

7.2 Administration_Record region

7.2.1 Overview

EDITOR'S NOTE #113. (Action required) In response to Issue 176, it was agreed to move 'administrative_status' from Administration_Record to 'Registration', to allow different *Registration Authorities* to record different *administrative_statuses* for the same *Administered_Item*. However, it has also be suggested that (1) we should treat administrative (event) data orthogonally to registration (quality) data (e.g. as a separate association class), and (2) that we should have the same flexibility for all attributes in *Administration_Record*.

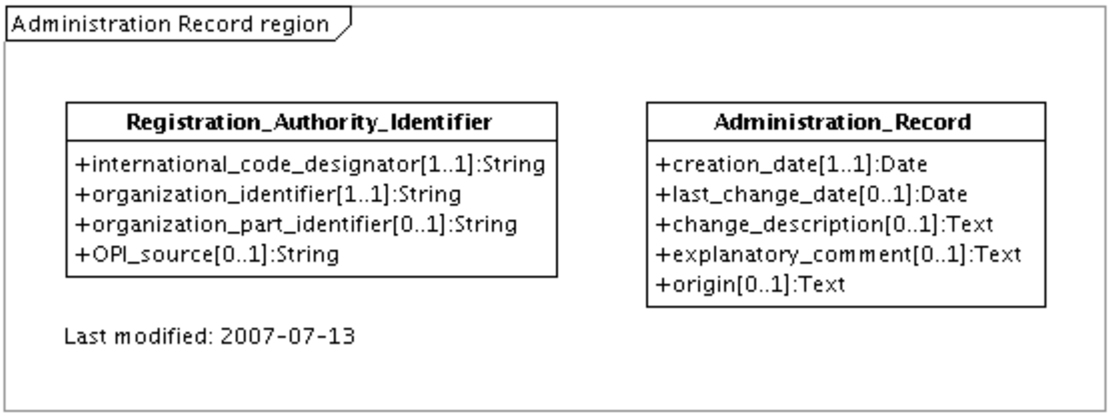

Administration Record region
Registration_Authority_Identifier
+international_code_designator[1..1]:String
+organization_identifier[1..1]:String
+organization_part_identifier[0..1]:String
+OPI_source[0..1]:String
Administration_Record
+creation_date[1..1]:Date
+last_change_date[0..1]:Date
+change_description[0..1]:Text
+explanatory_comment[0..1]:Text
+origin[0..1]:Text
Last modified: 2007-07-13

Figure 8 — Administration_Record region

7.2.2 Classes in the Administration_Record region

7.2.2.1 Administration_Record

An *Administration_Record* is a collection of administrative information for an *Administered_Item*.

Whenever an *Administered_Item* is modified, the *version* identifier should be updated and the change should be reflected in the corresponding *Administration_Record*. An *Administration_Record* contains:

— Exactly one *creation_date* of type *Date* that identifies the date and time that the *Administered_Item* was created.

— Zero or one *last_change_date* of type *Date* that specifies the date and time that the administered item was last changed.

— Zero or one *change descriptions* of type *Text* that describes what has changed in the *Administered_Item* since the prior version.

— Zero or one *explanatory_comment* of type *Text* that contains descriptive comments about the *Administered_Item*.

— Zero or one *origin* of type *Text* that describes the source (document, project, discipline or model)

EDITOR'S NOTE #114. (Action required) The above text has been added to describe the attributes of *Administration_Record*, which for other classes are simply listed in a table, like that below. We should be consistent in the way we describe the classes. Should we use text only, table only or both?

Attribute	Occurrences	Datatype
creation_date	One per *Administration record*	Date
last_change_date	Zero or one per *Administration record*	Date
change_description	Zero or one per *Administration record* conditional on presence of last_change_date	Text
explanatory_comment	Zero or one per *Administration record*	Text

Attribute	**Occurrences**	**Datatype**
origin	Zero or one per *Administration record*	Text

7.2.2.2 Registration_Authority_Identifier

The composite datatype *Registration_Authority_Identifier* is used to uniquely identify a *Registration_Authority*. The sources of values for each part of the identifier are specified in ISO/IEC 11179-6.

Attribute	**Occurrences**	**Datatype**
international_code_designator	One per *registration authority identifier*	String
organization_identifier	One per *registration authority identifier*	String
organization_part_identifier (OPI)	One per *registration authority identifier*	String
OPI_source	One per *registration authority identifier*	String

8 Relations Package

8.1 Relations Region

8.1.1 Overview

EDITOR'S NOTE #115. (Action required) This region has been extracted from the Concept_System package in order to eliminate dependencies of the Data Description package on the Concept_System package. The text needs to be reviewed and revised if necessary.

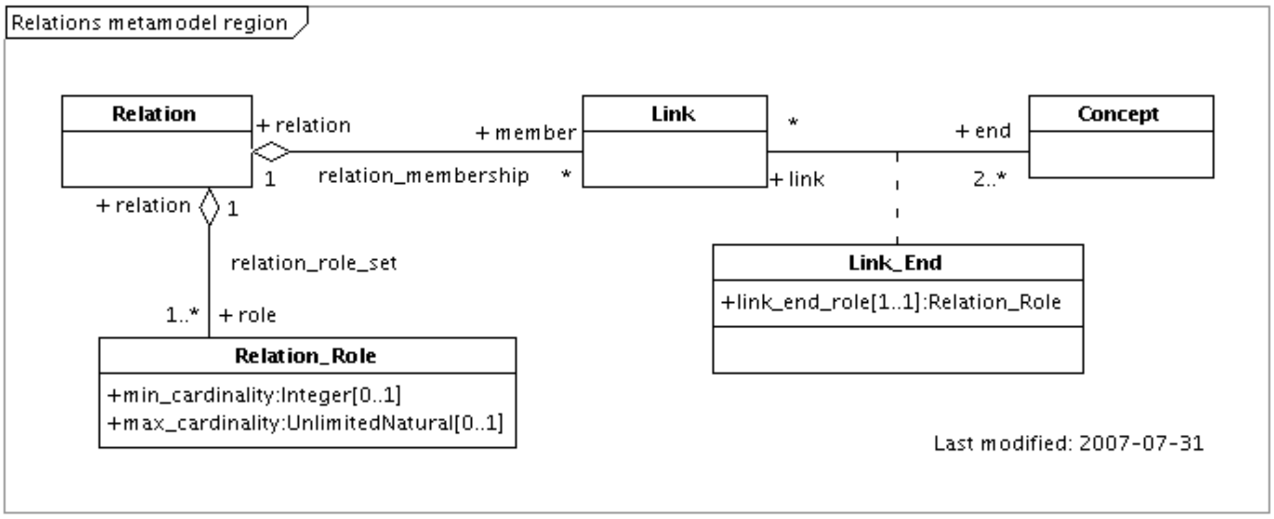

Relations metamodel region
Relation
+relation
+member
Link
*
+end
Concept
1
relation_membership
*
+link
2..*
+relation
1
relation_role_set
Link_End
+link_end_role[1..1]:Relation_Role
1..*
+role
Relation_Role
+min_cardinality:Integer[0..1]
+max_cardinality:UnlimitedNatural[0..1]
Last modified: 2007-07-31

Figure 9 — Relations region

8.1.2 Classes in the Relations Region

8.1.2.1 Concept

The *Concept* class models a unit of knowledge. *Concepts* are abstract, independent of representation.

A *Concept* may participate in the concept_membership association and/or the link_end association.

Concept may be both a *Registered_Item* and a *Hierarchy_Node* that models a unit of knowledge created by a unique combination of characteristics. *Concept* represents concepts which are recorded as *Registered_Items* within a registry. *Registered_Items* have detailed provenance tracking information recorded about them as they are installed or modified.

Data_Element_Concept, *Characteristic* and *Object_Class* are all subclasses of *Concept*.

The use of *Concept* in *Classification_Schemes*, *Concept_Systems* and *Ontologies* is described in clause 10.

8.1.2.2 Relation

EDITOR'S NOTE #116. (Action required) Special terms such as 'n-ary relation', 'n-tuple' and 'set' need to be defined, if we are to use them. Can we explain this concept without them?

EDITOR'S NOTE #117. Issue 140 proposes making *Relationship Type* (now called *Relation*) an Administered_Item. This CD has removed explicit sub-classing of any item type.

An *n*-ary relation on sets $A_1, ..., A_n$ is a set of ordered *n*-tuples $<a_1, ..., a_n>$ where a_i is an element of A_i for all *i*, *i between 1 and n* . Thus an *n*-ary relation on sets $A_1, ..., A_n$ is a subset of Cartesian product $A_1 \times ... \times A_n$. Membership of an n-tuple in the relation is specified by means of a predicate which must be true for the n-tuple to be a member of the corresponding relation. In our metamodel, relations are defined over sets of concepts.

Note: In this metamodel we actually use unordered n-tuples with named Relation_Roles rather than positional elements of the n-tuple.

A Relation class may participate in the following associations: relation_link, relation_role, and relation_membership.

A Relation class is a superclass of the Binary_Relation class.

Relation is further described in **Error! Reference source not found.**.

8.1.2.3 Relation_Role

The *Relation_Role* class models the distinct arguments which comprise a *Relation*. In relational DB terms, the *Relation_Role* represents a column in a relational table (for an asymmetric relation). Relation_Roles permit position independent naming of the arguments (columns) of a relation. This is similar to the distinction between positional arguments to procedures and named arguments of procedures in programming languages.

For Symmetric (Binary) Relations we reuse Relation_Roles to indicate multiple arguments (Link_Ends) since the arguments (Link_Ends) are to be treated identically.

The Relation_Role class has two attributes: min_cardinality and max_cardinality. Each of the cardinalities is constrained to the union type of non-negative integers or infinity (usually writeen as many), i.e., neither cardinalities may be negative. Each of these attributes is optional, i.e.,. a Relation Class may have zero or one min_cardinalities and zero or one max_cardinalities. It must be case the that max_cardinality is greater than or equal to min_cardinality.

The max_cardinality of a role (Relation_Role) is defined to be the maximum number of n-tuples (Links) in a Relation of arity n for for which all of the other n-1 roles have been given fixed values (i.e., the maximum taken with respect to all of the n-tuples of the projection of the Relation over the n-1 other roles). In our metamodel the values of Relation_Roles are Link_Ends (Concepts). Because Relations are sets of n-tuples, this definition is equivalent to the maximum number of values the role may take on given that all of the other n-1 roles have been given specific values (here Link_Ends of type Concept).

The min_cardinality of a role (Relation_Role) is defined somewhat similarly, as the minimum number of n-tuples (Links) in a relation of arity n for which all of the other n-1 roles have given fixed values (here again Link_Ends of type Concept). However, here we must take the minimum over all possible assignments of the values of the other n-1 roles. Because our metamodel lacks both explicit domain or inclusion dependency constraints (foreign key constraints) on roles the possible domains of the values for other n-1 roles are not well specified (they could be possible Concept).

For example if we have the husband_wife Relation, with min_cardinality of 1 for the wife role (i.e., all men have at least one wife), then we would need have an n-tuple (Link) for all men, but we have no way of specifying that the husband role has the domain of men. Thus we can not properly specify the definition of min_cardinality.

For example, if a role is a key of a relation its min_cardinality and max_cardinality will both be one., because there is a functional dependency from the key role to all of the other roles and because keys are constained to be non-null. If the relation role was wife and one lived in a society in which polygamy was banned then the max_cardinality of wife would be one; in a polygamous society the max_cardinality of wife would perhaps be four (or more).

Attribute	**Allowed Occurrences**	**Datatype**
min_cardinality	Zero or one per *Relation_Role*	*Non-negative integer or infinity (many)*
max_cardinality	Zero or one per *Relation_Role*	*Non-negative integer or infinity (many)*

8.1.2.4 Link

EDITOR'S NOTE #118. (Action required) It has been proposed that Link needs to be renamed, but there are problems with the names proposed to date:
Alternative 1: Relationship - there are two concerns with this alternative:
(a) it is too similar to Relation, and could be confused with it
(b) *relationship* is used in its UML sense within the definitions of *association* and *generalization* in the description of the metamodel.
Alternative 2: Relation instance – This term could lead to confusion because the instance (or member) is being stored at the same meta level as the Relation of which it is an instance. This is contrary to normal practice, but there is a need to store both in the registry.

The Link class models a member of a Relation. In common (relational) parlance a Link would be a tuple (row) in a relation (table).

A Link may participate in the following associations: link_membership, link_end, relation_links.

Link is further described in **Error! Reference source not found.**.

8.1.3 Association Classes in the Relations Region

8.1.3.1 Link_End Association Class

The link_end association class models the association between Links and Ends (link ends). This is used to represent the relationship between an n-tuple (row) of a relation and the values for the fields (arguments) of the n-tuple. Hence, a link_end association is used to model the instantiation of a Relation_Role for a particular Link (tuple, row) of a Relation.

The link_end association class has two roles: link (verb form: has_link) and end (verb form: has_end). The link role has a target of class Link. The end role has a target of class Concept. An end (Concept) may have zero or more links (Links). A link (Link) must have at least two, and possibly more ends (Concepts).

EDITOR'S NOTE #119. (Action required) The following sentence has been reworded by the Editor. It should be validated.

Finally, a link_end association class has an association to the Relation_Role which the end (Concept) is intended to fulfil within a Link.

8.1.4 Associations in the Relations Region

8.1.4.1 relation_membership Association

The *relation_membership* association is used to describe the membership of a *Link* within a *Relation*. The association has two roles: relation and member. A *Relation* may have zero or more members (*Links*). A member (*Link*) shall be contained in exactly one *Relation*.

The *relation_membership* association is a strong containment association. Deletion of the Relation will cause cascading deletions of its members (Links).

8.1.4.2 relation_role_set association

The *relation_role_set* association associates a *Relation* to the various *Relation_Roles* which comprise the relation. A *relation_role_set* association has two roles: role (verb form: has_role) and relation (containing_relation). The *role* role references the *Relation_Role* class. The *relation* role references the *Relation* class.

A relation (*Relation*) must have at least one role (*Relation_Role*); it may have more. A role (*Relation_Role*) must have exactly one relation (*Relation*).

The relation_role_set association is a strong containment relation. Hence, if a Relation is deleted all of its roles (Relation_Roles) are also deleted.

8.1.5 Integrity Constraints

8.1.5.1 Compatibility of Link_Ends and Relation_Roles

It must be the case that every role (Relation_Role) specified in the link_end association must correspond to a Relation_Role which is a role of the relation of the link to which the link_end is associated.

9 Data Description Package

EDITOR'S NOTE #120. (Action required) Comments on this clause from US NB noted in the attachments to Issue 248 have not yet been applied.

EDITOR'S NOTE #121. (Informational) Issue 114 has removed *Representation_Class*.

9.1 High-level Data Description metamodel

9.1.1 Overview

A high level overview of the metamodel can be found in Figure 10. It shows four classes: Conceptual_Domain, Value_Domain, and Data_Element and Data_Element_Concept. Figure 11 also shows four associations among the four classes: value_domain_meaning, data_element_domain, data_element_meaning, data_element_concept_domain.

The following text describes the classes and associations shown in Figure 10. It also describes a constraint on the high level metamodel not visible in the UML diagram. More detailed descriptions, e.g., of the class attributes, can be found elsewhere.

Figure 10 can be partitioned into two horizontal parts, one upper part comprised of Data_Element_Concept and Conceptual_Domain and a second lower part comprised of Data_Element and Value_Domain. This view effectively splits the metamodel between a conceptual (or semantic) level (at the top) and a representational level (below). The representational level describes the information artifacts (in contrast to the semantic constructs of the upper level).

This high-level metamodel omits many details, e.g. attributes and some associations, in the interest of clarity of exposition. For a complete characterization of the metamodel the reader must consult the more detailed discussions which follow.

EDITOR'S NOTE #122. (Action required) Issue 107 calls for improved support of Complex Data Types.

EDITOR'S NOTE #123. (Action required) Issue 111 calls for the addition of support for data groups such as database tables and record definitions. Such support should also consider inclusion, encapsulation, stereotyping and inheritance.

EDITOR'S NOTE #124. (Action required) Should we add support for XML documents and/or XML schemas?

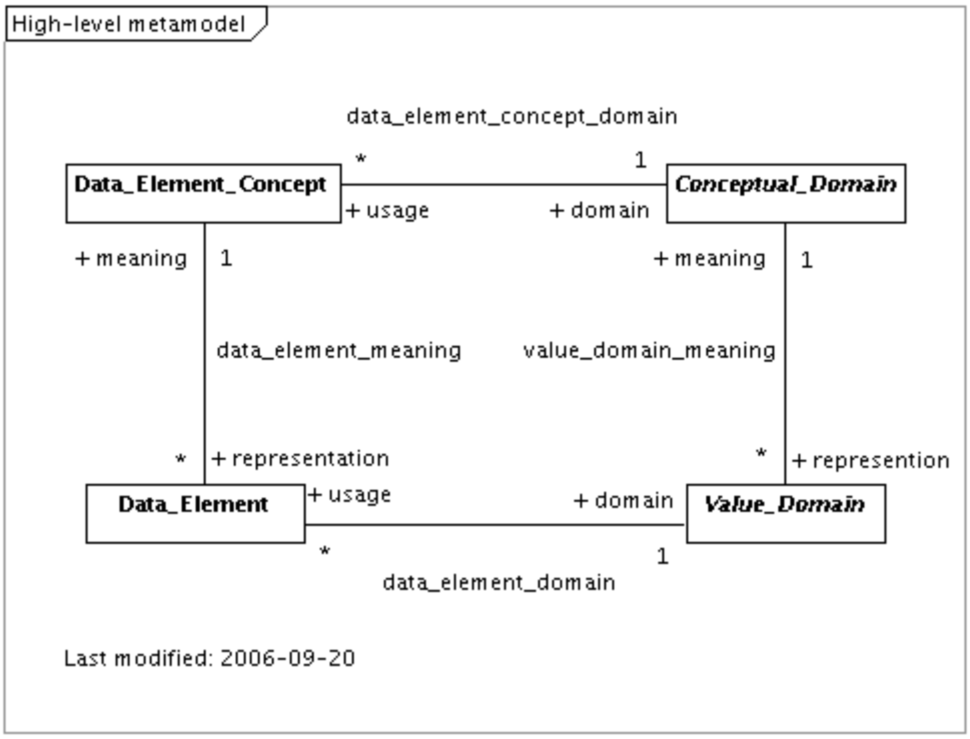


Figure 10 — High-level Data Description metamodel

9.1.2 Classes of High-level Data Description Metamodel

9.1.2.1 Overview

The classes shown in Figure 10 are described below starting with *Conceptual_Domain*, and proceeding clock-wise around the Figure.

9.1.2.2 Conceptual_Domain class

A *Conceptual_Domain* is a set of *Value_Meanings*, which may either be enumerated or expressed via a description.

For example, one possible *Conceptual_Domain* could be countries of the world. It might be associated with two *Value_Domains*: three letter country codes, and full country names. The *Conceptual_Domain* might be used in several *Data_Element_Concepts*, e.g., person's_country_of_residence, person's_country_of_birth, person's_country_of_citizenship.

A *Conceptual_Domain* is a class with two associations: data_element_concept_domain and value_domain_meaning.

Conceptual_Domain is further described in 9.3.2.1.

9.1.2.3 Value_Domain class

A *Value_Domain* is a collection of *Permissible_Values*. It provides representation, but has no implication as to what *Data_Element_Concept* the values are associated with nor what the values mean. *Permissible_Values* are designations, bindings of signs (values) to their corresponding *Value_Meanings*.

A *Value_Domain* is associated with a *Conceptual_Domain*. A *Value_Domain* provides a representation for the *Conceptual_Domain*.

An example of a *Conceptual_Domain* and a set of *Value_Domains* is ISO 3166, Codes for the representation of names of countries. For instance, ISO 3166 describes the set of seven *Value_Domains*: short name in English, official name in English, short name in French, official name in French, alpha-2 code, alpha-3 code, and numeric code.

Additional examples of *Value_Domains* would the Sex Value_Domain which contains two designations (*permissible values*), M -> Male and F-> Female, and the Parent Value_Domain which contains two designations (*permissible values*), M -> Mother and F -> Father.

Note that the two *Value_Domains* are defined over the same set of values (signs).

EDITOR'S NOTE #125. (Action required) The above note may need additional clarification. It is clear that both *Value_Domains* use the values (M, F), but the Edition 3 model explicitly store the values with the *Permissible_Value* instance, which can only be associated with a single *Value_Meaning*. Therefore, the values have to be repeated in the *Permissible_Values* of each *Value_Domain*. The sets are therefore different instances, even though they may be identical. To say that the *Value_Domains* are defined over the same set of values is therefore misleading. In contrast, Edition 2 had a separate *Value* class, which allowed the values to be stored once and referenced from multiple *Permissible_Values* in different *Value_Domains*. The note would therefore have been true in Edition 2, but it was not contained in Edition 2.

Value_Domain is a class with two associations: value_domain_meaning (which is described above), and data_element_domain (which is described below).

Value_Domains may be reused for multiple Data_Elements, see the discussion of countries of the world above.

Value_Domain is further described in 9.3.2.5.

9.1.2.4 Data_Element class

A *Data_Element* is considered to be a basic unit of data of interest to an organization. It is a unit of data for which the definition, identification, representation, and permissible values are specified by means of a set of attributes. Examples of Data_Elements 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.

Data_Element is a class with two associations: data_element_meaning and data_element_domain.

Data_Element is further described in 9.4.2.1.

9.1.2.5 Data_Element_Concept 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.

A Data_Element_Concept is a usage of a Conceptual_Domain, e.g., person's country_of_residence vs. country, which effectively narrows the meaning of the Conceptual_Domain.

A Data_Element_Concept is an abstraction of one or more Data_Elements. Each Data_Element addresses issues of concrete representation , e.g., codes, measurement units, etc. A Data_Element_Concept may be represented by multiple Data_Elements, which may vary in their Value_Domains.

Data_Element_Concept is a class with two associations: data_element_concept_domain (to Conceptual_Domain) and data_element_meaning (to Data_Element).

Data_Element_Concept is further described in 9.2.2.4.

9.1.3 Associations of the High Level Metamodel

9.1.3.1 value_domain_meaning Association

One association of a Conceptual_Domain is value_domain_meaning which links together Conceptual_Domains and Value_Domains. The association value_domain_meaning has two roles: meaning (verb form: means) and representation (verb form: represents).The meaning role (verb form means) specifies the Conceptual_Domain of a Value_Domain. The representation role (verb form: represents) specifies the Value_Domain(s) of a Conceptual_Domain. Each meaning (Conceptual_Domain) may have zero or more associated representations (Value_Domains). Each representation (Value_Domain) has exactly one associated meaning (Conceptual_Domain).

A Value_Domain is a collection of permissible values which are designations, the mappings between value meanings to values (signs). Note that the existence of a value_domain_meaning association between a Conceptual_Domain and a Value_Domain implies the existence of associations between the corresponding individual value meanings and values (these associations (designations) are recorded as permissible values in this metamodel).

Note that in this metamodel, Value_Domains are constrained to have a unique set of meanings (the associated Conceptual_Domain), i.e., a Value_Domain is a function from Values to Value_Meanings. If for some reason one wanted to reuse a Value_Domain (and the associated values, e.g., a code set) for more than one meaning, one is forced to create another Value_Domain and another set of Permissible_Values. This constraint is enforced so that within a Value_Domain one can unambiguously determine the value meanings (in Conceptual_Domain) for the values (in Value_Domain) associated with a Data_Element. (See discussion under Constraints in Section 4.7.3.3)

9.1.3.2 data_element_domain Association

The data_element_domain association connects a Data_Element to the values which may be stored in a Data_Element. Specifically, the data_element_domain association binds a Data_Element to its Value_Domain. The data_element_association has two roles: the usage (verb form: uses) role and the domain (verb form: has_domain) role. The usage role (verb form: uses) specifies the Data_Element which uses a Value_Domain. The domain role (verb form: has_domain) specifies the Value_Domain used for a Data_Element. A usage (Data_Element) has exactly one domain (Value_Domain) A domain (Value_Domain) may have zero or more usages (Data_Elements)

9.1.3.3 data_element_meaning Association

The data_element_meaning associationbinds a Data_Element to its Data_Element_ConceptThe association has two roles: meaning (verb form: means), which specifies a Data_Element_Concept, and representation (verb form: represents), which specifies a Data_Element. The Data_Element_Concept is said to provide the meaning for the Data_Element (the representation). The Data_Element is said to represent the Data_Element_Concept. Each representation (Data_Element) has exactly one meaning (Data_Element_Concept). However, a meaning (Data_Element_Concept) may have zero or more representations (Data_Elements) .

9.1.3.4 data_element_concept_domain Association

The data_element_concept_domain association binds a Data_Element_Concept to its Conceptual_Domain. The data_element_concept_domain association has two roles: the usage role (verb form: uses), which specifies the Data_Element_Concept which uses a Conceptual_Domain, and a second role, domain (verb form: has_domain) which specifies the Conceptual_Domain used by a Data_Element_Concept. Each usage (Data_Element_Concept) has exactly one domain (Conceptual_Domain). Each domain (Conceptual_Domain) may have zero or more associated usages (Data_Element_Concepts).

The data_element_concept_domain association narrows the scope (meaning) of a Conceptual_Domain to that of the Data_Element_Concept, e.g., person’s country of birth (Data_Element_Concept) vs. country (Conceptual_Domain).

9.1.4 Constraints of the High Level Metamodel

9.1.4.1 Equality of mappings from data element to conceptual domain

There are two paths in the metamodel from the Data_Element class to the Conceptual_Domain class. One can either proceed clockwise from Data_Element class via the data_element_meaning association to Data_Element_Concept class and then via data_element_concept_domain association to the Conceptual_Domain class. Alternatively, one can proceed counterclockwise from the Data_Element class via the data_element_domain association to the Value_Domain class and then via the value_domain_meaning association to the Conceptual_Domain class.

It must be the case, that if we start from a specific instance of Data_Element class, that we end at the same instance of the Conceptual_Domain class, regardless of whether we proceed clockwise or counterclockwise through the associations of the metamodel. This constraint is not visible in the UML model.

Formally, we assert that for every x such that x is a member of the Data_Element class, then the domain of the meaning of x must equal the meaning of the domain of x. Note that (unfortunately) domain and meaning are used here twice to refer to different roles (functions).

Note that the possible inverse constraint (starting from Conceptual_Domain) is not true, because the associations are not functions (uniquely valued) in the inverse directions.

9.2 Data_Element_Concept region

9.2.1 Overview

The data element concept region is illustrated in Figure 13. The purpose of the data element concept region is to maintain the information on the concepts related to data elements. The metadata objects in this region concern semantics. Concepts are independent of any internal or external physical representation. The metadata objects in this region are: *Registered_Concepts*, *Conceptual_Domains*, *Data_Element_Concepts*, *Object_Classes* and *Characteristics*. *Object_Classes* and *Characteristics* may be combined to form *Data_Element_Concepts*.

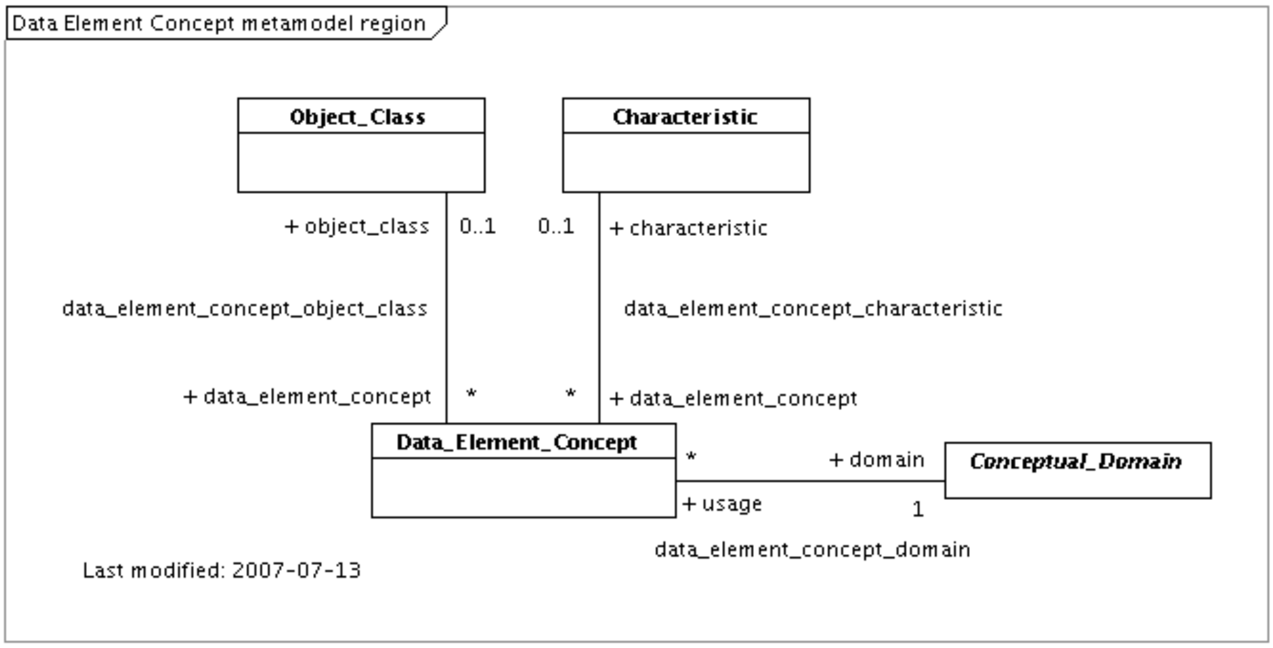


Figure 11 — Data_Element_Concept metamodel region

EDITOR'S NOTE #126. (Action required) This is a change from Edition 2, where Object_Class and property were shown as attributes of Data_Element_Concept. Either way works, but this way we have additional associations and roles to name. Are the names reasonable? Can they be improved?

9.2.2 Classes in the Data_Element_Concept region

9.2.2.1 Overview

Object_Class, Data_Element_Concept and *Characteristic* are all sub-classes of *Concept,* meaning that they are also *Registered_Items* within the metadata registry as well as being potential *Hierarchy_Nodes*.

9.2.2.2 Object_Class

Object_Class is a *Concept* 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. It may be either a single or a group of associated concepts, abstractions, or things.

EDITOR'S NOTE #127. (Action Required) In Edition 2, *Object_Class* was sub-typed as *Concept* and *Concept Relationship*. Now Object_Class is a sub-type of Concept, and can no longer represent *Concept Relationships*. We appear to have lost essential functionality. The following paragraph is no longer true.

An *Object_Class* may represent a single unit of thought (i.e., *Concept*)~~ or a set of *Concepts* in a relationship with each other to form a more complex concept (i.e., *Concept Relationship*). A *Concept* and a *Concept Relationship* are subtypes of an *Object_Class*. Each *Concept Relationship* carries a *concept relationship type description* that describes the nature of the relationship.~~

As a *Registered_Item,* an *Object_Class* is either directly or indirectly associated with an *Administration_Record*, and can be identified, named, defined. In addition, an *Object_Class* can be optionally classified as a *Classifiable_Item* within a *Classification_Scheme*.

An *Object_Class* may have a *data element concept object class* association with zero or more *Data_Element_Concepts*, where the *Object_Class* describes the ideas, abstractions or things in the real world that are represented by the *Data_Element_Concept*.

Example: The *Object_Class* "Person" could be represented by the *Data_Element_Concept* "Person Country of Residence".

9.2.2.3 Characteristic

A *Characteristic* is a *Concept* that represents an abstraction of a property of an object or set of objects. A *Characteristic* is common to all of the members of a given *Object_Class*. It may be any feature that humans naturally use to distinguish one individual object from another. It is the human perception of a single characteristic of an *Object_Class* in the real world. It is conceptual and thus has no particular associated means of representation by which the *Characteristic* can be communicated.

As a *Registered_Item*, a *Characteristic* is directly or indirectly associated with an *Administration_Record* and can be identified, named and defined. In addition, a *Characteristic* can be optionally classified as a *Classifiable_Item* within a *Classification_Scheme*.

EDITOR'S NOTE #128. (Action required) The following text, and Figure 13, introduces the idea that a Data_Element_Concept is a sub-division of a Conceptual_Domain, based on a Characteristic of an Object_Class. If this is true, it should not be buried here in the description of Characteristic, but given more prominence.

A *Characteristic* may have a *data_element_concept_characteristic* association with zero or more *subdivision Data_Element_Concepts*, where the *Characteristic* serves as the *criterion* of the subdivision of the *Conceptual_Domain*.

Example: The *Characteristic* "Residence" could be the *criterion Characteristic* for the *Data_Element_Concept* "Person Country of Residence". Note that the third component, *Country* could be supplied by a *data_element_concept_domain* association with a *Conceptual_Domain*.

9.2.2.4 Data_Element_Concept

Data_Element_Concept is a *Concept* that can be represented in the form of a *Data_Element*. A *Data_Element_Concept* may have a *data element concept object class* association with zero or one *Object_Class* and a *data_element_concept_characteristic* association with zero or one *Characteristic*. The union of a *Characteristic* and an *Object_Class* provides significance beyond either that of the *Characteristic* or the *Object_Class*. A *Data_Element_Concept* thus has a *Definition* independent from the *Definition* of the *Object_Class* or the *Characteristic.*

Every *Data_Element_Concept* must have exactly one *data_element_concept_domain* association with a *Conceptual_Domain*, where the *Data_Element_Concept* supplies a *usage* for the associated *Conceptual_Domain*. The *Conceptual_Domain* is described in **Error! Reference source not found.**.

Example: An association between the *Data_Element_Concept* "Person Country of Residence" and the *Conceptual_Domain* "Country".

As a *Registered_Item*, a *Data_Element_Concept* is directly or indirectly associated with an *Administration_Record*, and can be identified, named and defined. In addition, a *Data_Element_Concept* can be optionally classified as a *Classifiable_Item* within a *Classification_Scheme*.

EDITOR'S NOTE #129. (Action required) The *Data_Element_Concept Relationship* has been removed, and are replaced by more generic relations in the Relations region of the metamodel. We need to ensure that equivalent functionality is truly supported.

9.2.3 Associations in the Data_Element_Concept region

9.2.3.1 data_element_concept_characteristic

Data element concept characteristic is an association between a *Data_Element_Concept* and a *Characteristic* that provides a criterion for the subdivision of a *Conceptual_Domain*. *Data element concept characteristic* has two roles: *criterion* (verb form: *has criterion*) and *subdivision* (verb form: *subdivides*). The *criterion* role references a *Characteristic* and the *subdivision* role references a *Data_Element_Concept*. A *Data_Element_Concept* may be associated with zero or one *criterion Characteristics*. A *Characteristic* may be associated with zero or more *subdivision Data_Element_Concepts*.

9.2.3.2 data_element_concept_domain

A *data_element_concept_domain* is an association denoting the *Conceptual_Domain* that provides the domain for a *Data_Element_Concept.* Every *Data_Element_Concept* must be associated with exactly one *Conceptual_Domain*. A *Conceptual_Domain* may be associated with zero, one or many *Data_Element_Concepts*.

9.2.3.3 data_element_concept_object_class

Data element concept object class is an association between a **Data_Element_Concept** and an **Object_Class** that represent a particular set of ideas, abstractions, or things in the real world that whose properties and behaviour follow the a set of rules as represented by the **Data_Element_Concept**. *Data element concept object class* has two roles: *represented* (verb form: *represents*) and *representation* (verb form: *represented by*). The *represented* role references an **Object_Class** and the *representation* role references a **Data_Element_Concept**. A **Data_Element_Concept** may be associated with zero or one *represented* **Object_Classes**. An **Object_Class** may be associated with zero or more *representation* **Data_Element_Concepts.**

9.3 Conceptual and Value_Domain region

9.3.1 Overview

This region of the metamodel addresses the administration of *Conceptual_Domains* and *Value_Domains*. These domains can be viewed as logical code sets and physical code sets. *Conceptual_Domains* support *Data_Element_Concepts* and *Value_Domains* support *Data_Elements*. The region is illustrated in Figure 14.

EDITOR'S NOTE #130. (Action required) The relationships *Conceptual_Domain Relationship* and *Value_Domain_Relationship* have been removed by Issue 229. The rationale for *Conceptual_Domain Relationship* may be that Conceptual_Domains are sub-classes of Concept, and that the Relations regions allows concepts to be related. However, there does not seem to be equivalent functionality for *Value_Domains*. This removal is controversial and has not been agreed.

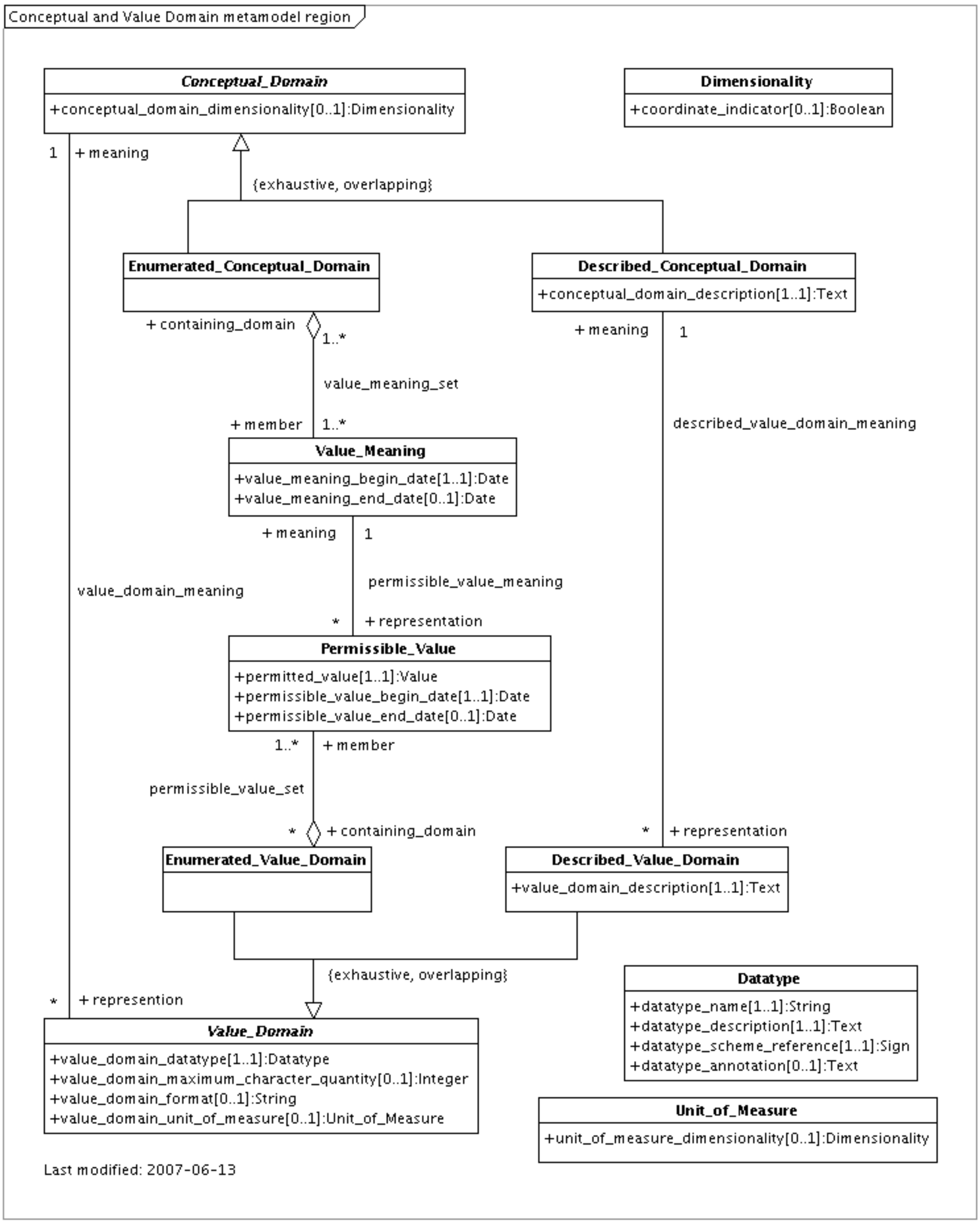

Conceptual and Value Domain metamodel region
Conceptual_Domain
+conceptual_domain_dimensionality[0..1]:Dimensionality
Dimensionality
+coordinate_indicator[0..1]:Boolean
1
+ meaning
{exhaustive, overlapping}
Enumerated_Conceptual_Domain
Described_Conceptual_Domain
+conceptual_domain_description[1..1]:Text
+ containing_domain
1..*
+ meaning
1
value_meaning_set
described_value_domain_meaning
+ member
1..*
Value_Meaning
+value_meaning_begin_date[1..1]:Date
+value_meaning_end_date[0..1]:Date
+ meaning
1
value_domain_meaning
permissible_value_meaning
*
+ representation
Permissible_Value
+permitted_value[1..1]:Value
+permissible_value_begin_date[1..1]:Date
+permissible_value_end_date[0..1]:Date
1..*
+ member
permissible_value_set
*
+ containing_domain
*
+ representation
Enumerated_Value_Domain
Described_Value_Domain
+value_domain_description[1..1]:Text
{exhaustive, overlapping}
*
+ represention
Datatype
+datatype_name[1..1]:String
+datatype_description[1..1]:Text
+datatype_scheme_reference[1..1]:Sign
+datatype_annotation[0..1]:Text
Value_Domain
+value_domain_datatype[1..1]:Datatype
+value_domain_maximum_character_quantity[0..1]:Integer
+value_domain_format[0..1]:String
+value_domain_unit_of_measure[0..1]:Unit_of_Measure
Unit_of_Measure
+unit_of_measure_dimensionality[0..1]:Dimensionality
Last modified: 2007-06-13

Figure 12 — Conceptual and value domain metamodel region

9.3.2 Classes in the Conceptual and Value_Domain region

Conceptual_Domain, *Dimensionalilty*, *Value_Meaning*, *Value_Domain* and *Unit_of_Measure* are each sub-classes of *Registered_Item*, and hence of *Identified_Item* and *Designatable_Item* (see Figure 10). Such are the mechanisms by which they are identified and named, respectively. As sub-classes of *Classifiable_Item*, they may also be classified within a Classification_Scheme.

9.3.2.1 Conceptual_Domain

A *Conceptual_Domain* is a set of *Value_Meanings*, which may either be enumerated or expressed via a description. *Conceptual_Domain* is an abstract class, which has two possible subtypes: *Enumerated_Conceptual_Domain* and *Described Conceptual_Domain.* A *Conceptual_Domain* instance must be either or both an *Enumerated_Conceptual_Domain* or a *Described Conceptual_Domain.*

NOTE In Figure 12, the use of *italics* in the name of *Conceptual_Domain* indicates that it is an abstract class.

The *Conceptual_Domain* class has one attribute, conceptual_domain_dimensionality, of type *Dimensionality*. The dimensionality attribute specifies the Dimensionality as elaborated in the discussion of the Dimensionality class below in Section

Attribute	**Allowed Occurrences**	**Datatype**
conceptual_domain_dimensionality	Zero or one per *Conceptual_Domain*	Dimensionality

When a *conceptual_domain_dimensionality* is specified, for any *Value_Domain* that is based on this *Conceptual_Domain,* any *Unit_of_Measure* specified shall be consistent with this *dimensionality*.

9.3.2.2 Enumerated_Conceptual_Domain

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.*

EXAMPLE: The notion of countries that is specified in ISO 3166, Codes for the representation of names of countries.

As a sub-type of *Conceptual_Domain,* an *Enumerated_Conceptual_Domain* inherits the attributes and relationships of the former.

9.3.2.3 Value_Meaning

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. Value_Meaning* is a subclass of *Concept*, and thus of *Registered_Item* (see Figure 11 on p.119).

EDITOR'S NOTE #131. (Action required) Because Value_Meaning was to be a sub-class of Registered_Item, *value meaning identifier* and *value meaning description* have been removed as direct attributes. The attributes and relationships of *Identified_Item*, and *Documentable Item* should be used instead. However, in this CD, the explicit sub-typing of Registered_Item has also been removed.

The *Value_Meaning* class has two attributes: *value_meaning_begin_date*, and *value_meaning_end_date*. The mandatory *value_meaning_begin_date* of type *Date* is used to specify the date at which this *Value_Meaning* became, or will become, a valid *Value_Meaning*. The optional *value_meaning_end_date* of type *Date* specifies the date on which the *Value_Meaning* ceased, or will cease, to be valid. The absence of the *value_meaning_end_date* indicates that the *Value_Meaning* is still valid.

Attribute	Allowed Occurrences	Datatype
value_meaning_begin_date	One per *Value_Meaning*	Date
value_meaning_end_date	Zero or one per *Value_Meaning*	Date

Value_Meanings may participate in the *value_meaning_set* association and the *permissible_value_meaning* association. See discussion below.

9.3.2.4 Described Conceptual_Domain

A *Conceptual_Domain* that cannot be expressed as a finite set of *Value_Meanings* is called a *Described Conceptual_Domain*. It may be expressed via a description or specification, such as a rule, a procedure, or a range (i.e., interval). As a sub-class of *Conceptual_Domain,* a *Described Conceptual_Domain* inherits the attributes and relationships of the former.

EDITOR'S NOTE #132. (Informational) The *described_conceptual_domain_description* is left as an attribute of *Described Conceptual_Domain* because it is mandatory, whereas *value meaning description* is replaced by the *Definition.definition_text* of a *Designatable_Item*.

The *Described Conceptual_Domain* class has one attribute: *described_conceptual_domain_description* which is of type *Text*. Each *Described Conceptual_Domain* class must have exactly one *described_conceptual_domain_description* attribute.

Attribute	Allowed Occurrences	Datatype
described_conceptual_domain_description	One per *Described Conceptual_Domain*	Text

9.3.2.5 Value_Domain

One of the key components of a representation is the *Value_Domain*. A *Value_Domain* provides representation, but has no implication as to the *Data_Element_Concept* with which the values are associated, nor what the values mean.

A *Value_Domain* is an abstract class which is used to denote a collection of *Permissible_Values* associated with a *Conceptual_Domain*. A *Value_Domain* has two possible subtypes (subclasses): an *Enumerated_Value_Domain* and a *Described_Value_Domain*. A *Value_Domain* must be either one or both an *Enumerated Valued* or a *Described_Value_Domain*.

NOTE In Figure 12, the use of *italics* in the name *Value_Domain* indicates that it is an abstract class.

A *Value_Domain* is associated with a *Conceptual_Domain*. A *Value_Domain* provides a representation for the *Conceptual_Domain*.

EXAMPLE: 'ISO 3166 Codes for the representation of names of countries' describes seven distinct *Value_Domains* for the single *Conceptual_Domain* 'names of countries'. The seven *Value_Domains* are: 'short name in English', 'official name in English', 'short name in French', 'official name in French', 'alpha-2 code', 'alpha-3 code' and 'numeric code'.

A *Value_Domain* has four attributes as listed below:

Attribute	Allowed Occurrences	Datatype
value_domain_datatype	One per *Value domain*	*Datatype*
value_domain_format	Zero or one per *Value_Domain*	String
value_domain_maximum_character_quantity	Zero or one per *Value_Domain*	Integer
value_domain_unit_of_measure	Zero or one per *Value_Domain*	*Unit_of_Measure*

The mandatory *value_domain_datatype* attribute of type *Datatype* specifies the datatype associated with all values in the *Value_Domain*.

The optional *value_domain_format* attribute of type *String* provides a template for the structure of the values in the *Value_Domain*.

The optional *value_domain_maximum_character_quantity* attribute of type *Integer* specifies the maximum number of characters that may be used to represent a value in the *Value_Domain*. This attribute applies only to character datatypes.

The optional *value_domain_unit_of_measure* attribute of type *Unit_of_Measure,* specifies the unit of measure used for values in the *Value_Domain.*

9.3.2.6 Enumerated_Value_Domain

An *Enumerated_Value_Domain* is one where the *Value_Domain* is expressed as an explicit set of two or more *Permissible_Values*. The *Enumerated_Value_Domain* class is a subclass of *Value_Domain*.

Each *Enumerated_Value_Domain* class may participate in the *permissible_value_set* association.

9.3.2.7 Permissible_Value

A *Permissible_Value* is an expression of a *Value_Meaning* within an *Enumerated_Value_Domain*. It is one of a set of such values that comprises an *Enumerated_Value_Domain*.

Each *Permissible_Value class* may participate in two associations: *permissible_value_meaning* and *permissible_value_set*.

The *Permissible_Value* class has three attributes as listed below:

Attribute	Allowed Occurrences	Datatype
permitted_value	One per *Permissible_Value*	*Value*
permissible_value_begin_date	One per *Permissible_Value*	Date
permissible_value_end_date	Zero or one per *Permissible_Value*	Date

The mandatory *permitted_value* attribute is of type *Value* and is used to specify the actual value of the *Permissible_Value*.

The mandatory *permissible_value_begin_date* specifies the date at which the *Permissible_Value* became valid. By imputation, this is also considered to be date at which the *Permissible_Value* was bound to the associated *Value_Meaning*, since the *permissible_value_meaning* association mandates that there must be exactly one meaning (*Value_Meaning*) for each representation (*Permissible_Value*).

The optional *permissible_value_end_date* specifies the date at which the *Permissible_Value* ceased to be valid, and (by imputation) ceased to be bound to its associated meaning (*Value_Meaning*) via the *permissible_value_meaning* association. The absence of the *permissible_value_end_date* attribute indicates that the *Permissible_Value* is still valid and (by imputation) still bound to its *Value_Meaning* via the *value_meaning* association.

9.3.2.8 Described_Value_Domain

A *Described_Value_Domain* is a concrete subclass of the abstract class *Value_Domain* which is characterized via a description or specification, such as a rule, a procedure, or a range (i.e., interval), rather than as an explicit set of *Permissible_Values*.. As a sub-class of *Value_Domain,* a *Described_Value_Domain* inherits the attributes and relationships of the former.

A *Described_Value_Domain* has one attribute:

Attribute	**Allowed Occurrences**	**Datatype**
described_value_domain_description	One per *Described_Value_Domain*	Text

The mandatory *described_value_domain_description* attribute of type Text, records the characterization (description) of the value domain.

9.3.2.9 Datatype

EDITOR'S NOTE #133. (Action required) In the following sentence, the text following 'for example' simply repeats the original statement but applied to a Data_Element, which is not so much an example, as how any value domain is supposed to be used. Some rewording would seem to be needed.

A *Value_Domain* is associated with a *Datatype* — a set of distinct values, characterized by properties of those values and by operations on those values, for example the category used for the collection of letters, digits, and/or symbols to depict values of a *Data_Element* determined by the operations that may be performed on the *Data_Element*.

EDITOR'S NOTE #134. (Action required) The Editor has extrapolated the datatype change made in *Designation.designation_name* to *Datatype.datatype_name* in the table below. If accepted, this change needs to be applied to the Figure as well.

EDITOR'S NOTE #135. (Action required) Should we make *Datatype* a subclass of *Designatable_Item,* in which case it would inherit *Designation* and *Definition* and we could eliminate the first two attributes below?x

A Datatype has four attributes as listed below:

Attribute	**Allowed Occurrences**	**Datatype**
datatype_name	One per *Datatype*	*Sign*
datatype_description	Zero or one per *Datatype*	*Text*
datatype_scheme_reference	One per *Datatype*	*Sign*
datatype_annotation	Zero or one per *Datatype*	*Text*

The mandatory *datatype_name* attribute, of type *Sign*, is used to designate the *Datatype*. The datatype_name is usually drawn from some external source, which in turn is designated by means of the mandatory *datatype_scheme_reference* of type *Sign*.

The characterization of the datatype is also specified by the reference to the mandatory *datatype_scheme_reference,* but this may be supplemented by the optional *datatype_description* attribute of type *Text*.

Finally, additional, optional information concerning the *Datatype* may be specified by means of the optional *datatype_annotation* attribute of type *Text*.

9.3.2.10 Unit_of_Measure

EDITOR'S NOTE #136. (Action required) The resolution of Issue 125 states to add "unit_of_measure_scheme_reference" to "Unit_of_Measure" class, to identify source of "unit_of_measure_name". Because Unit_of_Measure is now a sub-class of Registered_Item the "unit_of_measure_name" is removed and replaced by the common facilities of Designatable_Item. Should we still add a scheme reference?

EDITOR'S NOTE #137. (Action required) The resolution of Issue 125 states to move "unit_of_measure_precision" to Value_Domain, and make it optional. Instead, it has been moved to Data_Element.

If appropriate, a *Value_Domain* may be associated with a *Unit_of_Measure* — the units in which any associated *Data_Element* values are specified. Unit_of_Measure is a subclass of Registered_Item (see Figure 10).

The *Unit_of_Measure* class has one attribute:

Attribute	**Allowed Occurrences**	**Datatype**
unit_of_measure_dimensionality	Zero or one per *Unit_of_Measure*	*Dimensionality*

Note: While units of measure are commonly physical units of measure, they may also be currency units (in which the corresponding Dimensionality would be money).

9.3.2.11 Dimensionality

Dimensionality is the class used to represent a set of equivalent units of measure, where equivalence between two units of measure is determined by the existence of a quantity-preserving one-to-one correspondence between values measured in one unit of measure and values measured in the other unit of measure, independent of context, and where the characterizing operations are the same.

EXAMPLE: inches, feet, meters, and centimeters are all units of measure whose dimensionality is length. Other common dimensionalities include: mass, time, area, volume, etc.

NOTE 1 The equivalence defined here forms an equivalence relation on the set of all units of measure. Each equivalence class corresponds to a dimensionality. The units of measure "temperature in degrees Fahrenheit" and "temperature in degrees Celsius" have the same dimensionality, because given a value measured in degrees Fahrenheit there is a value measured in degrees Celsius that is the same quantity, and vice-versa. Quantity preserving one-to-one correspondences are the well-known equations $C^{\circ} = (5/9)*(F^{\circ} - 32)$ and $F^{\circ} = (9/5)*(C^{\circ}) + 32$. (Note that we have here assumed we are dealing with temperature coordinates. There is no offset when converting among temperature interval measures, e.g., the temperature difference between the coldest and hottest temperature on a day.)

NOTE 2 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.

EDITOR'S NOTE #138. (Action required) In NOTE 2, while 'currency' is fairly self-explanatory, 'quality indicator' and 'color intensity' probably require definition and further explanation if they are to be used as examples. It may be preferable to limit the example to currency.

NOTE 3 Quantities may be grouped together into categories of quantities which are mutually comparable. Lengths, diameters, distances, heights, wavelengths and so on would constitute such a category. Mutually comparable quantities usually have the same dimensionality (but see note 4) ISO 31-0 calls these "quantities of the same kind".

NOTE 4 The requirement of common “characterizing operations” for all units of measure with the same dimensionality is a stronger requirement than that commonly adopted in conventional dimensional analysis (where comparability and transformability usually suffice). Thus with respect to temperature, absolute temperature coordinates (e.g., Kelvins) are here 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.

NOTE 5 Dimensionalities for physical units of measurement are commonly specified as the products or quotients of powers of basis dimensions: mass, length, time… However, in this metamodel we do not dictate the specification of dimensionalities, only their names and coordinate status.

Dimensionality is a sub-class of Registered_Item (see Figure 10). It has one attribute:

Attribute	**Allowed Occurrences**	**Datatype**
coordinate	Zero or one per Dimensionality	Boolean

The *coordinate* attribute of a *Dimensionality* is of type *Boolean* and can occur zero or once. For dimensionalities of physical units, it is required.

EDITOR'S NOTE #139. (Action required) Do we need to specify the constraints more precisely? Can coordinate apply to anything other than physical units?

The *coordinate* attribute is a predicate on the *Dimensionality* whose value is true if the Dimensionality is a coordinate – an interval measure taken with respect to a specific origin of a frame of reference. Otherwise, e.g., if the Dimensionality refers to an interval measure, the value of the coordinate element is false.

EDITOR'S NOTE #140. (Action required) The following example suggests that Dimensionalities for all physical units essentially require two instances, one where *coordinate* is *True*, and one where it is *False*. Might it be better to put the *coordinate attribute* on *Unit_of_Measure*, or even on *Data_Element*, as we have done with *precision.*

EXAMPLE: There might be two Dimensionalities concerned with length: one a measure of the size of an object (hence an interval measure), the other a measure of the location of an object (hence a coordinate).

EDITOR'S NOTE #141. (Action required) The following paragraphs seems redundant with the notes above.

A very common example is the use of temperature to measure the absolute temperature of a point, or to measure the size of a temperature interval, e.g., the temperature difference across the wall of a furnace. Aside from the semantic difference, the function for converting units of measure, e.g., temperature, depends on whether it is a coordinate or an interval measure. For example when converting degrees Celsius to Kelvins, one must add 273.16 for temperature coordinates, but not for temperature interval measures.

Note however, that in the Dimensionality class we do not explicitly specify what the frame of reference is for the Dimensionality. For some units of measure, such as temperature in Kelvins, or degrees Celsius the frame of reference is implicit in the units of measure. Additional examples of coordinate Dimensionalities would include longitude and latitude. However, in many cases the frame of reference for a coordinate measurement is specified as part of the Data_Element. This is quite common in computer aided design applications.

9.3.3 Associations in the Conceptual and Value_Domain region

9.3.3.1 value_domain_meaning Association

The *value_domain_meaning* association has two roles: meaning (verb form: means) and representation (verb form: represents). The meaning role refers to a *Conceptual_Domain* class. The representation role refers to a *Value_Domain* class. Each representation (*Value_Domain*) must have exactly one meaning (*Conceptual_Domain*). However, each meaning (*Conceptual_Domain*) may have zero or more representations (*Value_Domains*).

NOTE This version of the metamodel lacks any mechanism to specify the valid dates for the value_domain_meaning association.

9.3.3.2 value_meaning_set Association

The *value_meaning_set* association has two roles: containing_domain (verb form: contains_domain) and member (verb form: has_member). The containing_domain role refers to an *Enumerated_Conceptual_Domain* class. The member role refers to a *Value_Meaning*. Each member (*Value_Meaning*) may have zero or more containing_domains (*Enumerated_Conceptual_Domain*). Each containing_domain (*Enumerated_Conceptual_Domain*) shall have one or more members (*Value_Meanings*). The value_meaning_set association is a weak containment association, which means that deletion of the containing *Enumerated_Conceptual_Domain* does not imply a cascading delete the contained *Value_Meanings*.

NOTE This version of the metamodel lacks any mechanism to specify the valid dates for the *value_meaning_set* association.

9.3.3.3 described_value_domain_meaning Association

EDITOR'S NOTE #142. (Informational) It has been suggested that the described_value_domain_meaning association is redundant w.r.t. the value_domain_meaning association. However, the Editor notes that it does explicitly restrict the association to the two 'Described' sub-classes. When a Conceptual_Domain and Value_Domain are a combination of Enumerated and Described domains, the two associations are not equivalent, though for navigation purposes they are still redundant. Should we include additional text explaining this?

The *described_value_domain_meaning* association has two roles: meaning (verb form: means) and representation (verb form: represents). The meaning role refers to a *Described_Conceptual_Domain* class. The representation role refers to a *Described_Value_Domain* class. Each representation (*Described_Value_Domain*) must have exactly one meaning (*Described_Conceptual_Domain*). However, each meaning (*Described Conceptual_Domain*) may have zero or more representations (*Described_Value_Domains*).

NOTE This version of the metamodel lacks any mechanism to specify the valid dates for the described_value_meaning association.

9.3.3.4 permissible_value_meaning Association

The *permissible_value_meaning* association has two roles: meaning (verb form: means) and representation (verb form: represents). The meaning role refers to a *Value_Meaning* class. The representation role refers to a *Permissible_Value* class. Each representation (*Permissible_Value*) must have exactly one meaning (*Value_Meaning*). However, each meaning (*Value_Meaning*) may have zero or more representations (*Permissible_Values*).

NOTE See discussion above under *Value_Meaning* for treatment of valid dates for *permissible_value_meaning* association. We impute valid dates for the *permissible_value_meaning* association from the *permissible_value_begin_date* and *permissible_value_end_date*.

9.3.3.5 permissible_value_set Association

The *permissible_value_set* association has two roles: member (verb form: has member) and containing_domain (verb form: contains_domain). The member role refers to a *Permissible_Value* class. The contains_domain role refers to an *Enumerated_Value_Domain* class. Each member (*Permissible_ Value*) may have zero or more containing_domains (*Enumerated_Value_Domains*). However, each containing_domain (*Enumerated_Value_Domain*) shall have one or more members (*Permissible_Values*). The *permissible_value_set* association is a weak containment relation, i.e., deletion of the containing domain does not cause a cascading delete of the members (*Permissible_Values*).

NOTE This version of the metamodel lacks any mechanism to specify the valid dates for the permissible_value_set association.

9.3.4 Additional Constraints of the Conceptual and Value_Domain region

9.3.4.1 Overview

This sub-clause specifies additional constraints that are not included in the UML diagram.

9.3.4.2 value_domain_meaning Association Constraints

EDITOR'S NOTE #143. (Action required) It may be useful to name or at least identify each of these constraints so that they can be referenced. The Editor has attempted to provide descriptive sub-headings, which might serve as names.

Constraint #1: Consistency of Enumeration, Description or combination for Conceptual and Value_Domains

Suppose that r is an instance of the class *Value_Domain* and s is an instance of the class *Conceptual_Domain*, such that s is the meaning of r according to the *value_domain_meaning* association. There must exist such an s for every r according to the cardinality constraints on the *value_domain_meaning* association. Then it is either the case that r is an instance of *Enumerated_Value_Domain* and s is an instance of *Enumerated_Conceptual_Domain* or it is the case that r is an instance of *Described_Value_Domain* and s is an instance of *Described Conceptual_Domain*. Since neither *Value_Domains*, nor *Conceptual_Domains* are disjoint w.r.t. the Enumerated and Described subclasses it may be that r and s are both Enumerated and Conceptual Value/Conceptual_Domains.

Constraint #2: Consistency of meanings reached by meaning associations

Suppose that there exists an instance x of the class *Described_Value_Domain*, such that the instance y is the meaning of x according to the *value_domain_meaning* association (since every instance of a *Described_Value_Domain* is also a *Value_Domain*) where y is some instance of a *Conceptual_Domain* (either a *Described Conceptual_Domain* or an *Enumerated_Conceptual_Domain*). There must exist such an instance y according to the cardinality constraints on the value_domain_meaning association.

EDITOR'S NOTE #144. (Action required) The parenthetical comment above (either a *Described Conceptual_Domain* or an *Enumerated_Conceptual_Domain*) appears incorrect, since the following paragraph requires the Conceptual_Domain to be a Described_Conceptual_Domain. Can we remove the parenthetical comment?

According to the cardinality constraints for the described_value_meaning association there must also exist an instance z of the Described_Conceptual_Domain such that z is the meaning of x. Then it must be the case that z is equal to y, i.e., the meaning of x must be same according to both the value_domain_meaning and described_value_domain meaning associations.

Constraint #3: Mapping Enumerated_Value_Domains across Enumerated_Conceptual_Domains

Suppose that there exists an instance *u* of the class *Enumerated_Value_Domain*, such that the instance *v* is the meaning of *u* according to the *value_domain_meaning* association (since every instance of a

Enumerated_Value_Domain is also a *Value_Domain*) where v is some instance of a *Conceptual_Domain* (either a *Described Conceptual_Domain* or an *Enumerated_Conceptual_Domain*). There must exist such an instance *v* according to the cardinality constraints on the *value_domain_meaning* association.

EDITOR'S NOTE #145. (Action required) The parenthetical comment above (either a *Described Conceptual_Domain* or an *Enumerated_Conceptual_Domain*) appears incorrect, since the following paragraph requires the Conceptual_Domain to be an Enumerated_Conceptual_Domain. Can we remove the parenthetical comment?

Now for each instance u of the class *Enumerated_Value_Domain* there must exist a non-null set W of the members of the *Permissible_Values* class according to the *permissible_value_set* association. For each element w_i of W there is an exactly one instance m_i of the class *Value_Meaning* such that m_i is the meaning of w_i according to the *permissible_value_meaning* association. Let M be the set union of these m_i.. Now consider the (possibly empty) sets E_i each of which is the unions of instances of the class *Enumerated_Conceptual_Domain* which are the containing domains of the various value meanings of each m_i. Then it must be the case that for every m_i in M there exists an instance e in the set E_i such that e is equal to v.

NOTE The final existential quantification (rather than universal quantification) over the elements of each set E_i arises because we no longer constrain *Value_Meanings* to exist in a single *Enumerated_Conceptual_Domain*.

9.3.4.3 Consistent Dimensionalities

Conceptual_Domains may have an attribute *conceptual_domain_dimensionality*. *Value_Domains* may have an attribute *value_domain_unit_of_measure* of type *Unit_of_Measure*. Suppose that we have an instance c of the class *Conceptual_Domain* and an instance v of a *Value_Domain* such that c is the meaning of v according to the *value_domain_meaning* association (or some equivalent path as above). Suppose that d (of type *Dimensionality*) is the *conceptual_domain_dimensionaility* attribute of the instance c. Suppose that e is the dimensionality of the *value_domain_unit_of_measure* of v. Then it must be the case that the d is equal to e.

In plain English, the *dimensionality* of the *unit_of_measure* of a *Value_Domain* must be the same as the *dimensionality* of the *Conceptual_Domain* which provides the meaning of the *Value_Domain*.

9.4 Data_Element region

9.4.1 Overview

The Data_Element metamodel region, illustrated in Figure 13 — Data_Element metamodel region, is used to address the administration of *Data_Elements*. *Data_Elements* provide the formal representations for some information (such as a fact, a proposition, an observation, etc.) about some concrete or abstract thing. *Data_Elements* are reusable and shareable representations of *Data_Element_Concepts*.

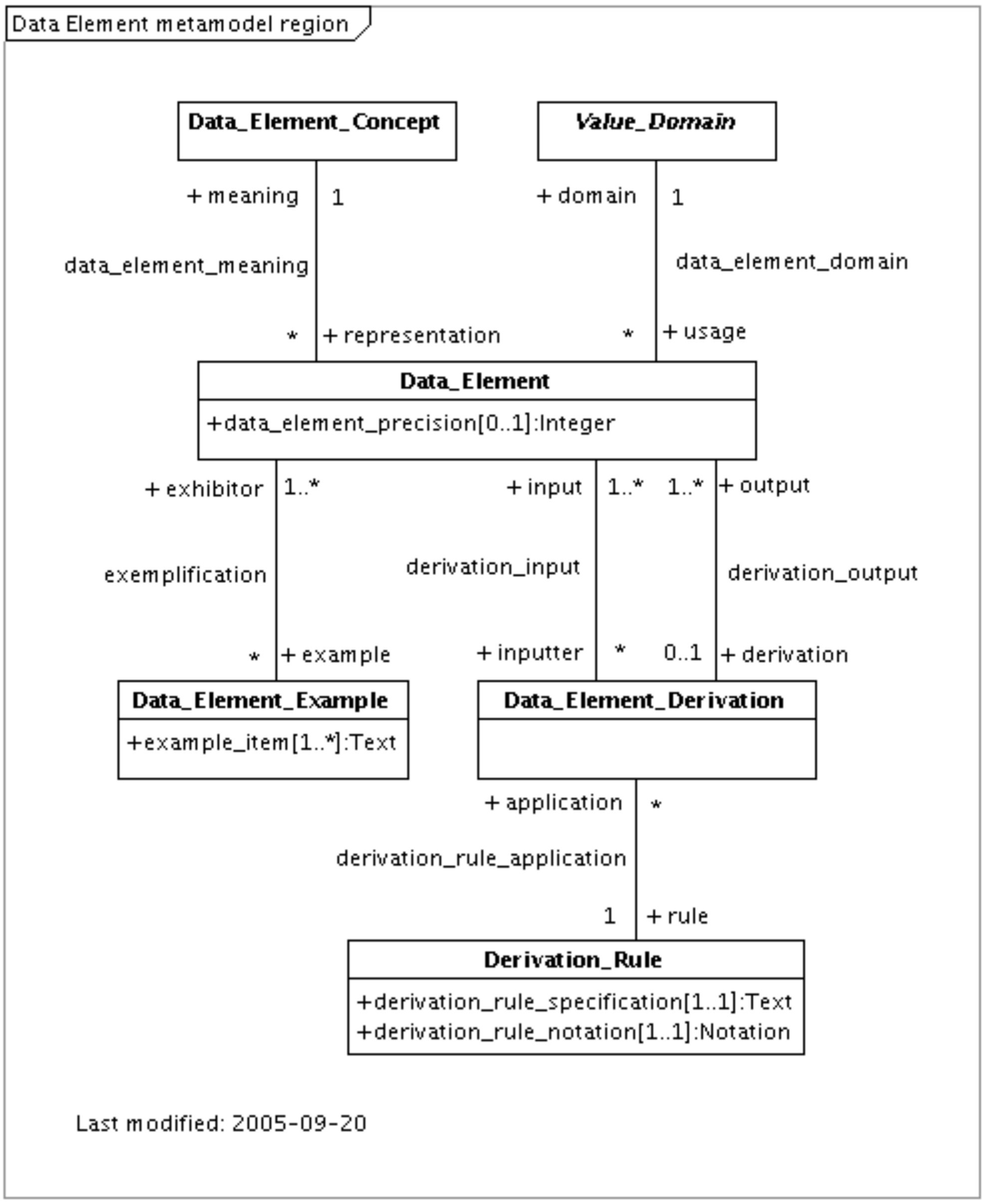


Figure 13 — Data_Element metamodel region

9.4.2 Classes in the Data_Element Region

9.4.2.1 Data_Element

A *Data_Element* is considered to be a basic unit of data of interest to an organization. It is a unit of data for which the definition, identification, representation, and permissible values are specified by means of a set of attributes.

NOTE In general usage, the term *data element* and *data element type* are used interchangeably. In this document, the shorter term *data element* is used.

As a *Registered_Item,* a *Data_Element* is directly or indirectly associated with an *Administration_Record,* allowing it to be identified, named and defined. In addition, a *Data_Element* can optionally be classified as a *Classifiable_Item* in a *Classification_Scheme*.

A *Data_Element* is formed when a *Data_Element_Concept* is assigned a representation. One of the key components of a representation is the *Value_Domain*, i.e., restricted valid values.

A *Data_Element* is the association of a *Data_Element_Concept* with a *Value_Domain*. A *Data_Element* cannot be registered as a *Registered_Item* without being associated with a *Data_Element_Concept* and a *Value_Domain*.

A *data_element_precision* may be used to specify the number of decimal places permitted in any associated data element values.

Attribute	Allowed Occurrences	Datatype
data_element_precision	Zero or one per *Data_Element*	Integer

9.4.2.2 Data_Element_Concept

Data_Element_Concept is described under the Data_Element_Concept region in **Error! Reference source not found.**. A *Data_Element_Concept* may be associated with several *Value_Domains* resulting in a different *Data_Element* for each association.

EDITOR'S NOTE #146. (Action required) The association between Data_Element and Value_Domain, currently named 'data_element_domain' would be more appropriately named 'data element value domain'. (Noted in response to issue 230.)

9.4.2.3 Value_Domain

Value_Domain is described under the Conceptual_Domain and Value_Domain region in 9.3.2.5. A *Value_Domain* provides representation, but has no implication as to what *Data_Element_Concept* the values are associated with, nor what the values mean. A *Value_Domain* may be associated with multiple *Data_Elements*.

9.4.2.4 Data_Element_Example

A *Data_Element_Example* provides representative illustration(s) of instances of a *Data_Element.* Every *Data_Element_Example* shall have an *exemplification* association with one or more *exhibitor Data_Elements*, where the *Data_Element_Example* serves as an *example* for the *Data_Element*.

A *Data_Element_Example* shall have one or more *example_item* attributes of type *Text* that provide representative illustrations of instances of a *Data_Element*.

Attribute	Allowed Occurrences	Datatype
data element example_item	One or more per *Data_Element_Example*	Text

9.4.2.5 Derivation_Rule

A *Derivation_Rule* specifies the logical, mathematical, and/or other operations for derivation. The *Derivation_Rule* may range from a simple operation such as subtraction to a very complex set of derivations (derivation being defined as a relationship between a *Derivation_Rule* and an input set upon which it acts). *Derivation_Rules* are not limited to arithmetic and logical operations.

As a *Registered_Item,* a *Derivation_Rule* is directly or indirectly associated with an *Administration_Record* and can be identified, named, defined and optionally classified as a *Classifiable_Item* in a *Classification_Scheme*. A *Derivation_Rule* may be registered as an *Administered_Item* without necessarily being associated with any *Data_Element_Derivation*.

A *Derivation_Rule* may have a *derivation_rule_application* association with zero or more *application Data_Element_Derivations*, where the *Derivation_Rule* provides the *rule* for the associated *Data_Element_Derivation*.

Every *Derivation_Rule* must have exactly one *derivation_rule_specification* of type *Text* that specifies the rule semantics.

Every *Derivation_Rule* must have exactly one *derivation_rule_notation* of type *Notation* that specifies the syntax and semantics used in the *derivation_rule_specification.*

A *Derivation_Rule* may be registered as a *Registered_Item* without necessarily being associated with any *Data_Element_Derivation*.

Attribute	Allowed Occurrences	Datatype
derivation_rule_specification	One per *Derivation_Rule*	Text

9.4.2.6 Data_Element_Derivation

A *Data_Element_Derivation* is the application of a *Derivation_Rule* to one or more input *Data_Elements*, to derive one or more output *Data_Elements.* A *Data_Element_Derivation* may have a *Derivation_Rule* that is a specification of derivation for the *Data_Element*.

Data_Element_Derivation is an object that describes the *Data_Element(s)* that serve as sources or inputs to a *Derivation_Rule* and the *Data_Element(s)* that are the products or outputs of the *Derivation_Rule*.

Every *Data_Element_Derivation* shall have one or more *derivation_input* associations with an *input Data_Element*, where the *Data_Element_Derivation* serves as the *inputter* for the associated *Data_Element*.

EDITOR'S NOTE #147. (Action required) Can we find a better name for the role *inputter?*

Every *Data_Element_Derivation* shall have a one or more *derivation_output* associations with an *output Data_Element*, where the *Data_Element_Derivation* serves as the *derivation* for the associated *Data_Element*.

9.4.3 Associations in the Data_Element region

9.4.3.1 data_element_domain Association

data_element_domain is an association between a *Data_Element* and a *Value_Domain* that describes a set of possible values that may be recorded in an instance of the *Data_Element*.

9.4.3.2 data_element_meaning Association

data_element_meaning is an association between a *Data_Element* and a *Data_Element_Concept* that identifies the *Data_Element_Concept* that provides the meaning for the *Data_Element*.

9.4.3.3 exemplification Association

Exemplification is an association between a *Data_Element* and a *Data_Element_Example* that provides an example instance or use of the *exhibitor Data_Element*. *Exemplification* has two roles: *exhibitor* and *example*. The *exhibitor* role refers to a *Data_Element* and the *example* role refers to a *Data_Element_Example*. An *exhibitor Data_Element* may be associated with zero or more *example Data_Element_Examples*. Every *Data_Element_Example* shall be associated with one or more *exhibitor Data_Elements*.

9.4.3.4 derivation_input Association

derivation_input is an association between a *Data_Element* and a *Data_Element_Derivation*. That indicates that the *input Data_Element* is a source for the *Data_Element_Derivation*. *Derivation_input* has two roles:

input and *inputter*. The *input* role refers to a *Data_Element* and the *inputter* role refers to a *Data_Element_Derivation*. An *input Data_Element* may be associated with zero or more *inputter Data_Element_Derivations*. Every *Data_Element_Derivation* shall be associated with one or more *input Data_Elements*.

9.4.3.5 derivation_output Association

derivation_output is an association between a *Data_Element* and a *Data_Element_Derivation* that indicates that the *output Data_Element* is the result of the application of a *Data_Element_Derivation*. *Derivation input* has two roles: *output* and *derivation*. The *output* role refers to a *Data_Element* and the *derivation* role refers to a *Data_Element_Derivation*. An *output Data_Element* may be associated with zero or more *derivation Data_Element_Derivations*. Every *Data_Element_Derivation* shall be associated with one or more *output Data_Elements*.

9.4.3.6 derivation_rule_application Association

derivation_rule_application is an association between a *Data_Element_Derivation* and a *Derivation_Rule* that specifies the *Derivation_Rule* that is utilized for the *Data_Element_Derivation*. *Derivation rule application* has two roles: *application* and *rule*. The *application* role refers to a *Data_Element_Derivation* and the *rule* role refers to a *Derivation_Rule*. Every *application Data_Element_Derivation* must be associated with exactly one *rule Derivation_Rule.* A *Derivation_Rule* may be associated with zero or more *application Data_Element_Derivations*.

9.4.4 Constraints in the Data_Element region

To be added.

EDITOR'S NOTE #148. (Action required) Are there any constraints to be specified?

9.5 Consolidated Data Description Metamodel

A consolidated metamodel is shown in Figure 14 — Consolidated Data Description metamodel. This combines the Data_Element_Concept, Data_Element, and Conceptual and Value_Domain regions of the model.

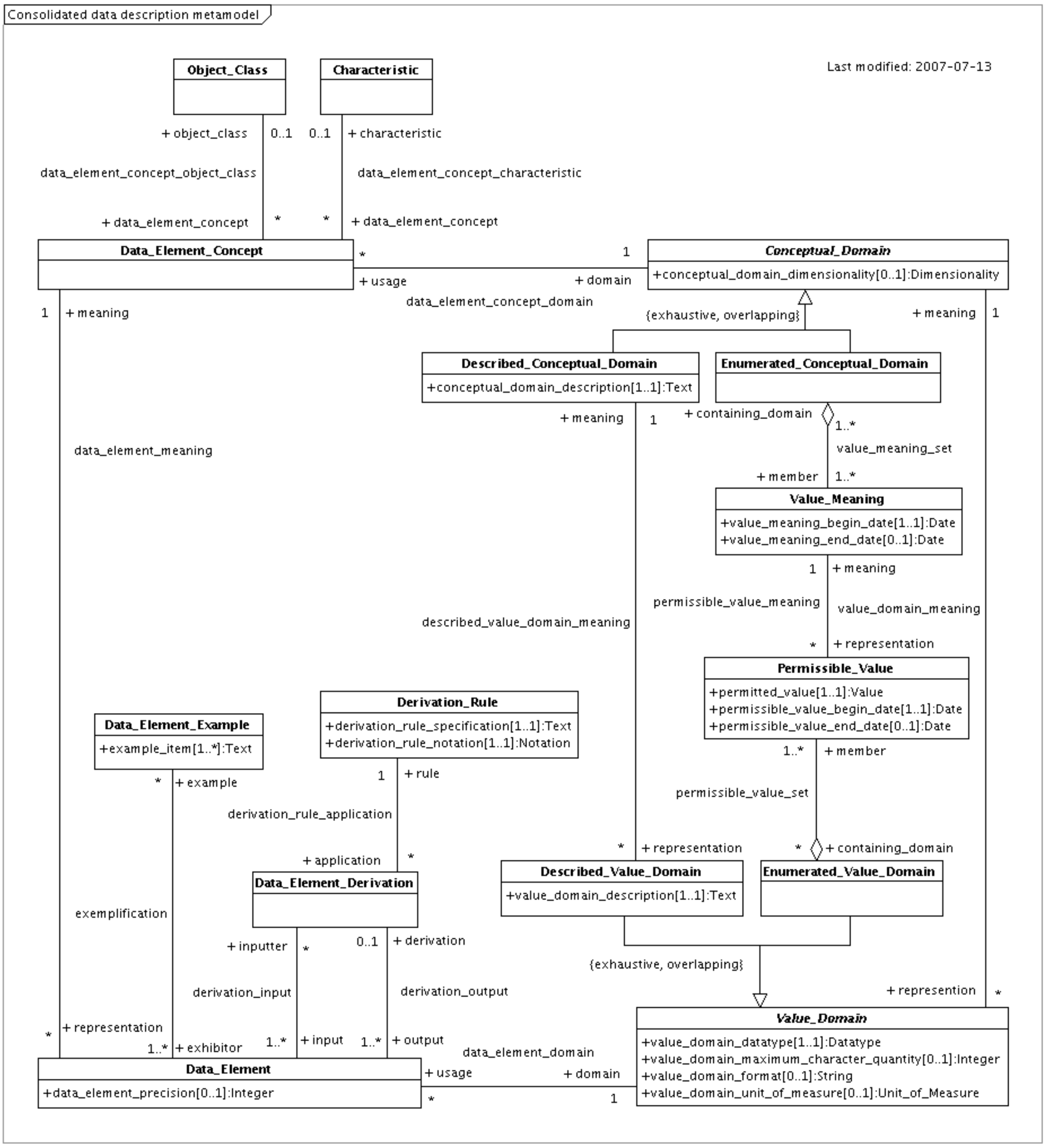

Consolidated data description metamodel
Last modified: 2007-07-13
Object_Class
Characteristic
+ object_class
0..1
0..1
+ characteristic
data_element_concept_object_class
data_element_concept_characteristic
+ data_element_concept
*
*
+ data_element_concept
Data_Element_Concept
*
+ usage
1
+ domain
data_element_concept_domain
Conceptual_Domain
+conceptual_domain_dimensionality[0..1]:Dimensionality
{exhaustive, overlapping}
1
+ meaning
+ meaning
1
Described_Conceptual_Domain
+conceptual_domain_description[1..1]:Text
Enumerated_Conceptual_Domain
+ meaning
1
+ containing_domain
1..*
value_meaning_set
data_element_meaning
+ member
1..*
Value_Meaning
+value_meaning_begin_date[1..1]:Date
+value_meaning_end_date[0..1]:Date
1
+ meaning
permissible_value_meaning
value_domain_meaning
described_value_domain_meaning
*
+ representation
Permissible_Value
+permitted_value[1..1]:Value
+permissible_value_begin_date[1..1]:Date
+permissible_value_end_date[0..1]:Date
Derivation_Rule
+derivation_rule_specification[1..1]:Text
+derivation_rule_notation[1..1]:Notation
Data_Element_Example
+example_item[1..*]:Text
1..*
+ member
*
+ example
1
+ rule
permissible_value_set
derivation_rule_application
*
+ representation
*
+ containing_domain
+ application
*
Described_Value_Domain
+value_domain_description[1..1]:Text
Enumerated_Value_Domain
Data_Element_Derivation
exemplification
+ inputter
*
0..1
+ derivation
{exhaustive, overlapping}
derivation_input
derivation_output
+ represention
*
Value_Domain
+value_domain_datatype[1..1]:Datatype
+value_domain_maximum_character_quantity[0..1]:Integer
+value_domain_format[0..1]:String
+value_domain_unit_of_measure[0..1]:Unit_of_Measure
*
+ representation
1..*
+ exhibitor
1..*
+ input
1..*
+ output
data_element_domain
Data_Element
+data_element_precision[0..1]:Integer
+ usage
*
+ domain
1

Figure 14 — Consolidated Data Description metamodel

10 Classification_Schemes, Concept_Systems and Ontologies

EDITOR'S NOTE #149. (Action required) Comments on this clause from US NB noted in the attachments to Issue 248 have not yet been applied.

10.1 Hierarchical Classification metamodel region

10.1.1 Overview

EDITOR'S NOTE #150. (Action Required) Issue 57 identifies the need to be able to classify an Administered_Item differently in different contexts. This issue has not been addressed.

EDITOR'S NOTE #151. Issue 159 proposes making *Classification_Scheme Item* (now called *Hierarchy_Node)* an Administered_Item. This CD has removed explicit sub-classing of any item type.

The Hierarchical Classification region is illustrated in Figure 15. The purpose of this region is to model *Hierarchical_Classification_Schemes*. *Hierarchical_Classification_Schemes* are intended to permit the classification of arbitrary objects into hierarchies (or partial orders), whereas *Concept_Systems* are used to enumerate and possibly classify *Concepts*.

Hierarchical_Classification_Schemes may be used to classify *Classifiable_Items* within a registry, but some *Hierarchical_Classification_Schemes* will be more applicable to classifying objects in the real world than items in a registry. Therefore, *Hierarchical_Classification_Schemes* may be specified in the registry without being used to classify registry items.

EDITOR'S NOTE #152. (Action Required) The following statement from Edition 2 is not true. ISO/IEC 11179-2 (edition 2) adds nothing new to what is specified in ISO/IEC 11179-3. We need to decide whether there is any benefit to retaining part 2.

ISO/IEC 11179-2 provides procedures and techniques for associating data with *Classification_Schemes*.

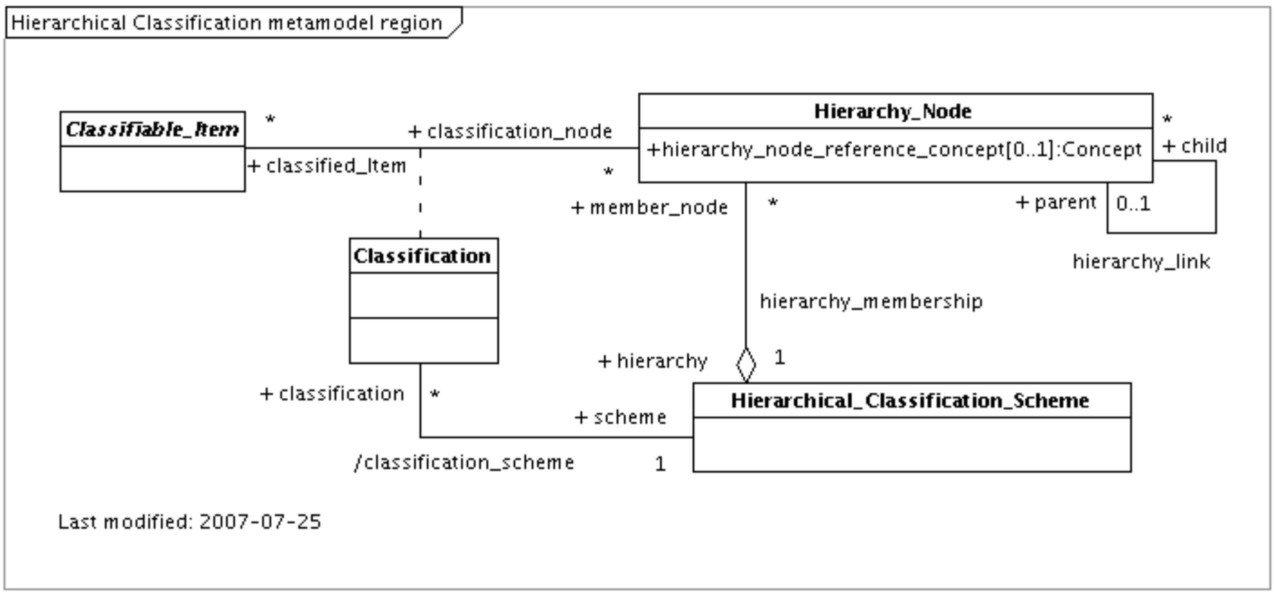


Figure 15 — Hierarchical Classification metamodel region

10.1.2 Classes in the Classification region

10.1.2.1 Classifiable_Item

Classifiable_Item is an abstract superclass of all classes which might be classified (organized into a hierarchical structure or partial order).

EDITOR'S NOTE #153. (Action required) *Classifiable_Item* is not explicitly sub-classed. We need to specify what classes are eligible for classification and how and where the sub-classing of *Classifiable_Item* is (to be) specified.

A *Classifiable_Item* may be classified in zero or more *Classification_Schemes*, by associating it with one or more *Hierarchy_Nodes* as represented by the *item classification* association in Figure 15. Such classification is optional.

10.1.2.2 Hierarchical_Classification_Scheme

A *Hierarchical_Classification_Scheme* is used to model the organization (grouping) of a collection of *Classifiable_Items* (things, individuals or concepts) into a hierarchy or partial order.

A *Hierarchical_Classification_Scheme* may participate in a *classification_scheme_node_membership* association.

A *Hierarchical_Classification_Scheme* may be a taxonomy, a network, an ontology, or any other terminological system. The classification may also be just a list of controlled vocabulary of property words (or terms). The list might be taken from the "leaf level" of a taxonomy.

EDITOR'S NOTE #154. (Action required) The following statement is no longer true, because the explicit sub-typing of Registered_Item has been removed.

A *Hierarchical_Classification_Scheme* is a sub-type of *Registered_Item*, inheriting its attributes and relationships, which allows it to be identified, named, defined and optionally classified.

A *Registered_Item* is named within a specific *Context*, and may have different *names* in different *Contexts*. As a *Registered_Item* itself, a *Hierarchical_Classification_Scheme* is also named within one or more *Contexts*. For a *Registered_Item* to be considered to have a *name* within a *Classification_Scheme*, the *Registered_Item* and the *Classification_Scheme* must share a common *Context*.

EDITOR'S NOTE #155. (Action required) Issue 57 suggests that we be able to relate the Classification_Scheme itself, not just its name, to a Context.

10.1.2.3 Hierarchy_Node

A *Hierarcy_Node* is a class used to model a single partition of *Classifiable_Items* which is homogeneous with respect some characteristic.

Note: Hierarchy_Nodes are commonly terms in a thesaurus, concepts in a taxonomy, or ontology, etc. Hierarchy_Nodes may also be Facets (property=value pairs).

Hierarcy_Node has one attribute which specifies the associated concept, if applicable.

Attribute	**Allowed Occurrences**	**Datatype**
hierarchy_node_reference_concept	*Zero or one per Hierarchy_Node*	*Concept*

Hierarcy_Node may participate in the role of parent in zero, one or more *hierarchy_link* associations with other *Hierarcy_Nodes* in the role of child. Each child *Hierarcy_Node* must be associated with only one *Hierarchy_Node* in the role of parent.

10.1.3 Associations Classes in the Classification Region

EDITOR'S NOTE #156. (Action required) Classification_Scheme_Item_Relationship association class that exists in Edition 2 has been replaced in Figure 15 by a simple association, which does not support the relationship type description that exists in Edition 2.

10.1.3.1 Classification association class

The *Classification* association is used to record the classification of a *Classifiable_Item* into a group designated by a *Hierarchy_Node* within a *Hierarchical_Classification_Scheme*.

A *Classification* association is itself associated with a *Hierarchical_Classification_Scheme* indirectly through the *Hierarchy_Node* and *hierarchy_membership* association, and directly through the derived *classification_scheme* association.

A *Classification* association has two roles: classified_item (verb form: has_classified_item) and classification_node (verb form: classified_as). The classified_item role references an instance of the *Classified_Item* class. The classification_node role references an instance of the *Hierarchy_Node* class.

The exact semantics of the *Classification* association are not specified by this standard, but will depend upon way in which the *Hierarchical_Classification_Scheme* is used. For example, if the *Hierarchy_Node* is a concept, the *Classification* association might signify either an "is-a" or an "instance-of" relationship.

A *Classifiable_Item* may be classified in zero or more Hierarchy_Nodes. A *Hierarchy_Node* may classify zero or more *Classifiable_Items*.

10.1.4 Associations in the Classification Region

10.1.4.1 hierarchy_link

The *hierarchy_link* association is used to relate a child *Hierarchy_Node* with its parent *Hierarchy_Node.*

10.1.4.2 hierarchy_membership association

The *hierarchy_membership* association is used to describe the membership (containment) of a *Hierarchy_Node* within a *Hierarchical_Classification_Scheme*. The association has two roles: hierarchy and member_node. A hierarchy (*Hierarchical_Classification_Scheme*) may have zero or more member_nodes (*Hierarchy_Nodes*). A member_node (*Hierarchy_Node*) must be contained in exactly one hierarchy (*Hierarchical_Classification_Scheme*).

10.1.4.3 classification_scheme association

The *classification_scheme* association is a derived association that records the membership (containment) of a *Classification* association class within a *Hierarchical_Classification_Scheme.*

NOTE The fact that the *classification_scheme* association is derived, is indicated in Figure 15 by the use of a '/' preceding the association name.

The association has two roles: classification and scheme. A scheme (*Hierarchical_Classification_Scheme*) may have zero or more *Classifications*. A *Classification* must be contained in exactly one scheme (*Hierarchical_Classification_Scheme*).

10.1.5 Integrity Constraints

10.1.5.1 Graph Theoretic Constraints on a Classification_Scheme

Classification schemes are constrained to be partial orders, i.e., the item_classification relation must not contain any cycles. Partial orders may be represented as directed acyclic graphs (DAGs). However, Classification_Schemes need not be restricted to a hierarchy (i.e., a tree).

The restriction of classification schemes to be partial orders is commonplace in the terminology and ontology communities.

10.2 Concept_System region

10.2.1 Overview

The Concept_System metamodel region is illustrated in Figure Figure 16.. The purpose of the Concept_System Metamodel Region is to describe Concepts (abstract units of knowledge) and the various Relations which may hold among Concepts. Ontologies (Concept_Systems with formal semantics) are described in Figure Figure 17.

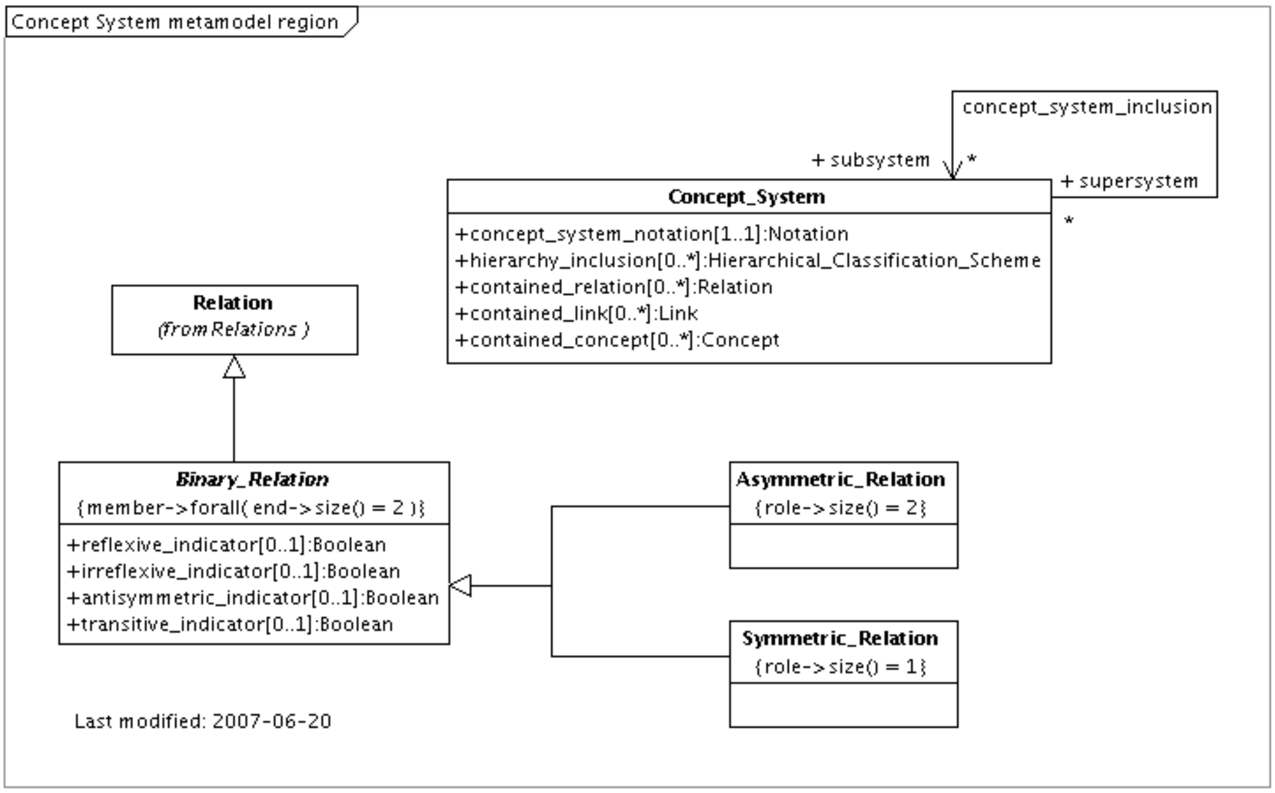

Concept System metamodel region
concept_system_inclusion
+ subsystem
*
+ supersystem
*
Concept_System
+concept_system_notation[1..1]:Notation
+hierarchy_inclusion[0..*]:Hierarchical_Classification_Scheme
+contained_relation[0..*]:Relation
+contained_link[0..*]:Link
+contained_concept[0..*]:Concept
Relation
(from Relations)
Binary_Relation
{member->forall(end->size() = 2)}
+reflexive_indicator[0..1]:Boolean
+irreflexive_indicator[0..1]:Boolean
+antisymmetric_indicator[0..1]:Boolean
+transitive_indicator[0..1]:Boolean
Asymmetric_Relation
{role->size() = 2}
Symmetric_Relation
{role->size() = 1}
Last modified: 2007-06-20

Figure 16 — Concept system metamodel region

EDITOR'S NOTE #157. (Action required) If Link in the Relations region is renamed, then *contained_link* and its datatype also need to be renamed.

10.2.2 Classes in the Concept_System region

10.2.2.1 Concept_System

A *Concept_System* is a class used to describe a domain of discourse which is independent of any particular application. A minimal Concept_System could simply be a collection of *Concepts*. A more elaborate

Concept_System could be a collection of Concepts which may be organized into a taxonomy or partonomy specified by means of various *Relations* (e.g., semantic relations) and *Links* amongst the Concepts. A much more elaborate subclass of Concept_System might be an (axiomatized) *Ontology* specified by means of predicates and axioms among the Concepts.

A *Concept_System* has exactly one *concept_system_notation* attribute of type *Notation*. The *concept_system_notation* attribute is used to record the Notation used to describe the Concept_System. Examples of such notations include XCL Common Logic (ISO 24707) or OWL-DL XML notation (Ontology Web Language from W3C).

EDITOR'S NOTE #158. (Action required) It has been suggested that **notation** should be an attribute of Ontology, instead of Concept_System.

A *Concept_System* may include zero, one or more *Hierarchies*, as specified by the *hierarchy_inclusion* attribute.

A *Concept_System* may include zero, one or more *Relations*, as specified by the *contained_relation* attribute.

A *Concept_System* may include zero, one or more *Links*, as specified by the *contained_link* attribute.

A *Concept_System* may include zero, one or more *Concepts*, as specified by the *contained_concept* attribute.

Attribute	**Allowed Occurrences**	**Datatype**
concept_system_notation	one per *Concept_System*	*Notation*
hierarchy_inclusion	zero, one or many per *Concept_System*	*Hierarchy*
contained_relation	zero, one or many per *Concept_System*	*Relation*
contained_link	zero, one or many per *Concept_System*	*Link*
contained_concept	zero, one or many per *Concept_System*	*Concept*

A *Concept_System* may participate in the *concept_system_inclusion* association, by which zero, one or more *subsystem Concept_Systems* may be included in a *supersystem Concept_System*.

10.2.2.2 Binary_Relation

Binary_Relation is a class which models Relations of arity 2 (having 2 link ends).

Most common semantic relations are binary, e.g., equals, less than, greater than, is-a, part-of, etc. A example of a relation which is not binary would be betweeness. Binary relations are commonly represented as edges (or directed edges for asymmetric binary relations) in graphs, cf. the RDF (Resource Description Framework) of the W3C.

Below is a table of examples of some binary relationships and their characterization.

Relation	**Asymmetric / Symmetric / Antisymmetric**	**Reflexive / Irreflexive**	**Transitive**
equals	symmetric	reflexive	transitive
not	symmetric	irreflexive	Not

Relation	Asymmetric / Symmetric / Antisymmetric	Reflexive / Irreflexive	Transitive
equals			transitive
less than	antisymmetric	irreflexive	Transitive
less than or equal	asymmetric	reflexive	Transitive
similar	symmetric	reflexive	Not transitive

The Binary_Relation class has four indicator attributes: reflexive, irreflexive, antisymmetric, and transitive.

Attribute	**Allowed Occurrences**	**Datatype**
reflexive_indicator	Zero or one per Binary_Relation.	*Boolean*
irreflexive_indicator	Zero or one per Binary_Relation	*Boolean*
antisymmetric_indicator	Zero or one per Binary_Relation	*Boolean*
transitive_indicator	Zero or one per Binary_Relation	*Boolean*

A Binary_Relation, R, is reflexive if for all x, R(x,x) is true. Equality is an example of a reflexive relation.

A Binary_Relation, R, is antireflexive if for all x, R(x,x) is false. Inequality is an example of an antireflexive relation.

A Binary_Relation, R, is anti-symmetric if for all x,y: R(x,y) implies not R(y,x). Less than is an example of an anti-symmetric relation. Note that an asymmetric relation is not necessarily anti-symmetric (consider less than or equals).

A Binary_Relation, R, is transitive, if for all x,y,z: R(x,y) and R(y,z) implies R(x,z). Examples of transitive relations include equality, less than, and less than or equals.

10.2.2.3 Symmetric_Relation

A Symmetric_Relation is a Binary_Relation, R, such that for all x, y: R(x,y) implies R(y,x). Examples of symmetric relations are equals, not equals, within-2-miles-of, etc. A Symmetric_Relation is a subclass of Relation.

In terms of this metamodel, a Symmetric_Relation has one Role which is used for both (two) Ends of each Link.

Note that symmetry does not imply reflexivity. Thus the inequality relation is symmetric, but irreflexive.

10.2.2.4 Asymmetric Relation

The Asymmetric Relation class is a subclass of a Binary_Relation, R, such that for all x,y: R(x,y) does not imply R(y,x).

In terms of this metamodel, Asymmetric Relations have two distinguishable (non-identical) roles, one for each End of each Link.

Examples of asymmetric relations include: less than, likes, father of, etc.

10.2.3 Associations of the Concept_System region

10.2.3.1 concept_system_inclusion Association

The *concept_system_inclusion association* describes the inclusion relations (i.e., part-of relations) among Concept_Systems. It is used to enable the modular construction of Concept_Systems.

The concept_system_inclusion association has two roles: subsystem (verb form: has_subsystem) and supersystem (verb form: has_supersystem). Both roles reference instances of the class Concept_System. A subsystem may be included in zero or more supersystems. A supersystem may include zero or more subsystems.

10.3 Ontology region

10.3.1 Overview

The Ontology Metamodel Region is illustrated in Figure 17 — Ontology metamodel region. The purpose of the Ontology Metamodel Region is to maintain information on *Ontologies*, a subclass of *Concept_Systems*. The metadata objects in this region are concerned with formal semantics. The metadata objects in this region are *Concept_Systems*, *Ontologies*, *Ontology_Entries*, and *Assertions*.

EDITOR'S NOTE #159. (Action required) The figure has been significantly changed from that included in WD4, and no accompanying text has been provided to justify the change. This region of the model appears to be very unstable. Is it really ready for inclusion in this standard?

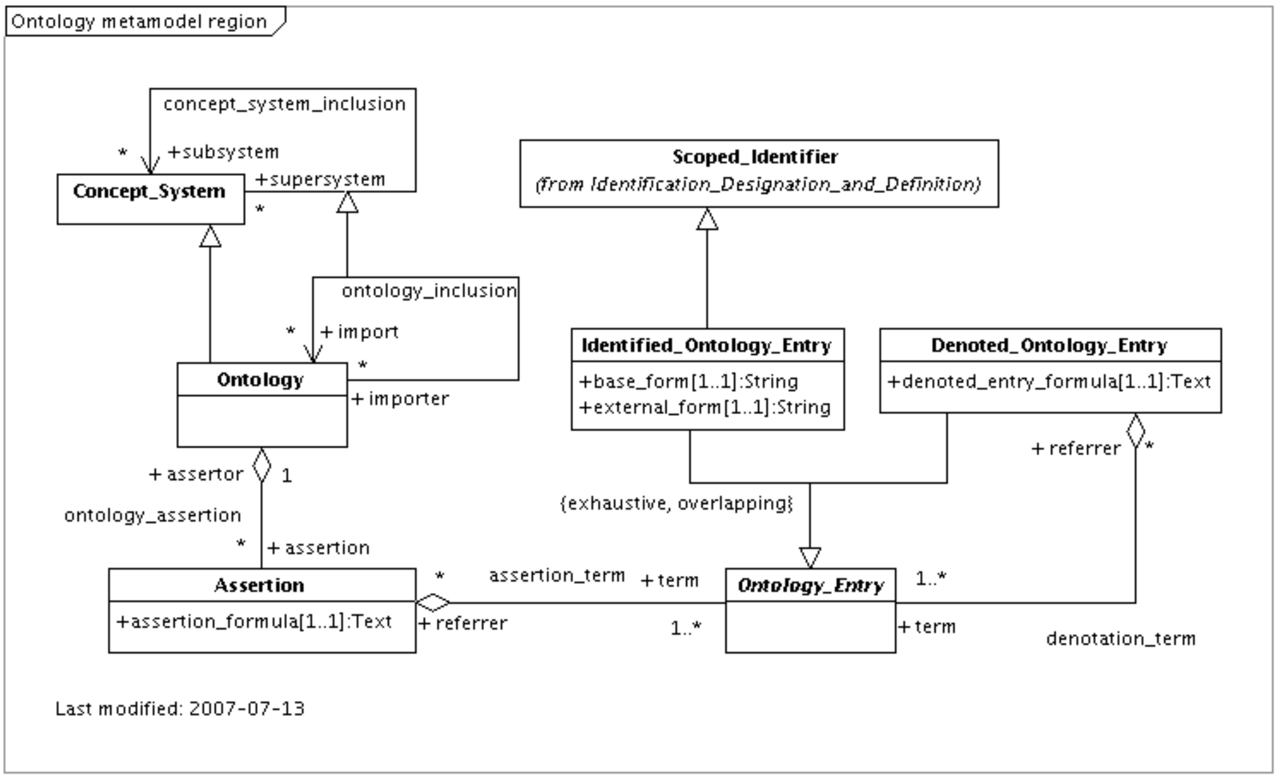


Figure 17 — Ontology metamodel region

10.3.2 Classes in the Ontology region

10.3.2.1 Concept_System

Concept_System is a class used to model collections of concepts, which may be organized into taxonomies, partonomies, ontologies, etc. Concept systems are intended to be more general than ontologies and may lack the formal axiomatization common to ontologies. Concept systems may be organized by means of semantic relations, or via collections of axioms (e.g., within a description logic).

The Concept_System class is a superclass of the Ontology class.

A Concept_System class may participate in the concept_system_inclusion association.

Concept_System is further described in **Error! Reference source not found.**.

10.3.2.2 Ontology

An ontology is a formal specification of the semantics for some domain of discourse.

EDITOR'S NOTE #160. (Action required) It has been suggested that the definition of ontology needs further work.

The Ontology class is a subclass of the Concept_System class.

A Ontology class may participate in the following associations: ontology_inclusion association, predicate_definition association, and the axiom_assertion association.

10.3.2.3 Assertion

EDITOR'S NOTE #161. (Action required) The editor requests text to complete this sub-clause.

10.3.2.4 Ontology_Entry

EDITOR'S NOTE #162. (Action required) The editor requests text to complete this sub-clause.

10.3.2.5 Identified_Ontology_Entry

EDITOR'S NOTE #163. (Action required) The editor requests text to complete this sub-clause.

10.3.2.6 Denoted_Ontology_Entry

EDITOR'S NOTE #164. (Action required) The editor requests text to complete this sub-clause.

10.3.2.7 Scope_Identifier

Scoped_Identifier is described in 6.1.2.4.

10.3.3 Associations in the Ontology region

10.3.3.1 assertion_term association

EDITOR'S NOTE #165. (Action required) The editor requests text to complete this sub-clause.

10.3.3.2 concept_system_inclusion association

The concept_system_inclusion association enables the modelling of inclusion relations (i.e., part-of relations) among Concept_Systems. It is used to enable the modular construction of Concept_Systems.

The concept_system_inclusion association has two roles: subsystem (verb form: has_subsystem) and supersystem (verb form: has_supersystem). Both roles reference instances of the class Concept_System. A subsystem may be included in zero or more supersystems. A supersystem may include zero or more subsystems.

The concept_system_inclusion association is a generalization of the ontology_inclusion association.

10.3.3.3 denotation_term association

EDITOR'S NOTE #166. (Action required) The editor requests text to complete this sub-clause.

10.3.3.4 ontology_assertion association

EDITOR'S NOTE #167. (Action required) The editor requests text to complete this sub-clause.

10.3.3.5 ontology_inclusion association

The ontology_inclusion association enables the modelling of inclusion relations (i.e., part-of relations) among Ontologies. It is used to enable the modular construction of Ontologies.

The concept_system_inclusion association has two roles: import (verb form: imported) and importer (verb form: importing). Both roles reference instances of the class Ontology. An importer ontology may import zero or more ontologies. An imported Ontology may be imported by zero or more Ontologies.

The ontology_inclusion association is a specialization of the concept_system_inclusion association.

11 Basic attributes

EDITOR'S NOTE #168. (Action required) This clause has been carried over from Edition 2 without significant modification. Changes will be required to reflect the changes in the model.

11.1 Use of basic attributes

This Clause is intended to provide continuity from ISO/IEC 11179-3:1994, which edition focused on basic attributes of data elements. However, the scope of this Clause extends beyond just data elements, to include: data element concepts, conceptual domains, value domains, permissible values and value meanings.

EDITOR'S NOTE #169. (Action required) The following sentence will need to be revised to reflect the changes made to the mappings among editions of the standard.

A mapping among the 1994 basic attributes, the 2002 basic attributes and the 2002 metamodel can be found in Annex C.

Clauses 5 through 10 describe a model for specifying metadata in a registry. However, sometimes the requirement for metadata specification exists outside the context of a registry, for example as part of an International Standard.

A specification of metadata consists of a set of attributes, and relationships among those attributes. This Clause specifies a set of *basic* attributes to be used in contexts other than a metadata registry. *Basic* means that they are frequently needed to specify a metadata item. The attributes specified in this Clause are also considered *basic* in the sense that additional attributes may be required when the metadata items are used in a particular context.

Basic does not imply that all standardized attributes presented in this Clause are required in all cases. Distinction is made between those basic attributes that are:

— mandatory: always required;

— conditional: required to be present under certain specified conditions;

— optional: permitted but not required.

NOTE The obligations specified for some basic attributes (especially identifiers) in contexts other than a registry are different from those specified for metadata items in a registry, as defined in Clause 4.

11.2 Common attributes

The attributes listed in this subclause are common to all types of Administered_Item. These attributes are further categorized as: Identifying, Definitional, Administrative, and Relational.

11.2.1 Identifying

EDITOR'S NOTE #170. (Action required) We should probably distinguish Identification from Designation, as we do in Clause 4.

Attribute	**Allowed Occurrences**
name	One or more per metadata item *(see note 1).*
context name	Zero or more per metadata item. Required if more than one *name* attribute exists.
context identifier	Zero or one per metadata item. Required if *context name* is not unique within its usage context (e.g. a standard).
context description	One per *context name*.
item identifier	Zero or one per metadata item. Required if *name* is not unique within a given *context (see note 2).*
item identifier – data identifier	One per *item identifier*. (The mandatory portion of an *item identifier*.)
item identifier – item registration authority identifier	Zero or one per *item identifier*. (The optional portion of an *item identifier* - *see note 3.)*
version	Zero or one per metadata item *(see note 4).*

NOTE 1 If more than one *name* is specified within a given *context*, it is usual nominate one name as "preferred", and the others as "synonyms".

NOTE 2 While *item identifier* is mandatory within a registry (see 4.8.1.4), it is only conditional in non-registry usages. The requirement for an *item identifier* can be eliminated by qualifying *name* and/or *context name* to ensure that the combination is unique.

NOTE 3 While *item registration authority identifier* is mandatory within a registry (see 4.8.1.4), it is optional in nonregistry settings.

NOTE 4 Within a registry, *version* is part of an *item identifier*. In non-registry settings, *version* may be used independently of *item identifier*.

11.2.2 Definitional

Attribute	**Allowed Occurrences**
definition	One for each *context* in which the metadata item is used *(see note 1).*
definition_language_identifier	Zero or one per *definition.*
definition_source_reference	Zero or one per *definition.*

NOTE Where multiple *definitions* are assigned to the same metadata item, the semantics of the *definition* should be the same across all *contexts*. (If the semantics are different, separate metadata items should be specified.) However, the terminology used to express the semantics may need to be different in different *contexts*, and thus separate *definitions* are permitted for each *context*.

11.2.3 Administrative

Administrative attributes are primarily associated with recording metadata items in a registry. They are therefore optional in non-registry settings.

Attribute	**Allowed Occurrences**
comments	Zero or one per metadata item.
registration_status	Zero or one per metadata item.
responsible organization name	Zero or one per metadata item.
submitting organization name	Zero or one per metadata item.

11.2.4 Relational

Attribute	**Allowed Occurrences**
classification scheme name	One for each *classification scheme* in which a metadata item is classified.
classification scheme identifier	Zero or one per *classification scheme name*. Required if *classification scheme name* is not unique within a *context*.
classification scheme type name	One for each *classification scheme* in which a metadata item is classified.
classification scheme item type name	Zero or one for each *classification scheme* in which a metadata item is classified (see note 1).
classification scheme item value	One for each *classification scheme item* by which a metadata item is classified.
related metadata reference	Zero or more per metadata item (see note 2).
type of relationship	One per *related metadata reference*.

NOTE 1 The metamodel in 0 treats *keywords* as a type of *classification scheme*.

NOTE 2 A *Registration_Authority* could choose to use a *Reference_Document*, an *administrative_note* or an *explanatory_comment* to record a *related metadata reference*.

11.3 Attributes specific to Data_Element_Concepts

The attributes listed in this subclause are specific to Data_Element_Concepts.

Attribute	**Allowed Occurrences**
object class name	One per *data element concept.*
object class identifier	Zero or one per *object class name.*
property name	One per *data element concept.*
property identifier	Zero or one per *property name.*

11.4 Attributes specific to Data_Elements

The attributes listed in this subclause are specific to Data_Elements.

EDITOR'S NOTE #171. (Action required) Issue 114 has removed *Representation_Class.* We need some text to explain the use of a *Classification_Scheme* instead.

Attribute	**Allowed Occurrences**
Value domain name	Zero or one per *data element.*
Value domain identifier	Zero or one per *data element.*
Datatype name	Zero or one per *data element.* Required if neither *value domain name* nor *value domain identifier* is not specified.
Datatype scheme reference	Zero or one per *datatype_name.*
Layout of representation	Zero or one per *data element.*
~~*Representation class*~~	~~Zero or one per *data element.*~~
Maximum size	Zero or one per *data element.*
Minimum size	Zero or one per *data element.*

11.5 Attributes specific to Conceptual_Domains

The attributes listed in this subclause are specific to Conceptual_Domains.

Attribute	**Allowed Occurrences**
dimensionality	Zero or one per *conceptual domain.*

11.6 Attributes specific to Value_Domains

The attributes listed in this subclause are specific to Value_Domains.

Attribute	**Allowed Occurrences**
datatype_name	One per *value domain.*
datatype_scheme_reference	Zero or one per *datatype_name.*
unit of measure name	Zero or one per *value domain.*

11.7 Attributes specific to Permissible_Values

The attributes listed in this subclause are specific to Permissible_Values.

Attribute	Allowed Occurrences
value	One per *permissible value.*
permissible_value_begin_date	Zero or one per *permissible value.*
permissible_value_end_date	Zero or one per *permissible value.*

11.8 Attributes specific to Value_Meanings

The attributes listed in this subclause are specific to Value_Meanings.

Attribute	Allowed Occurrences
value meaning description	One per *value meaning.*
value meaning identifier	Zero or one per *value meaning.*
value_meaning_begin_date	Zero or one per *value meaning.*
value_meaning_end_date	Zero or one per *value meaning.*

12 Conformance

This part of ISO/IEC 11179 prescribes a conceptual model, not a physical implementation. Therefore, the metamodel need not be physically implemented exactly as specified. However, it must be possible to unambiguously map between the implementation and the metamodel in both directions.

This part of ISO/IEC 11179 also prescribes a list of basic attributes for situation where a full conceptual model is not required or not appropriate.

Conformance may be claimed to either the conceptual model, or the basic attributes or both; see 5.2. Conformance claims shall specify a Degree and a Level of Conformance, as described below.

12.1 Degree of Conformance

The distinction between “strictly conforming” and “conforming” implementations is necessary to address the simultaneous needs for interoperability and extensions. This part of ISO/IEC 11179 describes specifications that promote interoperability. Extensions are motivated by needs of users, vendors, institutions, and industries, and:

a) are not directly specified by this part of ISO/IEC 11179,

b) are specified and agreed to outside this part of ISO/IEC 11179, and

c) may serve as trial usage for future editions of this part of ISO/IEC 11179.

A strictly conforming implementation may be limited in usefulness but is maximally interoperable with respect to this part of ISO/IEC 11179. A conforming implementation may be more useful, but may be less interoperable with respect to this part of ISO/IEC 11179.

12.1.1 Strictly conforming implementations

A strictly conforming implementation:

a) shall support all mandatory, optional and conditional data element attributes and relationships;

b) shall not use, test, access, or probe for any extension features nor extensions to data element attributes;

c) shall not recognize, nor act on, nor allow the production of data element attributes that are dependent on any unspecified, undefined, or implementation-defined behavior.

NOTE The use of extensions to the metamodel or the basic attributes may cause undefined behavior.

12.1.2 Conforming implementations

A conforming implementation:

a) shall support all mandatory, optional and conditional data element attributes and relationships;

b) as permitted by the implementation, may use, test, access, or probe for extension features or extensions to data element attributes;

c) may recognize, act on, or allow the production of data element attributes that are dependent on implementation-defined behavior.

NOTE 1 All strictly conforming implementations are also conforming implementations.

NOTE 2 The use of extensions to the metamodel or the basic attributes may cause undefined behavior.

12.2 Levels of Conformance

EDITOR'S NOTE #172. (Action required) We need to agree the levels of conformance we should provide in Edition 3. The Editor has included a 'straw' proposal for consideration.

12.2.1 Overview

This standard specifies multiple conformance levels for this Edition, and additionally lists conformance levels that approximate the levels found in prior Editions of the standard. An implementation may conform to one or more of these conformance levels.

Conformance Level	**Edition 3**	**Edition 2 Level 2**	**Edition 2 Level 1**	**Edition 1**
Basic package	Mandatory	Mandatory	N/a	N/a
Identification, Designation and Definition package	Mandatory	Mandatory	N/a	N/a
Registration package	Mandatory	Mandatory	N/a	N/a
Relations package	Optional	Mandatory	N/a	N/a
Data Description package	Optional	Mandatory	N/a	N/a
Classification schemes package	-	-	-	-
- Hierarchical classification schemes	Optional	Mandatory	N/a	N/a
- Concept systems	Optional	N/a	N/a	N/a
- Ontologies	Optional	N/a	N/a	N/a
Basic attributes	N/a	N/a	Mandatory	Mandatory

Conformance to the Relations package is shown as Optional on its own, but it is a prerequisite to any of the other Optional packages.

12.2.2 Conformance Levels for Edition 3

In this Edition of this standard, conformance may be claimed to some packages and not to others.

Conformance is mandatory to all metadata classes, associations and attributes specified in Clause 5 (Basic Package), Clause 6 (Identification, Designation and Definition Package) and Clause 7 (Registration Package).

Conformance is optional to Clause 8 (Relations Package) on its own, but support for this package is a pre-requisite for the following optional packages. If this package is supported, all metadata classes, associations and attributes specified in the package must be supported.

Conformance is optional to Clause 9 (Data Description Package). If this package is supported, all metadata classes, associations and attributes specified in the package must be supported.

Conformance is optional to Clause 10.1 (Hierarchical Classification metamodel region). If this package is supported, all metadata classes, associations and attributes specified in the package must be supported.

Conformance is optional to Clause 10.2 (Concept_System region). If this package is supported, all metadata classes, associations and attributes specified in the package must be supported.

Conformance is optional to Clause 10.3 (Ontology region). If this package is supported, all metadata classes, associations and attributes specified in the package must be supported.

12.2.3 Conformance Levels for Edition 2 Level 2

Metadata elements, relationships and properties specified in subclause 10.2 (Concept systems) and subclause 10.3 (Ontology) are **not supported**. All other metadata elements, associations and attributes specified in this standards are supported and may be used.

12.2.4 Conformance Levels for Edition 2 Level 1 and Edition 1

Only those metadata elements, relationships and properties specified in Clause 11 are supported and used.

12.3 Obligation

Properties and relationships specified in this part of ISO/IEC 11179 are stated to be Mandatory, Conditional or Optional.

For the purpose of conformance:

a) Mandatory properties and relationships shall exist, and shall conform to the provisions of this part of ISO/IEC 11179.

b) Anything specified as Conditional within this part of ISO/IEC 11179 shall be treated as Mandatory if the associated condition is satisfied, and shall otherwise be not present.

c) Optional properties and relationships are not required to exist, but if they do exist they shall conform to the provisions of this part of ISO/IEC 11179.

Such obligation is enforced if and only if the Registration Status of the associated metadata items is Recorded or higher.

12.4 Implementation Conformance Statement (ICS)

An implementation claiming conformance to this part of ISO/IEC 11179 shall include an Implementation Conformance Statement stating:

a) whether it conforms or strictly conforms (12.1 Degree of Conformance);

b) which packages/regions are supported (12.2 Levels of Conformance);

c) what extensions are supported or used.

12.5 Roles and Responsibilities for Registration

Conformance needs to be considered in the context of the roles and responsibilities of registration authorities, as covered by ISO/IEC 11179-6: Registration of data elements.

Extended conformance of systems requires formalisation of procedures, agreement of roles and responsibilities between parties, and guidelines addressing use of software products and conversions from other systems. The formalisation of these aspects must be consistent with the conformance requirements in the above Clauses, and roles of registration authorities as set out in ISO/IEC 11179-6.

Annex A
(informative)

Alphabetical List of Terms

Term	Defined in
Administered_Item	3.3.1
Administration_Record	3.3.2
administrative_note	3.3.3
administrative_status	3.3.4
antisymmetric indicator	3.3.5
Assertion	3.3.6
assertion_formula	3.3.7
assertion_term	3.3.8
association	3.1.1
association class	3.1.2
Asymmetric Relation	3.3.9
Attached_Item	3.3.10
attachment	3.3.11
attribute	3.1.3
attribute instance	3.2.1
attribute value	3.2.2
authority rule	3.3.12
base_form	3.3.13
basic attribute	3.2.3
Binary_Relation	3.3.14
binding	3.2.4
Boolean	3.3.15
CD	3.3.29, 3.4.1
change description	3.3.16
characteristic	3.2.5
Characteristic	3.3.17
class	3.1.4
Classifiable_Item	3.3.18
Classification	3.3.19
common attribute	3.2.6
composite attribute	3.1.5
composite datatype	3.1.6
Concept	3.3.20
Concept_System	3.3.21
concept_system_inclusion	3.3.22
concept_system_notation	3.3.23
conceptual data model	3.2.7
Conceptual_Domain (CD)	3.3.24
conceptual_domain_dimensionality	3.3.25
conditional	3.2.8
Contact	3.3.26
contact_individual	3.3.27
contact_mail_address	3.3.28
contact_organization	3.3.29
contact_phone	3.3.30
contact_title	3.3.31
contained_concept	3.3.32
contained_link	3.3.33
contained_relationed	3.3.34
Context	3.3.35
coordinate	3.2.9
coordinate_indicator	3.3.36
creation_date	3.3.37
data	3.2.10
Data_Element (DE)	3.3.38
Data_Element_Concept (DEC)	3.3.39
data_element_concept_characteristic	3.3.40
data_element_concept_domain	3.3.41
data_element_concept_object_class	3.3.42
Data_Element_Derivation	3.3.43
data_element_domain	3.3.44
Data_Element_Example	3.3.45
data_element_meaning	3.3.46
data model	3.2.11
datatype	3.1.7
Datatype	3.3.48
datatype_annotation	3.3.49
datatype_description	3.3.50
datatype_name	3.3.51
datatype_scheme_reference	3.3.52
Date	3.3.53
DE	3.3.38, 3.4.2
DEC	3.3.37, 3.4.3

Term	Defined in
definition	3.2.12
Definition	3.3.54
Definition_Context	3.3.55
definition_language	3.3.56
definition_source_reference	3.3.57
definition_text	3.3.58
derivation_input	3.3.59
derivation_output	3.3.60
Derivation_Rule	3.3.61
derivation_rule_application	3.3.62
derivation_rule_notation	3.3.63
derivation_rule_specification	3.3.64
Described Conceptual_Domain	3.3.65
described_conceptual_domain_description	3.3.66
Described_Value_Domain	3.3.67
described_value_domain_description	3.3.68
described_value_domain_meaning	3.3.69
designation	3.2.13
Designatable_Item	3.3.70
Designation	3.3.71
Designation_Context	3.3.72
designation_language	3.3.73
designation_sign	3.3.74
Designation_Space	3.3.75
designation_space_membership	3.3.76
Dimensionality	3.3.77
documentation_language_identifier	3.3.78
effective_date	3.3.79
entity	3.2.14
Enumerated_Conceptual_Domain	3.3.80
Enumerated_Value_Domain	3.3.81
example_item	3.3.82
exemplification	3.3.83
explanatory_comment	3.3.84
extension	3.2.15
extension_identifier	3.3.85
external_form	3.3.86
generalization	3.1.8
Hierarchical_Classification_Scheme	3.3.87
hierarchy_link	3.3.88
hierarchy_membership	3.3.89
Hierarchy_Node	3.3.90
hierarchy_node_reference_concept	3.3.91
identification	3.3.92
Identified_Item	3.3.93
identifier <Identified_Item>	3.3.94
identifier (in Metadata Registry)	3.1.9
Identifier_Space	3.3.95
Individual	3.3.96
Integer	3.3.97
international_code_designator	3.3.98
item_definition	3.3.99
item_designation	3.3.100
item_slot	3.3.101
language	3.2.16
Language_Identification	3.3.102
language_identifier	3.3.103
last_change_date	3.3.104
lexical_rule	3.3.105
Link	3.3.106
Link_End	3.3.107
link_end_role	3.3.108
mail_address	3.3.109
management	3.3.110
mandatory	3.2.17
mandatory_naming_convention_indicator	3.3.111
max_cardinality	3.3.112
MDR	3.2.22, 3.4.4
metadata	3.2.18
metadata item	3.2.19
metadata object	3.2.20
metadata register	3.2.21
Metadata Registry (MDR)	3.2.22
metadata set	3.2.23
metamodel	3.2.24
metamodel construct	3.2.25
min_cardinality	3.3.113
name	3.2.26
name (attribute of *Individual*)	3.3.114
name (attribute of *Organization*)	3.3.115
Namespace	3.3.116
Naming_Convention	3.3.117

Term	Defined in
naming_convention_conformance	3.3.118
naming_convention_utilization	3.3.119
Notation	3.3.120
object	3.2.27
Object_Class	3.3.121
one_item_per_name_indicator	3.3.122
one_name_per_item_indicator	3.3.123
Ontology	3.3.124
ontology_assertion	3.3.125
Ontology_Entry	3.3.126
ontology_inclusion	3.3.127
opi	3.3.130, 3.4.5
optional	3.2.28
Organization	3.3.128
organization_identifier	3.3.129
organization part	3.2.29
organization_part_identifier (opi)	3.3.130
organization_part_identifier_source	3.3.131
origin <Administered_Item>	3.3.132
Permissible_Value	3.3.133
permissible_value_begin_date	3.3.134
permissible_value_end_date	3.3.135
permissible_value_meaning	3.3.136
permissible_value_set	3.3.137
permitted_value	3.3.138
Phone_Number	3.3.139
Postal_Address	3.3.140
preferred_definition_indicator	3.3.141
preferred_designation_indicator	3.3.142
primitive datatype	3.1.10
quantity	3.2.30
RA	3.3.121, 3.4.6
ra_identifier	3.3.143
RDF	3.4.7
record	3.3.144
Reference	3.3.145
Reference_Document	3.3.146
reference_document_identifier	3.3.147
reference_document_language_identifier	3.3.148
reference_document_title	3.3.149
reference_document_type_description	3.3.150
reference_provider	3.3.151
reference_type	3.3.152
reflexive_indicator	3.3.153
region_identifier	3.3.154
Register	3.3.155
Registered_Item	3.3.156
Registrar	3.3.157
registrar_identifier	3.3.158
registration	3.2.31
Registration	3.3.159
Registration_Authority (RA)	3.3.160
Registration_Authority_Identifier	3.3.161
registration_authority_registrar	3.3.162
registration_status	3.3.163
registry item	3.2.32
registry metamodel	3.2.33
related metadata reference	3.2.34
Relation	3.3.164
relation_membership	3.3.165
Relation_Role	3.3.166
relationship (in registry metamodel)	3.1.11
scope_rule	3.3.167
Scoped_Identifier	3.3.168
script_identifier	3.3.169
semantic_rule	3.3.170
Sign	3.3.171
Slot	3.3.172
slot_name	3.3.173
slot_type	3.3.174
slot_value	3.3.175
stewardship (of Administered_Item)	3.3.176
stewardship (of metadata)	3.2.35
stewardship_contact	3.3.177
Stewardship_Record	3.3.178
String	3.3.179
submission (of Registered_Item)	3.3.180
submission_contact	3.3.181
Submission_Record	3.3.182
Symmetric_Relation	3.3.183
syntactic_rule	3.3.184
term_definition_pairing	3.3.185

Term	Defined in
Text	3.3.186
transitive_indicator	3.3.187
UML	3.4.8
Unit_of_Measure <Value_Domain>	3.3.188
unit_of_measure_dimensionality	3.3.189
UnlimitedNatural	3.3.190
unresolved_issue	3.3.191
until_date	3.3.192
Value	3.3.193
Value_Domain (VD)	3.3.194
value_domain_datatype	3.3.195
value_domain_format	3.3.196
value_domain_maximum_character_quantity	3.3.197
value_domain_meaning	3.3.198
value_domain_unit_of_measure	3.3.199
Value_Meaning	3.3.200
value_meaning_begin_date	3.3.201
value_meaning_end_date	3.3.202
value_meaning_set	3.3.203
variant_identifier	3.3.204
VD	3.3.194, 3.4.9
version	3.3.205
W3C	3.4.10
XML	3.4.11

Annex B
(informative)

Consolidated Class Hierarchy

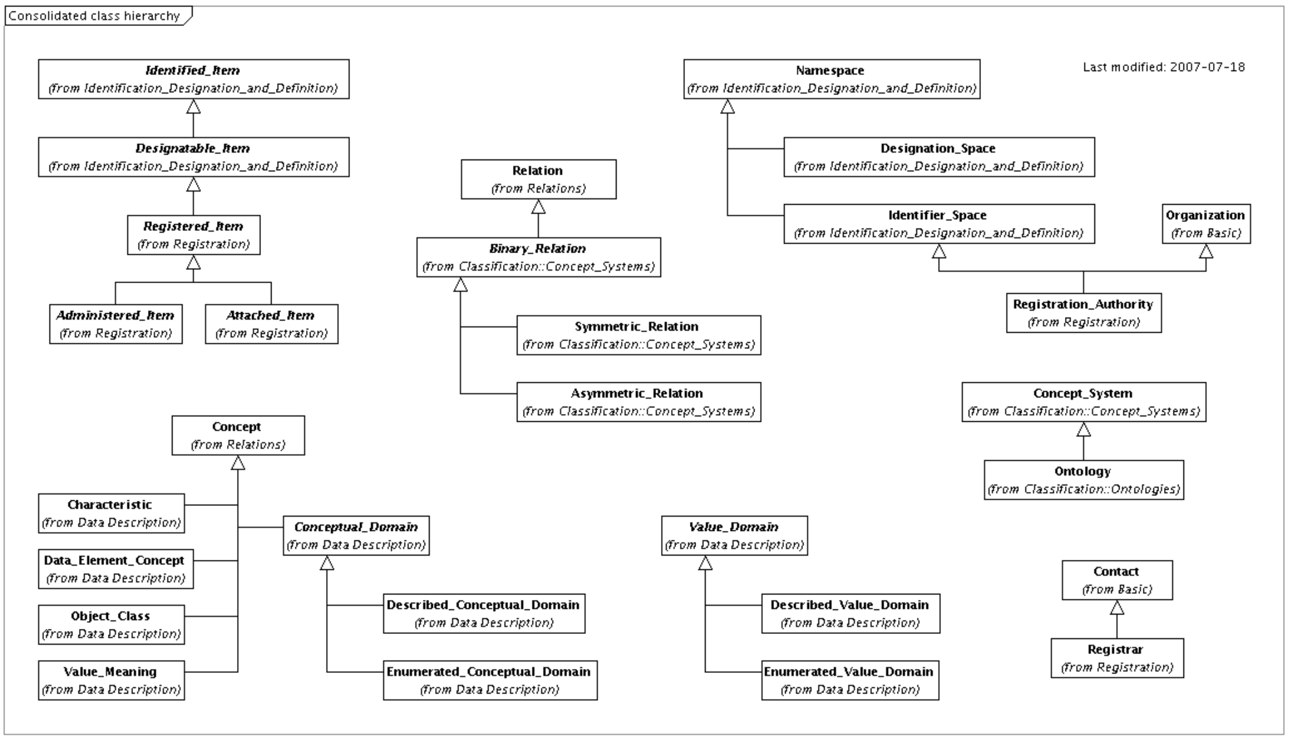

Consolidated class hierarchy
Last modified: 2007-07-18
Identified_Item (from Identification_Designation_and_Definition)
Designatable_Item (from Identification_Designation_and_Definition)
Registered_Item (from Registration)
Administered_Item (from Registration)
Attached_Item (from Registration)
Relation (from Relations)
Binary_Relation (from Classification::Concept_Systems)
Symmetric_Relation (from Classification::Concept_Systems)
Asymmetric_Relation (from Classification::Concept_Systems)
Namespace (from Identification_Designation_and_Definition)
Designation_Space (from Identification_Designation_and_Definition)
Identifier_Space (from Identification_Designation_and_Definition)
Organization (from Basic)
Registration_Authority (from Registration)
Concept_System (from Classification::Concept_Systems)
Ontology (from Classification::Ontologies)
Concept (from Relations)
Characteristic (from Data Description)
Data_Element_Concept (from Data Description)
Object_Class (from Data Description)
Value_Meaning (from Data Description)
Conceptual_Domain (from Data Description)
Described_Conceptual_Domain (from Data Description)
Enumerated_Conceptual_Domain (from Data Description)
Value_Domain (from Data Description)
Described_Value_Domain (from Data Description)
Enumerated_Value_Domain (from Data Description)
Contact (from Basic)
Registrar (from Registration)

Figure B.1 — Consolidated Class Hierarchy

Annex C
(informative)

Types of Registered_Items

EDITOR'S NOTE #173. (Action required) We need to explain how metadata items become Registered_Items. Figure C.1 shows what is currently defined in this CD. Figure C.2 shows what was defined in WD4. Figure C.2 would need to be modified to reflect other changes in this CD, but we need something like it in order to know what metadata we can expect any item to have.

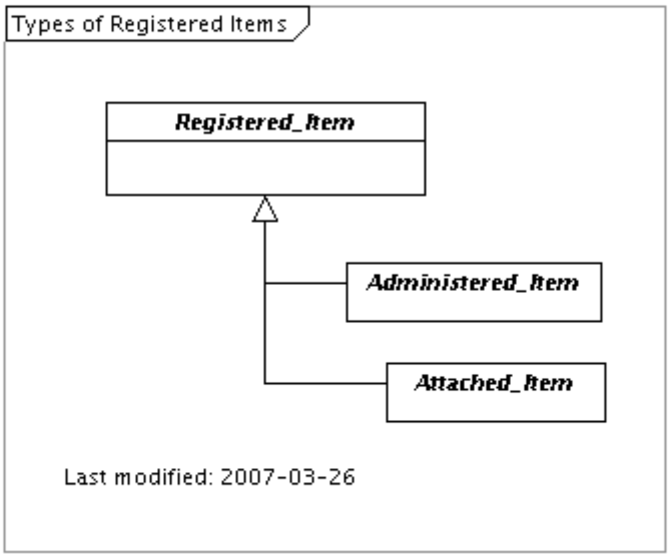


Figure C.1 — Types of Registered_Items as defined by this standard

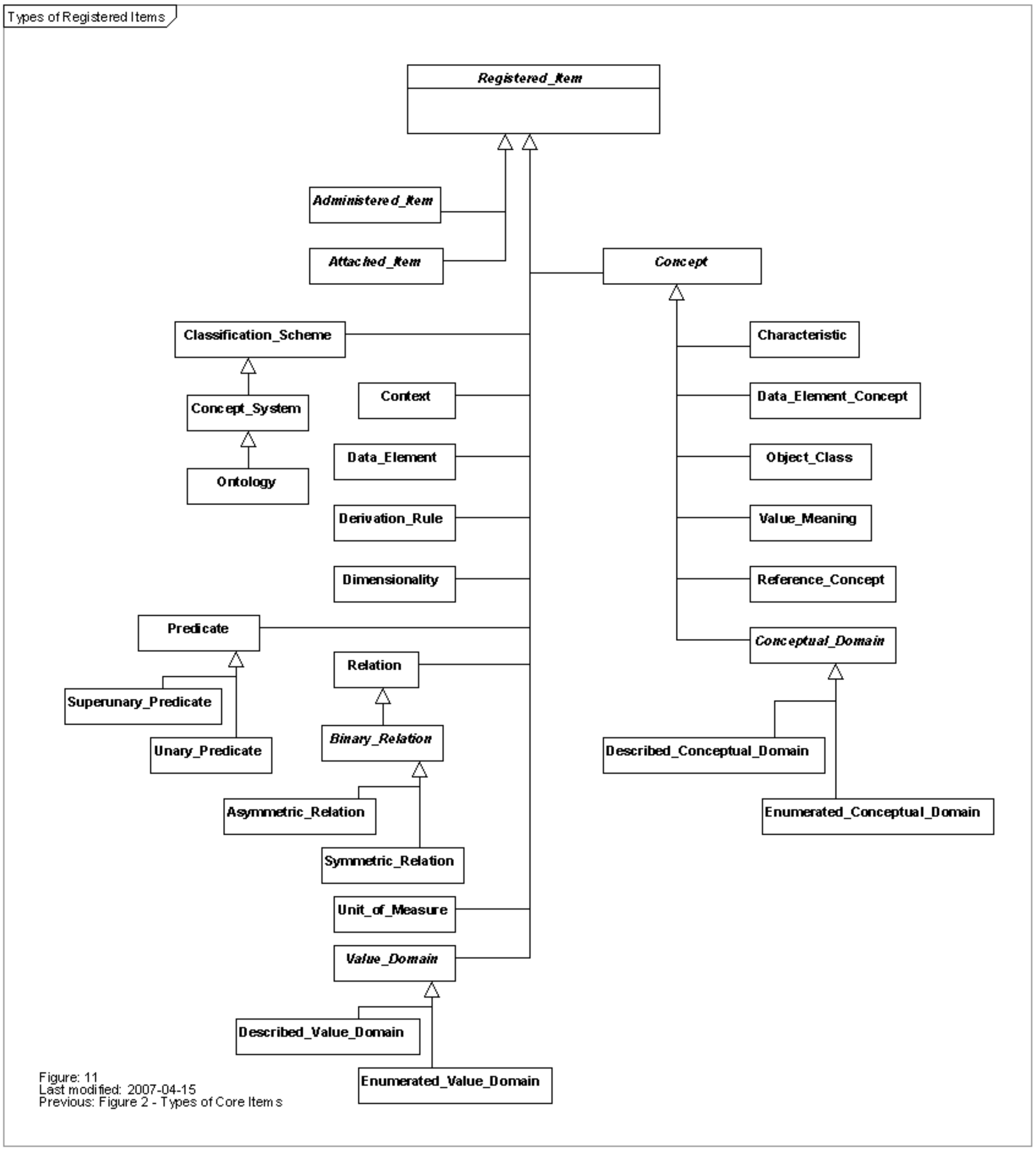


Figure C.2 — Types of Registered_Items extended to support defined metadata items

Annex D
(informative)

Mapping the ISO/IEC 11179-3:1994 basic attributes to the ISO/IEC 11179-3:200n metamodel and basic attributes

EDITOR'S NOTE #174. (Action required) In this edition we need to map from both Edition 1 and Edition 2 to Edition 3. This Annex has not yet been revised from Edition 2

D.1 Introduction

ISO/IEC 11179-3:1994 lists 23 basic attributes of data elements, as shown in Figure C.1.

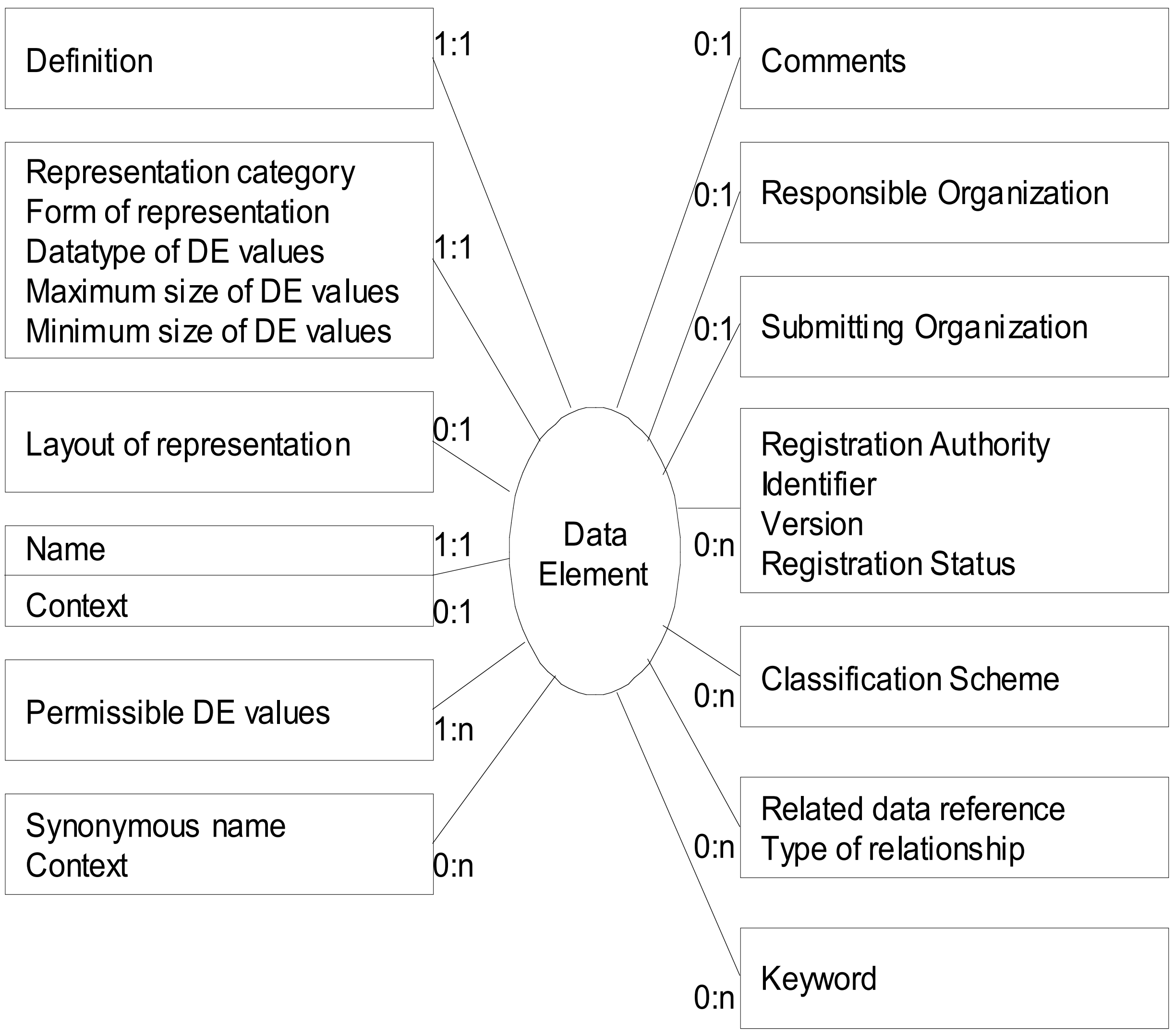


Figure D.1 — Basic Attributes of Data elements

This edition of the standard supports not only data elements, but also other metadata items associated with them, such as data element concepts, conceptual domains and value domains.

This annex maps the 1994 basic attributes to the metamodel in Clause 4, and the new basic attributes in Clause 5.

D.1.1 Description of Table Structures in this Annex

EDITOR'S NOTE #175. (Action required) The Editor proposes to use the following column headings when this Annex is revised:
Edition 1 Basic Attributes; Edition 2 Basic Attributes; Edition 3 Basic Attributes;
Edition 2 Model; Edition 3 Model.

The tables in this Annex are structured as follows:

	1994 Clause 6	2002 Clause 4	2002 Clause 5
Attribute name:			
Definition:			
Obligation:			
Condition:			
Datatype:			
Comment:			

Path from Administered_Item:

D.1.1.1 Description of the Columns

The columns in the table are used as follows:

- Column 1: Label for the row
- Column 2: What was specified in ISO/IEC 11179-3:1994 Clause 6
- Column 3: What is specified in ISO/IEC 11179-3:2002 Clause 4
- Column 4: What is specified in ISO/IEC 11179-3:2002 Clause 5

D.1.1.2 Description of the Rows

The rows in the table are used as follows, with the value in a particular cell coming from the Clause identified by the column (see above).

NOTE For the purposes of reference in the following text, the rows are numbered beginning at 1, and ignoring the column headings.

- Row 1: Attribute name - Contains the name of the attribute. For column 3, this is specified as: “Class name” “attribute name”, where “Class name” designates the Class in the metamodel that contains the attribute.
- Row 2: Definition – Contains the definition of the attribute.
- Row 3: Obligation – Contains the obligation of the attribute. (One of: Mandatory, Optional or Conditional.)
- Row 4: Condition – If the Obligation is “Conditional”, this row contains the condition that applies. (The entire row is omitted if it is not relevant for any column.)

- Row 5: Datatype – Contains the datatype of the attribute.
- Row 6: Comment – Contains any explanatory_comment. (The entire row is omitted if it is not relevant for any column.)

The notation "N/A" indicates that a row is "Not Applicable" for a particular column.

D.1.1.3 Specification of attribute name in row 1 column 3

For the old and new basic attributes (columns 2 and 4 respectively) the attribute name is straightforward. The equivalent attributes in the metamodel (column 3), need to be designated in the context of a particular class. The class that provides the context is named first, and then the attribute, using the "dot" notation:

> "Class Name" . "attribute name"
>
> e.g. “Item Identifier” . “version”

D.1.1.4 Specification of Path from Administered_Item to the named attribute

This information shows how the named attributed is related to an Administered_Item, and applies to column 3 only. It has been placed after the table to save space, and make the path easier to read. It specifies the path that needs to be navigated in the metamodel to reach the named attribute for any particular Administered_Item. (See below for an explanation of the notation.) Whenever the attribute is on the Administered_Item class, no navigation is necessary and this row is omitted.

In addition to designating the metamodel attribute in the context of a class (row 1 column 3), the “Path to Administered_Item” shows how the class is related an Administered_Item. It is necessary to navigate relationships and/or composite attributes within the model from one class to another. For common attributes (i.e. those that apply to any Administered_Item), the starting point for navigation is the supertype class “Administered_Item”. For attributes specific to a particular sub-type of Administered_Item, the starting point for navigation is that sub-type class (e.g. Data_Element). The “dot” notation is used as described below.

NOTE 1 The following notational convention is used:

- the names of classes and composite datatypes are capitalized e.g. "Item Identifier"
- the names of attributes are all lower case e.g. “version”
- the names of relationships are lower case and italicised e.g. “*name entry*”

NOTE 2 The use of *italics* to indicate a relationship applies only to the specification of the navigation path. In row 2 of the table (Definition), *italics* are used to distinguish the term from the definition.

Example 1: Attribute “version”

In this example, the attribute is a Common Attribute (i.e. it can apply to any type of Administered_Item), so the navigation starts from the super-type class “Administered_Item”.

> “Administered_Item”. “administered item administration record” .
> “Administration_Record”. “administered item identifier” .
> “Item Identifier”. “version”

specifies to follow the path in the model:

- from the class “Administered_Item” via its attribute “administered item administration record” to the composite datatype “Administration_Record”, then
- from the class “Administration_Record” via its attribute “administered item identifier” to the composite datatype “Item Identifier” and its attribute “version”.

Example 2: Attribute "datatype_name"

In this example, the attribute is specific to a Data_Element, so the navigation starts from the “Data_Element” sub-type class of Administered_Item.

> “Data_Element”. “*data element representation*”.
> “Value_Domain”. “value_domain_datatype”.
> “Datatype”. “datatype_name”

specifies to follow the path in the model:

— from the class “Data_Element” via its relationship “*data element representation*” to the related class “Value_Domain”, then

— from the class “Value_Domain” via its attribute “value_domain_datatype” to the composite datatype “Datatype” and its attribute “datatype_name”.

D.2 Mapping the Basic Attributes

The attributes are ordered in this Annex as in Clause 5 of this document.

D.2.1 Common Identifying attributes

D.2.1.1 Name

	1994 Clause 6	2002 Clause 4	2002 Clause 5
Attribute name:	Name	“Designation (of Administered_Item)”. “name”	name
Definition:	Single or multi word designation assigned to a data element.	A name by which an Administered_Item is known within a specific Context.	A name by which a metadata item is known within a specific context.
Obligation:	Mandatory	Mandatory	Mandatory
Data type:	Character string	String	String
Comment:		The attribute “preferred designation” may be used to specify the primary name if synonyms also exist in a particular context.	

D.2.1.1.1 Path from Administered_Item:

“Administered_Item”. “*administered item context*”. “Terminological Entry”. “*terminological entry languages*”. “Language Section”. “*name entry*”.

D.2.1.2 Synonymous name

	1994 Clause 6	2002 Clause 4	2002 Clause 5
Attribute name:	Synonymous name	“Designation (of Administered_Item)”. “name”	name
Definition:	Single word or multi word designation that differs from the given name, but represents the same data element concept.	A name by which an Administered_Item is known within a specific Context.	A name by which a metadata item is known within a specific context.
Obligation:	Optional	Optional	Optional
Data type:	Character string	String	String
Comment:	Synonymous names are often familiar names in a certain application environment. If this is the case use attribute 'Context' (6.1.6) to specify the context. If more synonymous names occur the attributes 'Synonymous name' and 'Context' shall be specified as a pair.	An Administered_Item may have multiple names in the same or different contexts. The distinction between “name” and “synonymous name” in a particular context may be specified by the attribute “preferred designation”, which should be set to “True” for the preferred name, and “False” for all synonyms.	A metadata item may have multiple names in the same or different contexts. The distinction between “name” and “synonymous name” in a particular context may be specified by the attribute “preferred designation”, which should be set to “True” for the preferred name, and “False” for all synonyms.

D.2.1.2.1 Path from Administered_Item:

“Administered_Item”. “*administered item context*”. “Terminological Entry”. “*terminological entry languages*”. “Language Section”. “*name entry*”.

D.2.1.3 Context name

	1994 Clause 6	2002 Clause 4	2002 Clause 5
Attribute name:	Context	“Designation (of Administered_Item)” . “name” NOTE The "Administered_Item" referred to here is the Context itself, not the Administered_Item to which context is being provided.	context name
Definition:	A designation or description of the application environment or discipline in which a name and/or synonymous name is applied or originates from. Note: The latest edition of the standard differentiates designations from descriptions.	*Context:* A universe of discourse in which a name or definition is used. *name:* A name by which an Administered_Item (in this case the Context) is known within a specific context (where the context for a Context is probably the registry).	*Context:* A universe of discourse in which a name or definition is used. *name:* A name by which a metadata item (in this case the Context) is known within a specific context (where the context for a context is the setting in which it is used).
Obligation:	Conditional	Mandatory	Conditional
Condition:	This attribute is mandatory for each occurrence of the attribute 'Synonymous name' (6.1.5). This attribute is mandatory when the attribute	N/A	Required if more than one *name* attribute exists for a particular metadata item.

	'Name' (6.1.1) occurs in an information exchange.		
Data type:	Character string	String	String
Comment:	Assignment of the attribute 'Context' to the attribute 'Name' may be made mandatory as part of the procedures of any Registration_Authority.	As an Administered_Item itself, any Context used within a registry must be given both a *name* and *definition*. A Context must itself exist within a Context, which for most will probably be the registry. (A Context may provide Context to itself.)	

D.2.1.3.1 Path from Administered_Item:

“Administered_Item”(1). “*administered item context*”.“Context”. “context administration record”. “Administration_Record” . “Administered_Item” (2). “*administered item context*”. “Terminological Entry”. “*terminological entry languages*”. “Language Section”. “name entry”.

NOTES (1) (2) The first “Administered_Item” is the one to which context is being provided. The second "Administered_Item" is the Context itself.

D.2.1.4 Context identifier

	1994 Clause 6	2002 Clause 4	2002 Clause 5
Attribute name:	Not supported.	“Context”. “context administration record”. “Administration_Record”. “administered item identifier”	context identifier
Definition:	N/A	*Context:* A universe of discourse in which a name or definition is used. *administered item identifier:* The unique identifier for an Administered_Item (in this case the Context).	*Context:* A universe of discourse in which a name or definition is used. *context identifier:* A unique identifier for the *Context* within its usage context.
Obligation:	N/A	Mandatory	Conditional
Condition:	N/A	N/A	Required if *context name* is not unique with its usage context.
Data type:	N/A	String	String
Comment:		As an Administered_Item itself, any Context used within a registry must be given an *administered item identifier*.	

D.2.1.4.1 Path from Administered_Item:

“Administered_Item”. “*administered item context*”.

D.2.1.5 Context description

	<u>1994 Clause 6</u>	<u>2002 Clause 4</u>	<u>2002 Clause 5</u>
Attribute name:	Context	“Context”. “context description”	context description
Definition:	A designation or description of the application environment or discipline in which a name and/or synonymous name is applied or originates from. Note: The new metamodel differentiates designations from descriptions.	*Context:* A universe of discourse in which a name or definition is used. *context description:* The textual description of the context.	*Context:* A universe of discourse in which a name or definition is used. *context description:* The textual description of the context.
Obligation:	Conditional	Mandatory	Conditional
Condition:	This attribute is mandatory for each occurrence of the attribute 'Synonymous name'. This attribute is mandatory when the attribute 'Name' occurs in an information exchange.	N/A	Required if *context name* is used.
Data type:	Character string	String	String
Comment:	Assignment of the attribute 'Context' to the attribute 'Name' may be made mandatory as part of the procedures of any Registration_Authority.	In this edition of this part of ISO/IEC 11179, *context description* and *context name* exist as two separate attributes.	In this edition of this part of ISO/IEC 11179, *context description* and *context name* exist as two separate attributes.

D.2.1.5.1 Path from Administered_Item:

“Administered_Item”. “*administered item context*”.

D.2.1.6 Item identifier – data identifier

	1994 Clause 6	**2002 Clause 4**	**2002 Clause 5**
Attribute name:	Identifier	"Item Identifier" . "data identifier"	item identifier – data identifier
Definition:	A language independent unique identifier of a data element within a Registration_Authority.	The unique identifier for an Administered_Item within a Registration_Authority.	The unique identifier for a metadata item within a specific context.
Obligation:	Conditional	Mandatory	Conditional
Condition:	If the attribute 'Name of data element' is not unique within a Registration_Authority this attribute is mandatory.	N/A	If the attribute *name* is not unique within a *context*, this attribute is mandatory.
Data type:	Character	String	String
Comment:	Assignment of a unique identifier may be made mandatory as part of the registration procedure of any Registration_Authority.		The requirement for an *item identifier* can be eliminated by qualifying *name* and/or *context name* to ensure that the combination is unique.

D.2.1.6.1 Path from Administered_Item:

"Administered_Item". "administered item administration record". "Administration_Record". "administered item identifier".

D.2.1.7 Item registration authority identifier

	1994 Clause 6	**2002 Clause 4**	**2002 Clause 5**
Attribute name:	Registration_Authority	"Item Identifier". "item registration authority identifier"	item identifier – item registration authority identifier
Definition:	Any organization authorized to register data elements.	An identifier (described in ISO/IEC 11179 Part 6) assigned to the Registration_Authority registering the item.	An identifier (described in ISO/IEC 11179 Part 6) assigned to the registration authority registering the item.
Obligation:	Conditional	Mandatory	Conditional
Condition:	One Registration_Authority shall be specified for each Identifier present.	N/A	Required if *item identifier – data identifier* is not unique within the usage context.
Data type:	Character string	String	String

D.2.1.7.1 Path from Administered_Item:

"Administered_Item". "administered item administration record". "Administration_Record". "administered item identifier".

D.2.1.8 Version

	1994 Clause 6	2002 Clause 4	2002 Clause 5
Attribute name:	Version	“Item identifier”. “version”	Version
Definition:	Identification of an issue of a data element specification in a series of evolving data element specifications within a Registration_Authority.	The unique version identifier of the Administered_Item.	The unique version identifier of the metadata item.
Obligation:	Conditional	Mandatory	Optional
Condition:	This attribute is mandatory if updates on attributes occur which meet the maintenance rules for allocating new versions as set by the Registration	N/A	N/A
Data type:	Character	String	String

D.2.1.8.1 Path from Administered_Item:

“Administered_Item”. “administered item administration record”. “Administration_Record”. “administered item identifier”.

D.2.2 Common Definitional attributes

D.2.2.1 Definition

	1994 Clause 6	2002 Clause 4	2002 Clause 5
Attribute name:	Definition	“Definition (of Administered_Item)”. “definition_text”	definition
Definition:	Statement that expresses the essential nature of a data element and permits its differentiation from all other data elements.	*Definition:* The definition of an Administered_Item within a Context. *Definition text:* The text of the definition.	The definition of an metadata item within a context.
Obligation:	Mandatory	Mandatory	Mandatory
Data type:	Character string	String	String
Comment:		Where more than one Definition is provided within a particular context, one of them may be specified as preferred by setting the attribute “preferred definition” to “True”.	

D.2.2.1.1 Path from Administered_Item:

“Administered_Item”. “*administered item context*”. “Terminological Entry”. “*terminological entry languages*”. “Language Section”. “*definition entry*”.

D.2.2.2 Definition language

	<u>1994 Clause 6</u>	<u>2002 Clause 4</u>	<u>2002 Clause 5</u>
Attribute name:	Not supported.	“Language Section”. “language section language_identifier”	definition_language_identifier
Definition:	N/A	The identifier of the language used within the *Terminological Entry*, which applies to both the name and the definition.	The identifier of the language used within the definition.
Obligation:	N/A	Optional	Optional
Data type:	N/A	String	String

D.2.2.2.1 Path from Administered_Item:

“Administered_Item”. “*administered item context*”. “Terminological Entry”.

D.2.2.3 Definition source reference

	<u>1994 Clause 6</u>	<u>2002 Clause 4</u>	<u>2002 Clause 5</u>
Attribute name:	Not supported.	“Definition”. “definition_source_reference”	definition_source_reference
Definition:	N/A	A reference to the source from which the definition is taken.	A reference to the source from which the definition is taken.
Obligation:	N/A	Optional	Optional
Data type:	N/A	String	String

D.2.2.3.1 Path from Administered_Item:

“Administered_Item”. “*administered item context*”. “Terminological Entry”. “*terminological entry languages*”. “Language Section”. “*definition entry*”.

D.2.3 Common Administrative attributes

D.2.3.1 Comments

	<u>1994 Clause 6</u>	<u>2002 Clause 4</u>	<u>2002 Clause 5</u>
Attribute name:	Comments	“Administration_Record”. “explanatory_comment”	Comments
Definition:	Remarks on the data element.	Descriptive comments about the Administered_Item.	Descriptive comments about the metadata item.
Obligation:	Optional	Optional	Optional
Data type:	Character string	String	String

D.2.3.1.1 Path from Administered_Item:

“Administered_Item”. “administered item administration record”.

D.2.3.2 Registration status

	1994 Clause 6	2002 Clause 4	2002 Clause 5
Attribute name:	Registration status	“Administration_Record”. “registration_status”	registration_status
Definition:	A designation of the position in the registration life-cycle of a data element.	A designation of the status in the registration life-cycle of an Administered_Item.	A designation of the status in the registration life-cycle of a metadata item.
Obligation:	Conditional	Mandatory	Optional
Condition:	This attribute is mandatory during the data element life-cycle specified by any Registration_Authority.	N/A	N/A
Data type:	Character	String	String
Comment:	The type of registration_status to be distinguished and the allocation of the registration_status shall follow the rules that are described in the procedures for the registration of data elements (see Part 6 of this International Standard).		

D.2.3.2.1 Path from Administered_Item:

Administered_Item”. “administered item administration record”.

D.2.3.3 Responsible organization

	1994 Clause 6	2002 Clause 4	2002 Clause 5
Attribute name:	Responsible organization	"Organization" . "organization name"	Responsible organization
Definition:	The organization or unit within an organization that is responsible for the contents of the mandatory attributes by which the data element is specified.	*Organization:* A unique framework of authority, within which a person or persons act, or are designated to act, towards some purpose. *stewardship:* The relationship of an Administered_Item, a Contact and an Organization involved in the stewardship of the metadata.	The organization or unit within an organization that is responsible for the contents of the mandatory attributes by which the metadata item is specified.
Obligation:	Optional	Mandatory	Optional
Data type:	Character string	String	String
Comment:	The organization shall be considered as 'owner' of the data element.		

D.2.3.3.1 Path from Administered_Item:

“Administered_Item” . “stewardship” .

D.2.3.4 Submitting organization

	1994 Clause 6	**2002 Clause 4**	**2002 Clause 5**
Attribute name:	Submitting organization	"Organization". "organization name"	Submitting organization
Definition:	The organization or unit within an organization that has submitted the data element for addition, change or cancellation/withdrawal in the data element dictionary.	*Organization:* A unique framework of authority, within which a person or persons act, or are designated to act, towards some purpose. *submission:* The relationship of an Administered_Item, a Contact and an Organization involved in a submission of metadata.	The organization or unit within an organization that has submitted the metadata item for addition, change or cancellation/withdrawal in a metadata registry.
Obligation:	Optional	Mandatory	Optional
Data type:	Character string	String	String

D.2.3.4.1 Path from Administered_Item:

"Administered_Item". "submission".

D.2.4 Common Relational attributes

D.2.4.1 Classification scheme name

	1994 Clause 6	2002 Clause 4	2002 Clause 5
Attribute name:	Classification scheme	“Designation (of Administered_Item)”. “name” NOTE The “Administered_Item” referred to here is the Classification_Scheme, not the Administered_Item which is being classified.	Classification scheme name
Definition:	A reference to (a) class(es) of a scheme for the arrangement or division of objects into groups based on characteristics that the objects have in common, e.g. origin, composition, structure, application, function etc.	*Classification_Scheme:* The descriptive information for an arrangement or division of objects into groups based on characteristics which the objects have in common. *name:* A name by which an Administered_Item (in this case the Classification_Scheme) is known within a specific Context.	The name of a particular arrangement or division of objects into groups based on characteristics which the objects have in common.
Obligation:	Optional	Conditional	Conditional
Condition:	N/A	If a Classification_Scheme is used, its name is mandatory.	If a Classification_Scheme is used, its name is mandatory.
Data type:	Character string	String	String
Comment	The definition does not specify whether the reference is by name or identifier.		

D.2.4.1.1 Path from Administered_Item:

“Administered_Item” (1). “*administered item classification*”. “Classification_Scheme Item”. “*classification scheme membership*”. “Classification_Scheme”. "classification scheme administration record”. “Administration_Record”. “Administered_Item” (2). “*administered item context*”. “Terminological Entry”. “*terminological entry languages*”. “Language Section”. “name entry”.

NOTES (1) (2) The first “Administered_Item” is the one which is being classified. The second “Administered_Item” is the Classification_Scheme itself.

D.2.4.2 Classification scheme identifier

	1994 Clause 6	2002 Clause 4	2002 Clause 5
Attribute name:	Classification scheme	“Classification_Scheme”. “classification scheme administration record”. “Administration_Record”. “administered item identifier”	classification scheme identifier
Definition:	A reference to (a) class(es) of a scheme for the arrangement or division of objects into groups based on characteristics that the objects have in common, e.g. origin, composition, structure, application, function etc.	*Classification_Scheme:* The descriptive information for an arrangement or division of objects into groups based on characteristics which the objects have in common. *administered item identifier:* An identifier for an Administered_Item (in this case the Classification_Scheme) within a Registration_Authority.	The identifier of a particular arrangement or division of objects into groups based on characteristics which the objects have in common.
Obligation:	Optional	Conditional	Optional
Condition	N/A	If a Classification_Scheme is used, its administered item identifier is mandatory.	N/A
Data type:	Character string	String	String
Comment	The definition does not specify whether the reference is by name or identifier.		

D.2.4.2.1 Path from Administered_Item:

“Administered_Item” . “*administered item classification*” . “Classification_Scheme Item” . “*classification scheme membership*” .

D.2.4.3 Classification scheme type name

	1994 Clause 6	2002 Clause 4	2002 Clause 5
Attribute name:	Not supported.	“Classification_Scheme”. “classification scheme type name”	Classification scheme type name
Definition:	N/A	The name of the type of classification scheme.	The name of the type of classification scheme.
Obligation:	N/A	Conditional	Optional
Condition	N/A	If Classification_Scheme is present, *classification scheme type name* is mandatory.	N/A
Data type:	N/A	String	String

D.2.4.3.1 Path from Administered_Item:

“Administered_Item”. “*administered item classification*”. “Classification_Scheme Item”. “*classification scheme membership*”.

D.2.4.4 Classification scheme item type name

	1994 Clause 6	2002 Clause 4	2002 Clause 5
Attribute name:	Not supported.	"Classification_Scheme Item" . "classification scheme item type name	classification scheme item type name
Definition:	N/A	The name of the type of the classification scheme item.	The name of the type of the classification scheme item.
Obligation:	N/A	Optional	Optional
Data type:	N/A	String	String

D.2.4.4.1 Path from Administered_Item:

"Administered_Item". "*administered item classification*".

D.2.4.5 Classification scheme item value

	1994 Clause 6	2002 Clause 4	2002 Clause 5
Attribute name:	Keyword	"Classification_Scheme Item". "classification scheme item value"	classification scheme item value
Definition:	One or more significant words used for retrieval of data elements.	An instance of a classification scheme item.	An instance of a classification scheme item.
Obligation:	Optional	Optional	Optional
Data type:	Character string	String	
Comment:	This attribute can be used for recording keywords (search keys) associated with the data element in question.	This edition of this part of ISO/IEC 11179 treats keywords as a type of classification scheme, with individual keywords being represented as classification scheme item values.	This edition of this part of ISO/IEC 11179 treats keywords as a type of classification scheme, with individual keywords being represented as classification scheme item values.

D.2.4.5.1 Path from Administered_Item:

"Administered_Item". "administered item classification".

D.2.4.6 Related metadata reference

	1994 Clause 6	2002 Clause 4	2002 Clause 5
Attribute name:	Related data reference	"Administration_Record". "administrative_note" OR "Administration_Record". "explanatory_comment" OR "Reference_Document". "reference_document_identifier"	Related metadata reference
Definition:	A reference between the data element and any related data.	*administrative_note:* any general note about the *Administered_Item* *expanatory comment:* descriptive comments about the *Administered_Item* *Reference_Document:* a document that provides pertinent details for consultation about a subject. *reference_document_identifier:* An identifier for the Reference_Document.	A reference from one metadata item to another.
Obligation:	Optional	Optional	Optional
Data type:	Character string	String	String
Comment:	If this attribute occurs it shall be specified in pair with the attribute 'Type of relationship'		

D.2.4.6.1 Path from Administered_Item:

For "adminisftrative note":

Administered_Item". "administered item administration record".

For "explanatory_comment":

Administered_Item". "administered item administration record".

For "reference_document_identifier":
"Administered_Item". "reference"

D.2.4.7 Type of relationship

	1994 Clause 6	2002 Clause 4	2002 Clause 5
Attribute name:	Type of relationship	“Reference_Document”. “reference_document_type_description”	Type of relationship
Definition:	An expression that characterizes the relationship between the data element and related data.	The description of the type of association with another data element concept that this data element concept modifies, is modified by, or is otherwise linked with.	The description of the type of relationship identified by the related metadata reference.
Obligation:	Conditional	Conditional OR Optional	Conditional
Condition:	This attribute is mandatory if the attribute 'related data reference' occurs.	“reference_document_type_description” is optional if Reference_Document is used.	This attribute is mandatory if the attribute 'related metadata reference' occurs.
Data type:	Character string	String	String
Comment:	Examples of type of relationships are: 'qualifier of', 'qualified by', 'subject of', 'part of', 'physical condition', 'external reference', 'higher standard', 'data element concept'.	See C.2.4.6 Related metadata reference.	

D.2.4.7.1 Path from Administered_Item:

“Administered_Item”. “reference”

D.2.5 Attributes specific to Data_Element_Concepts

D.2.5.1 Object class name

	1994 Clause 6	2002 Clause 4	2002 Clause 5
Attribute name:	Not supported.	“Designation (of Administered)”. “name” NOTE The Administered item referred to here is the Object_Class.	Object class name
Definition:	N/A	*data element concept object class:* the designation of an Object_Class for a Data_Element_Concept. *name:* A name by which an Administered_Item (in this case the Object_Class) is known within a specific context.	The designation of an *object class* for a *data element concept*.
Obligation:	N/A	Optional	Optional
Data type:	N/A	String	String

D.2.5.1.1 Path from Data_Element_Concept:

“Data_Element_Concept”. “data element concept object class”. “Object_Class”. “object class administration record”. “Administration_Record”. “Administered_Item”. “*administered item context*”. “Terminological Entry”. “*terminological entry languages*”. “Language Section”. “name entry”.

D.2.5.2 Object class identifier

	1994 Clause 6	2002 Clause 4	2002 Clause 5
Attribute name:	Not supported.	“Administration_Record”. “administered item identifier” NOTE The Administered item referred to here is the Object_Class.	Object class identifier
Definition:	N/A	*administered item identifier:* An identifier for an Administered_Item (in this case the Object_Class) within a Registration_Authority.	The identifier of an *object class* for a *data element concept.*
Obligation:	N/A	Optional	Optional
Data type:	N/A	String	String

D.2.5.2.1 Path from Data_Element_Concept:

“Data_Element_Concept”. “data element concept object class”. “Object_Class”. “object class administration record”.

D.2.5.3 Property name

	1994 Clause 6	2002 Clause 4	2002 Clause 5
Attribute name:	Not supported.	“Designation (of Administered)”. “name” NOTE The Administered item referred to here is the Property.	Property name
Definition:	N/A	*data element concept property:* the designation of a Property for a Data_Element_Concept. *name:* A name by which an Administered_Item (in this case the Property) is known within a specific context.	The designation of a *property* for a *data element concept.*
Obligation:	N/A	Optional	Optional
Data type:	N/A	String	String

D.2.5.3.1 Path from Data_Element_Concept:

“Data_Element_Concept”. “data element concept property”. "Property”. “property administration record". “Administration_Record”. “Administered_Item”. “*administered item context*”. “Terminological Entry”. “*terminological entry languages*”. “Language Section”. “name entry”.

D.2.5.4 Property identifier

	1994 Clause 6	2002 Clause 4	2002 Clause 5
Attribute name:	Not supported.	“Administration_Record”. “administered item identifier” NOTE The Administered item referred to here is the Property.	Property identifier
Definition:	N/A	*administered item identifier:* An identifier for an Administered_Item (in this case the Property) within a Registration_Authority.	The identifier of a *property* for a *data element concept*.
Obligation:	N/A	Optional	Optional
Data type:	N/A	String	String

D.2.5.4.1 Path from Data_Element_Concept:

“Data_Element_Concept”. “data element concept property”. “Property”. “property administration record”.

D.2.6 Attributes specific to Data_Elements

D.2.6.1 Representation category

	1994 Clause 6	2002 Clause 4	2002 Clause 5
Attribute name:	Representation category	Not supported.	Not supported.
Definition:	Type of symbol, character or other designation used to represent a data element.	N/A	N/A
Obligation:	Mandatory	N/A	N/A
Data type:	Character string	N/A	N/A
Comment:	The representation category shall be specified by the relevant standard. Examples of possible representation categories: — character representation (ISO/IEC 646) — character/symbol representation (ISO registration no. 143) — bar coded representation (EIA-556) — graphical representation		

D.2.6.2 Representation class

EDITOR'S NOTE #176. Issue 114 proposes to remove *Representation Class*. How should we map 'form of representation' in Edition 3?

	1994 Clause 6	**2002 Clause 4**	**2002 Clause 5**
Attribute name:	Form of representation	"Designation (of Administered)". "name" NOTE The Administered item referred to here is the Representation Class.	Representation class
Definition:	Name or description of the form of representation for the data element, e.g. 'quantitative value', 'code', 'text', 'icon'.	*Representation Class:* the classification of types of representations. *name:* A name by which an Administered_Item (in this case the Representation Class) is known within a specific context.	The name of the class of representation of a data element.
Obligation:	Mandatory	Optional	Optional
Data type:	Character string	String	String
Comment:	1. See ISO/IEC 11179-2 for appropriate terms ('property words' or 'class words') to be used. 2. Example 1: For the data element named: 'country of origin code' this attribute contains: 'code'. 3. Example 2: For the data element: 'product description' this attribute contains: 'text'. 4. Example 3: For the data element: 'weight of consignment' this attribute contains: 'quantitative value'.	See 4.13.1.4 for a list of Representation Class terms.	See 4.13.1.4 for a list of Representation Class terms.

D.2.6.2.1 Path from Data_Element

"Data_Element". "*data element representation class*". "Representation Class". "Administration_Record". "Administered_Item". "*administered item context*". "Terminological Entry". "*terminological entry languages*". "Language Section". "name entry".

D.2.6.3 Value domain name

	1994 Clause 6	2002 Clause 4	2002 Clause 5
Attribute name:	Not directly supported.	“Designation (of Administered)”. “name” NOTE The Administered item referred to here is the Value_Domain.	value domain name
Definition:	N/A	*Value domain:* A set of permissible values. It provides representation, but has no implication as to what data element concept the values may be associated with nor what the values mean. *name:* A name by which an Administered_Item (in this case the Value_Domain) is known within a specific context.	The name of the value domain that provides representation for the data element.
Obligation:	N/A	Mandatory	Optional
Data type:	N/A	String	String
Comment:	The closest equivalent is “permissible data element values” (see F.2.6.10), but this actually represents the values.		

D.2.6.3.1 Path from Data_Element

“Data_Element”. “*data element representation*”. “Value_Domain”. “value domain administration record”. “Administration_Record”. “Administered_Item”. “*administered item context*”. “Terminological Entry”. “*terminological entry languages*”. “Language Section”. “name entry”.

D.2.6.4 Value domain identifier

	1994 Clause 6	2002 Clause 4	2002 Clause 5
Attribute name:	Not directly supported.	“Administration_Record”. “administered item identifier”	value domain identifier
Definition:	N/A	*Value_Domain:* A set of permissible values. It provides representation, but has no implication as to what Data_Element_Concept the values may be associated with nor what the values mean. *administered item identifier:* An identifier for an administered item (in this case the Value_Domain) within a registration authority.	The identifier of the value domain that provides representation for the data element.
Obligation:	N/A	Mandatory	Optional
Data type:	N/A	String	String

Comment:	The closest equivalent is "permissible data element values" (see F.2.6.10), but this actually represents the values.		

D.2.6.4.1 Path from Data_Element

"Data_Element". "*data element representation*". "Value_Domain". "value domain administration record".

D.2.6.5 Datatype name

	1994 Clause 6	2002 Clause 4	2002 Clause 5
Attribute name:	Datatype of data element values	"Datatype" . "datatype_name"	datatype_name
Definition:	A set of distinct values for representing the data element value.	*Datatype:* A set of distinct values characterized by properties of those values and by operations on those values. *datatype_name:* A designation for the datatype.	*datatype_name:* A designation for the datatype.
Obligation:	Mandatory	Mandatory	Conditional
Condition	N/A	N/A	Required if neither *value domain name* nor *value domain identifier* is specified.
Data type:	Character string	String	String
Comment:	Examples: Possible instances are: 'character', 'ordinal number', 'integer', 'real', 'scaled', 'bit', 'rational'. Note: The examples suggest the attribute is intended to be the name of the datatype, whereas the definition implies it is a set of values.	In the metamodel, the datatype is an attribute of the value domain, not directly of the data element.	

D.2.6.5.1 Path from Data_Element

"Data_Element". "*data element representation*". "Value_Domain". "value_domain_datatype".

D.2.6.6 Datatype scheme reference

	1994 Clause 6	2002 Clause 4	2002 Clause 5
Attribute name:	Not supported.	"Datatype". "datatype_scheme_reference"	Datatype scheme reference
Definition:	N/A	A reference identifying the source of the Datatype specification.	A reference identifying the source of the datatype specification.
Obligation:	N/A	Mandatory	Conditional
Condition	N/A	N/A	Required if *datatype_name* is specified.
Data type:	N/A	String	String

Comment:		In the metamodel, the datatype is an attribute of the value domain, not directly of the data element.	

D.2.6.6.1 Path from Data_Element

"Data_Element". "*data element representation*". "Value_Domain". "value_domain_datatype".

D.2.6.7 Maximum size

	1994 Clause 6	**2002 Clause 4**	**2002 Clause 5**
Attribute name:	Maximum size of data element values	"Value_Domain". "value_domain_maximum_character_quantity"	Maximum size
Definition:	The maximum number of storage units (of the corresponding datatype) to represent the data element value.	The maximum number of characters to represent the data element value. NOTE Applicable only to character Datatypes.	The maximum number of storage units (of the corresponding datatype) to represent the data element value.
Obligation:	Mandatory	Optional	Optional
Data type:	Integer	Integer	Integer
Comment:	1. Example 1: For data element: 'invoice number' the attribute 'datatype' has instance 'character' and the attribute 'maximum size of data element value' has value: '17'. The data element value of 'invoice number' shall have a maximum of 17 characters. 2. The two attributes 'maximum and minimum (see 6.4.5) size of data element values' indicate whether data element values are 'fixed' (maximum and minimum size are equal) or 'variable' (maximum and minimum size vary).	This is not exactly equivalent, because it applies only to character datatypes.	

D.2.6.7.1 Path from Data_Element

"Data element". "data element representation".

D.2.6.8 Minimum size

	1994 Clause 6	**2002 Clause 4**	**2002 Clause 5**
Attribute name:	Minimum size of data element values.	Not supported.	Minimum size
Definition:	The minimum number of storage units (of the corresponding datatype) to represent the data element value.	N/A	The minimum number of storage units (of the corresponding datatype) to represent the data element value.
Obligation:	Mandatory	N/A	Optional
Data type:	Integer	N/A	Integer
Comment:	1. Example 1: For data element: 'product description' the attribute 'datatype' has instance 'character' and the attribute 'minimum size of data element value' has instance: '10'. The data element value of 'product description' shall have a minimum of 10 characters. 2. The two attributes 'maximum (see 6.4.4) and minimum size of data element values' indicate whether data element values are 'fixed' (maximum and minimum size are equal) or 'variable' (maximum and minimum size vary).		

D.2.6.9 Layout of representation

	1994 Clause 6	**2002 Clause 4**	**2002 Clause 5**
Attribute name:	Layout of representation	Not supported.	Layout of representation
Definition:	The layout of characters in data element values expressed by a character string representation.	N/A	The layout of characters in data element values expressed by a character string representation.
Obligation:	Conditional	N/A	Optional
Condition:	If the data element is of the class 'quantitative data' this attribute is mandatory. If the attribute 'form of representation' is 'code' the use of this attribute is recommended if the code representation has to have a specific structure or layout.		
Data type:	Character string		String
Comment:	1. For quantitative data it is necessary to distinguish between integers, decimal mark and floating point notations. Example: Integers may be indicated with 'n', for decimal mark the number of characters before and after the decimal mark are specified as: n(5).n(3), for floating point notations the layout convention for a value with exponents shall comply with ISO 6093: n(3).n(3)E2, where 'E2' stands for max. 2 digits for the power of 10. 2. For code representations having a specific structure or layout the type of character for each position in the code structure is important for validation purposes. Example: The data element 'flight number' has an international code representation structure consisting of two alphabetic characters of the airline company followed by a three-digit number identifying the flight (from starting-point to destination). The contents of the attribute: 'layout of representation' is: 'AA999'.		

D.2.6.10 Permissible data element values

	1994 Clause 6	2002 Clause 4	2002 Clause 5
Attribute name:	Permissible data element values	See Value_Domain for equivalent capability.	See Value_Domain for equivalent capability.
Definition:	The set of representations of permissible instances of the data element, according to the representation form, layout, datatype and maximum and minimum size specified in the corresponding attributes. The set can be specified by name, by reference to a source, by enumeration of the representation of the instances or by rules for generating the instances.	N/A	N/A
Obligation:	Mandatory	N/A	N/A
Data type:	Character string	N/A	N/A
Comment:	When the permissible data element values are an enumeration of coded representations each data element value and instance shall be presented as a pair.		

D.2.7 Attributes specific to Conceptual_Domains

D.2.7.1 Dimensionality

	1994 Clause 6	2002 Clause 4	2002 Clause 5
Attribute name:	Not supported.	"Conceptual_Domain" . "dimensionality"	dimensionality
Definition:	N/A	The dimensionality for a concept.	The dimensionality for a concept.
Obligation:	N/A	Optional	Optional
Data type:	N/A	String	String
Comment:		For example, length, mass, velocity, currency.	For example, length, mass, velocity, currency.

D.2.8 Attributes specific to Value_Domains

D.2.8.1 Datatype name

	1994 Clause 6	2002 Clause 4	2002 Clause 5
Attribute name:	See “Datatype of data element values” (F.2.6.5)	“Value_Domain”. “value_domain_datatype”. “Datatype”. “datatype_name”	datatype_name
Definition:	N/A	*Datatype:* A set of distinct values characterized by properties of those values and by operations on those values. *datatype_name:* A designation for the datatype.	*datatype_name:* A designation for the datatype.
Obligation:		Mandatory	Mandatory
Data type:		String	String

D.2.8.2 Datatype scheme reference

	1994 Clause 6	2002 Clause 4	2002 Clause 5
Attribute name:	Not supported.	“Value_Domain”. “value_domain_datatype”. “Datatype”. “datatype_scheme_reference”	Datatype scheme reference
Definition:		A reference identifying the source of the datatype specification.	A reference identifying the source of the datatype specification.
Obligation:		Mandatory	Optional
Data type:		String	String

D.2.8.3 Unit of measure name

	1994 Clause 6	2002 Clause 4	2002 Clause 5
Attribute name:	Not supported.	“Value_Domain”. “value_domain_unit_of_measure”. “Unit_of_Measure”. “unit of measure name”	unit of measure name
Definition:		The name of a unit of measure.	The name of a unit of measure.
Obligation:		Optional	Optional
Data type:		String	String

D.2.9 Attributes specific to Permissible_Values

D.2.9.1 Value

	1994 Clause 6	**2002 Clause 4**	**2002 Clause 5**
Attribute name:	See “permissible data element values” (F.2.6.10)	“Permissible_Value”. “permitted_value”. “Value”. “value item”	value
Definition:	N/A	A representation of a value meaning in a specific value domain. The actual value.	A representation of a value meaning in a specific value domain. The actual value.
Obligation:	N/A	Mandatory	Mandatory
Data type:	N/A	String	String
Comment:			

D.2.9.2 Permissible_Value Begin Date

	1994 Clause 6	**2002 Clause 4**	**2002 Clause 5**
Attribute name:	Not supported.	“Permissible_Value”. “permissible_value_begin_date”	permissible_value_begin_date
Definition:	N/A	The date this value became/becomes permissible in the value domain.	The date this value became/becomes permissible in the value domain.
Obligation:	N/A	Optional	Optional
Data type:	N/A	Date	Date

D.2.9.3 Permissible_Value End Date

	1994 Clause 6	**2002 Clause 4**	**2002 Clause 5**
Attribute name:	Not supported.	“Permissible_Value”. “permissible_value_end_date”	permissible_value_end_date
Definition:	N/A	The date this value became/becomes no longer permissible in the value domain.	The date this value became/becomes no longer permissible in the value domain.
Obligation:	N/A	Optional	Optional
Data type:	N/A	Date	Date

D.2.10 Attributes specific to Value_Meanings

D.2.10.1 Value meaning description

	1994 Clause 6	2002 Clause 4	2002 Clause 5
Attribute name:	Not supported.	"Value_Meaning". "value meaning description"	value meaning description
Definition:	N/A	A description of a value meaning.	A description of a value meaning.
Obligation:	N/A	Mandatory	Mandatory
Data type:	N/A	String	String

D.2.10.2 Value meaning identifier

	1994 Clause 6	2002 Clause 4	2002 Clause 5
Attribute name:	Not supported.	"Value_Meaning". "value meaning identifier"	value meaning identifier
Definition:	N/A	The unique identifier for a value meaning.	The unique identifier for a value meaning.
Obligation:	N/A	Mandatory	Optional
Data type:	N/A	String	String

D.2.10.3 Value meaning begin date

	1994 Clause 6	2002 Clause 4	2002 Clause 5
Attribute name:	Not supported.	"Value_Meaning". "value_meaning_begin_date"	value_meaning_begin_date
Definition:	N/A	The effective_date of this value meaning in the conceptual domain.	The effective_date of this value meaning in the conceptual domain.
Obligation:	N/A	Optional	Optional
Data type:	N/A	Date	Date

D.2.10.4 Value meaning end date

	1994 Clause 6	2002 Clause 4	2002 Clause 5
Attribute name:	Not supported.	"Value_Meaning". "value_meaning_end_date"	value_meaning_end_date
Definition:	N/A	The date this value meaning became/becomes invalid.	The date this value meaning became/becomes invalid.
Obligation:	N/A	Optional	Optional
Data type:	N/A	Date	Date

Annex E
(informative)

Mapping the ISO/IEC 11179-3:2002 metamodel to the ISO/IEC 11179-3:200n metamodel

EDITOR'S NOTE #177. Mapping from Edition 2 to Edition 3 to be added.

E.1 Introduction

This Annex explains how the Edition 2 metamodel relates to the Edition 3 metamodel.

E.2 Mapping the Edition 2 Common Facilities

To be added.

Edn. 2 clause #	Edition 2 metamodel object	Edn. 3 clause #	Edition 3 metamodel object

E.3 Mapping the Data Description Model

To be completed.

Edn. 2 clause #	Edition 2 metamodel object	Edn. 3 clause #	Edition 3 metamodel object
	Conceptual_Domain Relationship		'Relation' in the new 'Concept_System' metamodel region
	conceptual domain relationship type description		As a Registered_Item, 'Relation' can be designated and defined using the common facilities of the metamodel.
	conceptual domain representation		value_domain_meaning
	contact name		contact person
	context description		Definition for Context.
	context description language_identifier		Language of Definition of Context
	country identifier		region_identifier

Bibliography

[1] ISO/TR 9007:1987, *Information processing systems — Concepts and terminology for the conceptual schema and the information base*

TR 9007 provides information on conceptual modelling.

[2] ISO/IEC 10027:1990, *Information technology — Information Resource Dictionary System (IRDS) framework*

ISO/IEC 10027 describes the concept of levels of modelling.

[3] ISO/IEC TR 10032:2003, *Information technology — Reference model for data management*

ISO/IEC 10032 describes the concept of levels of modelling.

[4] ISO/IEC TR 20943-1 (2003), *Information technology — Achieving metadata registry content consistency — Part 1: Data elements*

TR 20943-1 provides guidelines for recording data elements in a 11179-3 metadata registry.

[5] ISO/IEC TR 20943-3 (2004), *Information technology — Achieving metadata registry content consistency — Part 3: Value domains*

TR 20943-3 provides guidelines for recording value domains in a 11179-3 metadata registry.