ISO/IEC JTC 1/SC 32 N 2271

Date: 2012-09-21

REPLACES: 32N1983

ISO/IEC JTC 1/SC 32

Data Management and Interchange

Secretariat: United States of America (ANSI)
Administered by Farance Inc. on behalf of ANSI

DOCUMENT TYPE	Text for FDIS ballot
TITLE	ISO/IEC FDIS 11179-3 Information technology - Metadata registries (MDR) - Part 3: Registry metamodel and basic attributes 3rd Edition
SOURCE	WG2 - Ray Gates - project editor
PROJECT NUMBER	1.32.15.03.03.00
STATUS	Disposition of comments on FCD N1983 is in N2272. This text is to be sent to ITTF for FDIS ballot.
REFERENCES	
ACTION ID.	ITTF
REQUESTED ACTION	
DUE DATE	--
Number of Pages	241
LANGUAGE USED	English
DISTRIBUTION	P & L Members SC Chair WG Conveners and Secretaries

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 N2271

Date: 2012-07-31

ISO/IEC FDIS 11179-3:2012(E)

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: (50) Approval
Document language: E

C:\Users\RayGates\Documents\Standards\ISO-IEC\JTC1\SC32\SC32-WG2-MetaData\WG2 documents\Editing 11179 Edition 3\Part 3\Post FCD Editing\Editors Working Drafts\32N2271T_FDIS_11179-3.doc STD Version 2.1c2

Contents

Table of Figures

Table of Tables

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, clauses / subclauses / tables / figures / annexes of which have been technically revised.

Edition 3 of this part of ISO/IEC 11179 includes several enhancements to Edition 2, both in terms of the presentation of the metamodel, and its capabilities, as follows:

From a presentation perspective, these include:

- use of UML 2.4.1 instead of UML 1.4 to describe the metamodel;
- use of UML packages to show dependencies between various regions of the metamodel. (See 5.3.3 and 5.3.4.)

From a capability perspective, these include:

- introduction of different types of metadata items (see 5.5);
- support for registration of Concept Systems (see 9.1);
- finer-grained conformance options (see 4.3).

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

— *Part 1: Framework*

— *Part 2: Classification*

— *Part 3: Registry metamodel and basic attributes*

— *Part 4: Formulation of data definitions*

— *Part 5: Naming and identification principles*

— *Part 6: Registration*

Introduction

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** (MDR).

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 5 through 11 of this part of ISO/IEC 11179 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 datatypes.

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.

In this part of ISO/IEC 11179, clause 12 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, application developers, business modellers and others who are responsible for making data understandable and shareable. ISO/IEC 11179 has broad applicability across subject area domains and information technologies.

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;

e) the registration and management of semantic artifacts that are useful for data management, data administration, and data analysis;

f) the interrelation and mapping of concept systems with other concept systems, e.g., to support efforts to converge on consistency through harmonization and vetting activities;

g) the interrelation of concept systems with data held in relational databases, XML databases, knowledgebases, text, and possibly graph databases deriving from natural language text understanding systems;

h) the provision of services for semantic computing: Semantics Service Oriented Architecture, Semantic Grids, semantics based workflows, Semantic Web, etc.;

i) support for addressing semantic web considerations such as AAA (anyone can say anything about anything), non-unique names, and open world assumption;

j) the capture of semantics with more formal techniques (in addition to natural language) -- First Order Logic (e.g., Common Logic), Description Logics (such as OWL-DL);

k) support of Application Development and Maintenance;

l) support of data migration, data mediation;

m) support of portals, data marts, and data warehouses;

n) support of data grids and online transaction networks;

o) ontological reasoning with metadata;

p) ontology entry point for browsing and searching metadata registries;

q) capture of associations between the published identifiers used in the ontology(s), and the concepts registered in the registry;

r) support for Ontology-driven Data Translation;

s) support for data integration & data interoperation.

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

1 Scope

1.1 Scope – Structure of a metadata registry

Clauses 5 through 11 specify the structure of a metadata registry in the form of a conceptual data model.

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.

1.2 Scope – Basic attributes of metadata items

Clause 12 specifies basic attributes which are required to describe metadata items, and which might be used in situations where a complete metadata registry is not appropriate (e.g. in the specification of other International Standards).

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.

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

3 Terms, definitions and abbreviated terms

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

NOTE 1 An alphabetical list of all terms used in this part of ISO/IEC 11179 is included in Annex A.

NOTE 2 Some definitions listed in this clause have one or more notes; some have a reference to another standard from which the definition is taken; and some have both notes and a reference. Where a definition has both one or more notes and a reference, notes that precede the reference come from the referenced source; notes that follow the reference have been added by this part of ISO/IEC 11179.

3.1 Terms and definitions of metamodel constructs used in this part of ISO/IEC 11179

NOTE This subclause defines the **metamodel constructs** (3.2.81 - units of notation for modelling) used in specifying the registry metamodel in Clauses 6 through 11.

3.1.1
abstract class
〈metamodel〉 **class** (3.1.5) that does not provide a complete declaration and typically cannot be instantiated

NOTE 1 An abstract class is intended to be used by other classes as a general class for **specialization** (3.1.16);

NOTE 2 Adapted from ISO/IEC 19505-1:2012, 9.19.1 Classifier.isAbstract.

NOTE 3 Cf. **concrete class** (3.1.8)

3.1.2
association
〈metamodel〉 semantic **relationship** (3.1.15) between two **class**es (3.1.5)

NOTE 1 An association is a type of relationship;

NOTE 2 Adapted from ISO/IEC 19505-2:2012, 7.3.3.

3.1.3
association class
〈metamodel〉 **association** (3.1.2) that is also a **class** (3.1.5)

NOTE 1 An association class not only connects a set of classes, but also defines a set of features that belong to the association itself;

NOTE 2 Adapted from ISO/IEC 19505-2:2012, 7.3.4.

3.1.4
attribute
〈metamodel〉 **characteristic** (3.2.14) of an **object** (3.2.87) or set of objects

3.1.5
class
〈metamodel〉 description of a set of **object**s (3.2.87) that share the same **attribute**s (3.1.4), operations, methods, **relationship**s (3.1.15), and semantics

NOTE Adapted from ISO/IEC 19505-2:2012, 7.3.7.

3.1.6
composite attribute
〈metamodel〉 **attribute** (3.1.4) whose **datatype** (3.1.8) is non-atomic

EXAMPLE See *Registration.registration_state* (8.1.5.1.2.1), where *registration_state* is a composite attibute with *Registration_State* (8.1.2.6) as its **composite datatype** (3.1.7).

3.1.7
composite datatype
⟨metamodel⟩ **datatype** (3.1.8) that is also a **class** (3.1.5)

NOTE A composite datatype is used as a datatype for a **composite attribute** (3.1.6).

EXAMPLE See *Registration.registration_state* (8.1.5.1.2.1) where *Registration_State* (8.1.2.6) serves as the composite datatype for the composite attribute *registration_state*.

3.1.8
concrete class
⟨metamodel⟩ **class** (3.1.5) that can be instantiated

NOTE Cf. **abstract class** (3.1.1).

3.1.9
datatype
set of distinct values, characterized by properties of those values and by operations on those values

[ISO/IEC 11404:2007, 3.12]

3.1.10
generalization
⟨metamodel⟩ **relationship** (3.1.15) between a more general class (the parent) and a more specific class (the child) that is fully consistent with the general class and that adds additional information

NOTE 1 A generalization is a type of **relationship** (3.1.15);

NOTE 2 The more general class is referred to as the **superclass** (3.1.18);

NOTE 3 The more specific class is referred to as a **subclass** (3.1.17);

NOTE 4 A generalization is directed from the **subclass** to the **superclass**;

NOTE 5 'Fully consistent' means that the **subclass** has all of the **attribute**s (3.1.4) and **relationships** (3.1.15) of the **superclass**;

NOTE 6 cf. **specialization** (3.1.16); generalization is the inverse of specialization;

NOTE 7 Adapted from ISO/IEC 19505-2:2012, 7.3.20.

3.1.11
identifier
⟨metamodel⟩ sequence of characters, capable of uniquely identifying that with which it is associated, within a specified context

NOTE A **name** (3.2.83) should not be used as an identifier because it is not linguistically neutral.

3.1.12
metamodel region
sub-division of a **package** (3.1.13) used to organize **metadata objects** (3.2.76) for ease of explanation

3.1.13
package
grouping of **metadata objects** (3.2.76) that provides a **namespace** (3.2.84) for the grouped objects, and allows them to be referenced as a group

NOTE 1 Adapted from ISO/IEC 19505-2:2012, 7.3.37.

3.1.14
primitive datatype
datatype (3.1.8) that cannot be decomposed into other datatypes without loss of associated semantics

NOTE Adapted from ISO/IEC 11404:2007, 3.44.

3.1.15
relationship
⟨metamodel⟩ connection among model elements

NOTE 1 In this part of ISO/IEC 11179, a relationship is one of: an **association** (3.1.2), a **generalization** (3.1.10) or a **specialization** (3.1.16);

NOTE 2 Adapted from ISO/IEC 19505-2:2012, 7.3.47.

3.1.16
specialization
⟨metamodel⟩ **relationship** (3.1.15) between a more general class (the parent) and a more specific class (the child) that is fully consistent with the general class and that adds additional information

NOTE 1 A specialization is a type of **relationship** (3.1.15);

NOTE 2 The more general class is referred to as the **superclass** (3.1.18);

NOTE 3 The more specific class is referred to as a **subclass** (3.1.17);

NOTE 4 A specialization is directed from the **superclass** to the **subclass**;

NOTE 5 'Fully consistent' means that the **subclass** has all of the **attribute**s (3.1.4) and **relationships** (3.1.15) of the **superclass**;

NOTE 6 cf. **generalization** (3.1.10); specialization is the inverse of generalization**;**

NOTE 7 Adapted from ISO/IEC 19505-2:2012, 7.3.20.

3.1.17
subclass
class (3.1.5) that is a **specialization** (3.1.16) of another class, its **superclass** (3.1.18)

NOTE 1 In UML, **subclass**es of a **superclass** are by default not **disjoint** (3.2.59). This part of ISO/IEC 11179 specifies when subclasses are required to be disjoint. Further, when the list of subclasses is intended to be exhaustive, this part of ISO/IEC 11179 shows the superclass as abstract, thus preventing any other subclass being instantiated. An **abstract class** (3.1.1) is indicated by showing the class name in *italics* in any class diagram that uses it.

NOTE 2 A particular class may be a subclass with respect to one relationship, but a superclass with respect to another relationship.

3.1.18
superclass
class (3.1.5) that is a **generalization** (3.1.10) of one or more other classes, its **subclass**es (3.1.17)

NOTE 1 In UML, **subclass**es of a **superclass** are by default not **disjoint** (3.2.59). This part of ISO/IEC 11179 specifies when subclasses are required to be disjoint. Further, when the list of subclasses is intended to be exhaustive, this part of ISO/IEC 11179 shows the superclass as abstract, thus preventing any other subclass being instantiated. An **abstract class** (3.1.1) is indicated by showing the class name in *italics* in any class diagram that uses it.

NOTE 2 A particular class may be a superclass with respect to one relationship, but a subclass with respect to another relationship.

3.2 Terms for concepts used in this part of ISO/IEC 11179

NOTE This subclause provides definitions for terms which are concepts used in the specification of the metadata model in Clauses 5 through 10. Data definitions are included in those clauses. An alphabetical list of terms with links to the corresponding definitions is included in Annex A.

3.2.1
acceptability rating
scale of acceptability

NOTE 1 The following values shall be used: preferred, admitted, deprecated, obsolete and superseded

NOTE 2 Adapted from ISO 10241 :1992, 5.2.2.

3.2.2
administered item
registered item (3.2.105) for which **administrative information** (3.2.3) is recorded

3.2.3
administrative information
<metadata registry> information about the administration of an item in a **metadata registry** (3.2.78)

EXAMPLES creation date, last change date, origin, change description, explanatory comment

3.2.4
arity
number of arguments that a function takes

3.2.5
assertion
sentence or proposition in logic which is asserted (or assumed) to be true

3.2.6
attached item
registered item (3.2.105) for which **administrative information** (3.2.3) is recorded in another registered item

NOTE This is often a member of a group of registered items that is managed as a whole.

3.2.7
attribute instance
specific instance of an **attribute** (3.1.4)

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

3.2.8
attribute value
value associated with an **attribute instance** (3.2.7)

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

3.2.9
basic attribute
<metadata> **attribute** of a **metadata item** commonly needed in its specification

3.2.10
binary relation
relation (3.2.119) with **arity** (3.2.4) equal to 2 (i.e. whose members all have two ends)

NOTE 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*. (e.g. A is between B and C.)

3.2.11
binding
mapping from one framework or specification to another, enabling **data** (3.2.27) and/or commands to be passed between them

3.2.12
boolean
mathematical **datatype** (3.1.8) associated with two-valued logic

[ISO/IEC 11404:2007, 8.1.1 Boolean]

3.2.13
cardinality
number of elements in a set

cf. **multiplicity** (3.2.82)

NOTE Adapted from ISO/IEC 19501:2005, Glossary

3.2.14
characteristic
abstraction of a **property** (3.2.100) of an **object** (3.2.87) or of a set of objects

NOTE Characteristics are used for describing **concept**s (3.2.18).

[ISO 1087-1:2000, 3.2.4]

3.2.15
classifiable item
metadata item (3.2.75) of a type for which classification is supported in a given **metadata registry** (3.2.78)

3.2.16
classification scheme
descriptive information for an arrangement or division of **object**s (3.2.87) into groups based on criteria such as **characteristic**s (3.2.14), which the objects have in common

EXAMPLE Origin, composition, structure, application, function, etc.;

3.2.17
common attribute
<metadata> **basic attribute** (3.2.9) that is applicable to many or all types of **metadata item** (3.2.75)

3.2.18
concept
unit of knowledge created by a unique combination of **characteristic**s (3.2.14)

NOTE Concepts are not necessarily bound to particular languages. They are, however, influenced by the social or cultural background which often leads to different categorizations.

[ISO 1087-1:2000, 3.2.1]

NOTE 2 A concept is independent of its representation.

3.2.19
concept system
set of **concept**s (3.2.18) structured according to the **relation**s (3.2.119) among them

[ISO 1087-1:2001, 3.2.11]

3.2.20
conceptual data model
data model (3.2.36) that represents an abstract view of the real world

3.2.21
conceptual domain
CD
concept (3.2.18) that expresses its description or valid instance meanings

3.2.22
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** (3.2.71) and **optional** (3.2.89).

NOTE 2 Obligation statuses apply to metadata items with a **registration status** (3.2.112) of "Recorded" or higher.

3.2.23
contact
instance of a **role** (3.2.121) of an **individual** (3.2.65) or **organization** (3.2.90) (or **organization part** (3.2.93) or **organization Person** (3.2.95)) to or from whom an information item(s), a material object(s) and/or person(s) can be sent in a specified context

3.2.24
contact information
information to enable a **contact** to be located or communicated with

3.2.25
context
<designation and definition> setting in which a **designation** (3.2.51) or **definition** (3.2.39) is used

3.2.26
coordinate
measurement from the origin of a frame of reference

3.2.27
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]

3.2.28
data element
<in organization of data> unit of **data** (3.2.27) that is considered in context to be indivisible

EXAMPLE The data element “age of a person” with values consisting of all combinations of 3 decimal digits.

[ISO/IEC 2382-4:1999, 04.07.01]

NOTE The definition states that a data element is “indivisible” in some context. This means that it is possible that a data element considered indivisible in one context (e.g., telephone number) may be divisible in another context, (e.g., country code, area code, local number).

3.2.29
data element concept
concept (3.2.18) that is an **association** (3.1.2) of a **property** (3.2.100) with an **object class** (3.2.88)

NOTE 1 A data element concept is implicitly associated with both the property and the object class whose combination it expresses.

NOTE 2 A data element concept may also be associated with zero or more **conceptual domains** (3.2.21) each of which expresses its **value meanings** (3.2.141).

NOTE 3 A data element concept may also be associated with zero or more **data elements** (3.2.28) each of which provide representation for the data element concept via its associated **value domain** (3.2.140).

3.2.30
data element concept property
property (3.2.100) of a **data element concept** (3.2.29)

3.2.31
data element concept domain
conceptual domain (3.2.21) of a **data element concept** (3.2.29)

3.2.32
data element concept object class
object class (3.2.88) of a **data element concept** (3.2.29)

3.2.33
data element derivation
application of a **derivation rule** (3.2.45) to one or more input **data element**s (3.2.28) to derive one or more output data elements

3.2.34
data element example
representative illustration of a **data element** (3.2.28)

3.2.35
data element precision
degree of specificity for a **data element** (3.2.28)

NOTE Expressed as a number of decimal places to be used in any associated data element values.

3.2.36
data model
graphical and/or lexical representation of **data** (3.2.27), specifying their properties, structure and inter-relationships

3.2.37
date
datatype (3.1.8) whose values are points in time to the resolution: year, month, day

NOTE 1 Adapted from ISO/IEC 11404:2007, 8.1.6 date and time;

NOTE 2 See also 6.2.3 Date datatype.

3.2.38
datetime
datatype (3.1.8) whose values are points in time to the resolution: year, month, day, hour, minute, second, and optionally fractions thereof

NOTE 1 Adapted from ISO/IEC 11404:2007, 8.1.6 date and time;

NOTE 2 See also 6.2.4 Datetime datatype.

3.2.39
definition
representation of a **concept** (3.2.18) by a descriptive statement which serves to differentiate it from related concepts

[ISO 1087-1:2000, 3.3.1]

3.2.40
definition
<designatable item> representation of a **designatable item** (3.2.50) by a descriptive statement which, in a given **language** (3.2.68) and **context**(s) (3.2.25) serves to differentiate it from related designatable items

NOTE See also **definition** (3.2.39 above).

3.2.41
definition context
<designatable item> **context** (3.2.25) in which the **definition** (3.2.39) is applicable

3.2.42
definition language
language (3.2.68) used to write the **definition text** (3.2.44)

3.2.43
definition source reference
reference to the source from which the **definition** (3.2.39) is taken

3.2.44
definition text
text of the **definition** (3.2.39)

3.2.45
derivation rule
logical, mathematical, and/or other operations specifying derivation

3.2.46
derivation rule notation
notation (3.2.86) used to specify the **derivation rule** (3.2.45)

3.2.47
derivation rule specification
text of a specification of **data element derivation** (3.2.33)

3.2.48
described conceptual domain
conceptual domain (3.2.21) that is specified by a description or specification, such as a rule, a procedure, or a range (i.e. interval)

3.2.49
described value domain
value domain (3.2.140) that is specified by a description or specification, such as a rule, a procedure, or a range (i.e. interval)

3.2.50
designatable item
identified item (3.2.64) which can have **designation**s (3.2.51) and/or **definition**s (3.2.40)

3.2.51
designation
representation of a **concept** (3.2.18) by a **sign** (3.2.123) which denotes it

[ISO 1087-1:2000, 3.4.1]

NOTE See also 3.2.52 designation <designatable item>

3.2.52
designation
<designatable item> representation of a **designatable item** (3.2.50) by a **sign** (3.2.123) which denotes it

NOTE This contextualized designation is as specified in **designation** (3.2.51) or a **name** (3.2.83).

3.2.53
designation acceptability
rating of the acceptability of the **designation** (3.2.51) in the specified **context** (3.2.25)

3.2.54
designation context
<designatable item> **context** (3.2.25) in which a **designation** (3.2.51) is applicable

3.2.55
designation language
<designatable item> **language** (3.2.68) or dialect in which a **sign** (3.2.123), usually a **name** (3.2.83), is expressed

3.2.56
designation namespace
namespace (3.2.84) to which a **designation** (3.2.51) is bound

3.2.57
designation sign
<designatable item> **sign** (3.2.123) of the **designation** (3.2.51)

3.2.58
dimensionality
set of equivalent **unit**s **of measure** (3.2.138)

NOTE 1 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 2 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 ºC = (5/9)*(ºF - 32) and ºF = (9/5)*(ºC) + 32.
b) the same operations can be performed on both values.

NOTE 3 The units of measure "temperature in degrees Celsius" and "temperature in kelvins" do not belong to the same dimensionality. Even though it is easy to convert quantities from one unit of measure to the other (ºC = K - 273.15 and K = ºC + 273.15), the characterizing operations in kelvins include taking ratios, whereas this is not the case for degrees Celsius. For instance, 20K is twice as warm as 10K, but 20ºC is not twice as warm as 10ºC.

NOTE 4 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 5 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 6 ISO 31-0 specifies physical dimensions (e.g. length, mass, velocity). This part of ISO/IEC 11179 also permits 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.

3.2.59
disjoint
<set theory> having no elements in common

3.2.60
enumerated conceptual domain
conceptual domain (3.2.21) that is specified by a list of all its **value meaning**s (3.2.141)

NOTE No ordering of the value meanings is implied.

3.2.61
enumerated value domain
value domain (3.2.140) that is specified by a list of all its **permissible value**s (3.2.96)

NOTE No ordering of the permissible values is implied.

3.2.62
extension
<11179-3> feature not defined by this part of ISO/IEC 11179

<registry metamodel> **class** (3.1.5), **attribute** (3.1.4) or **relationship** (3.1.15) that an implementation of a **metadata registry** (3.2.78) provides that is not defined by this part of ISO/IEC 11179

NOTE This part of ISO/IEC 11179 specifies **slot**s (3.2.124) as a mechanism for extending **identified items** (3.2.64).

3.2.63
identification scheme
system for allocating **identifiers** (3.1.11) to registered **objects** (3.2.87)

[ISO/IEC 6523-1:1998, 3.6]

3.2.64
identified item
metadata item (3.2.75) identified in a **metadata registry** (3.2.78)

3.2.65
individual
single human being

3.2.66
integer
mathematical **datatype** (3.1.8) comprising the exact integral values

[ISO/IEC 11404:2007, 8.1.7]

3.2.67
international code designator
identifier (3.1.11) of an **organization identification scheme** (3.2.91)

NOTE 1 Adapted from ISO/IEC 6523-1:1998, 3.8.

NOTE 2 See also ISO/IEC 11179-6.

3.2.68
language
system of **sign**s (3.2.123) for communication, usually consisting of a vocabulary and rules

[ISO 5127:2001, 1.1.2.01]

3.2.69
link
member of a **relation** (3.2.119)

3.2.70
link end
end of a **link** (3.2.69), identifying the **relation role** (3.2.119) played by a **concept** (3.2.18) in the link

3.2.71
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** (3.2.22) and **optional** (3.2.89);

NOTE 2 Obligation statuses apply to **metadata items** (3.2.75) with a **registration status** (3.2.112) of "recorded" or higher.

3.2.72
measure class
set of equivalent **units of measure** (3.2.138) for association with one or more **dimensionalities** (3.2.58)

3.2.73
meronomy
type of hierarchy which deals with part-whole relationships

NOTE cf. **taxonomy** (3.2.135)

3.2.74
metadata
data (3.2.27) that defines and describes other data

3.2.75
metadata item
instance of a **metadata object** (3.2.76)

NOTE 1 In all parts of ISO/IEC 11179, this term is applied only to instances of metadata objects described by the metamodel in Clauses 5 through 11 of this part of ISO/IEC 11179. Examples include instances of *Data_Elements* (11.5.2.1), *Data_Element_Concepts* (11.2.2.3), *Permissible_Values* (11.3.2.7) etc.

NOTE 2 A metadata item has associated **attributes** (3.1.4), as appropriate for the metadata object it instantiates.

3.2.76
metadata object
object type defined by a **metamodel** (3.2.80)

NOTE In all parts of ISO/IEC 11179, this term is applied only to metadata objects described by the metamodel in Clauses 5 through 11 of this part of ISO/IEC 11179. Examples include *Data_Elements* (11.5.2.1), *Data_Element_Concepts* (11.2.2.3), *Permissible_Values* (11.3.2.7) etc.

3.2.77
metadata register
information store or database maintained by a **metadata registry** (3.2.78)

3.2.78
metadata registry
MDR
information system for registering **metadata** (3.2.74)

NOTE The associated information store or database is known as a **metadata register** (3.2.77).

3.2.79
metadata registry product
particular information system for implementing a **metadata registry** (3.2.78)

3.2.80
metamodel
model that specifies one or more other models

3.2.81
metamodel construct
unit of **notation** (3.2.86) for modelling

NOTE The metamodel constructs used in this part of ISO/IEC 11179 are defined in 3.1.

3.2.82
multiplicity
specification of the range of allowable **cardinalities** (3.2.13) that a set may assume

NOTE 1 Multiplicity specifications may be given for roles within **association**s (3.1.2)

NOTE 2 A multiplicity is a (possibly infinite) subset of the nonnegative integers

NOTE 3 Adapted from ISO/IEC 19501:2005, Glossary

3.2.83
name
designation (3.2.51) of an **object** (3.2.87) by a linguistic expression

[ISO 1087:1990, 5.3.1.3 and ISO/IEC 15944-1:2002, 3.35]

3.2.84
namespace
set of **designation**s (3.2.51) and/or **scoped identifier**s (3.2.122) for a particular business need

NOTE The term **namespace** is used in this International Standard because it is in common use, even though the concept is being applied to identifiers as well as names.

3.2.85
naming convention
specification of how **signs** (3.2.123) of **designation**s (3.2.51) and/or **scoped identifier**s (3.2.122) are formulated

NOTE A naming convention can apply to scoped identifiers when they are included in the associated **namespace** (3.2.84)

3.2.86
notation
formal syntax and associated semantics

EXAMPLES UML, MOF, OCL, OWL/RDF, SKOS, CGIF, XCL, XTM, or ISO/IEC 11404

NOTE Formal syntax is often intended for machine processing.

3.2.87
object
anything perceivable or conceivable

NOTE 1 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);

NOTE 2 Adapted from ISO 1087-1:2000, 3.1.1.

3.2.88
object class
set of ideas, abstractions or things in the real world that are identified with explicit boundaries and meaning and whose properties and behaviour follow the same rules

3.2.89
optional
permitted but not required

NOTE 1 One of three obligation statuses applied to the **attribute**s (3.1.4) of **metadata item**s (3.2.75), indicating the conditions under which the attribute is required. See also **conditional** (3.2.22) and **mandatory** (3.2.71).

NOTE 2 Obligation statuses apply to **metadata item**s (3.2.75) with a **registration status** (3.2.112) of "recorded" or higher.

3.2.90
organization
unique framework of authority within which **individual**s (3.2.65) act, or are designated to act, towards some purpose

NOTE 1 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.

NOTE 2 Adapted from ISO/IEC 6523-1:1998, 3.1

3.2.91
organization identification scheme
identification scheme (3.2.63) dedicated to the unique identification of **organization**s (3.2.90)

[ISO/IEC 6523-1:1998, 3.7]

3.2.92
organization identifier
identifier (3.1.11) assigned to an **organization** (3.2.90) within an **organization identification scheme** (3.2.91), and unique within that scheme

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

3.2.93
organization part
any department, service or other entity within an **organization** (3.2.90) which needs to be identified for information exchange

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

3.2.94
organization part identifier
opi
identifier (3.1.11) allocated to a particular **organization part** (3.2.93)

NOTE See also ISO/IEC 11179-6.

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

3.2.95
organization Person
organization part (3.2.93) which has the properties of a **Person** (3.2.97) and thus is able to make commitments on behalf of that **organization** (3.2.90)

NOTE 1 An **organization** can have one or more **organization Person**s.

NOTE 2 An **organization Person** is deemed to represent and act on behalf of the **organization** and to do so in a specified capacity.

NOTE 3 An **organization Person** can be a "natural person" such as an employee or officer of the **organization**.

NOTE 4 An **organization Person** can be a “legal person”, i.e., another **organization**.

[ISO/IEC 15944-1:2002, 3.46]

3.2.96
permissible value
designation (3.2.51) of a **value meaning** (3.2.141)

NOTE A permissible value may be associated with one or more **enumerated value domain**s (3.2.61).

3.2.97
Person
entity, i.e., a natural or legal person, recognized by law as having legal rights and duties, able to makecommitment(s), assume and fulfil resulting obligation(s), and able of being held accountable for its action(s)

NOTE 1 Synonyms for "legal person" include "artificial person", "body corporate", etc., depending on the terminology used in competent jurisdictions.

NOTE 2 Person is capitalized to indicate that it is being utilized as formally defined in the standards and to differentiate it from its day-to-day use.

NOTE 3 Minimum and common external constraints applicable to a business transaction often require one to differentiate among three common subtypes of Person, namely “individual”, “organization”, and “public administration”

[ISO/IEC 15944-1:2002, 3.47]

3.2.98
phone number
telephone number

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

3.2.99
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]

3.2.100
property
quality common to all members of an **object class** (3.2.88)

3.2.101
quantity
value with an associated **unit of measure** (3.2.138)

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

3.2.102
reference document
document that provides pertinent details for consultation about a subject

3.2.103
reflexivity
characterization of a **binary relation** (3.2.10) as: reflexive, irreflexive or antireflexive

NOTE 1 A binary relation, R, is reflexive if for all x, R(x,x) is true. Equality is an example of a reflexive relation.

NOTE 2 A binary relation, R, is irreflexive if it is not reflexive. i.e., R(x,x) is not necessarily true for all x.

NOTE 3 A binary relation, R, is antireflexive if for all x, R(x,x) is false. Inequality is an example of an antireflexive relation. An antireflexive relation is also irreflexive, but antireflexive is a more specific characterization.

3.2.104
register
information store or database maintained by a **registry** (3.2.113)

3.2.105
registered item
metadata item (3.2.75) that is recorded and managed in a **metadata registry** (3.2.78)

3.2.106
registrar
representative of a **registration authority** (3.2.109)

3.2.107
registrar identifier
identifier (3.1.11) for the **registrar** (3.2.106)

3.2.108
registration
<generic>inclusion of an item in a **registry** (3.2.113)

<metadata registry> inclusion of a **metadata item** (3.2.75) in a **metadata registry** (3.2.78)

3.2.109
registration authority
RA
organization (3.2.90) responsible for maintaining a **register** (3.2.104)

3.2.110
registration authority identifier
identifier (3.1.11) assigned to a **registration authority** (3.2.109)

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

3.2.111
registration state
<registration> information about the **registration** (3.2.108) of an **administered item** (3.2.2)

3.2.112
registration status
designation (3.2.51) of the status in the **registration** (3.2.108) life-cycle of an **administered item** (3.2.2)

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

3.2.113
registry
information system for **registration** (3.2.108)

3.2.114
registry instance
<generic> implementation of a **registry product** (3.2.117) and instance of a **registry** (3.2.113)

<metadata registry> implementation of a **metadata registry product** (3.2.79) and instance of a **metadata registry** (3.2.78)

3.2.115
registry item
<generic> item recorded in a **registry** (3.2.113)

<metadata registry> **metadata item** (3.2.75) recorded in a **metadata registry** (3.2.78)

3.2.116
registry metamodel
metamodel (3.2.80) specifying the model for a **registry** (3.2.113)

3.2.117
registry product
particular information system for implementing a **registry** (3.2.113)

3.2.118
related metadata reference
reference from one **metadata item** (3.2.75) to another

NOTE A **registration authority** (3.2.109) could choose to use a *Reference_Document* (8.1.3.3), an *administrative_note* (8.1.2.6.2.4) or an *explanatory_comment* (8.1.2.2.3.4) to record a related metadata reference.

3.2.119
relation
sense in which **concept**s (3.2.18) may be connected, via constituent roles.

EXAMPLE causality is a relation with two constituent roles: cause and effect.

NOTE The related concepts may be general or individual concepts

3.2.120
relation role
role that a **concept** (3.2.18) plays in a **relation** (3.2.119)

3.2.121
role
specified responsibilities

3.2.122
scoped identifier
identifier (3.1.11) of an **identified item** (3.2.64) within a specified **namespace** (3.2.84)

NOTE A **namespace** provides the scope within which the **scoped identifier** uniquely identifies the **identified item**.

3.2.123
sign (noun)
textual string or symbol that can be used to denote a **concept** (3.2.18)

3.2.124
slot
container for an **extension** (3.2.62) to an **identified item** (3.2.64)

3.2.125
stewardship
<metadata> responsibility for the maintenance of **administrative information** (3.2.3) applicable to one or more **administered item**s (3.2.2)

NOTE The responsibility for the **registration** (3.2.108) of **metadata** (3.2.74) may be different from the responsibility for **stewardship** of **metadata**.

3.2.126
stewardship contact
contact (3.2.23) information associated with a **stewardship** (3.2.125)

3.2.127
stewardship organization
organization (3.2.90) that maintains **stewardship** (3.2.125) of an **administered item** (3.2.2)

3.2.128
stewardship record
record of a **stewardship organization** (3.2.127) and a **stewardship contact** (3.2.126) involved in the **stewardship** (3.2.125) of an **administered item** (3.2.2)

3.2.129
string
family of **datatype**s (3.1.8) which represent strings of symbols from standard character-sets

[ISO/IEC 11404:2007 10.1.5 Character String]

NOTE The syntax and semantics of the String datatype (6.2.11) are as defined in ISO/IEC 11404:2007 10.1.5 Character String

3.2.130
submission
act of submitting a **metadata item** (3.2.75) for registration in a **metadata registry** (3.2.78)

3.2.131
submission contact
contact (3.2.23) information associated with a **submission** (3.2.130)

3.2.132
submission organization
organization (3.2.90) that submits a **metadata item** (3.2.75) for **registration** (3.2.108)

3.2.133
submission record
record of a **submission organization** (3.2.132) and a **submission contact** (3.2.131) involved in the **submission** (3.2.130) of a **metadata item** (3.2.75) for **registration** (3.2.108)

3.2.134
symmetry
characterization of a **binary relation** (3.2.10) as: symmetric, asymmetric or antisymmetric

NOTE 1 A binary relation, R, is symmetric if for all x, y: R(x,y) implies R(y,x).

EXAMPLE Symmetric relations include: 'equals', 'not equals', 'within-2-miles-of', etc.

NOTE 2 Symmetry does not imply **reflexivity** (3.2.103).

EXAMPLE the 'inequality' relation is symmetric, but antireflexive.

NOTE 3 A binary relation, R, is asymmetric if for all x,y: R(x,y) does not imply R(y,x).

EXAMPLE Asymmetric relations include: less than, likes, father of, etc.

NOTE 4 A binary relation, R, is anti-symmetric if for all x,y: R(x,y) implies not R(y,x). 'An antisymmetric relation is also asymmetric, but antisymmetric is a more specific characterization.

EXAMPLE 'less than' is an antisymmetric relation.

NOTE 5 An asymmetric relation is not necessarily antisymmetric

EXAMPLE Less than or equals.

3.2.135
taxonomy
type of hierarchy which deals with generalization/specialization relationships.

NOTE cf. **meronomy** (3.2.73)

3.2.136
text
paragraph, page or document that can be used to define or describe an **object** (3.2.87)

NOTE 1 Text is **data** (3.2.27) 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

NOTE 2 Adapted from ISO/IEC 2382-23:1994

NOTE 3 See also the Text datatype (6.2.12).

3.2.137
transitive relation
binary relation (3.2.10) *R* over a set *X* is transitive if whenever an element *a* is related to an element *b*, and *b* is in turn related to an element *c*, then *a* is also related to *c*.

3.2.138
unit of measure
⟨value domain⟩ actual units in which the associated values are measured

NOTE 1 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. This part of ISO/IEC 11179 permits the use of any appropriate system of measurement.

NOTE 2 The **dimensionality** (3.2.58) of the associated **conceptual domain** (3.2.21) must be appropriate for the specified **unit of measure**.

3.2.139
unit of measure dimensionality
dimensionality (3.2.58) that specifies the equivalence relation that applies to all values representing a particular unit

3.2.140
value domain
VD
set of **permissible value**s (3.2.96)

NOTE 1 The **value domain** provides representation, but has no implication as to what **data element concept** (3.2.29) the values might be associated with nor what the values mean

NOTE 2 The **permissible value**s may either be enumerated or expressed via a description.

3.2.141
value meaning
semantic content of a possible value

NOTE The representation of value meanings in a **registry** (3.2.113) shall be independent of (and shall not constrain) their representation in any corresponding **value domain** (3.2.140).

3.2.142
version
unique version **identifier** (3.1.11) of the **scoped identifier** (3.2.122)

3.3 Abbreviated terms

3.3.1
CD
Conceptual Domain

3.3.2
CL
Common Logic

3.3.3
CLIF
Common Logic Interchange Format

3.3.4
DE
Data Element

3.3.5
DEC
Data Element Concept

3.3.6
MDR
metadata registry

3.3.7
mece
mutually exclusive and collectively exhaustive

3.3.8
opi
organization part identifier

3.3.9
ORM
Object Role Modelling

3.3.10
OWL
Web Ontology Language

3.3.11
OWL-DL
OWL Description Logic

3.3.12
RA
Registration Authority

3.3.13
RDF
Resource Description Framework

3.3.14
SBVR
Semantics of Business Vocabulary and Business Rules

3.3.15
SKOS
Simple Knowledge Organization System

3.3.16
Turtle
Terse RDF Triple Language

3.3.17
UML
Unified Modeling Language

3.3.18
UPU
Universal Postal Union

3.3.19
URI
Universal Resource Identifier

3.3.20
VD
Value Domain

3.3.21
W3C
World Wide Web Consortium

3.3.22
XCL
eXtended Common Logic markup language

3.3.23
XML
eXtensible Markup Language

3.3.24
XTML
eXtensible Topic Maps

4 Conformance

4.1 Overview of 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 (see clause 12) for situations 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. Conformance claims shall specify a Degree and a Level of Conformance, as described below.

Conformance statements with respect to this standard must also be explicit as to which portions of this standard conformity is being claimed. This may be done in some cases simply by reference to one or more of the clauses. In other cases, conformance may instead be claimed to one or more of the standard profiles (see Annex H), which specify combinations of multiple clauses, and how they are to be fitted together.

When a **registry product** (3.2.117) makes a conformance claim, the product must support all the associated functionality, and must enable the enforcement of the associated constraints. When a **registry instance** (3.2.114) makes a conformance claim, it must actually enforce the associated constraints.

4.2 Degree of conformance

4.2.1 General

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 might be limited in usefulness but is maximally interoperable with respect to this part of ISO/IEC 11179. A conforming implementation might be more useful, but might be less interoperable with respect to this part of ISO/IEC 11179.

4.2.2 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 might cause undefined behavior.

4.2.3 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 might cause undefined behavior.

4.3 Conformance by clause

Conformance claims may be limited to individual clauses 6-12 of this standard. Clauses 7-11 are all dependent upon one or more other clauses of this standard (see Figure 1 on p. 26), so conformance to any of these clauses must be understood to imply conformance also to relevant provisions specified in one or more preceding clauses.

Conformance may be claimed for a set of data structures and/or datatypes for clauses:

— clause 7 Identification, Designation and Definition

— clause 9 Concepts

— clause 10 Binary Relations

— clause 11 Data Description .

Conformance may be claimed for a register or registry (a set of administered metadata), or for a registry software system for clauses:

— clause 8 Registration

— clause 12 Basic attributes.

4.4 Registry conformance

4.4.1 Overview

Registries shall claim conformance to either clause 8 or clause 12. Registries conforming to clause 8 may additionally claim conformance to one of the standard profiles specified in clause Annex H, each of which incorporates some subset of the clauses 7 and 9-11.

4.4.2 Standard profiles for edition 3 registries

Implementers of either a generic registry platform (software system) which is customizable for a range of registered content types, or of a registry or registry software system supporting some specific range of registered content types not specified in this standard, might simply claim conformance to clause 8. Alternatively, the following standard profiles are defined, specifying how additional clauses should be combined with clause 8:

— **Concept Systems Registry**: Implements clauses 7-9, and also satisfies the following additional provisions:

  — All Concepts and Concept_Systems must be Registered_Items.

  — All Registered_Items must also be Designatable_Items and Classifiable_Items.

— **Extended Concept Systems Registry**: Implements clause 10, in addition to all provisions of the Concept Systems Registry profile.

— **Metadata Registry**: Implements clause 11, in addition to all provisions of the Concept Systems Registry profile, and also satisfies the following additional provision:

  — All Data_Elements, Value_Domains, and Derivation_Rules must be Registered_Items.

— **Extended Metadata Registry**: Implements all of clauses 7-11, and also satisfies all provisions of the Metadata Registry profile.

4.5 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.

4.6 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;

b) which clauses are supported;

c) which registry type is supported

d) what extensions, if any, are supported or used.

4.7 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.

5 Structure of a metadata registry

5.1 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 information model 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 class. The metamodel need not be physically implemented as specified.

The structure described by this metamodel may be distributed over several implementations. These implementations might be databases, data repositories, metadata registers, metadata registries, dictionaries, etc. No particular technology is implied. Implementations may utilize technologies including, but not limited to: relational database, XML database, object oriented systems, or RDF/OWL.

The model shows constraints on minimum and maximum occurrences (the obligation) 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.

5.2 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;

— to promote a common understanding of the data described;

— to provide the specification at a conceptual level to facilitate 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 particular metadata management tool sets can be developed. The metamodel constructs can be translated into the language of each tool set, preserving the concepts represented in the metamodel.

An implementer may 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.

5.3 Specification of the metamodel

5.3.1 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 part of ISO/IEC 11179 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) Version 2.4.1. This part of ISO/IEC 11179 uses the term **metamodel construct** (3.2.81) for the UML model constructs it uses, but **metadata object** (3.2.76) for the model constructs it specifies. The metamodel constructs used are: classes, associations, association classes, attributes, composite attributes and composite datatypes, and these are defined in 3.1. The specified metadata objects are defined in clauses 6 through 11. The concepts that the metadata objects represent are defined in clause 3.2.

However, there are certain parallels between the two metamodels. For example, the *Object_Class* (11.2.2.1) specified in the registry metamodel is similar to the metamodel construct **class** (3.1.5), and the *Property* (11.2.2.2) specified in the registry metamodel is similar to the metamodel construct **attribute** (3.1.4). 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 part of ISO/IEC 11179 uses at both levels is “datatype”, but the level to which it applies should be apparent from the context in which it is used.

5.3.2 Choice of fonts

In clauses 6 through 11 this specification uses:

— **bold** font to highlight terms which represent concepts defined in clause 3;

— *italic* font to highlight terms which represent metadata objects specified by the metamodel.

EXAMPLE *Concept* (9.1.2.1) is a class each instance of which models a **concept** (3.2.18).

5.3.3 Use of UML Packages

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

- Basic package (clause 6) – contains simple classes that are reused by other packages

- Identification, designation and definition package (clause 7) – contains classes that enable the contents of a registry to be identified, named or otherwise designated, and defined. This package is sub-divided into two regions:
  - Identification metamodel region (see 7.2)
  - Designation and Definition metamodel region (see 7.3)
- Registration package (clause 8) – contains classes that enable metadata items to be registered. These are specified as the Registration metamodel region (see 8.1).
- Concepts package (clause 9) – contains classes that enable concepts to be related. This package is sub-divided into two regions:
  - Concepts metamodel region (see 9.1)
  - Classification metamodel region (see 9.2)
- Binary Relations package (clause 10) – contains a specialization of the Relations class from the Concepts package, specifically for binary relations. The separation is for conformance purposes.
- Data Descriptions package (clause 11) – contains classes that enable the description of specific metadata objects:
  - Data Element Concept metamodel region (see 11.2)
  - Conceptual and Value_Domain metamodel region (see 11.3)
  - Measurement metamodel region (see 11.4)
  - Data_Element metamodel region (see11.5).

5.3.4 Package dependencies

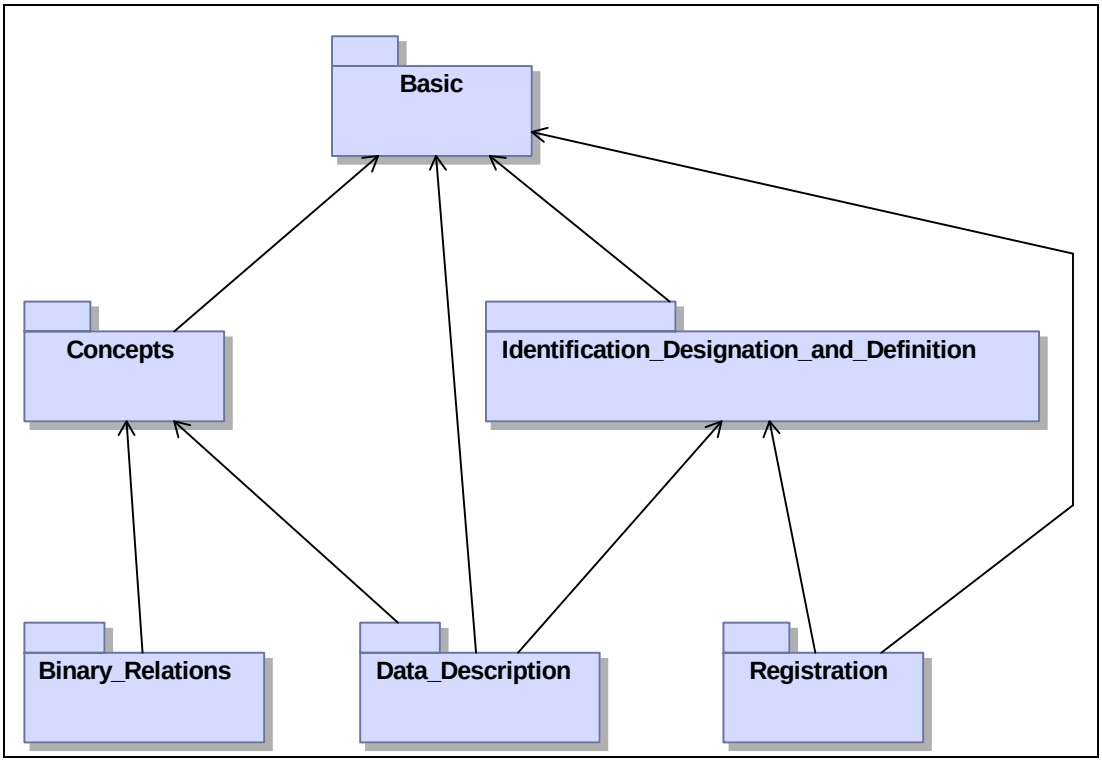

Figure 1 — Package dependencies

Figure 1 illustrates the dependencies among the packages. The lines in the figure illustrate dependencies in the direction of the arrow. In order to implement a package that has dependencies, the packages on which it is dependent must also be implemented. The dependencies are of three types:

a) Subclassing from classes in another package,
e.g. *Conceptual_Domain* (11.2.2.4), *Data_Element_Concept* (11.2.2.3), *Object_Class* (11.2.2.1) and *Property* (11.2.2.2) in the Data Element Concept region (11.2) of the Data_Description package (11) are all subclassed from the Concept class (9.1.2.1) in the Concepts package (9).

b) Relationship between classes,
e.g. *Registered_Item* (8.1.2.1) in the Registration package (8) has a relationship with *Reference_Document* (6.3.7) in the Basic package (6).

c) Some attributes use a predefined datatype or a class from another package as a datatype
e.g. the *contact* attribute (8.1.2.7.2.2) of the *Stewardship_Record* class (8.1.2.7) in the Registration package (8) uses the *Contact* class (6.3.2) of the Basic package (6) as a datatype.

Conformance options are specified in clause 4 and standard conformance profiles in Annex H.

5.3.5 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. Further, if a conflict exists between a formal definition and other normative text, the formal definition takes precedence until such time as a correction is made to make them consistent. For associations, the description of the association itself takes precedence over any description in the associated classes.

A consolidated UML class hierarchy is included as Annex B.

5.4 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.

NOTE In UML, **subclass**es (3.1.17) of a **superclass** (3.1.18) are by default not **disjoint** (3.2.59). This standard specifies when subclasses are required to be disjoint. Further, when the list of subclasses is intended to be exhaustive, this standard shows the superclass as abstract, thus preventing any other subclass being instantiated. An abstract class is indicated by showing the class name in *italics* in any class diagram that uses it.

Clauses 6 through 11 of this part of ISO/IEC 11179 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* (referred to as *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 metadata object types.

NOTE ISO/IEC TR 10032:2003 Reference model for data management explains the concepts of different levels of modelling, as does ISO/IEC 10027:1990 IRDS Framework.

5.5 Types of items in an ISO/IEC 11179 metadata registry

5.5.1 Overview of types of items

Figure 2 shows the types of items specified by this Part of this International Standard. These types are explained in subsequent clauses. In the Figure, the notation <<type>> indicates the use of the <<type>> stereotype as specified in ISO/IEC 19505-2:2012 OMG UML Part 2: Superstructure Annex C.1 Standard Profile L2.

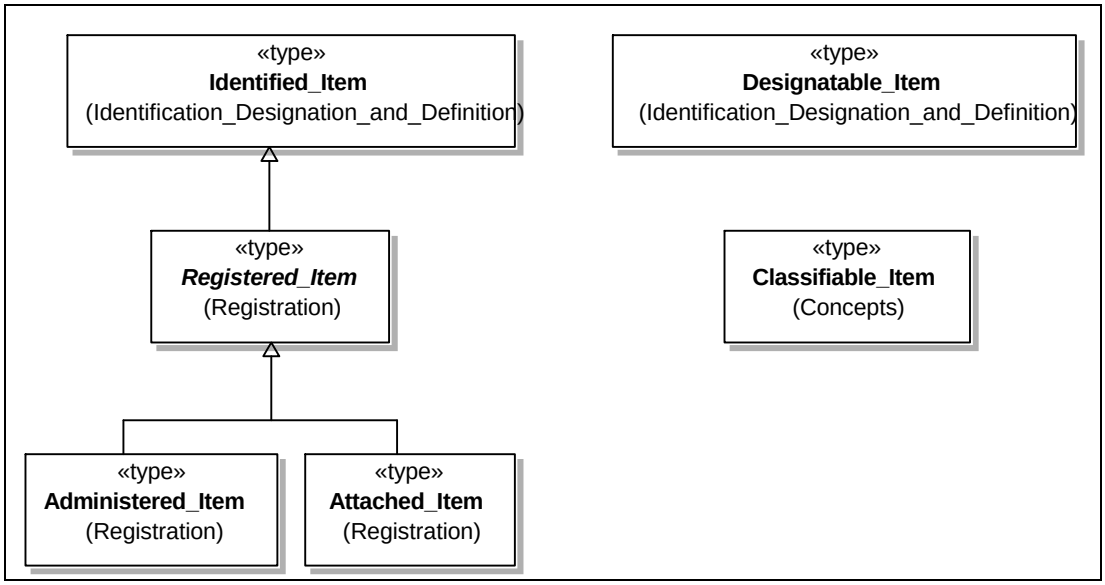

Figure 2 — Types of items

Any **metadata item** (3.2.75) entered into a **metadata registry** (3.2.78) may be extended by one or more of the above types, as follows:

— Any metadata item that is to be retrieved directly (as opposed to indirectly through a related item), shall be an *Identified_Item* (see 7.2.2.1), so the item can be referenced. An example of metadata items that might not be explicitly identified are the *Permissible Values* (11.3.2.7) within a *Value Domain* (11.3.2.5).

— Any metadata item that is to be designated (named) and/or defined shall be a *Designatable_Item* (7.3.2.2).

NOTE The separation of Designation and Definition from Identification has been done to better harmonize with ISO/IEC 19763-2, in which ModelElements are identified and are administered, but are not (required to be) designated or defined.

— Any metadata item that is to be registered in the registry shall be a *Registered_Item* (8.1.2.1). *Registered_Item* is an abstract class, which means that each such item must be instantiated as one of the subclasses: *Administered_Item* (8.1.2.2), or *Attached_Item* (8.1.2.3). These subclasses are mutually exclusive and collectively exhaustive (mece).

— Any metadata item that is to be classified in a **classification scheme** (3.2.16) shall be a *Classifiable_Item* (see 9.2.2.1).

A *Registration_Authority* (8.1.2.5) responsible for the registry shall determine which metadata items should become *Identified_Items* (7.2.2.1), *Registered_Items* (8.1.2.1), *Designatable_Items* (7.3.2.2) and/or *Classifiable_Items* (9.2.2.1), within the contraints of any conformance claim (see clause 4.) that is made for the registry.

NOTE The precise mechanism by which metadata items are extended by the above types is implementation-defined.

Any metadata item that has been extended by one or more of the above types shall inherit the relationships of the type(s), which can thus be navigated from the item (e.g. the *identification* (7.2.3.1) association from *Identified_Item* (7.2.2.1).

5.5.2 Rules for types of items

Table 1 – Rules for Types of Items

Rule #	Rule Description
1	A metadata item is identified if it has one or more *identification* (7.2.3.1) associations, each with a *Scoped_Identifier* (7.2.2.2) class.
2	A metadata item is designated if it has one or more *item_designation* (7.3.4.4) associations, each with a *Designation* (7.3.2.3) class.
3	A metadata item is defined if it has one or more *item_definition* (7.3.4.3) associations, each with a *Definition* (7.3.2.4) class.
4	A metadata item is classified if it has one or more *Classification* (9.2.3.1) association classes, each with a *Concept* (9.1.2.1) class in a *Concept_System* (9.1.2.2).
5	A metadata item is registered if it has one or more *submission* (8.1.6.5) associations, each with a *Submission_Record* (8.1.2.8) class. The metadata item must also be identified as described by rule 1 and either administered as described by rule 6, or attached as described by rule 7.
6	A metadata item is administered if it has exactly one *stewardship* (8.1.6.4) association with a *Stewardship_Record* (8.1.2.7) class, and one or more *Registration* (8.1.5.1) association classes, each with a *Registration_Authority* (8.1.2.5) class. The metadata item must also be registered as described by rule 5. The metadata item cannot also be attached as described by rule 7.
7	A metadata item is attached if it has exactly one *attachment* (8.1.6.1) association with an *Administered_Item* (8.1.2.2) class. The metadata item cannot also be administered as described by rule 6.

Table 2 on p.10 is a compressed decision table that shows the rules that apply to the various types of item.

The upper section of the table lists the associations that apply to each rule. The lower section of the table indicates the state of a metadata item that conforms to each rule.

For example, reading from the top section of the table, Rule 1 states that if a metadata item has one or more instances of an *identification* association with a *Scoped_Identifier* class, then according to the bottom section of the table for Rule 1, the metadata item is identified.

Table 2 – Rules for Types of Items as a Decision Table

Metadata item has instance of:	**Rule 1**	**Rule 2**	**Rule 3**	**Rule 4**	**Rule 5**	**Rule 6**	**Rule 7**
identification association with *Scoped_Identifier* class	one or more				one or more	one or more	one or more
item_designation association with *Designation* class		one or more					
item_definition association with *Definition* class			one or more				
Classification association class with a *Concept* class in a *Concept_System*				one or more			
submission association with a *Submission_Record* class					one or more	one or more	one or more
stewardship association with a *Stewardship_Record* class						one	none
Registration association class with a *Registration_Authority* class						one or more	none
attachment association with an *Administered_Item* class						none	one
Metadata item is identified	Yes				Yes	Yes	Yes
Metadata item is designated		Yes					
Metadata item is defined			Yes				
Metadata item is classified				Yes			
Metadata item is registered					Yes	Yes	Yes
Metadata item is administered						Yes	
Metadata item is attached							Yes

5.6 Extensibility

It is not expected that this metamodel will completely accommodate all users. Particular sectors, such as document management, scientific data and statistical data, require metadata attributes not addressed in this standard. This standard provides *Slots* (see 7.2.2.4) as a mechanism to extend metadata items with custom attributes. Classes, relationships, and attributes may be added as extensions to existing packages in this conceptual data model, or complete new packages may be added.

Implementers of this standard may include extensions as part of an implementation, and/or they may provide facilities to enable a registry user to define their own extensions, such as classes and/or packages. An implementation with such extensions shall be considered conformant if it does not violate any of the rules inherent in the structure and content as specified by the metamodel in this standard.

5.7 Date references

In this standard, dates are important attributes of an *Administered_Item* (8.1.2.2) and of operations of a registry. For the purpose of this standard, "date" refers to Gregorian calendar date {see ISO 8601:2004}. (See also 6.2.3 for the specification of the associated datatypes.)

6 Basic package

6.1 Overview of Basic package

The Basic package specifies common datatypes and classes for use elsewhere in the metamodel. The contents of the package are grouped into two regions: "basics types" and "basic classes", as described below.

6.2 Basic Types metamodel region

6.2.1 Overview of Basic Types

The Basic Types metamodel region specifies the **datatype**s (3.1.8) used in the specification of the **metamodel** (3.2.80). A datatype is a set of distinct values, characterized by properties of those values and by operations on those values (ISO/IEC 11404). 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.

NOTE The datatypes that are described in this section are used in specification of the metamodel itself, and are not intended to constrain the datatypes that may be used in 11.3.2.9.

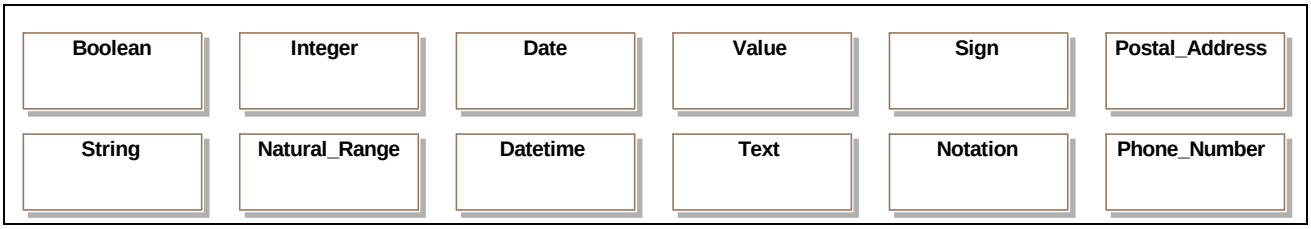

Figure 3 — Basic types metamodel region

6.2.2 Boolean datatype

A mathematical **datatype** (3.1.8) associated with two-valued logic. [ISO/IEC 11404:2007, 8.1.1 Boolean].

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

6.2.3 Date datatype

A **datatype** (3.1.8) whose values are points in time to the resolution: year, month, and day

[ISO/IEC 11404:2007, 8.1.6 Date-and-Time].

6.2.4 Datetime datatype

A **datatype** (3.1.8) whose values are points in time to the resolution: year, month, day, hour, minute, second, and optionally fractions of seconds.

[ISO/IEC 11404:2007, 8.1.6 Date-and-Time].

6.2.5 Integer datatype

A mathematical **datatype** (3.1.8) comprising the exact integral values [ISO/IEC 11404:2007, 8.1.7 Integer].

NOTE Both the notation and semantics of the Integer datatype is as specified in ISO/IEC 11404:2007:8.1.7.

6.2.6 Natural_Range datatype

Natural_Range is a **datatype** (3.1.8) comprising a range of “natural numbers“, i.e. the positive integers, including zero. Any instance of *Natural_Range* is one of:

— a constant non-negative Integer

— a bounded range of non-negative Integers defined by a minimum and a (strictly larger) maximum value

— an unbounded range defined by only a non-negative minimum (e.g., 0..*, 1..*, 2..*).

NOTE *Natural_Range* is used as the type of both *multiplicity* (an attribute of *Relation Roles*) and *arity* (an attribute of *Relations*) in the Concepts metamodel region, but this in no way constrains how it may be used by a Registration Authority.

6.2.7 Notation datatype

Notation denotes a **notation** (3.2.86) defined elsewhere, but used by an item within the registry. A notation defines a formal syntax and semantics, meant for machine processing. In this metamodel, *Notation* is used by *Concept_System* (9.1.2.2), *Derivation_Rule* (11.5.2.6) and *Reference_Document* (6.3.7).

EXAMPLES XCL Common Logic (ISO/IEC 24707) or OWL-DL XML notation.

6.2.8 Phone_Number datatype

A **phone number** (3.2.98) 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/IEC 19773 Information technology – Metadata registries (MDR) Modules – Module 17: Data structure for ITU-T E.164 phone number data.

NOTE ISO/IEC 19773 is referenced but not required.

6.2.9 Postal_Address datatype

A **postal address** (3.2.99) enables 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.

NOTE ISO/IEC 19773 is referenced but not required.

6.2.10 Sign datatype

A **sign** (3.2.123) may be a character string, graphic image, sound clip or other symbol that can be used to denote or designate a **concept** (3.2.18). The *Sign* **datatype** (3.1.8) may be implemented using the Reflit(of_type) data structure (see ISO/IEC 19773:2011 10.4.2), where the list of supported types is implementation defined. At a minimum, datatype String must be supported.

6.2.11 String datatype

String is a family of **datatype**s (3.1.8) which represent strings of symbols from standard character-sets. The syntax and semantics of the *String* datatype are as defined in ISO/IEC 11404:2007 10.1.5 Character String.

6.2.12 Text datatype

Text is **data** (3.2.27) 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.

6.2.13 Value datatype

A *Value* represents any instance of any **datatype** (3.1.8).

6.3 Basic Classes metamodel region

6.3.1 Overview of Basic Classes

The Basic Classes metamodel region specifies classes which are used as datatypes within the metamodel.

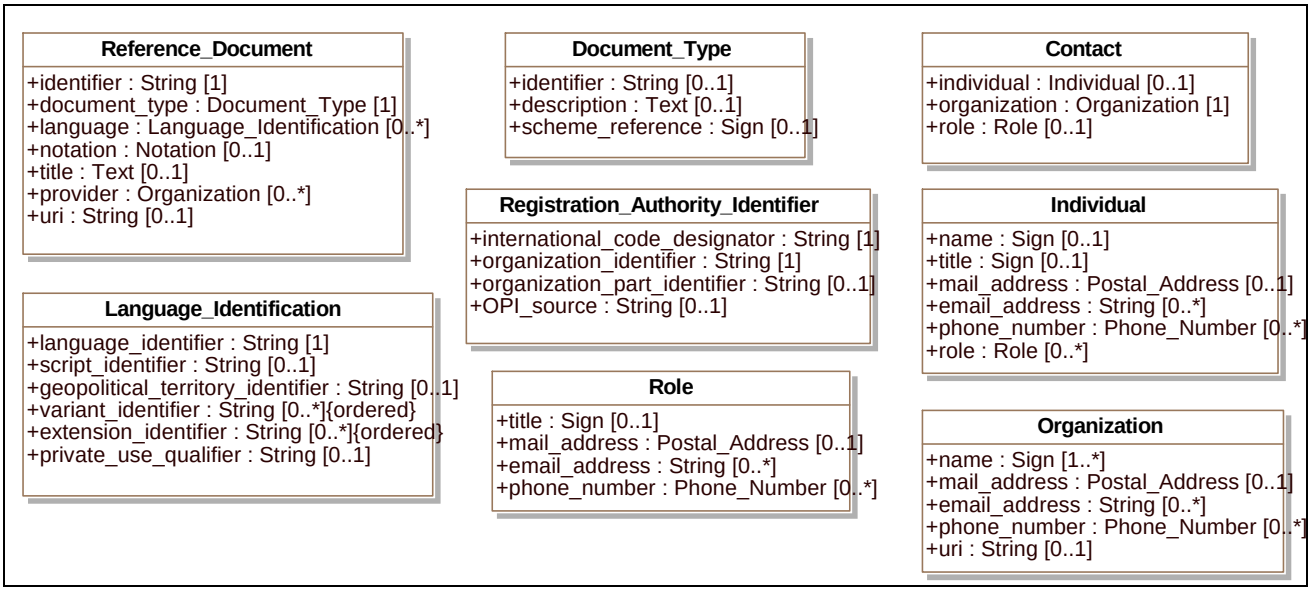

Figure 4 — Basic classes metamodel region

6.3.2 Contact class

6.3.2.1 Description of *Contact*

Contact is a class each instance of which models a **contact** (3.2.23), which specifies a **role** (3.2.121) and/or an **individual** (3.2.65) within an **organization** (3.2.90) or an **organization part** (3.2.93) to whom information item(s), material object(s) and/or person(s) can be sent to or from. *Registrar* (8.1.2.4) is a subclass of *Contact*.

The attributes of the *Contact* class are summarized here and specified more formally in 6.3.2.2.

— Every *Contact* shall have exactly one *organization* (6.3.2.2.2) of type *Organization* (6.3.6) for which the *Contact* is a representative.

— Every *Contact* shall have one *individual* (6.3.2.2.1) of type *Individual* (6.3.4) or one *role* (6.3.2.2.3) of type *Role* (6.3.9), or both. The individual is the *Individual* that is the *Contact*. The role is the *Role* that is the *Contact*.

6.3.2.2 Attributes of *Contact*

6.3.2.2.1 individual

Attribute name:	*individual*
Definition:	*Individual* that is the *Contact*
Obligation:	Conditional
Multiplicity:	*0..1*
Datatype:	*Individual* (6.3.4)
Condition:	Either an *Individual* or a *Role* or both shall be specified.

6.3.2.2.2 organization

Attribute name:	*organization*
Definition:	*Organization* for which the *Contact* acts as a representative
Obligation:	Mandatory
Multiplicity:	*1*
Datatype:	*Organization* (6.3.6)

6.3.2.2.3 role

Attribute name:	*role*
Definition:	Specified responsibilities of the *Contact*
Obligation:	Conditional
Multiplicity:	0..1
Datatype:	*Role* (6.3.9)
Condition:	Either an *Individual* or a *Role* or both shall be specified.

—— End of attributes of *Contact* ——

6.3.3 Document_Type class

6.3.3.1 Description of *Document_Type*

The composite **datatype** (3.1.8) *Document_Type* is used to specify the document type of a *Reference_Document* (6.3.7). The document type may be specified using an *identifier* (6.3.3.2.1) or a *description* (6.3.3.2.2).

The attributes of the *Document_Type* class are summarized here and specified more formally in 6.3.3.2.

— A *Document_Type* shall have either one *identifier* of type *String,* or one *description* of type *Text,* or both.

— A *Document_Type* may have a *scheme_reference* (6.3.3.2.3) of type *Sign* which identifies the scheme from which the identifier and/or description are taken.

6.3.3.2 Attributes of *Document_Type*

6.3.3.2.1 identifier

Attribute name:	*identifier*
Definition:	identifies the type of document
Obligation:	Conditional
Multiplicity:	*0..1*
Datatype:	*String* (6.2.11)
Condition:	Either the *identifier* or the *description* or both shall be specified.

6.3.3.2.2 description

Attribute name:	*description*
Definition:	describes the type of document
Obligation:	Conditional
Multiplicity	*0..1*
Datatype:	*Text* (6.2.12)
Condition:	Either the *identifier* or the *description* or both shall be specified.

6.3.3.2.3 scheme_reference

Attribute name:	*scheme_reference*
Definition:	identification scheme from which the identifier and/or description are drawn.
Obligation:	Optional
Multiplicity	*0..1*
Datatype:	*Sign* (6.2.10)

—— End of attributes of *Document_Type* ——

6.3.4 Individual class

6.3.4.1 Description of *Individual*

Individual is a class each instance of which models an **individual** (3.2.65), a single human being. The *Individual* class has the following attributes:

— Every *Individual* shall have exactly one *name* (6.3.4.2.1) of type *Sign* (6.2.10).

— An *Individual* may have zero or one *title* (6.3.4.2.2) of type *Sign*.

— A *Individual* may have zero or one *mail_address*es (6.3.4.2.3) of type *Postal_Address* (6.2.9), where the individual can be contacted by postal mail.

— A *Individual* may have zero or more *email_address*es (6.3.4.2.4) of type *String* (6.2.11), where the individual may be contacted by e-mail.

— A *Individual* may have zero or more *phone_numbers* (6.3.4.2.5) of type *Phone_Number* (6.2.8) where the individual may be contacted by phone.

6.3.4.2 Attributes of *Individual*

6.3.4.2.1 name

Attribute name:	*name*
Definition:	**sign** (3.2.123) that designates the *Individual*.
Obligation:	Mandatory
Multiplicity:	*1*
Datatype:	*Sign* (6.2.10)

6.3.4.2.2 title

Attribute name:	*title*
Definition:	**name** (3.2.83) of the position held by the *Individual*
Obligation:	Optional
Multiplicity:	0..1
Datatype:	*Sign* (6.2.10)

6.3.4.2.3 mail address

Attribute name:	*mail_address*
Definition:	postal address for the *Individual*
Obligation:	Optional
Multiplicity:	*0..1*
Datatype:	*Postal_Address* (6.2.9)

6.3.4.2.4 email_address

Attribute name:	*email_address*
Definition:	email address of the *Individual*
Obligation:	Optional
Multiplicity:	0..*
Datatype:	*String* (6.2.11)

6.3.4.2.5 phone_number

Attribute name:	*phone_number*
Definition:	phone numbers for the *Individual*
Obligation:	Optional
Multiplicity:	0..*

Datatype: *Phone_Number* (6.2.8)

6.3.4.2.6 role

Attribute name: ***role***

Definition: specified responsibilities of the *Individual*

Obligation: Optional

Multiplicity: 0..*

Datatype: *Role* (6.3.9)

—— End of attributes of *Individual* ——

6.3.5 Language_Identification class

6.3.5.1 Description of *Language_Identification*

The composite **datatype** (3.1.8) *Language_Identification* serves as an **identifier** (3.1.11) for a **language** (3.2.68). *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.

The identifier is comprised of the following parts, or attributes, which are based on IETF RFC 5646. IETF RFC 5646 refers to these attributes as 'language subtags':

— a mandatory *language_identifier* (6.3.5.2.1) that identifies the primary language

— an optional *script_identifier* (6.3.5.2.2) that identifies the set of graphic characters used for the written form of one or more languages

— an optional *geopolitical_territory_identifier* (6.3.5.2.3) that denotes the area or region in which a word, term, phrase or language variant is used.

— zero or more *variant_identifiers* (6.3.5.2.4) 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* (6.3.5.2.5) that denote extensions to a given language. Extensions consist of key-value pairs, which may be order dependent.

— an optional *private_use_qualifier* (6.3.5.2.6) that provides additional qualification for specific non-standardized purposes and uses.

NOTE 1 The W3C has a description of the use of the IETF language subtags at: http://www.w3.org/International/articles/language-tags/Overview.en.php.

NOTE 2 IANA maintains a registry of language subtags at: http://www.iana.org/assignments/language-subtag-registry.

NOTE 3 RFC 5646 requires the extension identifiers to be prefixed by a single character that identifies the registration authority that has registered the extension.

6.3.5.2 Attributes of *Language_Identification*

6.3.5.2.1 language_identifier

Attribute name: *language_identifier*

Definition: **identifier** (3.1.11) for the *Language*

Obligation:	Mandatory
Multiplicity	*1*
Datatype:	*String* (6.2.11)
NOTE	Use of the three character alphabetic codes from ISO 639-2/Terminology, with extensions if needed, is recommended but not required..

6.3.5.2.2 script_identifier

Attribute name:	*script_identifier*
Definition:	identifies the set of graphic characters used for the written form of one or more languages
Obligation:	Optional
Multiplicity	*0..1*
Datatype:	*String* (6.2.11)
NOTE	Use of the four character codes from ISO 15924:2004 codes for the representation of the names of scripts is recommended but not required.

6.3.5.2.3 geopolitical_territory_identifier

Attribute name:	*geopolitical_territory_identifier*
Definition:	identifies a specific country, territory, or region whose linguistic variations apply
Obligation:	Optional
Multiplicity	*0..1*
Datatype:	*String* (6.2.11)
NOTE	Use of the three digit numeric codes from ISO 3166-1, with extensions if needed, is recommended but not required.

6.3.5.2.4 variant_identifier

Attribute name:	*variant_identifier*
Definition:	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
Obligation:	Optional
Multiplicity	*0..**
Datatype:	*String* (6.2.11) [ordered]
NOTE	In RFC 5646, variant identifiers are typically represented as dates and are used to 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 tags reference IANA variant subtags. Use of RFC 5646 is recommended but not required.

6.3.5.2.5 extension_identifier

Attribute name:	*extension_identifier*
Definition:	identifies an extension to a *language_identifier*
Obligation:	Optional
Multiplicity	*0..**
Datatype:	*String* (6.2.11) [ordered]
NOTE 1	In RFC 5646, 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. Whitespace separates the

identifiers. Use of RFC 5646 is recommended but not required.

NOTE 2 As of 2012-07-31, no extensions have been registered.

6.3.5.2.6 private_use_qualifer

Attribute name:	*private_use_qualifer*
Definition:	qualifier whose meaning is defined solely by private agreement.
Obligation:	Optional
Multiplicity	*0..1*
Datatype:	*String* (6.2.11)
NOTE	Definition derived from IETF RFC 5646. Use of RFC 5646 is recommended but not required.

—— End of attributes of *Language_Identification* ——

6.3.6 Organization class

6.3.6.1 Description of *Organization*

Organization is a class each instance of which models an **organization** (3.2.90), a unique framework of authority within which **individuals** (3.2.65) act, or are designated to act, towards some purpose.

The role of *Organization* in the Registration Package is further described in 8.1.3.2.

The attributes of the *Organization* class are summarized here and specified more formally in 6.3.6.2.

— Every *Organization* shall have one or more *name*s (6.3.6.2.1) of type *Sign* (6.2.10).

— An *Organization* may have zero or one *mail_address*es (6.3.6.2.2) of type *Postal_Address* (6.2.9), where the *Organization* can be contacted by postal mail.

— An *Organization* may have zero or more *phone_numbers* (6.3.6.2.4) of type *Phone_Number* (6.2.8) where the *Organization* may be contacted by phone.

— An *Organization* may have zero or more *email_address*es (6.3.6.2.3) of type *String* (6.2.11), where the *Organization* may be contacted by e-mail.

— An *Organization* may have zero or one *uri* (6.3.6.2.5) of type *String* (6.2.11), where the *Organization* may be contacted via the web.

6.3.6.2 Attributes of *Organization*

6.3.6.2.1 name

Attribute name:	*name*
Definition:	**sign** (3.2.123) that designates the *Organization.*
Obligation:	Mandatory
Multiplicity:	*1..**
Datatype:	*Sign* (6.2.10)

6.3.6.2.2 mail address

Attribute name:	*mail_address*
Definition:	postal address for the *Organization.*

Obligation: Optional

Multiplicity: *0..1*

Datatype: *Postal_Address* (6.2.9)

6.3.6.2.3 email_address

Attribute name: *email_address*

Definition: email addresses of the *Organization.*

Obligation: Optional

Multiplicity: 0..*

Datatype: *String* (6.2.11)

6.3.6.2.4 phone_number

Attribute name: *phone_number*

Definition: phone numbers for the *Organization*

Obligation: Optional

Multiplicity: 0..*

Datatype: *Phone_Number* (6.2.8)

6.3.6.2.5 uri

Attribute name: *uri*

Definition: uri for *Organization*

Obligation: Optional

Multiplicity: *0..1*

Datatype: *String* (6.2.11)

—— End of attributes of *Organization* ——

6.3.7 Reference_Document class

6.3.7.1 Description of *Reference_Document*

A *Reference_Document* is a document that provides pertinent details for consultation about a subject.

The attributes of the *Reference_Document* class are summarized here and specified more formally in 6.3.7.2.

The following attributes are Mandatory:

— Every *Reference_Document* shall have exactly one *identifier* (6.3.7.2.1) of type *String* (6.2.11), which is an unambiguous identifier for the document.

— Every *Reference_Document* shall have exactly one *type_description* (6.3.7.2.2) of type *Document_Type* (6.3.3), which describes the type of the document.

— Every *Reference_Document* shall have one or more *provider*s (6.3.7.2.6) of type *Organization* (6.3.6), *each of* which shall identify an *Organization* that maintains or carries an official copy of the *Reference_Document*.

The following attributes are Optional:

— A *Reference_Document* may have zero, one or more *language_identifier*s (6.3.7.2.3), which specify the language or languages used in the *Reference_Document.*

— A *Reference_Document* may optionally have a *notation* (6.3.7.2.4) of type *Notation* (6.2.7).

— A *Reference_Document* may optionally have a *title* (6.3.7.2.5) of type *Text* (6.2.12).

— A *Reference_Document* may optionally have a *uri* (6.3.7.2.7) of type *String* (6.2.11).

6.3.7.2 Attributes of *Reference_Document*

6.3.7.2.1 identifier

Attribute name:	*identifier*
Definition:	**identifier** (3.1.11) for the *Reference_Document*
Obligation:	Mandatory
Multiplicity:	*1*
Datatype:	*String* (6.2.11)

6.3.7.2.2 type_description

Attribute name:	*type_description*
Definition:	description of the type of *Reference_Document*
Obligation:	Mandatory
Multiplicity:	*1*
Datatype:	*Document_Type* (6.3.3)

6.3.7.2.3 language_identifier

Attribute name:	*language_identifier*
Definition:	**identifier** (3.1.11) of the natural language used in the *Reference_Document*
Obligation:	Optional
Multiplicity:	*0..**
Datatype:	*Language_Identification* (6.3.5)
NOTE	Absence of a *Reference_Document language_identifier* implies use of the language specified by *Registry_Specification registry_primary_language.*

6.3.7.2.4 notation

Attribute name:	*notation*
Definition:	formal syntax and semantics used within the *Reference_Document*
Obligation:	Optional
Multiplicity:	*0..1*
Datatype:	*Notation* (6.2.7)

6.3.7.2.5 title

Attribute name:	*title*
Definition:	title of the *Reference_Document.*
Obligation:	Optional
Multiplicity:	*0..1*
Datatype:	*Text* (6.2.12)

6.3.7.2.6 provider

Attribute name:	*provider*

Definition:	*Organization* that maintains or carries an official copy of the *Reference_Document*.
Obligation:	Optional
Multiplicity:	*0..**
Datatype:	*Organization* (6.3.6)

6.3.7.2.7 uri

Attribute name:	*uri*
Definition:	uri for *Reference_Document*
Obligation:	Optional
Multiplicity:	*0..1*
Datatype:	*String* (6.2.11)

—— End of attributes of *Reference_Document* ——

6.3.8 Registration_Authority_Identifier class

6.3.8.1 Description of *Registration_Authority_Identifier*

The composite datatype *Registration_Authority_Identifier* is used to uniquely identify a *Registration_Authority* (8.1.2.5). The sources of values for each part of the identifier are specified in ISO/IEC 6523-1 and further explained for the metadata registry in ISO/IEC 11179-6.

A *Registration_Authority_Identifier* consists of the following parts, which are summarized here and specified more formally in 6.3.8.2.

The following parts are mandatory:

— Every *Registration_Authority_Identifier* shall have exactly one *international_code_designator* (6.3.8.2.1) of type *String* (6.2.11).

— Every *Registration_Authority_Identifier* shall have exactly one *organization_identifier* (6.3.8.2.2) of type *String*.

The following parts are optional:

— A *Registration_Authority_Identifier* may optionally have an *organization_part_identifier (OPI)* (6.3.8.2.3) of type *String*.

— A *Registration_Authority_Identifier* may conditionally have an *OPI_source* (6.3.8.2.4) of type *String*. If the *organization_part_identifier* is present, then the *OPI_source* shall be present.

6.3.8.2 Attributes of *Registration_Authority_Identifier*

6.3.8.2.1 international_code_designator

Attribute name:	*international_code_designator*
Definition:	**identifier** (3.1.11) of an organization identification scheme
Obligation:	Mandatory
Multiplicity:	*1*
Datatype:	*String* (6.2.11)

6.3.8.2.2 organization_identifier

Attribute name:	*organization_identifier*
Definition:	**identifier** (3.1.11) assigned to an *Organization* within an organization identification scheme, and unique within that scheme
Obligation:	Mandatory
Multiplicity:	*1*
Datatype:	*String* (6.2.11)

6.3.8.2.3 organization_part_identifier (OPI)

Attribute name:	*organization_part_identifier*
Definition:	**identifier** (3.1.11) allocated to a particular organization part
Obligation:	Optional
Multiplicity:	*0..1*
Datatype:	*String* (6.2.11)

6.3.8.2.4 opi_source

Attribute name:	*opi_source*
Definition:	source for the *organization_part_identifier*
Obligation:	Conditional
Multiplicity:	*0..1*
Datatype:	*String* (6.2.11)
Condition:	If *organization_part_identifier* is present, then *opi_source* shall be present.

—— End of attributes of *Registration_Authority_Identifier* ——

6.3.9 Role class

6.3.9.1 Description of *Role*

Role is a class each instance of which models a role which an *Individual* (6.3.4) may play as a *Contact* (6.3.2) within an *Organization* (6.3.6).

The attributes of the *Role* class are summarized here and specified more formally in 6.3.9.2.

— A *Role* may have zero or one *title* (6.3.9.2.1) of type *Sign* (6.2.10).

— A *Role* may have zero or one *mail_address*es (6.3.9.2.2) of type *Postal_Address* (6.2.9), where the *Role* can be contacted by postal mail.

— A *Role* may have zero or more *email_address*es (6.3.9.2.3) of type *String* (6.2.11), where the *Role* may be contacted by e-mail.

— A *Role* may have zero or more *phone_numbers* (6.3.9.2.4) of type *Phone_Number* (6.2.8) where the *Role* may be contacted by phone.

6.3.9.2 Attributes of *Role*

6.3.9.2.1 title

Attribute name:	*title*
Definition:	**name** (3.2.83) of the position that fulfils the *Role*

Obligation: Optional

Multiplicity: 0..1

Datatype: *Sign* (6.2.10)

6.3.9.2.2 mail address

Attribute name: *mail_address*

Definition: postal address for the *Role*

Obligation: Optional

Multiplicity: *0..1*

Datatype: *Postal_Address* (6.2.9)

6.3.9.2.3 email_address

Attribute name: *email_address*

Definition: email addresses of the *Role*

Obligation: Optional

Multiplicity: 0..*

Datatype: *String* (6.2.11)

6.3.9.2.4 phone_number

Attribute name: *phone_number*

Definition: phone numbers for the *Role*

Obligation: Optional

Multiplicity: 0..*

Datatype: *Phone_Number* (6.2.8)

—— End of attributes of *Role* ——

7 Identification, Designation and Definition package

7.1 Overview of this package

This package is divided into two regions:

- 7.2 Identification metamodel region
- 7.3 Designation and Definition metamodel region

NOTE: A description of the distinction between identification and designation is available in ISO/IEC 15944-1:2002, Annex C. (See Bibliography.)

The table below illustrates the differences between a *Designation* (7.3.2.3) and a *Scoped_Identifier* (7.2.2.1).

Table 3 – Comparison of Designation to Scoped_Identifier

	Designation	**Scoped_Identifier**
Scope	Namespace and Context, independently	Namespace
Occurrence	Many allowable per *Designatable_Item*	Many allowable per *Identified_Item*
Language dependent	Yes	No
Datatype	*Sign* (6.2.10)	*String* (6.2.11)

7.2 Identification metamodel region

7.2.1 Overview

The Identification region of the model specifies how **metadata items** (3.2.75) are identified in the **metadata registry** (3.2.78). A metadata item that is to be identified shall be assigned the type *Identified_Item* (7.2.2.1).

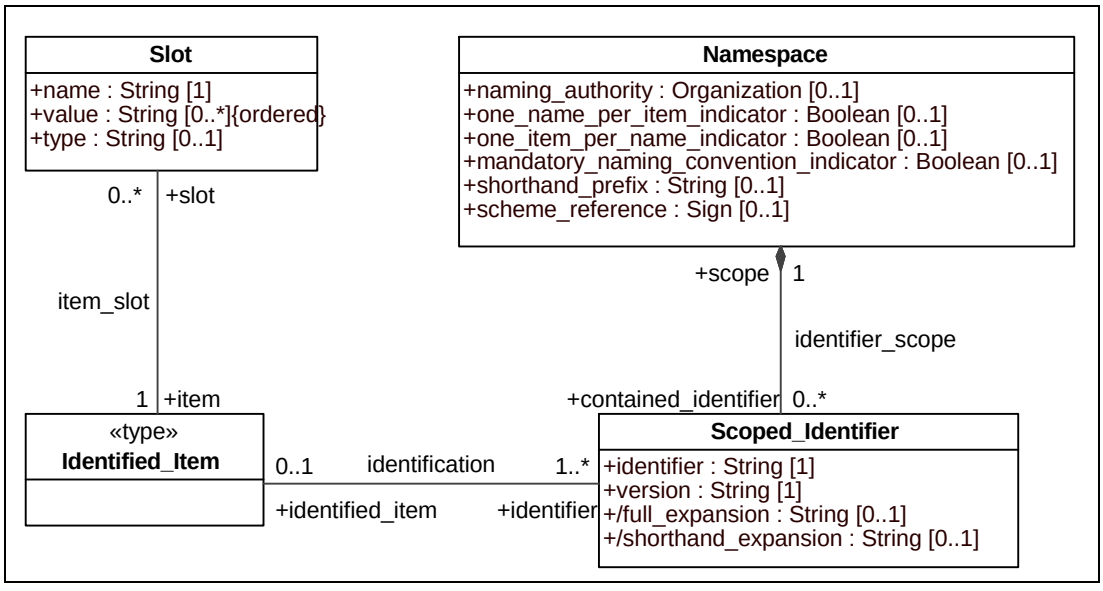

Figure 5 — Identification metamodel region

7.2.2 Classes in the Identification metamodel region

7.2.2.1 Identified_Item class

Identified_Item is a class each instance of which models an **identified item** (3.2.64), a **metadata item** (3.2.75) that is identified in a **metadata registry** (3.2.78).

Identified_Item shall participate in the following association:

— *identification* (7.2.3.1) which references one or more *Scoped_Identifiers* (7.2.2.2), each of which provides an *identifier* (7.2.2.2.2.1) for the *Identified_Item* within a specific *Namespace* (7.2.2.3).

Identified_Item may participate in the following association:

— *item_slot* (7.2.3.3) which references zero or more *Slot*s (7.2.2.4) which extend the *Identified_Item*.

Registered_Item (8.1.2.1) is a subclass of *Identified_Item*. *Identified_Item* has no attributes.

7.2.2.2 Scoped_Identifier class

7.2.2.2.1 Description of *Scoped_Identifier*

Scoped_Identifier is a class each instance of which models a **scoped identifier** (3.2.122), an *identifier* (7.2.2.2.2.1) with a particular scope provided by a *Namespace* (7.2.2.3).

Scoped_Identifier shall participate in the following association:

— *identifier_scope* (7.2.3.2) which references a *Namespace* which provides the scope for the *Scoped_Identifier*.

Scoped_Identifier may participate in the following association:

— *identification* (7.2.3.1) which references zero or one *Identified_Items* (7.2.2.1), which are unambiguously identified by the *Scoped_Identifier* within the *Namespace*.

The attributes of the *Scoped_Identifier* class are summarized here and specified more formally in 7.2.2.2.2.

— *Scoped_Identifier* shall have exactly one *identifier* of type *String* (6.2.11), which may be used as an unambiguous identifier for an *Identified_Item* within a particular *Namespace*.

— *Scoped_Identifier* shall have exactly one *version* of type *String*. *Scoped_Identifier.version* allows more than one version of an *Identified_Item* to be identified within a particular *Namespace*.

— *Scoped_Identifier* may have zero or one derived attribute *full_expansion* (7.2.2.2.2.2) of type *String*, formed by prefixing the unique *identifier* of *Namespace* to the *identifier* of this *Scoped_Identifier.*

NOTE a unique *identifier* for the *Namespace* exists only if the *Namespace* is itself an *Identified_item* with exactly one *Scoped_Identifier*. *full_expansion* is not defined if *Namespace* has more than one *identifier*, or none at all.

— *Scoped_Identifier* may have zero or one derived attribute *short_expansion* (7.2.2.2.2.4) of type *String*, formed by prefixing the *shorthand_prefix* (7.2.2.3.2.5) of *Namespace* to the *identifier* of this *Scoped_Identifier. short_expansion* will exist if and only if the corresponding *shorthand_prefix* exists.

7.2.2.2.2 Attributes of *Scoped_Identifier*

7.2.2.2.2.1 identifer

Attribute name: *identifier*

Definition:	*String* used to unambiguously denote an *Identified_Item* within the scope of a specified *Namespace.*
Obligation:	Mandatory
Multiplicity:	*1*
Datatype:	*String* (6.2.11)

7.2.2.2.2.2 version

Attribute name:	*version*
Definition:	unique version identifier of the *Scoped_Identifier* which identifies an *Identified_Item*
Obligation:	Mandatory
Multiplicity:	*1*
Datatype:	*String* (6.2.11)

7.2.2.2.2.3 full_expansion

Attribute name:	*full_expansion*
Definition:	*String* representation of a *Scoped_Identifier*, in which the unique *identifier* of the associated *Namespace* is combined in some way with the *identifier* of the *Scoped_Identifier* to fully specify the scope*.*
Obligation:	Optional, derived.
Multiplicity:	*0..1*
Datatype:	*String* (6.2.11)
Comment:	The precise manner of derivation might vary depending on the type of namespace. *full_expansion* is defined only when *Namespace* has exactly one identifier.

7.2.2.2.2.4 shorthand_expansion

Attribute name:	*shorthand_expansion*
Definition:	*String* representation of a *Scoped_Identifier* in which a *shorthand_prefix* from the associated *Namespace* has been prepended to the *identifier* to indicate the *scope*.
Obligation:	Conditional, derived.
Multiplicity:	*0..1*
Datatype:	*String* (6.2.11)
Condition:	*shorthand_expansion* will exist if and only if the corresponding *shorthand_prefix* (7.2.2.3.2.5) exists.

—— End of attributes of *Scoped_Identifier* ——

7.2.2.3 Namespace class

7.2.2.3.1 Description of *Namespace*

Namespace is a class each instance of which represents a **namespace** (3.2.84). *Namespace* is a scoping construct used to group sets of *Designation*s (7.3.2.3) and/or *Scoped_Identifier*s (7.2.2.2) used in a **metadata registry** (3.2.78). Distinct *Namespaces* permit independent development of metadata collections and/or ontologies*.* They permit enforcement of uniqueness constraints on *identifier*s (7.2.2.2.2.1) or *designation_sign*s (7.3.2.3.2.1) within a specific *Namespace* without central coordination.

A *Namespace* may contain a set of *Designations,* a set of *Scoped_Identifier*s or a combination of the two.

NOTE 1 The term **namespace** is used in this International Standard because it is in common use, even though the concept is applied to identifiers as well as names.

NOTE 2 These are NOT XML Namespaces. However, it might be possible to add additional subclasses of Namespaces to model XML Namespaces.

NOTE 3 See also 7.3.2.6 for further description of Namespace in the Designation and Definition metamodel region, and 8.1.4.1 for further description of Namespace in the Registration metamodel region.

Namespace may participate in the following association:

— *identifier_scope* (7.2.3.2) which references contained *Scoped_Identifiers* (7.2.2.2).

As a **metadata item** (3.2.75) itself, a *Namespace* may be assigned a type of:

— *Identified_Item* (7.2.2.1), enabling it to be identified;

— *Designatable_Item* (7.3.2.1), enabling it to be named and/or defined;

— *Classifiable_Item* (9.2.2.1), enabling it to be classified.

The attributes of the *Namespace* class are summarized here and specified more formally in 7.2.2.3.2.

— *Namespace* may have zero or one *naming_authority* (7.2.2.3.2.1) that specifies the *Organization* (6.3.6) that has authority for naming in the *Namespace*.

— *Namespace* may have zero or one *one_name_per_item_indicator* (7.2.2.3.2.2), signifying whether or not:

  — many *Designations* (7.3.2.3) from the *Namespace* may be associated with (bound to) one *Designatable_Item* (7.3.2.2), and

  — many *Scoped_Identifiers* (7.2.2.2) from the *Namespace* may be associated with (bound to) one *Identified_Item* (7.2.2.1).

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

— *Namespace* may have zero or one *one_item_per_name_indicator* (7.2.2.3.2.3), signifying whether or not:

  — a given *Designation* (7.3.2.3) may denote many *Designatable_Items* (7.3.2.2), *and*

  — a given *Scoped_Identifier* (7.2.2.2) must identify only a single *Identified_Item* (7.2.2.1).

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

— *Namespace* may have zero or one *mandatory_naming_convention_indicator* (7.2.2.3.2.4) that determines whether or not all *Designations* (7.3.2.3) in a *Namespace* shall conform to exactly one *Naming_Convention* (7.3.2.7).

— *Namespace* may have zero or one *shorthand_prefix* (7.2.2.3.2.5), used as shorthand for the *Namespace* in text intended for human consumption.

7.2.2.3.2 Attributes of *Namespace*

7.2.2.3.2.1 naming_authority

Attribute name: *naming_authority*

Definition: *Organization* that has the authority for naming in the *Namespace*.

Obligation: Optional

Multiplicity: *0..1*

Datatype: *Organization* (6.3.6)

7.2.2.3.2.2 one_name_per_item_indicator

Attribute name: *one_name_per_item_indicator*

Definition: indicator that denotes whether more than one *Designation* and/or *Scoped_Identifier* within the *Namespace* may be associated with any single item (*Designatable_Item* and/or *Identified_Item*). If the indicator is *true*, then at most one *Designation* and/or *Scoped_Identifier* within the *Namespace* may be associated with any single item.

Obligation: Optional

Multiplicity: *0..1*

Datatype: *Boolean* (6.2.2)

Comment: If the indicator is *true*, then the registry shall enforce the rule for the *Namespace*.

7.2.2.3.2.3 one_item_per_name_indicator

Attribute name: *one_item_per_name_indicator*

Definition: indicator that denotes whether the *Namespace* may contain more than one *Designation* and/or *Scoped_Identifier* having the same sign and/or identifier. If the indicator is *true*, then at most one *Designation* and/or *Scoped_Identifier* having the same sign and/or identifier is permitted within the *Namespace*.

Obligation: Optional

Multiplicity: *0..1*

Datatype: *Boolean* (6.2.2)

Comment: If the indicator is *true*, then the registry shall enforce the rule for the *Namespace*.

7.2.2.3.2.4 mandatory_naming_convention_indicator

Attribute name: *mandatory_naming_convention_indicator*

Definition: indicator specifying whether all *Designations* in this *Namespace* shall conform to one of the acceptable *Naming_Conventions.*

Obligation: Optional

Multiplicity: *0..1*

Datatype: *Boolean* (6.2.2)

NOTE If *mandatory_naming_convention_indicator* is true:
(a) there must be at least one acceptable convention *Naming_Convention* associated with this *Namespace* in the *naming_convention_utilization* (7.3.4.6) association, and
(b) every binding *Designation* must have a *naming_convention_conformance* (7.3.4.5) association with one of the same *Naming_Conventions* 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.

7.2.2.3.2.5 shorthand_prefix

Attribute name: *shorthand_prefix*

Definition: prefix conventionally used as shorthand for a namespace, for greater readability, in text for human consumption.

NOTE In the case of URL prefixes as defined in XML, a final colon (:) should be included here as part of the shorthand prefix.

Obligation: Optional

Multiplicity: *0..1*

Datatype: *String* (6.2.11)

7.2.2.3.2.6 scheme_reference

Attribute name: *scheme_reference*

Definition: reference identifying the type of the *Namespace* specification

Obligation: Optional

Multiplicity: *0..1*

Datatype: *Sign* (6.2.10)

EXAMPLE 1 For XML Namespaces, specify: http://www.w3.org/TR/1999/REC-xml-names-19990114/

EXAMPLE 2 For UML Namespaces, specify: http://www.omg.org/spec/UML/2.4.1/Core/Namespace

—— End of attributes of *Namespace* ——

7.2.2.4 Slot class

7.2.2.4.1 Description of *Slot*

Slot instances provide a dynamic way to add arbitrary attributes to instances of *Identified_Item* (7.2.2.1). A *Slot* instance is associated with exactly one *Identified_Item* instance, through the association *item_slot* (7.2.3.3). An *Identified_Item* instance may be associated with zero, one or more *Slot* instances. All *Slot* instances associated with a particular *Identified_Item* instance must have a distinct *name* (7.2.2.4.2.1) to allow each *Slot* instance to be unambiguously identified.

For example, if a company wants to add a “copyright” attribute to each *Identified_Item* instance that it submits, it can do so by adding a slot with name “copyright” and value containing the copyrights statement.

The attributes of the *Slot* class are summarized here and specified more formally in 7.2.2.4.2.

— A *Slot* shall have exactly one *name* of type *String* (6.2.11), that unambiguously identifies the *Slot* among several associated with an *Identified_Item.*

— A *Slot* may have zero or one *type* (7.2.2.4.2.2) of type *String*, that specifies the datatype to be used to interpret and constrain the *slot_value*. For example, the *type* may specify that the string ‘1998-12-31’ in *value* should be interpreted as a *Date*.

— A *Slot* may have zero, one or more *value*s (7.2.2.4.2.3) of type *String*, that provide the value(s) associated with the *name*.

NOTE *Slot* is modelled after the *Slot* class of ebXML RIM (see Bibliogrpahy [36]), but it also maps directly to the ‘slot tuple’ of ISO/IEC 19773 (see Bibliography [29]), where:

— *Slot.name* maps to slot_tuple.identifer

— *Slot.type* maps to slot_tuple.kind

— *Slot.value* maps to slot_tuple.value

7.2.2.4.2 Attributes of *Slot*

7.2.2.4.2.1 name

Attribute name:	*name*
Definition:	**name** (3.2.83) of the *Slot*
Obligation:	Mandatory
Multiplicity:	*1*
Datatype:	*String* (6.2.11)

7.2.2.4.2.2 type

Attribute name:	*type*
Definition:	categorization of the *value* of the *Slot*
Obligation:	Optional
Multiplicity:	*0..1*
Datatype:	*String* (6.2.11)
Comment:	*type* may be used to categorize slot values in some way, including but not limited to specifying the datatype of the value.

7.2.2.4.2.3 value

Attribute name:	*value*
Definition:	value assigned to the *Slot*
Obligation:	Optional
Multiplicity:	*0..**
Datatype:	*String* (6.2.11) (ordered)

—— End of attributes of *Slot* ——

7.2.3 Associations in the Identification metamodel region

7.2.3.1 identification association

The *identification* association specifies the *Scoped_Identifier* (7.2.2.2) that identifies an *Identified_Item* (7.2.2.1).

identification has two roles:

— identifier (verb form: identifies) which references a *Scoped_Identifier*;

— identified_item (verb form: identified_by).which references an *Identified_Item*

Every *Identified_Item* must have one or more *identification* associations with a *Scoped_identifier* that provides an *identifier* (7.2.2.2.2.1) for the *Identified_Item*.

7.2.3.2 identifier_scope association

identifier_scope associates zero, one or more *Scoped_Identifier*s (7.2.2.2) with exactly one *Namespace* (7.2.2.3).

identifier_scope has two roles:

— scope (verb form: provides_scope_for) which references a *Namespace*;

— contained_identifier (verb form: contained_in) which references a *Scoped_Identifier*.

7.2.3.3 item_slot association

item_slot associates an *Identified_Item* (7.2.2.1) with zero, one or more *Slots* (7.2.2.4) which extend the *Identified_item*.

item_slot has two roles:

— item (verb form: extended by) which references an *Identified_Item;*

— slot (verb form: extends) which references a *Slot*.

7.3 Designation and Definition metamodel region

7.3.1 Overview

The Designation and Definition region is used to manage the *Designations* (7.3.2.3) and *Definitions* (7.3.2.4) of *Designatable_Items* (7.3.2.1) and the *Contexts* (7.3.2.5) for the *Designations* and *Definitions*. A *Designatable_Item* may have many *Designation.sign*s (7.3.2.3.2.1) that will vary depending on discipline, locality, technology, etc. This sub-clause describes the classes, associations, and association classes of this region.

Figure 6 on p. 53 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.

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_Item*s within a *Context*.

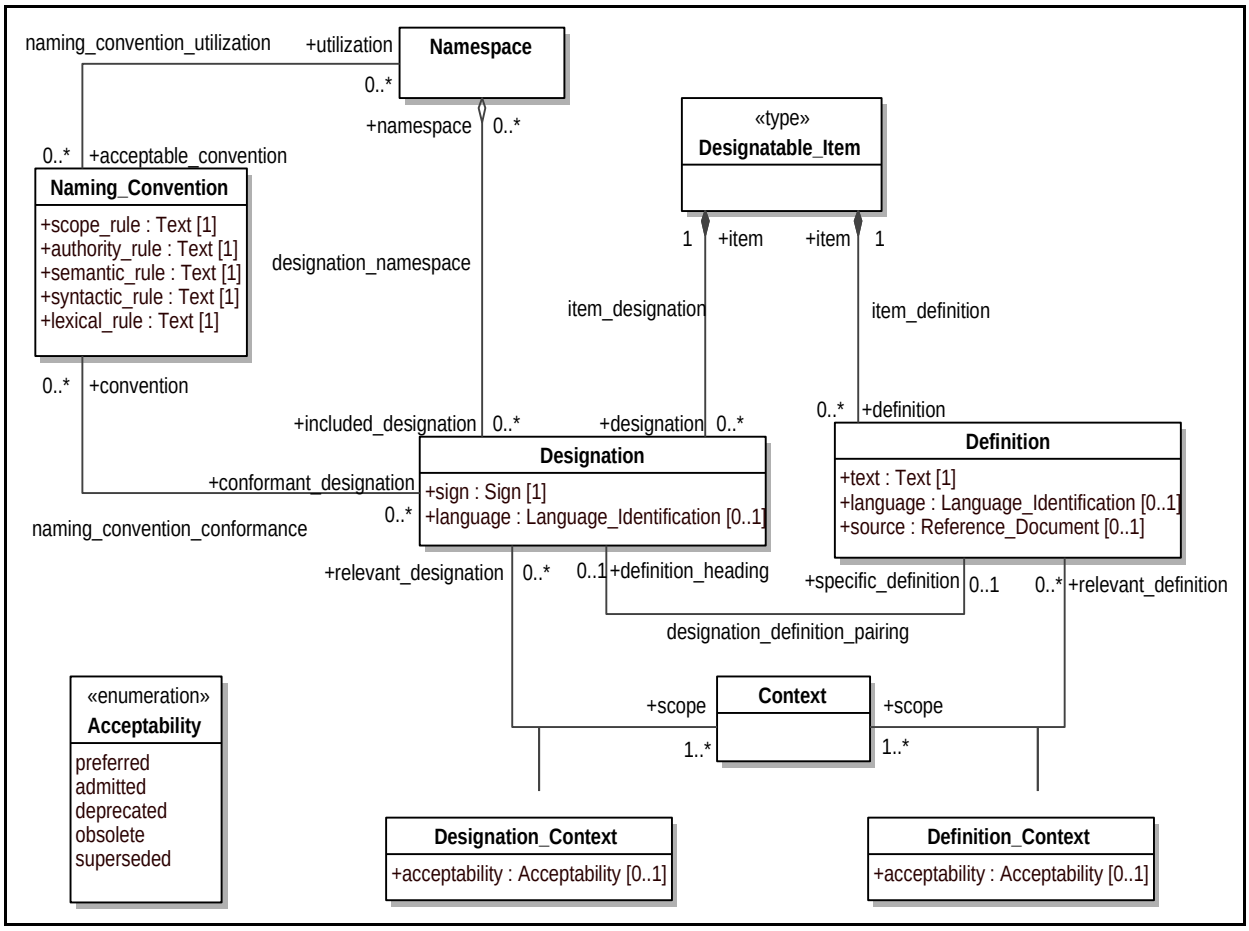

Figure 6 — Designation and Definition metamodel region

7.3.2 Classes in the Designation and Definition metamodel region

7.3.2.1 Acceptability enumeration

Acceptability models a scale of **acceptability rating**s (3.2.1) comprised of: preferred, admitted, deprecated, obsolete and superseded. This enumeration is used as a **datatype** (3.1.8) for the attributes *Designation.acceptability* (7.3.3.2.2.1) and *Definition.acceptability* (7.3.3.1.2.1).

7.3.2.2 Designatable_Item class

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. *Designatable_Items* may have many different (or identical) *signs* in various *languages*, *Contexts* (7.3.2.5), *Namespaces* (7.3.2.6) and *Naming_Conventions* (7.3.2.7).

A *Designatable_Item* may have zero or more *item_designation* associations (7.3.4.4) with *Designation*s (7.3.2.3).

A *Designatable_Item* may have zero or more *item_definition* associations (7.3.4.3) with *Definition*s (7.3.2.4).

NOTE It is because the associations are optional that the class is called *Designatable_Item* rather than *Designated_Item*.

7.3.2.3 Designation class

7.3.2.3.1 Description of *Designation*

The *Designation* class records the binding of a pair comprised of a *sign* (7.3.2.3.2.1) and its *language* (7.3.2.3.2.2) to a *Designatable_Item* (7.3.2.1). Each *Designation* is situated with respect to a *Context* (7.3.2.5), a *Naming_Convention* (7.3.2.7), a *Namespace* (7.3.2.6), and may be paired with a *Definition* (7.3.2.4).

A *Designation* shall participate in:

— exactly one *item_designation* (7.3.4.4) association with a *Designatable_Item*;

— one or more *designation_context* (7.3.3.2) associations with a *Context*;

A *Designation* may participate in the associations:

— *designation_definition_pairing* (7.3.4.1) with zero or one *Definition*.

— *designation_namespace* (7.3.4.2) with zero or more *Namespaces* that provide scope for the *Designation*;

— *naming_convention_conformance* (7.3.4.5) with zero or more *Naming_Conventions* that provide naming rules for the *Designation*;

The attributes of the *Designation* class are summarized here and specified more formally in 7.3.2.3.2.

— A *Designation* shall have exactly one *sign* (7.3.2.3.2.1) attribute of type *Sign* (6.2.10), which is used to designate a *Designatable_Item*, e.g., a name of an object or concept. The *sign* may be a word or phrase in a natural language (as specified by the *language* (7.3.2.3.2.2)), or it may be an icon or other symbol.

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).

— A *Designation* may have zero or one *language* (7.3.2.3.2.2) attribute of type *Language_Identification* (6.3.5), which is used to record the language or dialect in which the *sign* (usually a name) is used, when the *sign* has an associated *language*. Usually the *language* will refer to a natural human language.

7.3.2.3.2 Attributes of *Designation*

7.3.2.3.2.1 sign

Attribute name:	*sign*
Definition:	**sign** (3.2.123) of the *Designation*
Obligation:	Mandatory
Multiplicity:	*1*
Datatype:	*Sign* (6.2.10)

7.3.2.3.2.2 language

Attribute name:	*language*
Definition:	**language** (3.2.68) or dialect in which the *Sign* (usually a *name*) is expressed
Obligation:	Conditional
Multiplicity:	*0..1*
Datatype:	*Language_Identification* (6.3.5)
Condition	*language* is conditional because it might not be applicable (e.g. if the *sign* is an icon). If it is applicable, it must be specified. *Designation.language* shall not default to *Registry_Specification.registry_primary_language*, as

Definition.language does.

—— End of attributes of *Designation* ——

7.3.2.4 Definition class

7.3.2.4.1 Description of *Definition*

The *Definition* class provides the definition *text* (7.3.2.4.2.1) for a *Designatable_Item* (7.3.2.2) as it applies in one or more *Contexts* (7.3.2.5). Each *Designatable_Item* may be associated with zero, one or more *Definitions*, each *Definition* being specified in a particular *language* (7.3.2.4.2.2).

The *Definition* class records the binding of a definition *text* and its *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 *source* (7.3.2.4.2.3) for the *text*.

A *Definition* shall participate in:

— exactly one *item_definition* (7.3.4.3) association with a *Designatable_Item*;

— one or more *Definition_Context* (7.3.3.1) associations with a *Context*;

A *Definition* may participate in:

— zero or one *designation_definition_pairing* (7.3.4.1) associations with a *Designation* (7.3.2.3).

The attributes of the *Definition* class are summarized here and specified more formally in 7.3.2.4.2.

The *Definition* class has the following attributes:

— A *Definition* shall have exactly one *text* attribute of datatype *Text* (6.2.12) which contains the text which constitutes the definition.

— A *Definition* may have zero or one *language* attribute of datype *Language_Identification* (6.3.5). The *language* attribute records the language in which the *text* is written.

— A *Definition* may have zero or one *source* attribute of datatype *Reference_Document* (6.3.7). The *source* attribute is used to record the origin of the *text*.

7.3.2.4.2 Attributes of *Definition*

7.3.2.4.2.1 text

Attribute name:	*text*
Definition:	text of the *Definition*
Obligation:	Mandatory
Multiplicity:	*1*
Datatype:	*Text* (6.2.12)

7.3.2.4.2.2 language

Attribute name:	*language*
Definition:	**language** (3.2.68) used to write the definition *text*
Obligation:	Conditional
Multiplicity:	*0..1*

Datatype:	*Language_Identification* (6.3.5)
Condition	If *Registry_Specification.registry_primary_language* (see 8.1.2.9.2.7) is specified, then *language* may be omitted, implying that the language is that specified by the *Registry_Specification.registry_primary_ language.* If *Registry_Specification.registry_primary_language* is not specified, then *language* must be specified.

7.3.2.4.2.3 source

Attribute name:	*source*
Definition:	reference to the source from which the definition *text* is taken
Obligation:	Optional
Multiplicity:	*0..1*
Datatype:	*Reference_Document* (6.3.7)

—— End of attributes of *Definition* ——

7.3.2.5 Context class

Context is a class each instance of which models a **context** (3.2.25), the setting in which *Designations* (7.3.2.3) are used to designate and *Definitions* (7.3.2.4) are used to define a set of *Designatable_Items* (7.3.2.2). Each *Designatable_Item* may be designated and/or defined within one or more *Contexts*.

A *Context* defines the setting 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 **stewardardship organization** (3.2.127) responsible for the *Context*, or the **registration authority** (3.2.109) responsible for the registry. Each *Context* may itself be made a *Designatable_Item* within the registry and be given a *designation* and/or a *definition*.

A *Context* may participate in the following associations:

— *Definition_Context* (7.3.3.1) with zero or more *relevant_definitions* of type *Definition* (7.3.2.4) where the *Context* provides the *scope* of the associated *Definition*;

— *Designation_Context* (7.3.3.2) with zero or more *relevant_designations* of type *Designation* (7.3.2.3) where the *Context* provides the *scope* of the associated *Designation*.

NOTE The requirement that all *Designations* and *Definitions* be associated with at least one *Context* applies even when the item being designated and defined is a *Context*, and this does not create a problem of infinite regress. For example, one straightforward way to satisfy this requirement is to include, within each *Context*, the *Designation*(s) and *Definition*(s) for itself; another is to place all *Designation*s and *Definition*s for *Context*s in a “registry context” (including the *Designation*(s) and *Definition*(s) for the registry context itself).

7.3.2.6 Namespace class

Namespace is described in 7.2.2.3. 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 *Namespace* to be associated with zero or more acceptable conventions and/or a *conformant_designation Designation* to conform to more than one convention.

A *Namespace* may have zero or more *designation_namespace* associations with an *included_designation* of type *Designation* providing the *Namespace* of the associated *Designation*.

One use of *Namespaces* is to permit the *Designatable_Item* represented by a *Designation.sign* to be uniquely determined for a *sign* within a particular *Namespace*.

A *Namespace* may participate in the following associations:

— naming_convention_utilization

— designation_space_membership.

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

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

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

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 *Namespace*.

Unambiguous names implies a functional mapping from *Designation.signs* to *Designatable_Items*.

7.3.2.7 Naming_Convention class

7.3.2.7.1 Description of *Naming_Convention*

The *Naming_Convention* class provides the specification by which the *sign* of a *Designation* is developed. The *Naming_Convention* class records a set of rules for constructing *Designation.signs* (names) to designate *Designatable_Items*. *Naming_Convention*s may range in complexity from very simple to very complex. The semantic, syntactic, and lexical_rules may each have their own complexity.

As a metadata item itself, a *Naming_Convention* may be assigned a type of:

— *Identified_Item,* enabling it to be identified;

— *Designatable_Item*, enabling it to be named and/or defined;

— *Classifiable_Item*, enabling it to be classified.

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

The attributes of the *Naming_Convention* class are summarized here and specified more formally in 7.3.2.7.2.

— A *Naming_Convention* shall have exactly one *scope_rule* of type *Text*, by which the scope of the naming convention shall be specified.

— A *Naming_Convention* shall have exactly one *authority_rule* of type *Text*, by which the authority of the naming convention shall be specified.

— A *Naming_Convention* shall have exactly one *semantic_rule* of type *Text*, by which the semantics of the *designation_signs* conforming to the naming convention shall be specified.

— A *Naming_Convention* shall have exactly one *syntactic_rule* of type *Text*, by which the syntax of the *designation_signs* conforming to the naming convention shall be specified.

— A *Naming_Convention* shall have exactly one *lexical_rule* of type *Text*, by which the appearance of the *designation_signs* conforming to the naming convention shall be specified.

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

7.3.2.7.2 Attributes of *Naming_Convention*

7.3.2.7.2.1 scope_rule

Attribute name:	*scope_rule*
Definition:	rule specifying the range within which the *naming_convention* is in effect.
Obligation:	Mandatory
Multiplicity:	*1*
Datatype:	*Text* (6.2.12)
NOTE	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.

7.3.2.7.2.2 authority_rule

Attribute name:	*authority_rule*
Definition:	rule identifying the authority that assigns *Designation.signs* (names) and/or enforces *naming conventions*
Obligation:	Mandatory
Multiplicity:	*1*
Datatype:	*Text* (6.2.12)
NOTE	Examples of authorities include information technology standards committees or nomenclature standardization bodies (e.g., in biology).

7.3.2.7.2.3 semantic_rule

Attribute name:	*semantic_rule*
Definition:	rule specifying the meanings of parts of a *Designation.sign* (name) and possibly separators that delimit them in a *Naming_Convention*
Obligation:	Mandatory
Multiplicity:	*1*
Datatype:	*Text* (6.2.12)
NOTE	The rule should specify whether or not names convey meaning, and if so how.

7.3.2.7.2.4 syntactic_rule

Attribute name:	*syntactic_rule*
Definition:	rule specifying the arrangement of parts of a *Designation.sign* (name) and the separators that delimit them in a *Naming_Convention*
Obligation:	Mandatory
Multiplicity:	*1*
Datatype:	*Text* (6.2.12)
NOTE	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.

7.3.2.7.2.5 lexical_rule

Attribute name:	*lexical_rule*
Definition:	rule specifying the appearance of *Designation.signs* (names): preferred and non-preferred terms, synonyms, abbreviations, part length, spelling, permissible character set, case sensitivity, etc. [Derived from ISO/IEC 11179-5]
Obligation:	Mandatory
Multiplicity:	*1*
Datatype:	*Text* (6.2.12)
NOTE	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.

—— End of attributes of *Naming_Convention* ——

7.3.3 Association Classes in the Designation and Definition metamodel region

7.3.3.1 Definition_Context association class

7.3.3.1.1 Description of *Definition_Context*

The *Definition_Context* association class records the *Context* (7.3.2.5) in which a *Definition* (7.3.2.4) occurs.

The association has two roles:

— relevant_definition (verb form: includes_relevant_definition) which references a *Definition;*

— scope (verb form: occurs_in_scope) which references a *Context*.

A relevant_definition may occur within zero or more scopes. A scope may include zero or more relevant_definitions.

Definition_Context may have zero or one *acceptability*, specifying the acceptability of a particular *Definition* within the specified *Context.*

7.3.3.1.2 Attributes of *Definition_Context*

7.3.3.1.2.1 acceptability

Attribute name:	*acceptabilty*
Definition:	**acceptability rating** (3.2.1) of the *Definition* in the specified *Context*.
Obligation:	Optional
Multiplicity:	*0..1*
Datatype:	*Acceptability* (7.3.2.1)

—— End of attributes of *Definition_Context* ——

7.3.3.2 Designation_Context association class

7.3.3.2.1 Description of *Designation_Context*

The *Designation_Context* association class records the *Context* (7.3.2.5) in which a *Designation* (7.3.2.3) occurs.

The association has two roles:

— relevant_designation (verb form: includes_relevant_designation) which references a *Designation* class;

— scope (verb form: occurs_in_scope) which references a *Context* class.

A relevant_designation may have zero or more scopes. A scope may include zero or more relevant_designations.

Designation_Context may have zero or one *acceptability*, specifying the acceptability of a particular *Designation* within the specified *Context.*

7.3.3.2.2 Attributes of *Designation_Context*

7.3.3.2.2.1 acceptability

Attribute name:	*acceptabilty*
Definition:	**acceptability rating** (3.2.1) of the *Designation* in the specified *Context*.
Obligation:	Optional
Multiplicity:	*0..1*
Datatype:	*Acceptability* (7.3.2.1)

—— End of attributes of *Designation_Context* ——

7.3.4 Associations in the Designation and Definition metamodel region

7.3.4.1 designation_definition_pairing association

The *designation_definition_pairing* association records the bindings of zero or one *Designation* (7.3.2.3) to zero or one *Definition* (7.3.2.4).

NOTE 1 If there is a need to pair a designation with different definitions in different contexts, then this requires the use of separate *Designatable_Items* (7.3.2.2)*, Designations* and *Definitions*, even though the *Sign* (6.2.10) used by each *Designation* may be identical.

NOTE 2 The requirement that a *Designation* be associated with a *Designatable_Item*, means that it is not possible to use the registry simply to record terms as designations with associated definitions without reference to some item. It is anticipated that such a requirement would be met by the use of a *Concept_System* (9.1.2.2), where each term would be specified as a *Concept* (9.1.2.1).

The association has two roles:

— definition_heading (verb form: used_for_definition_heading) which references a *Designation*;

— specific_definition (verb form: defined_as) which references a *Definition*.

7.3.4.2 designation_namespace association

The *designation_namespace* association records the bindings of zero or more *Designations* (7.3.2.3) to zero or more *Namespaces* (7.3.2.6).The association is used to record the *Namespace*s in which a *Designation* is valid.

The association has two roles:

— namespace (verb form: occurs_in_namespace) which references a *Namespace;*

— binding (verb form: binds_to) which references a *Designation*.

7.3.4.3 item_definition association

The *item_definition* association records the binding of exactly one *Designatable_Item* (7.3.2.2) to zero or more *Definitions* (7.3.2.4)*,* recording also that the *Designatable_Item* is defined by the *Definitions.* The association records all of the definitions for a specific *Designatable_Item*.

The association has two roles:

— item (verb form: used_for_item) which references a *Designatable_Item;*

— definition (verb form: has_definition) which references a *Definition*.

A *Definition* is used for exactly one *Designatable_item*. Definitions may not be not be reused across multiple *Designatable_Item*s because each such item should be distinct and distinguishable, and this should be reflected in the *Definition*, if not in the text itself, then in the associated *Context*.

7.3.4.4 item_designation association

The *item_designation* association records the bindings of exactly one *Designatable_Item* (7.3.2.2) to zero or more *Designations* (7.3.2.3)*,* records all of the *Designation*s (sign + language pairs) of a *Designatable_Item.*

The association has two roles:

— item (verb form: used_for_item)

— designation (verb form: has_designation).

The item role references a *Designatable_Item.* The designation role references a *Designation*. Each designation shall be used for exactly one item. An item may have zero or more designations.

*Designation*s may not be not be reused across multiple *Designatable_Items* because each such item should be distinct and distinguishable, and this should be reflected in the *Designation*, if not in the *Sign* itself, then in the associated *Context* or *Namespace*.

7.3.4.5 naming_convention_conformance association

The *naming_convention_conformance* association between a *Designation* (7.3.2.3) and zero or more *Naming_Conventions* (7.3.2.7) records the *Naming_Convention*s (if any) to which a particular *Designation* conforms.

The association has two roles:

— convention (verb form: conforms_to)

— conformant_designation (verb form: has_conformant_designation).

The convention role refers to a *Naming_Convention*. The conformant_designation role refers to a *Designation*. Each conformant_designation may have zero or more conventions. Each convention may have zero or more conformant_designations.

7.3.4.6 naming_convention_utilization association

The *naming_convention_utilization* association between a *Namespace* (7.3.2.6) and zero or more *Naming_Conventions* (7.3.2.7) records the *Naming_Convention*s (if any) used by a *Namespace*.

The association has two roles:

— utilization (verb form: utilized_by);

— acceptable_convention (verb form: accepted_convention).

The utilization role refers to a *Namespace*. The acceptable_convention role refers to a *Naming_Convention*. Each utilization may utilize zero or one accepted_conventions. Each accepted_convention has zero or more utilizations.

8 Registration package

8.1 Registration metamodel region

8.1.1 Overview

The Registration region supports the registration of items in a registry. ISO/IEC 11179-6 further describes the registration of *Administered_Items* (8.1.2.2).

Figure 7 on p.63 shows the classes, relationships, attributes and composite attributes that support Registration.

8.1.2 Classes in the Registration region

8.1.2.1 Registered_Item class

8.1.2.1.1 Direct superclass

Identified_Item (7.2.2.1).

8.1.2.1.2 Description of *Registered_Item*

Registered_Item is a class each instance of which models a **registered item** (3.2.105). *A* registered item is an **identified item** (3.2.64) that is registered and managed in a **metadata registry** (3.2.78).

A *Registered_Item* may also be a *Designatable_Item* (7.3.2.2), having one or more *Designations* (7.3.2.3) and/or *Definitions* (7.3.2.4). A **registration authority** (3.2.109) may specify that a *Registered_Item* is required to have at least one *Designation* and *Definition*.

A *Registered_Item* may also be a *Classifiable_Item* (9.2.2.1), associated with zero or more *Concepts* (9.1.2.1) in one or more *Classifications* (9.2.3.1).

A *Registered_Item* must be either an *Administered_Item* (8.1.2.2) or an *Attached_Item* (8.1.2.3) but not both.

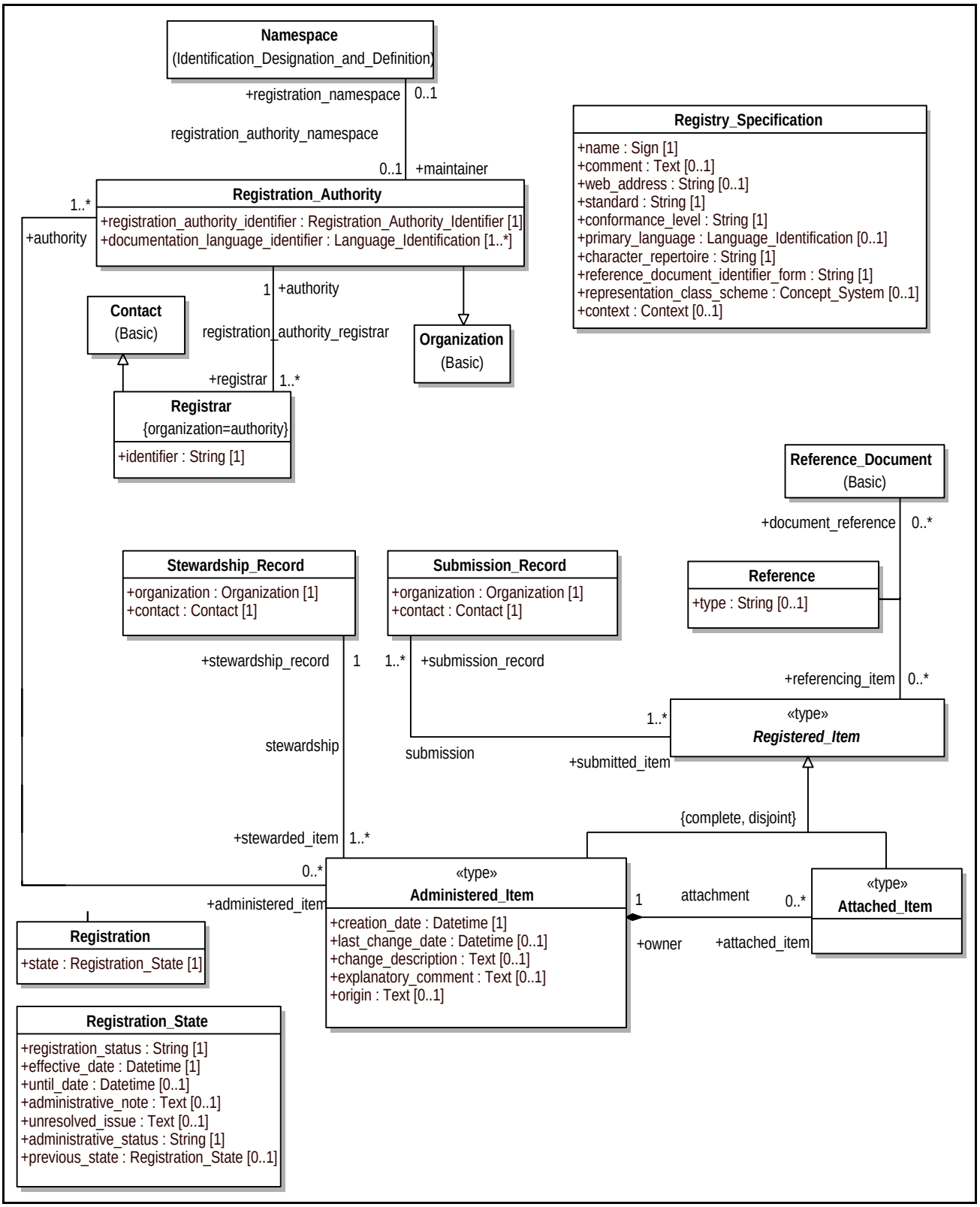

Figure 7 — Registration metamodel region

A *Registered_Item* shall have a *submission* association (8.1.6.5) with one or more *Submission_Record*(s) (8.1.2.8) which identifies a submission *organization* (8.1.2.8.2.1) of type *Organization* (8.1.3.2), and a submission *contact* (8.1.2.8.2.2) of type *Contact* (8.1.3.1). The *organization* is the *Organization* that has submitted the *Registered_Item* for addition, change or cancellation/withdrawal within a **metadata registry** (3.2.78). The *contact* is the *Contact* at the *Organization* for issues related to the *submission.*

A *Registered_Item* may be described by zero or more *Reference_Documents* (8.1.3.3) as represented by the association class *Reference*(8.1.5.2).

8.1.2.2 Administered_Item class

8.1.2.2.1 Direct superclass

Registered_Item (8.1.2.1).

8.1.2.2.2 Description of *Administered_Item*

Administered_Item is a class each instance of which models an **administered item** (3.2.2). An administered item is a **registered item** (3.2.105) for which **administrative information** (3.2.3) is recorded by a **registration authority** (3.2.109).

Every *Administered_Item* must have one or more *Registration* (8.1.5.1) associations with one or more *Registration_Authority* (8.1.2.5).

Every *Administered_Item* shall have exactly one *stewardship* association (8.1.6.4) with a *Stewardship_Record* (8.1.2.7), which identifies a stewardship *organization* (8.1.2.7.2.1) of type *Organization* (8.1.3.2) and a stewardship *contact* (8.1.2.7.2.2) of type *Contact* (8.1.3.1), which is responsible for maintaining the item's administrative information.

An *Administered_Item* may have an *attachment* association (8.1.6.1) with zero or more *Attached_Items* (8.1.2.3). The set of *Attached_Items* that participate in this association are administered collectively under the one *Administered_Item* – they all share the same *stewardship*, *Registration*s, and administrative information.

NOTE *Attached_Items* may share the same *Administered_Item* and still have different *submission_organizations* (8.1.2.8.2.1).

The attributes of the *Administered_Item* class are summarized here and specified more formally in 8.1.2.2.3.

An *Administered_Item* contains:

— Exactly one *creation_date* (8.1.2.2.3.1) of type *Datetime* (6.2.4) that identifies the date and time that the *Administered_Item* was created.

— Zero or one *last_change_date* (8.1.2.2.3.2) of type *Datetime* (6.2.4) that specifies the date and time that the *Administered_Item* was last changed.

— Zero or one *change descriptions* (8.1.2.2.3.3) of type *Text* (6.2.12) that describes what has changed in the *Administered_Item* since the prior version.

— Zero or one *explanatory_comment* (8.1.2.2.3.4) of type *Text* (6.2.12) that contains descriptive comments about the *Administered_Item*.

— Zero or one *origin* (8.1.2.2.3.5) of type *Text* (6.2.12) that describes the source (document, project, discipline or model) of the *Administered_Item*.

8.1.2.2.3 Attributes of *Administered_Item*

8.1.2.2.3.1 creation_date

Attribute name:	*creation_date*
Definition:	date the *Administered_Item* was created
Obligation:	Mandatory
Multiplicity:	*1*

Datatype: *Datetime* (6.2.4)

8.1.2.2.3.2 last_change_date

Attribute name: *last_change_date*

Definition: date the *Administered_Item* was last changed

Obligation: Optional

Multiplicity: *0..1*

Datatype: *Datetime* (6.2.4)

8.1.2.2.3.3 change_description

Attribute name: *change_description*

Definition: description of what has changed since the prior version of the *Administered_Item*

Obligation: Optional

Multiplicity: *0..1*

Datatype: *Text* (6.2.12)

Comment: Previous versions of the *Administered_Item* might or might not be maintained in the registry.

8.1.2.2.3.4 explanatory_comment

Attribute name: *explanatory_comment*

Definition: descriptive comments about the *Administered_Item*

Obligation: Optional

Multiplicity: *0..1*

Datatype: *Text* (6.2.12)

8.1.2.2.3.5 origin

Attribute name: *origin*

Definition: the source (e.g. document, project, discipline or model) for the *Administered_Item*

Obligation: Optional

Multiplicity: *0..1*

Datatype: *Text* (6.2.12)

—— End of attributes of *Administered_Item* ——

8.1.2.3 Attached_Item class

8.1.2.3.1 Direct superclass

Registered_Item (8.1.2.1).

8.1.2.3.2 Description of *Attached_Item*

Attached_Item is a class each instance of which models an **attached item** (3.2.6). An attached item is a **registered item** (3.2.105) for which **administrative information** (3.2.3) is recorded in another registered item (an **administered item** (3.2.2)). Every *Attached_Item* has an *attachment* (8.1.6.1) association with an *owning Administered_Item* (8.1.2.2), which supplies the administrative information.

Attached_Item provides a mechanism by which to administer a set of *Registered_Items* (8.1.2.1) together, as a group, rather than maintaining separate administrative information for every individual item.

EXAMPLE 1 the *Value_Meanings* (11.3.2.3) within a *Conceptual_Domain* (11.3.2.1) may be attached to, and thus administered with, the containing *Conceptual_Domain*.

EXAMPLE 2 all the *Assertions* (9.1.2.3) and *Concepts* (9.1.2.1) within a *Concept_System* (9.1.2.2) may be attached to, and thus administered with, the containing *Concept_System*.

8.1.2.4 Registrar class

8.1.2.4.1 Direct superclass

Contact (8.1.3.1).

8.1.2.4.2 Description of *Registrar*

Registrar is a class each instance of which represents a **registrar** (3.2.106), a **contact** (3.2.23) that is a representative of the **registration authority** (3.2.109). A *Registration_Authority* (8.1.2.5) is represented by one or more *Registrars*. A *Registrar* has a *registration_authority_registrar* association (8.1.6.3) with exactly one *authority Registration_Authority.*

Registrars are the persons who perform the administrative steps to register **administered items** (3.2.2) in a **metadata registry** (3.2.78).

Registrar has one mandatory attribute *identifier* (8.1.2.4.3.1) of type *String* (6.2.11) that identifies a *Registrar*.

8.1.2.4.3 Attributes of *Registrar*

8.1.2.4.3.1 identifier

Attribute name:	*identifier*
Definition:	**identifier** (3.1.11) for the *Registrar*
Obligation:	Mandatory
Multiplicity:	*1*
Datatype:	*String* (6.2.11)

—— End of attributes of *Registrar* ——

8.1.2.4.4 Constraint on *Registrar*

The *organization* (6.3.2.2.2) that *Registrar* inherits from *Contact* (8.1.3.1) shall be the same *Organization* (8.1.3.2) as the associated *Registration_Authority* (8.1.2.5).

8.1.2.5 Registration_Authority class

8.1.2.5.1 Direct superclass

Organization (8.1.3.2).

8.1.2.5.2 Description of Registration_Authority

Registration_Authority is a class each instance of which models a **registration authority** (3.2.109), an **organization** (3.2.90) responsible for maintaining a **register** (3.2.104).

A registration authority may register many **administered items** (3.2.2) as shown by the *Registration* (8.1.5.1) association class.

A *Registration_Authority* shall have a *registration_authority_namespace* association (8.1.6.2) with a *Namespace* (8.1.4.1) that provides the scope for the *registration_authority_identifier* (8.1.2.5.3.1) for the *Registration_Authority.*

The attributes of the *Registration_Authority* class are summarized here and specified more formally in 8.1.2.5.3.

— A *Registration_Authority* shall have an *registration_authority_identifier* (8.1.2.5.3.1) of type *Registration_Authority_Identifier* (6.3.8), which is the identifier for the *Registration_Authority.*

— A *Registration_Authority* shall have one or more *documentation_language_identifiers* (8.1.2.5.3.2) of type *Language_Identification* (6.3.5), which specify the language(s) used for documentation by the *Registration_Authority.*

8.1.2.5.3 Attributes of Registration_Authority

8.1.2.5.3.1 registration_authority_identifier

Attribute name:	*registration_authority_identifier*
Definition:	**identifier** (3.1.11) of a *Registration_Authority*
Obligation:	Mandatory
Multiplicity:	*1*
Datatype:	*Registration_Authority_Identifier* (6.3.8)

8.1.2.5.3.2 documentation_language_identifier

Attribute name:	*documentation_language_identifier*
Definition:	**identifier** (3.1.11) of the *Language* used for documentation by the *Registration_Authority*
Obligation:	Mandatory
Multiplicity:	*1..**
Datatype:	*Language_Identification* (6.3.5)
Comment:	A registration authority may choose to default this attribute to *Registry_Specification.registry_primary_language* (8.1.2.9.2.7).

—— End of attributes of *Registration_Authority* ——

8.1.2.6 Registration_State class

8.1.2.6.1 Description of Registration_State

A *Registration_State* is a collection of information about the *Registration* (8.1.5.1) of an *Administered Item* (8.1.2.2).

The attributes of the *Registration_State* class are summarized here and specified more formally in 8.1.2.6.2.

A *Registration_State* shall have:

— a *registration_status* (8.1.2.6.2.1) of type *String* (6.2.11) which designates the status of an *Administered_Item* (8.1.2.2) in the registration life-cycle;

— an *effective_date* (8.1.2.6.2.2) of type *Datetime* (6.2.4) that identifies the date and time that an *Administered_Item* (8.1.2.2) became or will become available to registry users.

A *Registration_State* may have:

— an *administrative_status* (8.1.2.6.2.6) of type *String* (6.2.11) which designates the status of an *Administered_Item* (8.1.2.2) in the administrative process of a *Registration_Authority* (8.1.2.5);

— an *until_date* (8.1.2.6.2.3) of type *Datetime* (6.2.4) that identifies the date and time that an *Administered_Item* (8.1.2.2) is or will no longer be effective in the registry;

— an *administrative_note* (8.1.2.6.2.4) of type *Text* (6.2.12) that contains general comments and instructions about the *Administered_Item* (8.1.2.2)*;*

— an *unresolved_issue* (8.1.2.6.2.5) of type *Text* (6.2.12) that documents any problem that remains unresolved regarding proper documentation of the *Administered_Item* (8.1.2.2);

— a *previous_state* (8.1.2.6.2.7) of type *Registration_State* (8.1.2.6) which records the immediately prior state of state of the registration.

8.1.2.6.2 Attributes of Registration_State

8.1.2.6.2.1 registration_status

Attribute name:	*registration_status*
Definition:	**designation** (3.2.51) of the status in the registration life-cycle of an *Administered_Item*
Obligation:	Mandatory
Multiplicity:	*1*
Datatype:	*String* (6.2.11)
NOTE	Designation values are described in ISO/IEC 11179-6.

8.1.2.6.2.2 effective_date

Attribute name:	*effective_date*
Definition:	date and time an *Administered_Item* became/becomes available to registry users
Obligation:	Mandatory
Multiplicity:	*1*
Datatype:	*Datetime* (6.2.4)

8.1.2.6.2.3 until_date

Attribute name:	*until_date*
Definition:	date and time the *Registration* of an *Administered_Item* by a *Registration_Authority* in a registry is no longer effective
Obligation:	Optional
Multiplicity:	*0..1*
Datatype:	*Datetime* (6.2.4)

8.1.2.6.2.4 administrative_note

Attribute name:	*administrative_note*
Definition:	general note(s) about the *Registration*
Obligation:	Optional
Multiplicity:	*0..1*
Datatype:	*Text* (6.2.12)

8.1.2.6.2.5 unresolved_issue

Attribute name: *unresolved_issue*

Definition: any problem(s) that remains unresolved regarding proper documentation of the *Administered_Item*

Obligation: Optional

Multiplicity: *0..1*

Datatype: *Text* (6.2.12)

8.1.2.6.2.6 administrative_status

Attribute name: *administrative_status*

Definition: **designation** (3.2.51) of the status in the administrative process of a *Registration_Authority*

Obligation: Optional

Multiplicity: *0..1*

Datatype: *String* (6.2.11)

NOTE The values and associated meanings of the *administrative_status* are determined by each *Registration_Authority*. cf. *registration_status*.

8.1.2.6.2.7 previous_state

Attribute name: *previous_state*

Definition: immediately prior collection of **administrative information** (3.2.3) about registration.

Obligation: Optional

Multiplicity: *0..1*

Datatype: *Registration_State* (8.1.2.6)

—— End of attributes of *Registration_State* ——

8.1.2.7 Stewardship_Record class

8.1.2.7.1 Description of Stewardship_Record

Stewardship_Record is a class each instance of which identifies both a stewardship *organization* (8.1.2.7.2.1) of type *Organization* (8.1.3.2) and a stewardship *contact* (8.1.2.7.2.2) of type *Contact* (8.1.3.1) at the *Organization*, responsible for the **stewardship** (3.2.125) of one or more *Administered_Items* (8.1.2.2) as represented by the s*tewardship* association (8.1.6.4).

The attributes of the *Stewardship_Record* class are summarized here and specified more formally in 8.1.2.7.2.

A *Stewardship_Record* shall have:

— an *organization* (8.1.2.7.2.1) of type *Organization* (6.3.6)*;*

— a *contact* (8.1.2.7.2.2) of type *Contact* (6.2.3)*.*

8.1.2.7.2 Attributes of Stewardship_Record

8.1.2.7.2.1 organization

Attribute name: *organization*

Definition: *Organization* that maintains *stewardship* of an *Administered_Item.*

Obligation: Mandatory

Multiplicity: *1*

Datatype: *Organization* (6.3.6)

8.1.2.7.2.2 contact

Attribute name: *contact*

Definition: Contact information associated with a *Stewardship*

Obligation: Mandatory

Multiplicity: *1*

Datatype: *Contact* (6.2.3)

—— End of attributes of *Stewardship_Record* ——

8.1.2.8 Submission_Record class

8.1.2.8.1 Description of Submission_Record

Each *Registered_Item* (8.1.2.1) must have a *submission* association (8.1.6.5) with one or more *Submission_Record*(s) which identifies a submission *organization* (8.1.2.8.2.1) of type *Organization* (8.1.3.2), and a submission *contact* (8.1.2.8.2.2) of type *Contact* (8.1.3.1). The submission *organization* is the organization that has submitted the *Registered_Item* for addition, change or cancellation/withdrawal within a **metadata registry** (3.2.78). The submission *contact* is the *Contact* at the *Organization* for issues related to the *submission.*

NOTE A *Submission_Record* is required for *Attached_Items* (8.1.2.3) as well as to *Administered_Items* (8.1.2.2) because they can be submitted separately. However, one submission record can be used for multiple items submitted together.

The attributes of the *Submission_Record* class are summarized here and specified more formally in 8.1.2.8.2.

A *Submission_Record* shall have:

— an *organization* of type *Organization;*

— a *contact* of type *Contact.*

8.1.2.8.2 Attributes of Submission_Record

8.1.2.8.2.1 organization

Attribute name: *organization*

Definition: *Organization* which has submitted a *Registered_Item* for inclusion in a register.

Obligation: Mandatory

Multiplicity: *1*

Datatype: *Organization* (6.3.6)

8.1.2.8.2.2 contact

Attribute name: *contact*

Definition: Contact information associated with a *Submission.*

Obligation: Mandatory

Multiplicity: *1*

Datatype: *Contact* (6.2.3)

—— End of attributes of *Submission_Record* ——

8.1.2.9 Registry_Specification class

8.1.2.9.1 Description of Registry_Specification

Registry_Specification describes the environment in which a **registry** (3.2.113) operates. If this standard is being applied in an environment that does not use a registry, then *Registry_Specification* is not required. In an environment with multiple registries, there should be one *Registry_Specification* per registry.

The attributes of the *Registry_Specification* class are summarized here and specified more formally in 8.1.2.9.2.

A *Registry_Specification* shall have:

— a *name* (8.1.2.9.2.1.1) of type *Sign* (6.2.10)

— a *web_address* (8.1.2.9.2.2) of type *String* (6.2.11)

— a *standard* (8.1.2.9.2.3) of type *String* (6.2.11)

— a *conformance_level* (8.1.2.9.2.4) of type *String* (6.2.11)

— a *character_repertoire* (8.1.2.9.2.5) of type *String* (6.2.11)

— a *reference_document_identifier_form* (8.1.2.9.2.6) of type *String* (6.2.11).

A *Registry_Specification* may have:

— a *primary_language* (8.1.2.9.2.7) of type *Language_Identification* (6.3.5)

— a *representation_class_scheme* (8.1.2.9.2.8) of type *Concept_System* (9.1.2.2)

— a *context* (8.1.2.9.2.9) of type *Context* (7.3.2.5)

— a *comment* (8.1.2.9.2.10) of type *Text* (6.2.12).

8.1.2.9.2 Attributes of Registry_Specification

8.1.2.9.2.1.1 name

Attribute name: *name*

Definition: **name** (3.2.83) by which the *Registry* is commonly known.

Obligation: Mandatory

Multiplicity: *1*

Datatype: *Sign* (6.2.10)

EXAMPLE US EPA Environmental Data Registry

8.1.2.9.2.2 web_address

Attribute name: *web_address*

Definition: The World Wide Web uniform resource locator (url) for the registry.

Obligation: Optional

Multiplicity: *0..1*

Datatype: *String* (6.2.11)

EXAMPLE www.epa.gov/edr

8.1.2.9.2.3 standard

Attribute name: *standard*

Definition: The standard to which this registry complies.

Obligation: Mandatory

Multiplicity: *1*

Datatype: *String* (6.2.11)

EXAMPLE ISO/IEC 11179-3:2003 with Cor 1:2004

8.1.2.9.2.4 conformance_level

Attribute name: *conformance_level*

Definition: The conformance level of the registry as described in the standard.

Obligation: Mandatory

Multiplicity: *1*

Datatype: *String* (6.2.11)

EXAMPLE Conformance level 2

8.1.2.9.2.5 character_repertoire

Attribute name: *character_repertoire*

Definition: The character reperoire that is used by the registry for internal operation.

Obligation: Mandatory

Multiplicity: *1*

Datatype: *String* (6.2.11)

EXAMPLE ISO/IEC 646:1991 or ISO/IEC 10646-1:2000

8.1.2.9.2.6 reference_document_identifier_form

Attribute name: *reference_document_identifier_form*

Definition: Specification of the form of **identifier** (3.1.11) used for identifying *Reference_Document*s in the registry.

Obligation: Mandatory

Multiplicity: *1*

Datatype: *String* (6.2.11)

NOTE Some registries might use URIs. Other registries might utilize an external document management system which provides unstructured, opaque identifiers. Yet other registries might define a structured identifier form which embeds an identifier type within each identifier in the registry. This attribute specifies the form of identifiers used for *Reference_Document*s in this particular registry.

8.1.2.9.2.7 primary_language

Attribute name: *registry_primary_language*

Definition: The primary and/or default language that is used for the registry.

Obligation: Optional

Multiplicity: *0..1*

Datatype: *Language_Identification* (6.3.5)

EXAMPLE *eng-840 or en-US*

8.1.2.9.2.8 representation_class_scheme

Attribute name: *registry_representation_class_scheme*

Definition: *Concept_System* used by the registry to capture representation classes.

Obligation: Optional

Multiplicity: *0..1*

Datatype: *Concept_System* (9.1.2.2)

NOTE Edition 2 had a separate structure to record representation classes. In Edition 3, these are considered to be just another classification scheme, which in turn is considered a *Concept_System*. This attribute allows the registry to specify which *Concept_System* is used for this purpose.

8.1.2.9.2.9 context

Attribute name: *context*

Definition: *Context* which represents the *Registry* itself.

Obligation: Optional

Multiplicity: *0..1*

Datatype: *Context* (7.3.2.5)

Comment: It might sometimes be useful to reference a default *Context*, rather than create a new one. The *Registry_Specification.context* is one possible choice for such a default.

8.1.2.9.2.10 comment

Attribute name: *comment*

Definition: any comment that should be noted about the registry.

Obligation: Optional

Multiplicity: *0..1*

Datatype: *Text* (6.2.12)

—— End of attributes of *Registry_Specification* ——

8.1.3 Classes referenced from the Basic package

8.1.3.1 Contact class

Contact is a class each instance of which models a **contact** (3.2.23), which specifies a **role** (3.2.121) and/or an **individual** (3.2.65) within an **organization** (3.2.90) or an **organization part** (3.2.93) to whom information item(s), material object(s) and/or person(s) can be sent to or from. *Registrar* (8.1.2.4) is a subclass of *Contact*. *Contact* is further described in 6.3.2.

8.1.3.2 Organization class

Organization is a class each instance of which models an **organization** (3.2.90), a unique framework of authority within which **individual**s (3.2.65) act, or are designated to act, towards some purpose.

An *Organization* can play one or more **role**s (3.2.121) with respect to a **metadata registry** (3.2.78). All *Registration Authorities* (8.1.2.5) are *Organizations*, but not all *Organizations* are necessarily *Registration Authorities*. An *Organization* may be a submission *organization* (8.1.2.8.2.1) within a *Submission_Record*

(8.1.2.8), with zero or more *submitted item Registered_Items* (8.1.2.1). An *Organization* may also have a *stewardship* association (8.1.6.4) with zero or more *stewarded_item Administered_Items* (8.1.2.2) where the *Organization* acts as the stewardship *organization* (8.1.2.7.2.1) for the item.

Organization is further described in 6.3.6.

8.1.3.3 Reference_Document class

Reference_Document is a class each instance of which models a **reference document** (3.2.102), a document that provides pertinent details for consultation about a subject. *Reference_Document* is specified in 6.3.7.

A *Registered_Item* (8.1.2.1) may reference zero or more *Reference_Documents* as shown by the association class *Reference* (8.1.5.2) in Figure 7 on p. 63 .

8.1.4 Classes referenced from the Identification, Designation and Definition package

8.1.4.1 Namespace class

Namespace is described in 7.2.2.3. A *Namespace* is a scoping construct used to group sets of *Designation*s (7.3.2.3) and/or *Scoped_Identifier*s (7.2.2.2) used in a **metadata registry** (3.2.78). A *Registration_Authority* (8.1.2.4.4) may have a *registration_authority_namespace* association (8.1.6.2) with a *Namespace* that provides the *registration_namespace* for the *Registration_Authority.*

8.1.5 Association Classes in the Registration region

8.1.5.1 Registration association class

8.1.5.1.1 Description of Registration

Registration is an association between a *Registration_Authority* (8.1.2.4.4) and an *Administered_Item* (8.1.2.2) where the *Registration_Authority* manages the *Administered_Item* in a **metadata register** (3.2.77).

Registration is also a class, and has an attribute *registration_state* of type *Registration_State*, as specified below.

8.1.5.1.2 Attributes of Registration

8.1.5.1.2.1 registration_state

Attribute name:	*registration_state*
Definition:	current collection of administrative data about *Registration*
Obligation:	Mandatory
Multiplicity:	*1*
Datatype:	*Registration_State* (8.1.2.6)

—— End of attributes of *Registration* ——

8.1.5.2 Reference association class

8.1.5.2.1 Description of Reference

A *Reference* is the association between a *Reference_Document* (8.1.3.3) and a *Registered_Item* (8.1.2.1).

A *Reference* is also a class, and may have zero or one *type* of type *String*, as specified below.

8.1.5.2.2 Attributes of Reference

8.1.5.2.2.1 type

Attribute name:	*type*
Definition:	specification of the type of *Reference*
Obligation:	Optional
Multiplicity:	*0..1*
Datatype:	*String* (6.2.11)

—— End of attributes of *Reference* ——

8.1.6 Associations in the Registration region

8.1.6.1 attachment association

The *attachment* association records the binding of exactly one *Administered_Item* (8.1.2.2) with zero or more *Attached_Item* (8.1.2.3) indicating that the *Attached_Item* shares all of the **administrative information** (3.2.3) of the *Administered_Item*. *attachment* enables collections of *Registered_Items* (8.1.2.1) to be administered collectively as a block.

attachment has two roles:

— *owner* (verb form: *has owner*) references an *Administered_Item*;

— *attached item* (verb form: *attached to*) references an *Attached_Item*.

8.1.6.2 registration_authority_namespace association

The *registration_authority_namespace* association records the binding of zero or one *Registration_Authority* (8.1.2.4.4) with zero or one *Namespace* (8.1.4.1), that provides the scope for the *registration_authority_identifier* (8.1.2.5.3.1) for the *Registration_Authority.*

The association has two roles:

— registration_namespace (verb form: has_registration_namespace) which references a *Namespace*;

— maintainer (verb form: maintained_by) which references a *Registration_Authority*

8.1.6.3 registration_authority_registrar association

The *registration_authority_registrar* association records the binding of exactly one *Registration_Authority* (8.1.2.4.4) with one or more *Registrars* (8.1.2.4) indicating that the *Registrar* is a representative of the *Registration_Authority*.

The association has two roles:

— authority (verb form: authorized_by) which references a *Registration_Authority*;

— registrar (verb form: has_registrar)

8.1.6.4 stewardship association

The *stewardship* association records the binding of exactly one *Stewardship_Record* (8.1.2.7) to one or more *Administered_Items* (8.1.2.2).

The association has two roles:

— stewardship_record (verb form: stewarded_by) which references a *Stewardship_Record*

— stewarded_item (verb form: stewards) which references one or more *Administered_Items.*

8.1.6.5 submission association

The *submission* association records the binding of exactly one *Submission_Record* (8.1.2.8) with one or more *Registered_Items* (8.1.2.1).

The association has two roles:

— submission_record (verb form: submitted by) which references a *Submission_Record*

— submitted_item (verb form: submits) which references one or more *Registered_Items.*

9 Concepts package

9.1 Concepts metamodel region

9.1.1 Overview

The Concepts metamodel region is illustrated in Figure 8. The purpose of the Concepts Metamodel Region is to describe *Concept*s (9.1.2.1) (abstract units of knowledge) and the various *Relation*s (9.1.2.4) which might hold among Concepts. Ontologies are supported as *Concept_Systems* (9.1.2.2) with formal semantics through the use of *Assertions* (9.1.2.3). Annex E provides examples using SKOS, ORM, OWL and CLIF.

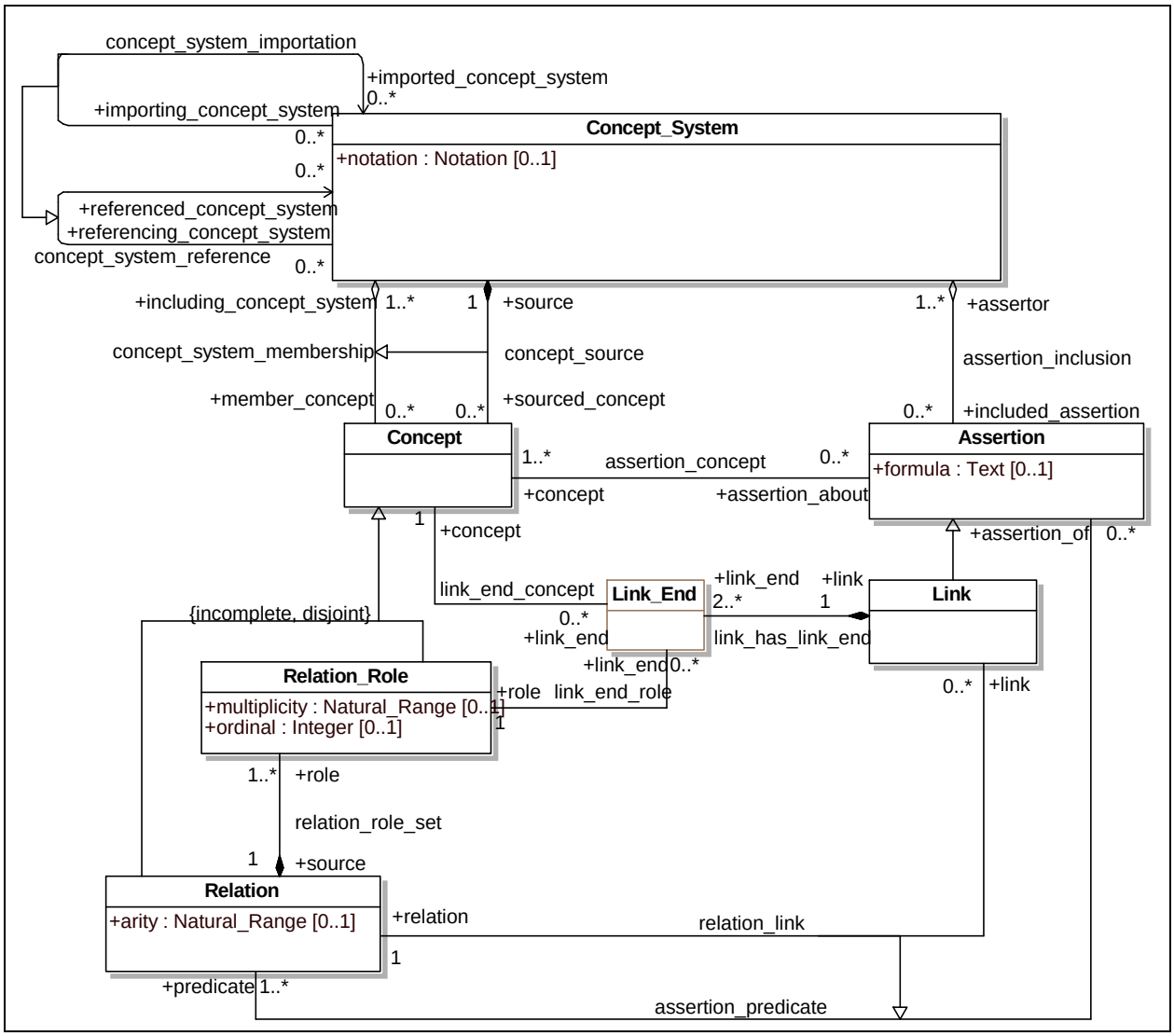

Figure 8 — Concepts metamodel region

9.1.2 Classes in the Concepts metamodel region

9.1.2.1 Concept class

Concept is a class each instance of which models a **concept** (3.2.18), a unit of knowledge created by a unique combination of **characteristic**s (3.2.14). A **concept** is independent of representation.

Concept is a **superclass** (3.1.18) of both *Relation* (9.1.2.4) and *Relation_Role* (9.1.2.5).

A *Concept* shall participate in the following associations:

— *concept_system_membership* (9.1.3.1) by which zero or more *Concepts* may be included in one or more *Concept_Systems* (9.1.2.2). Each *Concept* shall be a member of at least one *Concept_System*.

NOTE The registry may be specified as a *Concept_System* if no more appropriate *Concept_System* is defined.

— *concept_source* (9.1.3.2) by which exactly one *Concept_System* shall be specified as the source of each *Concept.*

A *Concept* may participate in the following associations:

— *assertion_concept* (9.1.3.6) with zero or more *Assertion*s (9.1.2.3) where each *Assertion* asserts something about the referenced *Concept*.

— *link_end_concept* (9.1.3.11) with zero or more *Link_End*s (9.1.2.7) where each *Concept* represents an *end* of the associated *Link* (9.1.2.6).

Several of the classes in the Data Description package (11) are subclasses of *Concept* – see Figure 17 on p.123.

NOTE As for any metadata item, a *Concept* may be defined by making it a *Designatable_Item* (7.3.2.2).

9.1.2.2 Concept_System class

9.1.2.2.1 Description of Concept_System

Concept_System is a class each instance of which models a **concept system** (3.2.19), a set of **concept**s (3.2.18) structured according to the **relation**s (3.2.119) among them.

A minimal concept system could simply be a collection of concepts. A more elaborate concept system could be a collection of concepts which might be organized into a **taxonomy** (3.2.135) or **meronomy** (3.2.73) specified by means of various relations (e.g., semantic relations) and **link**s (3.2.69) amongst the concepts.

NOTE 1 Examples of concept systems are included in Annex E.

NOTE 2 The use of concept systems as **classification scheme**s (3.2.16) is described in clause 9.2.

A *Concept_System* may participate in the following associations:

— *concept_system_reference* (9.1.3.3) by which zero or more referenced *Concept_Systems* may be referenced by zero or more referencing *Concept_System*s.

— *concept_system_importation* (9.1.3.4) by which zero or more imported *Concept_Systems* may be imported into zero or more importing *Concept_System*s. *concept_system_importation* is a specialization of *concept_system_reference.*

— *concept_system_membership* (9.1.3.1) by which zero or more *Concepts* (9.1.2.1) may be included in one or more *Concept_Systems.* Each *Concept* shall be a member of at least one *Concept_System*. Since

Relations (9.1.2.4) and *Relation_Roles* (9.1.2.5) are **subclasses** (3.1.17) of *Concepts*, these too are included in one or more *Concept_Systems*.

— *concept_source* (9.1.3.2) by which exactly one *Concept_System* shall be specified as the source of each *Concept.*

— *assertion_inclusion* (9.1.3.5) by which zero or more *Assertions* (9.1.2.3) may be included in one or more *Concept_Systems.* Since a *Link* (9.1.2.6) is a **subclass** (3.1.17) of *Assertion*, *Links* are also included in one or more *Concept_Systems.*

A *Concept_System* has exactly one *notation* attribute of type *Notation*. The *notation* attribute is used to record the **notation** (3.2.86) used to describe the *Concept_System*.

NOTE 3 Examples of such notations include XCL Common Logic (ISO/IEC 24707) or OWL-DL XML notation (Ontology Web Language from W3C).

NOTE 4 If a **concept system** is available in multiple **notations**, it is good practice to register the *Concept_System* only once, and to use *Reference_Documents* to record the various notations.

9.1.2.2.2 Attributes of Concept_System

9.1.2.2.2.1 notation

Attribute name:	*notation*
Definition:	formal syntax and semantics used in the *Concept_System*.
Obligation:	Optional
Multiplicity:	*0..1*
Datatype:	*Notation* (6.2.7)

—— End of attributes of *Concept_System* ——

9.1.2.3 Assertion class

9.1.2.3.1 Description of Assertion

Assertion is a class each instance of which models an **assertion** (3.2.5)*,* a sentence or proposition in logic which is asserted (or assumed) to be true.

An *Assertion* shall participate in the following associations:

— *assertion_inclusion* (9.1.3.5) with one or more *Concept_Systems* (9.1.2.2) in an *assertor* role;

— *assertion_concept* (9.1.3.6) with one or more *Concepts* (9.1.2.1) in a *concept* role;

— *assertion_predicate* (9.1.3.7) with one or more *Relations* (9.1.2.4) in a *predicate* role.

Assertion has one attribute, *formula* which expresses the assertion.

9.1.2.3.2 Attributes of Assertion

9.1.2.3.2.1 formula

Attribute name:	*formula*
Definition:	text which expresses an **assertion** (3.2.5)
Obligation:	Optional

Multiplicity: *0..1*

Datatype: *Text* (6.2.12

—— End of attributes of *Assertion* ——

9.1.2.3.3 Constraint on Assertions

If *formula* is used, all *Assertions* within a *Concept_System* shall use the same **notation** (3.2.86) for the formulas.

9.1.2.4 Relation class

9.1.2.4.1 Direct superclass

Concept (9.1.2.1)

9.1.2.4.2 Description of Relation

Relation is a class each instance of which models a **relation** (3.2.119), a sense in which **concepts** (3.2.18) may be connected via constituent **relation roles** (3.2.120).

Relation may participate in the following associations:

— *relation_role_set* (9.1.3.9) with one or more *Relation_Roles* (9.1.2.5), each of which specifies the *role* of an element in the *Relation*. The number of *Relation_Roles* is specified by the *arity* (9.1.2.4.3.1) of the *Relation*;

— *relation_link* (9.1.3.8) with zero or more *Links* (9.1.2.6) as members of the *Relation;*

— *assertion_predicate* (9.1.3.7) with zero or more *Assertions* (9.1.2.3) which reference the *predicate Relation*.

Relation is a superclass of the *Binary_Relation* class which is described in 10.1.2.2. *Relation* has one attribute *arity* (9.1.2.4.3.1), which specifies the number of elements in the *relation*.

NOTE 1 A *Relation* is a subset of the powerset of RxUD, for some role set R, where UD is the universe of discourse.

NOTE 2 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 x ... x A_n . Membership of an n-tuple in the *Relation* is specified through *Assertions*, the simplest form of which is a *Link* representing exactly one tuple of the *Relation*. In this metamodel, *Relations* are defined over sets of *Concepts*.

NOTE 3 In this metamodel we actually use unordered n-tuples with named *Relation_Roles* rather than positional elements of the n-tuple. The ordering can optionally be specified using the *ordinal* attribute on *Relation_Role.*

NOTE 4 Any relation is also a concept in its own right (and hence *Relation* is a subclass of *Concept).* A unary relation is just a concept and should be registered as such. Only relations with arity 2 or greater should be registered as *Relations*.

NOTE 5 A reflexive relation (i.e. one which refers to itself) needs to be modelled using an *Assertion*.

9.1.2.4.3 Attributes of Relation

9.1.2.4.3.1 arity

Attribute name: *arity*

Definition: number of elements in the relation

Obligation:	Optional
Multiplicity:	*0..1*
Datatype:	*Natural_Range* (6.2.6)
EXAMPLE	A *binary relation* has an arity of 2.
Constraint:	Arity should be >= 2.

—— End of attributes of *Relation* ——

9.1.2.5 Relation_Role class

9.1.2.5.1 Direct superclass

Concept (9.1.2.1)

9.1.2.5.2 Description of Relation_Role

Relation_Role is a class each instance of which models a **relation role** (3.2.120). A relation role is an argument (element) of a **relation** (3.2.119).

NOTE 1 In relational database terms, the relation role represents a column in a relational table (for an asymmetric relation).

Relation_roles permit position independent naming of the arguments of a *Relation*.

NOTE 2 This is similar to the distinction between positional and named arguments to procedures in programming languages.

For symmetric **binary relations** (3.2.10) we reuse relation roles to indicate multiple arguments (**link ends** (3.2.70)), since the arguments (link ends) are to be treated identically.

Relation_Role shall participate in the following association:

— *relation_role_set* (9.1.3.9) which specifies the set of *Relation_Roles* belonging to a *Relation.*

Relation_Role may participate in the following association:

— *link_end_role* (9.1.3.12) which identifies the role that a *Concept* (9.1.2.1) plays in a *Link_End* (9.1.2.7).

The *Relation_Role* class has two attributes: *multiplicity* and *ordinal*.

9.1.2.5.3 Attributes of Relation_Role

9.1.2.5.3.1 multiplicity

Attribute name:	*multiplicity*
Definition:	number of *links* which must (logically) be members of the source *relation* of this role, differing only by an *end* with this *role* as an *end_role*. For example, if a relation *purchase* with an arity of 3 has roles *buyer*, *seller*, and *item*, then a multiplicity of 0..1 on the *buyer* role means that it is not permitted for more than one member of the *purchase* relation to involve both the same *seller* and the same *item* (differing only in the *buyer*).
Obligation:	Optional
Multiplicity:	*0..1*
Datatype:	*Natural_Range* (6.2.6)

9.1.2.5.3.2 ordinal

Attribute name:	*ordinal*
Definition:	order of the *relation role* among other *relation roles* in the *relation*.
Obligation:	Optional
Multiplicity:	*0..1*
Datatype:	*Integer* (6.2.5)
Comment:	*ordinal* allows the ordering of the *Concepts,* represented in the *Relation* by the *Relation_Roles,* to be specified. This might be necessary if the ordering of the *Concepts* changes the meaning of the *Relation*.

—— End of attributes of *Relation_Role* ——

9.1.2.6 Link class

9.1.2.6.1 Direct superclass

Assertion (9.1.2.3)

9.1.2.6.2 Description of Link

Link is a class each instance of which models a **link** (3.2.69). A link is a member of a **relation** (3.2.119) (not an instance of a relation). In relational database parlance, a link would be a tuple (row) in a relation (table). *Link* is a subclass of *Assertion* (9.1.2.3), and as such is included in one or more *Concept_Systems* (9.1.2.2) through the *assertion_inclusion* (9.1.3.5) association.

Link shall participate in the following associations:

— *assertion_inclusion* (9.1.3.5) with one or more *Concept_Systems* in which it is included;

— *relation_link* (9.1.3.8) with exactly one *Relation* of which it is a *link*;

— *link_has_link_end* (9.1.3.10) with two or more *Link_Ends* (9.1.2.7).

NOTE A *Link* can have two or more *Link_Ends*, depending on the *arity* of the *relation*.

Link has no attributes.

9.1.2.6.3 Constraint on Link class

The number of *Link_Ends* (9.1.2.7) associated with a *Link* must match the *arity* (9.1.2.4.3.1) of the *Relation* (9.1.2.4) of which the *Link* is a member, if the *arity* is specified.

9.1.2.7 Link_End class

9.1.2.7.1 Description of Link_End

Link_End is a class each instance of which models a **link end** (3.2.70). A link end identifies the **relation role** (3.2.119) played by a **concept** (3.2.18) in the associated **link** (3.2.69).

The *Link_End* class models the association among *Links* (9.1.2.6), *Concepts* (9.1.2.1) (ends) and *Relation_Roles* (9.1.2.5). 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* is used to model the instantiation of a *Relation_Role* (9.1.2.5) for a particular *Link* (tuple, row) of a *Relation* (9.1.2.4).

NOTE A symmetric **binary relation** (3.2.10) may use the same *relation role* for both *link ends*.

Link_End shall participate in the following associations:

— *link_has_link_end* (9.1.3.10) with exactly one *Link* (9.1.2.6) for which it is an end

— *link_end_concept* (9.1.3.11) with exactly one *Concept* (9.1.2.1)

— *link_end_role* (9.1.3.12) with exactly one *Relation_Role* (9.1.2.5)

Link_End has no attributes.

9.1.2.7.2 Constraint on Link_End class

It must be the case that the *Relation_Role* specified in the *link_end_role* association must correspond to a *Relation_Role* which is a role of the *Relation* of the *Link* to which the *Link_End* is associated via the *link_has_link_end* association.

9.1.3 Associations of the Concepts metamodel region

9.1.3.1 concept_system_membership association

The *concept_system_membership* association specifies the inclusion of zero or more *Concepts* (9.1.2.1) in one or more *Concept_Systems* (9.1.2.2).

concept_system_membership has two roles:

— *including_concept_system* (verb form: *is_included_in*) which references a *Concept_System;*

— *member_concept* (verb form: *has_member_concept*) which references a *Concept*.

Each *Concept* shall have a *concept_system_membership* association with at least one *Concept_System*.

9.1.3.2 concept_source association

The *concept_source* association specifies the *Concept_System* (9.1.2.2) that is the source of a *Concept* (9.1.2.1).

concept_source has two roles:

— *source* (verb form: *has_source*) which references a C*oncept_System;*

— *specified_concept* (verb form: *specifies_concept*) which references a *Concept*.

Each *Concept* .shall have exactly one *Concept_System* specified as its source.

The *source Concept_System* establishes an explicit minimum scope within which the identity of the *Concept* can be taken to have been determined, in the sense that there should be no other *Concept* within that same scope which represents the same meaning. In some registries (including presumably all Edition 2 implementations), this scope might always be the registry *Concept_System*, thus making explicit the expectation that there shall always be at most one *Concept* within that entire registry representing any given meaning. In other registries the scope might generally be much narrower, reflecting a lack of determination having (necessarily) been made as to whether the same meaning may or may not also be represented by one or more other *Concepts* in the registry, from a different source(s).

The source *Concept_System* also provides a scoping mechanism for assertions pertaining to that *Concept* (generally including many *Assertions* (9.1.2.3) that the *Concept* does not participate in directly). In some registries in which all *Concepts* have a distinguished registry *Concept_System* as their source, all *Assertions* may also be included in that same registry *Concept_System*, thus indicating that all assertions pertaining to any *Concept* are valid across the whole registry context.

NOTE within the ontology community, this is called a ‘uniform ontological commitment’.

In other registries, all *Concepts* may have the registry as their source, but discrimination may be made between *Assertions* included in that registry *Concept_System*, and other *Assertions* which are registered may be valid only in (a) narrower context(s). In yet other registries there might be many different *Concept* instances representing either similar or even arguably identical meanings, but with some potentially critical difference(s) in semantics represented by the distinct sets of *Assertions* included in their respective source *Concept_Systems*.

9.1.3.3 concept_system_reference association

The *concept_system_reference* association specifies the reference of zero or more referenced *Concept_Systems* (9.1.2.2) by zero or more referencing *Concept_System*s.

concept_system_reference has two roles, both of which reference instances of the class *Concept_System*:

— *referenced_concept_system* (verb form: *has_ referenced_concept_system*);

— *referencing_concept_system* (verb form: *has_ referencing_concept_system*).

A *referenced_concept_system* may be referenced by zero or more *referencing_concept_systems*. A *referencing_concept_system* may reference zero or more *referenced_concept_systems*.

A *referenced_concept_system* is not considered to be part of the *referencing_concept_system.*

Navigation from the *referencing_concept_system* to the *referenced_concept_system* shall be supported by a conforming or strictly conforming implementation. Navigation from the *referenced_concept_system* to the *referencing_concept_system* is permitted but not required.

9.1.3.4 concept_system_importation association

The *concept_system_importation* association specifies the importation of zero or more imported *Concept_Systems* (9.1.2.2) by zero or more importing *Concept_System*s. Such importation specifies that all *Concepts* (9.1.2.1) and *Assertions* (9.1.2.3) included in the imported Concept_System are also to be included in the importing Concept_System.

concept_system_importation has two roles, both of which reference instances of the class *Concept_System*:

— *imported_concept_system* (verb form: *has_ imported_concept_system*);

— *importing_concept_system* (verb form: *has_ importing_concept_system*).

An *imported_concept_system* may be imported by zero or more *importing_concept_system*s.

An *importing_concept_system* may import zero or more *imported_concept_system*s.

An *imported_concept_system* is considered to be an integral part of the *importing_concept_system.*

Navigation from the *importing_concept_system* to the *imported_concept_system* shall be supported by a conforming or strictly conforming implementation. Navigation from the *imported_concept_system* to the *importing_concept_system* is permitted but not required.

9.1.3.5 assertion_inclusion association

The *assertion_inclusion* association specifies the inclusion of an *Assertion* (9.1.2.3) in one or more *Concept_Systems* (9.1.2.2).

assertion_inclusion has two roles:

— *assertor* (verb form: *asserted_by*) which references a C*oncept_System;*

— *included_assertion* (verb form: *includes*) which references an *Assertion*.

An *included_assertion* shall be *asserted_by* one or more *Concept_Systems.*

An *assertor Concept_System* may include zero or more *Assertions*.

9.1.3.6 assertion_concept association

The *assertion_concept* association specifies the use of one or more *Concepts* (9.1.2.1) in zero or more *Assertions* (9.1.2.3)*.*

assertion_concept has two roles:

— *concept* (verb form: *uses*) which references a C*oncept;*

— *assertion_about* (verb form: *used_by*) which references an *Assertion*.

An *assertion* shall use one or more *concepts.* A *concept* may be used by zero or more *assertions*.

9.1.3.7 assertion_predicate association

The *assertion_predicate* association specifies the use of one or more *Relations* (9.1.2.4) as predicates in zero or more *Assertions* (9.1.2.3).

assertion_predicate has two roles:

— *predicate* (verb form: *is predicate of*) which references a *Relation*;

— *assertion of* (verb form: *is asserted by*) which references an *Assertion*.

9.1.3.8 relation_link association

The *relation_link* association specifies the membership of zero or more *Links* (9.1.2.6) as *links* in exactly one *Relation* (9.1.2.4)*.*

relation_link has two roles:

— *relation* (verb form: *member of*) which references a *Relation*;

— *link* (verb form: *has link*) which references a *Link*.

A *link* shall be a member of exactly one *relation.* A *relation* may have zero or more *links*.

9.1.3.9 relation_role_set association

The *relation_role_set* association specifies the *Relation_Roles* (9.1.2.5) that participate in the *Relation* (9.1.2.4)*.*

relation_role_set has two roles:

— *role* (verb form: *has_role*) which references a *Relation_Role*;

— *source* (verb form: *has_source*) which references a *Relation*.

A *role* shall have exactly one *source.* A s*ource* shall have one or more *roles*.

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.

9.1.3.10 link_has_link_end association

The *link_has_link_end* association specifies the bind of a *Link* (9.1.2.6) to two or more *Link_Ends* (9.1.2.7).

link_has_link_end has two roles:

— *link* (verb form: *is_link_for)* which references a *Link*

— *link_end* (verb form::*is_end_for*) which references a *Link_End*

9.1.3.11 link_end_concept association

The *link_end_concept* association specifies the *Concept* (9.1.2.1) that plays the associated *link_end_role* (9.1.3.12) for the *Link_End* (9.1.2.7).

link_end_concept has two roles:

— concept (verb form: *is_concept_for*)

— link_end (verb form::*is_end_for*) which references a *Link_End.*

9.1.3.12 link_end_role association

The *link_end_role* association specifies the *Relation_Role* (9.1.2.5) that is the *end_role* for a *Link_End* (9.1.2.7).

link_end_role has two roles:

— *role* (verb form: *is_role_for*) which references exactly one *Relation_Role*;

— *link_end* (verb form: *is_end_for*) which references zero or more *Link_Ends*.

9.2 Classification metamodel region

9.2.1 Overview

The Classification region is illustrated in Figure 9 below. The purpose of this region is to provide a facility to use *Concept_Systems* (9.1.2.2) to model *classification schemes*.

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*. However, since the structures are similar, we use the *Concept_System* structures of the metamodel to model *classification schemes* as well.

Concept_Systems may be used as *classification schemes* to classify *Classifiable_Items* within a registry, but some *classification schemes* will be more applicable to classifying objects in the real world than items in a registry. If the objects to be classified are not in the registry, the classification scheme may still be recorded using the *Concept_System* structures.

A *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.

Annex F illustrates the use of *Concept_System* to implement a classification scheme for representation classes.

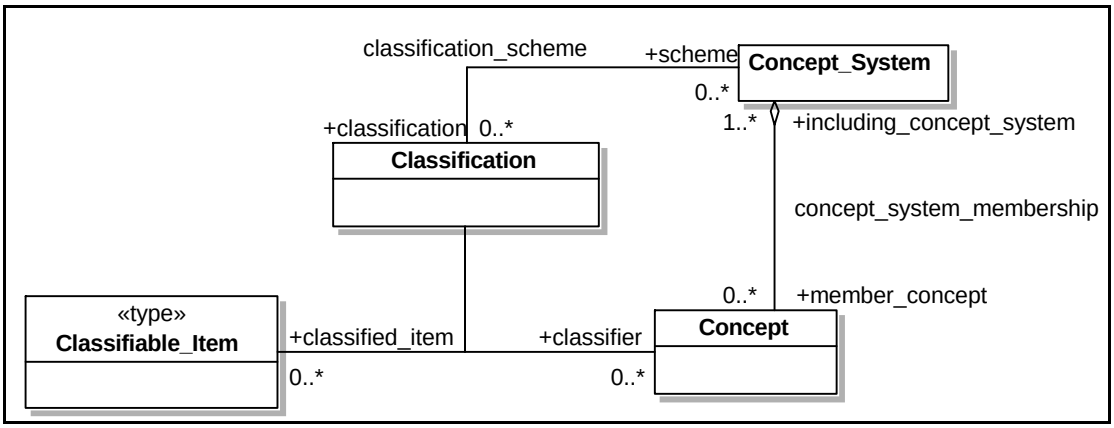

Figure 9 — Classification metamodel region

9.2.2 Classes in the Classification metamodel region

9.2.2.1 Classifiable_Item class

Classifiable_Item is an abstract supertype of all metadata items which might be classified (organized into a hierarchical structure or partial order). (See 5.5.)

A *Classifiable_Item* may be classified in zero or more *classification_schemes*, by associating it with one or more *classifier Concepts* as represented by the *Classification* association class in Figure 9 above. Such classification is optional.

NOTE It is because the associations are optional that the class is called *Classifiable_Item* rather than *Classified_Item*.

9.2.2.2 Concept_System class

Concept_System is specified in 9.1.2.2. *Concept_System* is used to model a *classification scheme* by defining the nodes of the *classification scheme* as *Concepts* (9.1.2.1) in the *Concept_System*. Hierarchical ordering or other relationships among the nodes of the *classification scheme* can be specified using *Relations* (9.1.2.4) among the *Concepts*. The *Relations* may be named and defined by making them *Designatable_Items* (7.3.2.2).

Constraint: *Concept_Systems* used as *classification schemes* shall be constrained to be partial orders, i.e., the *Relation*s among the *Concepts* must not contain any cycles. Partial orders may be represented as directed acyclic graphs. However, the *Concept_System* 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.

9.2.2.3 Concept class

Concept is specified in 9.1.2.1. For purposes of modelling a *classification scheme*, *Concept* is used to represent a node in the *classification scheme.* The *Concept* represents a partition of the *classification scheme* that is homogeneous with respect to some *characteristic*.

9.2.3 Associations Classes in the Classification metamodel region

9.2.3.1 Classification association class

The *Classification* association class is used to record the classification of a *Classifiable_Item* into a group designated by a *Concept* (9.1.2.1) in a *Concept_System* (9.1.2.2).

A *Classification* association has two roles:

— classified_item (verb form: has_classified_item) which references an instance of *Classified_Item;*

— classifier (verb form: classified_as) which references an instance of *Concept*.

The exact semantics of the *Classification* association are not specified by this standard, but will depend upon way in which the *classification scheme* is used. For example, the *Classification* association might signify either an "is-a" or an "instance-of" relationship.

A *Classifiable_Item* may be classified by zero or more *Concepts*. A *Concept* may classify zero or more *Classifiable_Items*.

9.2.4 Associations in the Classification metamodel region

9.2.4.1 classification_scheme association

classification_scheme associates a *Classification* association class with zero or more *Concept_Systems* within which the classification occurs.

The association has two roles:

— classification (verb form: has_classification) which references zero or more *Classifications*

— scheme (verb form: has_scheme) which references zero or more *Concept_Systems*.

A *scheme* may have zero or more *Classifications*. A *Classification* may have zero or more *schemes*.

Constraint: It must be the case that the *Concept_System(s)* associated with the *Classification* through the *classification_scheme* association are the same *Concept_System(s)* associated with the *Concept(s)* that participate in the *Classification*.

9.2.4.2 concept_system_membership association

The *concept_system_membership* association is described in 9.1.3.1.

10 Binary Relations package

10.1 Binary Relations metamodel region

10.1.1 Overview

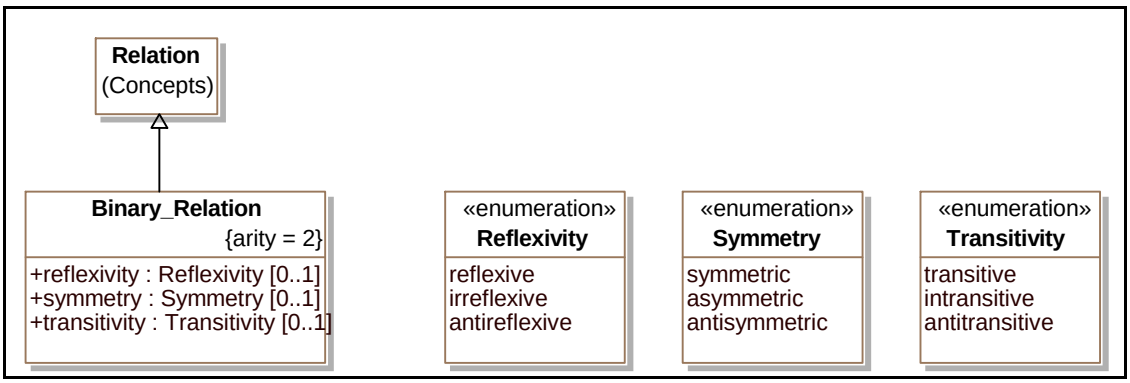

Figure 10 — Binary Relations metamodel region

10.1.2 Classes in the Binary_Relations metamodel region

10.1.2.1 Relation class

The *Relation* class is described in 9.1.2.4.

10.1.2.2 Binary_Relation class

10.1.2.2.1 Direct superclass

Relation (9.1.2.4)

10.1.2.2.2 Description of Binary_Relation

Binary_Relation is a class each instance of which models a **binary relation** (3.2.10), a **relation** (3.2.119) of **arity** (3.2.4) 2 (i.e. having two **link ends** (3.2.70)).

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.

Table 4 – Examples of binary relations and their characterization

Relation	Symmetry	Reflexivity	Transitivity
equals	symmetric	reflexive	transitive
not equals	symmetric	antireflexive	intransitive
less than	antisymmetric	antireflexive	transitive

Relation	Symmetry	Reflexivity	Transitivity
less than or equal	asymmetric	reflexive	transitive
similar	symmetric	reflexive	intransitive

The *Binary_Relation* class has three enumeration attributes: reflexivity, symmetry, and transitivity.

10.1.2.2.3 Attributes of Binary_Relation

10.1.2.2.3.1 reflexivity

Attribute name: *reflexivity*

Definition: characterization of the *Binary_Relation* as: reflexive, irreflexive or antireflexive.

Obligation: Optional

Multiplicity: *0..1*

Datatype: *Reflexivity* (10.1.2.3)

NOTE 1 A *Binary_Relation*, R, is reflexive if for all x, R(x,x) is true. Equality is an example of a reflexive relation.

NOTE 2 A *Binary_Relation*, R, is irreflexive if it is not reflexive. i.e., R(x,x) is not necessarily true for all x.

NOTE 3 A *Binary_Relation*, R, is antireflexive if for all x, R(x,x) is false. Inequality is an example of an antireflexive relation.

NOTE 4 An antireflexive relation is also irreflexive, but antireflexive is a more specific characterization.

10.1.2.2.3.2 symmetry

Attribute name: *symmetry*

Definition: characterization of the *Binary_Relation* as: symmetric, asymmetric or antisymmetric

Obligation: Optional

Multiplicity: *0..1*

Datatype: *Symmetry* (10.1.2.4)

NOTE 1 A *Binary_Relation*, R, is symmetric if for all x, y: R(x,y) implies R(y,x). Examples of symmetric relations are ‘equals’, ‘not equals’, ‘within-2-miles-of’, etc. Symmetry does not imply reflexivity. For example, the inequality relation is symmetric, but antireflexive.

NOTE 2 A *Binary_Relation*, R, is asymmetric if 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.

NOTE 3 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 antisymmetric relation.

NOTE 4 An antisymmetric relation is also asymmetric, but antisymmetric is a more specific characterization. An asymmetric relation is not necessarily antisymmetric (consider less than or equals).

10.1.2.2.3.3 transitivity

Attribute name: *transitivity*

Definition:	characterization of the *Binary_Relation* as: transitive, intransitive or antitransitive
Obligation:	Optional
Multiplicity:	*0..1*
Datatype:	*Transitivity* (10.1.2.5)
NOTE 1	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.
NOTE 2	A Binary_Relation, R, is intransitive if it is not transitive i.e., R(x,y) and R(y,z) does not imply R(x,z).
NOTE 3	A Binary_Relation, R, is antitransitive if for all x,y,z: R(x,y) and R(y,z) implies not R(x,z).
NOTE 4	An antitransitive relation is also intransitive, but antitransitive is a more specific characterization.

—— End of attributes of *Binary_Relation* ——

10.1.2.3 Reflexivity enumeration

Reflexivity is an enumeration of the values: *reflexive, irreflexive, antireflexive*. *Reflexivity* is used as the datatype of the *reflexivity* attribute (10.1.2.2.3.1) of *Binary_Relation* (10.1.2.2).

10.1.2.4 Symmetry enumeration

Symmetry is an enumeration of the values: *symmetric, asymmetric, antisymmetric*. *Symmetry* is used as the datatype of the *symmetry* attribute (10.1.2.2.3.2) of *Binary_Relation* (10.1.2.2).

10.1.2.5 Transitivity enumeration

Transitivity is an enumeration of the values: *transitive, intransitive, antitransitive*. *Transitivity* is used as the datatype of the *transitivity* attribute (10.1.2.2.3.3) of *Binary_Relation* (10.1.2.2).

11 Data Description package

11.1 High-level Data Description metamodel region

11.1.1 Overview

A high level overview of the metamodel can be found in Figure 11. It shows four classes: *Conceptual_Domain* (11.1.2.2), *Value_Domain* (11.1.2.3), *Data_Element* (11.1.2.4), and *Data_Element_Concept* (11.1.2.5). The Figure also shows four associations among the four classes: *value_domain_meaning* (11.1.3.1), *data_element_domain* (11.1.3.2), *data_element_meaning* (11.1.3.3), and *data_element_concept_domain* (11.1.3.4).

The following text describes the classes and associations shown in the Figure. 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, follow in subsequent subclauses.

Figure 11 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. A consolidated detailed metamodel is presented in 11.6.

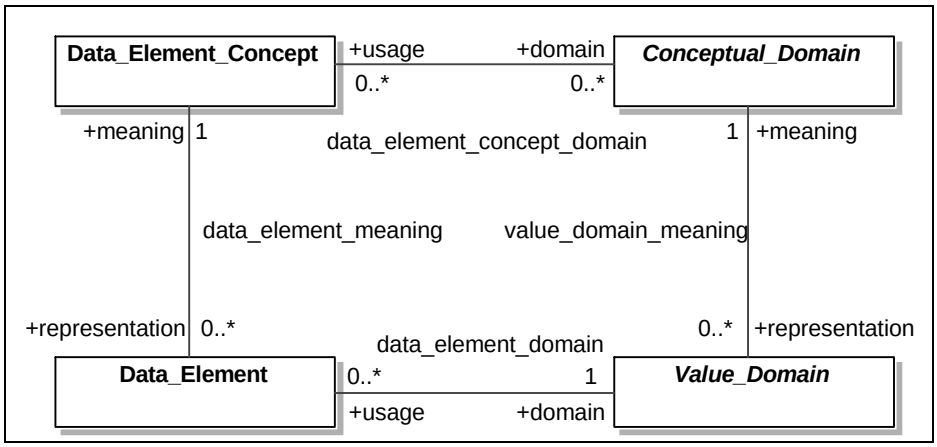

Figure 11 — High-level Data Description metamodel

11.1.2 Classes of High-level Data Description metamodel

11.1.2.1 Overview

The classes shown in Figure 11 are described below starting with *Conceptual_Domain* (11.1.2.2), and proceeding clock-wise around the Figure.

11.1.2.2 Conceptual_Domain class

11.1.2.2.1 Direct superclass

Concept (9.1.2.1)

11.1.2.2.2 Description of Conceptual_Domain

Conceptual_Domain is a class each instance of which models a **conceptual domain** (3.2.21), a set of **value meaning**s (3.2.141) 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 domain**s (3.2.140): three letter country codes, and full country names. The conceptual domain might be used in several **data element concept**s (3.2.29), e.g., "person's country of residence", "person's country of birth", "person's country of citizenship".

A conceptual domain can facilitate the mapping of equivalent values of two or more value domains that share the conceptual domain.

Conceptual_Domain may participate in the following associations:

— *data_element_concept_domain* (11.1.3.4) to zero or more *Data_Element_Concept*s (11.1.2.5) which reference the domain for its associated value meanings;

— *value_domain_meaning* (11.1.3.1) to zero or more *Value_Domain*s (11.1.2.3) which provide representation for the conceptual domain.

Conceptual_Domain is further described in 11.3.2.1.

11.1.2.3 Value_Domain class

Value_Domain is a class each instance of which models a **value domain** (3.2.140), a collection of **permissible value**s (3.2.96). A value domain provides representation, but has no implication as to what **data element concept** (3.2.29) the values are associated with, nor what the values mean. Permissible values are **designation**s (3.2.51), bindings of **sign**s (3.2.123) (values) to their corresponding **value meaning**s (3.2.141).

Value_Domain is associated with a *Conceptual_Domain* (11.1.2.2) through the association *value_domain_meaning* (11.1.3.1). Through this association, a value domain provides a representation for the **conceptual domain** (3.2.21) and the conceptual domain provides meaning for the value 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 are:

— Sex, which contains two designations (permissible values), M -> Male and F-> Female;

— Parent, which contains two designations (permissible values), M -> Mother and F -> Father.

NOTE These two value domains are defined over the same set of values (signs), but they are mapped to separate conceptual domains.

Value_Domain shall participate in the following association:

— *value_domain_meaning* (11.1.3.1) to exactly one *Conceptual_Domain* (11.1.2.2) which provides meaning to the value domain.

Value_Domain may participate in the following association:

— *data_element_domain* (11.1.3.2) to zero or more *Data_Element*s (11.1.2.4) for which the value domain provides a set of permissible values.

Value_Domain is further described in 11.3.2.5.

Value domains may be reused for multiple data elements - see the discussion of countries of the world in 11.1.2.2 above.

11.1.2.4 Data_Element class

Data_Element is a class each instance of which models a **data element** (3.2.28), a unit of **data** (3.2.27) that is considered in context to be indivisible. A data element is a basic unit of data of interest to an organization, for which the definition, identification, representation, and permissible values are specified by means of a set of attributes. Examples of data element include: a column in a table of a relational database, a field in a record or form, an XML element, the attribute of a Java class, or a variable in a program. The description of data elements is a major purpose of ISO/IEC 11179 Metadata Registries.

Data_Element shall participate in the following associations:

— *data_element_meaning* (11.1.3.3) to exactly one *Data_Element_Concept* (11.1.2.5) which the data element represents, and which gives meaning to the data element;

— *data_element_domain* (11.1.3.2) to exactly one *Value_Domain* (11.1.2.3) which specifies the permissible values that the data element may assume.

Data_Element is further described in 11.5.2.1.

11.1.2.5 Data_Element_Concept class

11.1.2.5.1 Direct superclass

Concept (9.1.2.1)

11.1.2.5.2 Description of Data_Element_Concept

Data_Element_Concept is a class each instance of which models a **data element concept** (3.2.29), a **concept** (3.2.18) that is an **association** (3.1.2) of a **property** (3.2.100) with an **object class** (3.2.88). A data element concept is a concept that can be represented in the form of a **data element** (3.2.28), described independently of any particular representation.

A data element concept is a usage of a **conceptual domain** (3.2.21), 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 domain**s (3.2.140).

A data element concept can facilitate the mapping of equivalent values of two or more data elements that share the data element concept.

Data_Element_Concept may participate in the following associations:

— *data_element_concept_domain* (11.1.3.4) to zero or more *Conceptual_Domain*s (11.1.2.2);

— *data_element_meaning* (11.1.3.3) to zero or more *Data_Element*s (11.1.2.4).

Data_Element_Concept is further described in 11.2.2.3.

11.1.3 Associations of the High Level Data Description metamodel

11.1.3.1 value_domain_meaning association

The association *value_domain_meaning* binds a *Value_Domain* (11.1.2.3) to a *Conceptual_Domain* (11.1.2.2). The association has two roles:

— *meaning* (verb form: gives meaning to) which references a *Conceptual_Domain;*

— *representation* (verb form: provides representation for) which references a *Value_Domain*.

The *meaning* role specifies the **conceptual domain** (3.2.21) of a **value domain** (3.2.140). The *representation* role specifies the value domain(s) of a conceptual domain. Each *meaning* (*Conceptual_Domain*) may have zero or more associated *representations* (*Value_Domain*s). Each *representation* (*Value_Domain*) has exactly one associated *meaning* (*Conceptual_Domain*).

A value domain is a collection of **permissible values** (3.2.96) which are *designations*, the mappings between **value meaning**s (3.2.141) and values (**sign***s* 3.2.123). 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 (**designation**s 3.2.51) are recorded as *Permissible_Values* (11.3.2.7) in this metamodel.

NOTE 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 (e.g. see the additional examples in 11.1.2.3), 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 the *Conceptual_Domain*) for the values (in the *Value_Domain*) associated with a *Data_Element*. (See discussion under Constraints in 11.1.4.1)

11.1.3.2 data_element_domain association

The *data_element_domain* association binds a *Data_Element* (11.1.2.4) to a *Value_Domain* (11.1.2.3) which specifies the **permissible value**s (3.2.96) which may be stored in the **data element** (3.2.28). The association has two roles:

— *usage* (verb form: uses), which specifies a *Data_Element* which *uses* a *Value_Domain*;

— *domain* (verb form: provides_values_for), which specifies a *Value_Domain* which *provides values for* the *Data_Element*.

A *usage* (*Data_Element*) has exactly one *domain* (*Value_Domain*). A *domain* (*Value_Domain*) may have zero or more *usages* (*Data_Elements*)

11.1.3.3 data_element_meaning association

The *data_element_meaning* association binds a *Data_Element* (11.1.2.4) to its *Data_Element_Concept* (11.1.2.5). The association has two roles:

— *meaning* (verb form: provides meaning for), which specifies the *Data_Element_Concept* which *provides meaning for* a *Data_Element*;

— *representation* (verb form: represents), which specifies a *Data_Element* which *represents* 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*).

11.1.3.4 data_element_concept_domain association

The *data_element_concept_domain* association binds a *Data_Element_Concept* (11.1.2.5) to its *Conceptual_Domain* (11.1.2.2). The association has two roles:

— *usage* (verb form: uses), which specifies the *Data_Element_Concept* which *uses* a *Conceptual_Domain*;

— *domain* (verb form: provides_domain_for) which specifies the *Conceptual_Domain* which *provides the domain for* a Data_Element_Concept.

Each *usage* (*Data_Element_Concept*) may have zero or more *domains* (*Conceptual_Domains*). Each *domain* (*Conceptual_Domain*) may have zero or more associated *usages* (*Data_Element_Concepts*). For example, a *Conceptual_Domain* may be registered without registering an associated *Data Element Concept*, and vice versa.

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).

NOTE ISO/IEC 11179-3 Edition 2 required a *Data_Element_Concept* to be associated with exactly one *Conceptual_Domain*. This association has been relaxed in Edition 3 because of feedback from organizations using Edition 2. The lower bound has been relaxed from one to zero to allow a *Data_Element_Concept* to be recorded without an associated *Conceptual_Domain*. The upper bound has been increased from one to many to avoid a proliferation of similar *Data_Element_Concept*s. For example, the *Data_Element_Concept*: Person’s marital status, might in some context be associated with a *Conceptual_Domain* of { single , married }, and in another context with a *Conceptual_Domain* of { single , married , divorced , widowed }. Edition 2 required that separate *Data_Element_Concept*s be specified. Edition 3 allows an organization to choose whether to specify separate *Data_Element_Concept*s, or to share a single *Data_Element_Concept*. Separate *Data_Element*s (11.1.2.4) should still be defined and associated with the appropriate *Value_Domain*s (11.1.2.3).

11.1.4 Constraints of the High Level Metamodel

11.1.4.1 Equality of mappings from data element to conceptual domain

There are two paths in the metamodel in Figure 11 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; or 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 the *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.

NOTE The possible inverse constraint (starting from Conceptual_Domain) is not true, because the associations are not functions (uniquely valued) in the inverse directions.

11.2 Data Element Concept metamodel region

11.2.1 Overview

The **Data Element Concept** region is illustrated in Figure 12. The purpose of this region is to maintain the information on the **concept**s (3.2.18) related to **data element**s (3.2.28). The **metadata object**s (3.2.76) in this region concern semantics. *Concepts* (9.1.2.1) are independent of any internal or external physical representation. The metadata objects in this region are: *Conceptual_Domains* (11.2.2.4), *Data_Element_Concepts* (11.2.2.3), *Object_Classes* (11.2.2.1) and *Properties* (11.2.2.2). *Object_Classes* and *Properties* may be combined to form *Data_Element_Concepts*. All of these metadata objects are subclasses of *Concept* (see 11.7).

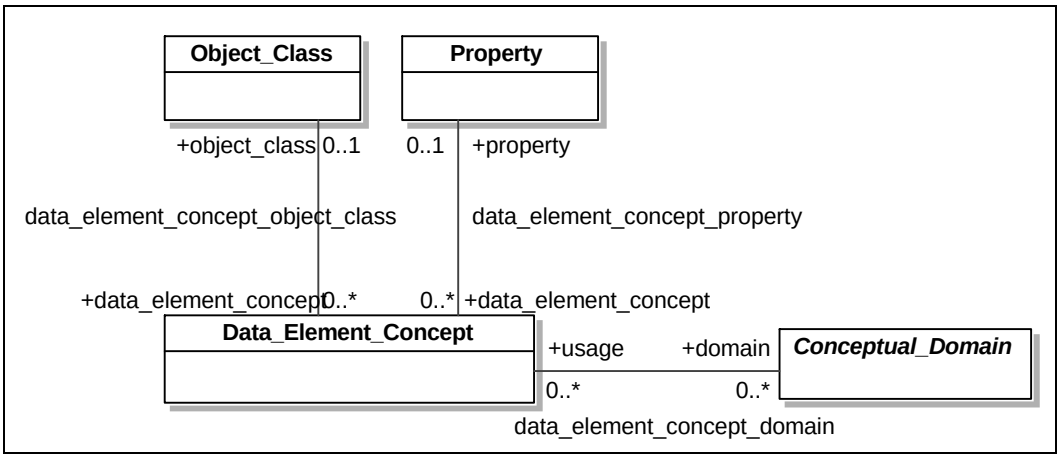

Figure 12 — Data_Element_Concept metamodel region

11.2.2 Classes in the Data_Element_Concept region

11.2.2.1 Object_Class class

11.2.2.1.1 Direct superclass

Concept (9.1.2.1)

11.2.2.1.2 Description of Object_Class

Object_Class is a class each instance of which models an **object class** (3.2.88). An object class is a **concept** (3.2.18) that represents a set of ideas, abstractions, or things in the real world that can be identified with explicit boundaries and meaning and whose properties and behavior follow the same rules. It may be either a single or a group of associated concepts, abstractions, or things.

An *Object_Class* may have a *data_element_concept_object_class* association (11.2.3.3) with zero or more *Data_Element_Concepts* (11.2.2.3), where the object class describes the ideas, abstractions or things in the real world that are represented by the **data element concept**s (3.2.29).

EXAMPLE The *Object_Class* “Person” could be associated with the *Data_Element_Concept* “Person height”.

11.2.2.2 Property class

11.2.2.2.1 Direct superclass

Concept (9.1.2.1)

11.2.2.2.2 Description of Property

Property is a class each instance of which models a **property** (3.2.100), a quality common to all members of an **object class** (3.2.88). A property may be any feature that humans naturally use to distinguish one individual object from another. It is the human perception of a single quality of an object class in the real world. It is conceptual and thus has no particular associated means of representation by which the property can be communicated.

A *Property* may have a *data_element_concept_property* association (11.2.3.1) with zero or more *Data_Element_Concept*s (11.2.2.3).

11.2.2.3 Data_Element_Concept class

11.2.2.3.1 Direct superclass

Concept (9.1.2.1)

11.2.2.3.2 Description of Data_Element_Concept

Data_Element_Concept is a class each instance of which models a **data element concept** (3.2.29). A data element concept is a specification of a **concept** (3.2.18) independent of any particular representation. A data element concept can be represented in the form of a **data element** (3.2.28).

A *Data_Element_Concept* may have a *data_element_concept_object_class* association (11.2.3.3) with zero or one *Object_Class* (11.2.2.1) and a *data_element_concept_property* association (11.2.3.1) with zero or one *Property* (11.2.2.2).

The combination of a **property** (3.2.100) and an **object class** (3.2.88) provides significance beyond either that of the property or the object class. A *Data_Element_Concept* thus has a *Definition* (7.3.2.4) independent of the *Definition* of the *Object_Class* or the *Property.*

Every *Data_Element_Concept* must have exactly one *data_element_concept_domain* association (11.2.3.2) with a *Conceptual_Domain* (11.3.2.1), where the *Data_Element_Concept* supplies a *usage* for the associated *Conceptual_Domain*.

EXAMPLE An association between the *Data_Element_Concept* “Person Country of Residence” and the *Conceptual_Domain* “Country”.

11.2.2.4 Conceptual_Domain class

Conceptual_Domain is described in 11.3.2.1 as part of the Conceptual and Value Domain region (11.3).

11.2.3 Associations in the Data_Element_Concept region

11.2.3.1 data_element_concept_property association

The association *data_element_concept_property* binds a *Data_Element_Concept* (11.2.2.3) and a *Property* (11.2.2.2). The association has two roles:

— *property* (verb form: *has property*) which references a *Property* that is part of the specification of the *Data_Element_Concept;*

— *data element concept* (verb form: *has data element concept*) which references a *Data_Element_Concept* that the *Property* helps to specify.

A *Data_Element_Concept* may be associated with zero or one *Property*. A *Property* may be associated with zero or more *Data_Element_Concepts*.

11.2.3.2 data_element_concept_domain association

The association *data_element_concept_domain* binds a *Conceptual_Domain* (11.3.2.1) and a *Data_Element_Concept* (11.2.2.3). The association has two roles:

— *domain* (verb form: *provides domain for*) which references a *Conceptual_Domain* that *provides the domain for* the *Data_Element_Concept;*

— *usage* (verb form: *uses*) which references a *Data_Element_Concept* that *uses* the *Conceptual_Domain*.

A *Data_Element_Concept* may be associated with zero or more *Conceptual_Domains*. A *Conceptual_Domain* may be associated with zero or more *Data_Element_Concepts*.

11.2.3.3 data_element_concept_object_class association

The association *data_element_concept_object_class* binds a *Data_Element_Concept* (11.2.2.3) and an *Object_Class* (11.2.2.1) that represents a particular set of ideas, abstractions, or things in the real world whose properties and behaviour follow a set of rules as represented by the *Data_Element_Concept*.

The association has two roles:

— *object_class* (verb form: *has object class*) which references an *Object_Class* that is part of the specification of the *Data_Element_Concept;*

— *data_element_concept* (verb form: *has data element concept*) which references a *Data_Element_Concept* that the *Object_Class* helps to specify.

A *Data_Element_Concept* may be associated with zero or one *Object_Classes*. An *Object_Class* may be associated with zero or more *Data_Element_Concepts***.**

11.3 Conceptual and Value_Domain metamodel region

11.3.1 Overview

This region of the metamodel addresses the administration of *Conceptual_Domains* (11.3.2.1) and *Value_Domains* (11.3.2.5). These domains can be viewed as logical code sets and physical code sets. *Conceptual_Domains* support *Data_Element_Concepts* (11.2.2.3) and *Value_Domains* support *Data_Elements* (11.5.2.1). The region is illustrated in Figure 13 below.

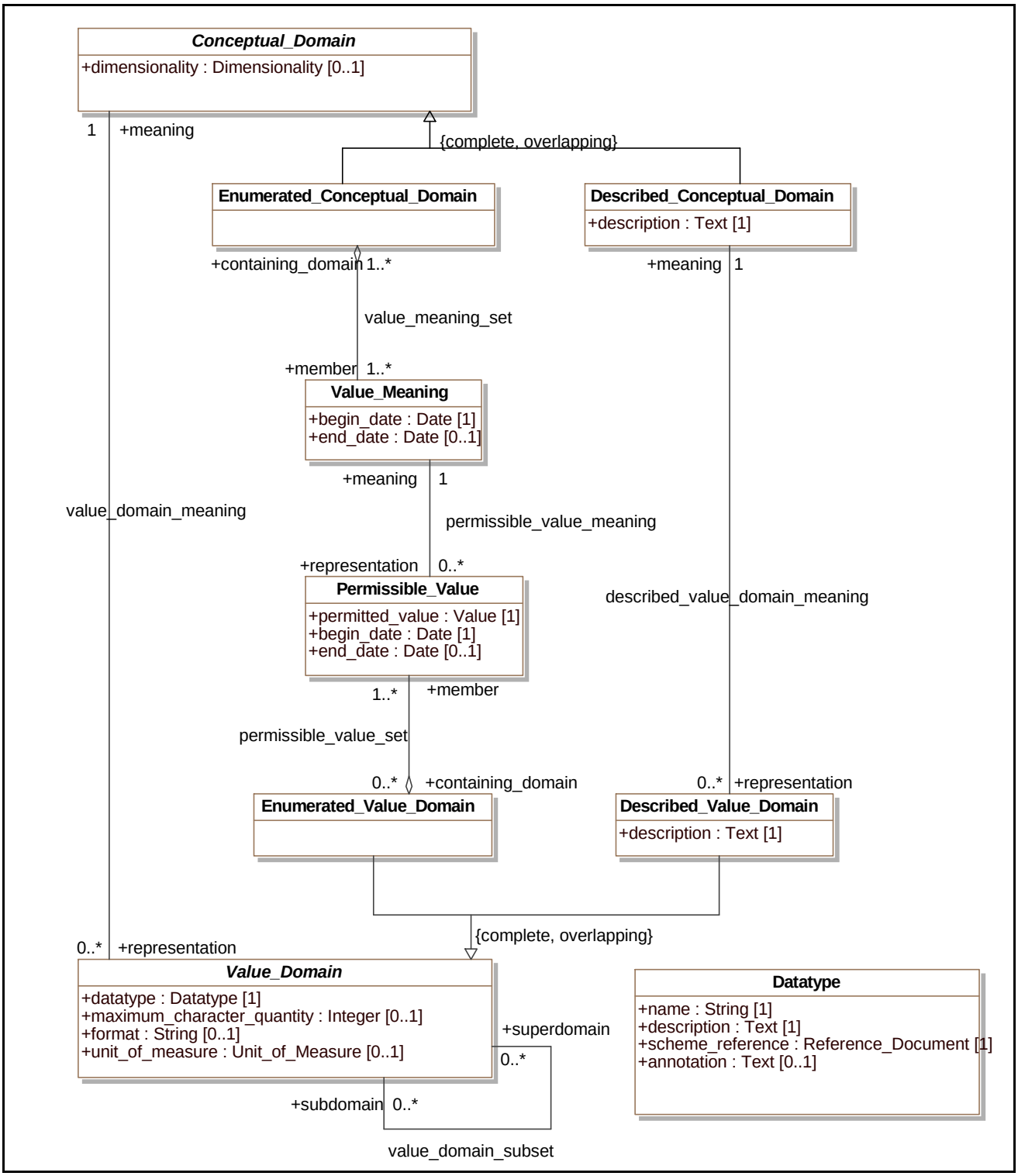

Figure 13 — Conceptual and value domain metamodel region

NOTE In Figure 13, the use of *italics* in the names of *Conceptual_Domain* and *Value_Domain* indicates that they are abstract classes, meaning that only their subclasses may be instantiated.

11.3.2 Classes in the Conceptual and Value_Domain region

11.3.2.1 Conceptual_Domain class

11.3.2.1.1 Direct superclass

Concept (9.1.2.1)

11.3.2.1.2 Description of Conceptual_Domain

Conceptual_Domain is a class, each instance of which models a **conceptual domain** (3.2.21), a set of **value meanings** (3.2.141) which may either be enumerated or expressed via a description. *Conceptual_Domain* is an abstract class, which has two possible subclasses: *Enumerated_Conceptual_Domain* (11.3.2.2) and *Described Conceptual_Domain* (11.3.2.4). Every *Conceptual_Domain* instance must be either an *Enumerated_Conceptual_Domain* or a *Described Conceptual_Domain* or a combination of the two.

Conceptual_Domain may participate in the following associations:

— *data_element_concept_domain* (11.2.3.2) (not shown in Figure 13) to zero or more *Data_Element_Concept*s (11.2.2.3) which reference the domain for its associated value meanings;

— *value_domain_meaning* (11.3.3.1) to zero or more *Value_Domain*s (11.3.2.5) which provide representation for the conceptual domain.

The *Conceptual_Domain* class has one attribute, *dimensionality* (11.3.2.1.3.1), of type *Dimensionality* (11.4.2.3). The *dimensionality* attribute specifies the **dimensionality** (3.2.58) as elaborated in the discussion of the *Dimensionality* class below in 11.4.2.3.

11.3.2.1.3 Attributes of Conceptual_Domain

11.3.2.1.3.1 dimensionality

Attribute name:	*dimensionality*
Definition:	expression of measurement without units
Obligation:	Optional
Multiplicity:	*0..1*
Datatype:	*Dimensionality* (11.4.2.3)
NOTE 1	When a *dimensionality* is specified, then any *Unit_of_Measure* (11.4.2.1) specified for any *Value_Domain* (11.3.2.5) that is based on this *Conceptual_Domain,* shall be consistent with this *dimensionality*.
NOTE 2	ISO 31-0 specifies physical dimensions (e.g. length, mass, velocity). This part of ISO/IEC 11179 also permits non-physical dimensions (e.g. value dimensions such as: currency, quality indicator)

—— End of attributes of *Conceptual_Domain* ——

11.3.2.2 Enumerated_Conceptual_Domain class

11.3.2.2.1 Direct superclass

Conceptual_Domain (11.3.2.1)

11.3.2.2.2 Description of 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 subclass of *Conceptual_Domain,* an *Enumerated_Conceptual_Domain* inherits the attributes and relationships of the former.

11.3.2.3 Value_Meaning class

11.3.2.3.1 Direct superclass

Concept (9.1.2.1)

11.3.2.3.2 Description of Value_Meaning

Value_Meaning is a class each instance of which models a **value meaning** (3.2.141), which provides semantic content of a possible value.

Each member of an *Enumerated_Conceptual_Domain* (11.3.2.2) 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* (11.3.2.5). A particular *Value_Meaning* may have more than one means of representation by *Permissible_Values* (11.3.2.7) — each from a distinct *Enumerated_Value_Domain* (11.3.2.6).

A *Value_Meaning* shall participate in the following association:

— *value_meaning_set* (11.3.3.2) with one or more *containing_domain Enumerated_Conceptual_Domain*s (11.3.2.2) of which the *Value_Meaning* is a *member*.

A *Value_Meaning* may participate in the following association:

— *permissible_value_meaning* (11.3.3.4) with zero or more *Permissible_Value*s (11.3.2.7) which provide *representation* for the *Value_Meaning.*

The *Value_Meaning* class has two attributes:

— *begin_date* (11.3.2.3.3.1) of type *Date* (6.2.3);

— *end_date* (11.3.2.3.3.2) of type *Date* (6.2.3).

The description of the *Value_Meaning* is specified by making the *Value_Meaning* a *Designatable_Item* (7.3.2.2) and using an associated *Definition* (7.3.2.4).

11.3.2.3.3 Attributes of Value_Meaning

11.3.2.3.3.1 begin_date

Attribute name:	*begin_date*
Definition:	date at which this *Value_Meaning* became, or will become, a valid *Value_Meaning*
Obligation:	Mandatory
Multiplicity:	*1*
Datatype:	*Date* (6.2.3)
NOTE	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.

11.3.2.3.3.2 end_date

Attribute name:	*end_date*
Definition:	date on which the *Value_Meaning* ceased, or will cease, to be valid.
Obligation:	Optional
Multiplicity:	*0..1*
Datatype:	*Date* (6.2.3)
NOTE 1	The absence of the *value_meaning_end_date* indicates that the *Value_Meaning* is still valid.
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.

—— End of attributes of *Value_Meaning* ——

11.3.2.4 Described_Conceptual_Domain class

11.3.2.4.1 Direct superclass

Conceptual_Domain (11.3.2.1)

11.3.2.4.2 Description of Described_Conceptual_Domain

Described_Conceptual_Domain is a class each instance of which models a **described conceptual domain** (3.2.48), a **conceptual domain** (3.2.21) that is specified by a description or specification, such as a rule, a procedure, or a range (i.e. interval), because it cannot be expressed as a finite set of **value meanings** (3.2.141).

As a subclass of *Conceptual_Domain,* a *Described Conceptual_Domain* inherits the attributes and relationships of the former.

A *Described_Conceptual_Domain* may participate in the following association:

— *described_value_domain_meaning* (11.3.3.3) with zero or more *Described_Value_Domain*s (11.3.2.8) which provide *representation* for the *Described_Conceptual_Domain*.

The *Described_Conceptual_Domain* class has one attribute:

— *description* (11.3.2.4.3.1) of type *Text* (6.2.12).

Each *Described_Conceptual_Domain* instance must have exactly one *description* attribute.

11.3.2.4.3 Attributes of Described_Conceptual_Domain

11.3.2.4.3.1 description

Attribute name:	*description*
Definition:	description or specification of a rule, reference, or range for a set of all *Value_Meanings* for a *Conceptual_Domain.*
Obligation:	Mandatory
Multiplicity:	*1*
Datatype:	*Text* (6.2.12)

—— End of attributes of *Conceptual Domain* ——

11.3.2.5 Value_Domain class

11.3.2.5.1 Description of Value_Domain

Value_Domain is a class each instance of which models a **value domain** (3.2.140), a set of **permissible value**s (3.2.96). A value domain provides representation, but has no implication as to the **data element concept** (3.2.29) 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* (11.3.2.1). A *Value_Domain* has two possible subclasses: an *Enumerated_Value_Domain* (11.3.2.6) and a *Described_Value_Domain* (11.3.2.8). A *Value_Domain* must be either one or both an *Enumerated Valued* or a *Described_Value_Domain*.

A *Value_Domain* shall participate in the following association:

— *value_domain_meaning* (11.3.3.1) with exactly one *Conceptual_Domain* (11.3.2.1) which provides *meaning* for the *Value_Domain*, while the *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'.

The *Value_Domain* class has the following attributes, which are summarized here and specified more formally in 11.3.2.5.2:

— *datatype* (11.3.2.5.2.1) of type *Datatype* (11.3.2.9);

— *format* (11.3.2.5.2.2) of type *String* (6.2.11);

— *maximum_character_quantity* (11.3.2.5.2.3) of type *Integer* (6.2.5);

— *unit_of_measure* (11.3.2.5.2.4) of type *Unit_of_Measure* (11.4.2.1).

11.3.2.5.2 Attributes of Value_Domain

11.3.2.5.2.1 datatype

Attribute name:	*datatype*
Definition:	*Datatype* used in a *Value_Domain*
Obligation:	Mandatory
Multiplicity:	*1*
Datatype:	*Datatype* (11.3.2.9)
NOTE	applies to all values in the *Value_Domain.*

11.3.2.5.2.2 format

Attribute name:	*format*
Definition:	template for the structure of the presentation of the **value**(s) EXAMPLE – YYYY-MM-DD for a date.
Obligation:	Optional
Multiplicity:	*0..1*
Datatype:	*String* (6.2.11)

11.3.2.5.2.3 maximum_character_quantity

Attribute name:	*maximum_character_quantity*
Definition:	maximum number of characters available to represent the *Data_Element* value
Obligation:	Optional
Multiplicity:	*0..1*
Datatype:	*Integer* (6.2.5)
NOTE	Applicable only to character datatypes.

11.3.2.5.2.4 unit_of_measure

Attribute name:	*unit_of_measure*
Definition:	**unit of measure** (3.2.138) used in a *Value_Domain*
Obligation:	Optional
Multiplicity:	*0..1*
Datatype:	*Unit_of_Measure* (11.4.2.1)
NOTE 1	Applies to all values in the *Value_Domain*.
NOTE 2	Constraints on the unit of measure dimensionality are specified in 11.3.4.4.

—— End of attributes of *Value_Domain* ——

11.3.2.6 Enumerated_Value_Domain class

11.3.2.6.1 Direct superclass

Value_Domain (11.3.2.5)

11.3.2.6.2 Description of Enumerated_Value_Domain

Enumerated_Value_Domain is a class each instance of which models an **enumerated value domain** (3.2.61), a **value domain** (3.2.140) that is specified by a list of all its **permissible value**s (3.2.96). The *Enumerated_Value_Domain* class is a concrete subclass of the abstract class *Value_Domain*.

Each *Enumerated_Value_Domain* class shall participate in the association:

— *permissible_value_set* (11.3.3.5) with one or more *Permissible_Value*s (11.3.2.7) which specify the values within the domain.

11.3.2.7 Permissible_Value class

11.3.2.7.1 Description of Permissible_Value

Permissible_Value is a class each instance of which models a **permissible value** (3.2.96), the **designation** (3.2.51) of a **value meaning** (3.2.141). A *Permissible_Value* is an expression of a *Value_Meaning* within zero or more *Enumerated_Value_Domain*s.

Each *Permissible_Value* shall participate in the following association:

— *permissible_value_meaning* (11.3.3.4) with exactly one *Value_Meaning* (11.3.2.3).

Each *Permissible_Value* may participate in the following association:

— *permissible_value_set* (11.3.3.5) with zero or more *Enumerated_Value_Domains* (11.3.2.6). It is one of a set of such values that comprises an *Enumerated_Value_Domain*.

The *Permissible_Value* class has three attributes, which are summarized here and specified more formally in 11.3.2.7.2:

— exactly one *permitted_value* (11.3.2.7.2.1) of type *Value* (6.2.13);

— exactly one *begin_date* (11.3.2.7.2.2) of type *Date* (6.2.3);

— zero or one *end_date* (11.3.2.7.2.3) of type *Date* (6.2.3).

11.3.2.7.2 Attributes of Permissible_Value

11.3.2.7.2.1 permitted_value

Attribute name:	*permitted_value*
Definition:	the actual value of the *Permissible_Value*
Obligation:	Mandatory
Multiplicity:	*1*
Datatype:	*Value* (6.2.13)

11.3.2.7.2.2 begin_date

Attribute name:	*begin_date*
Definition:	date at which the *Permissible_Value* became valid
Obligation:	Mandatory
Multiplicity:	*1*
Datatype:	*Date* (6.2.3)
NOTE	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*).

11.3.2.7.2.3 end_date

Attribute name:	*end_date*
Definition:	date at which the *Permissible_Value* ceased to be valid
Obligation:	Optional
Multiplicity:	*0..1*
Datatype:	*Date* (6.2.3)
NOTE 1	By imputation, this is also considered to be date at which the *Permissible_Value* ceased to be bound to its associated meaning (*Value_Meaning*) via the *permissible_value_meaning* association.
NOTE 2	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.

—— End of attributes of *Permissible_Value* ——

11.3.2.7.3 Example of Permissible_Values

The following example from ISO 3166-3 illustrates how permissible values may be valid for specified period of time only.

ISO 3166-3 NEWSLETTER No. I-1 Date: 2002-11-15 announced a change of country name and associated codes from:

Former country name: East Timor

Former alpha-2 code: TP

Former alpha-3 code: TMP

Former numeric code: 626

Validity start date: 1974-01-01

Validity end date: 2002-11-14

To:

New country name: Timor Leste

New alpha-2 code: TL

New alpha-3 code: TLS

New numeric code: 626

Validity start date: 2002-11-15

Validity end date: (null)

11.3.2.8 Described_Value_Domain class

11.3.2.8.1 Direct superclass

Value_Domain (11.3.2.5)

11.3.2.8.2 Description of Described_Value_Domain

Described_Value_Domain is a class each instance of which models a **described value domain** (3.2.49), a **value domain** (3.2.140) that is specified by 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** (3.2.96). It is a concrete subclass of the abstract class *Value_Domain.* As a subclass of *Value_Domain,* a *Described_Value_Domain* inherits the attributes and relationships of the former.

The *Described_Value_Domain* class has one attribute:

— *description* (11.3.2.8.3.1) of type *Text* (6.2.12).

Each *Described_Value_Domain* instance must have exactly one *description* attribute.

11.3.2.8.3 Attributes of Described_Value_Domain

11.3.2.8.3.1 description

Attribute name:	*description*
Definition:	description or specification of a rule, reference, or range for a set of all *Permissible_Values* for a *Value_Domain*
Obligation:	Mandatory
Multiplicity:	*1*
Datatype:	*Text* (6.2.12)

—— End of attributes of *Described_Value_Domain* ——

11.3.2.9 Datatype class

11.3.2.9.1 Description of Datatype

Datatype is a class, each instance of which models a **datatype** (3.1.9), 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* (11.5.2.1) determined by the operations that may be performed on the *Data_Element*.

This part of 11179 is intended to accommodate datatypes from datatype schemes specified in external standards. It is also intended to accommodate other non-standard datatype schemes. Possible standardized datatype schemes include:

— ISO/IEC 9075 (SQL datatype);

— ISO/IEC 11404 (General Purpose Datatypes);

— ISO 21090 Health Informatics datatypes;

— C programming language datatypes;

— XML Schema datatypes;

etc.

The *Datatype* class has the following attributes, which are summarized here and specified more formally in 11.3.2.9.2:

— *name* (11.3.2.9.2.1) of type *String* (6.2.11);

— *description* (11.3.2.9.2.2) of type *Text* (6.2.12);

— *scheme_reference* (11.3.2.9.2.3) of type *Reference_Document* (6.3.7);

— *annotation* (11.3.2.9.2.4) of type *Text* (6.2.12).

11.3.2.9.2 Attributes of Datatype

11.3.2.9.2.1 name

Attribute name:	*name*
Definition:	designation for the *Datatype*
Obligation:	Mandatory
Multiplicity:	*1*
Datatype:	String (6.2.11)
NOTE 1	The *name* is usually drawn from some external source, which in turn is designated by means of the mandatory *scheme_reference*.
NOTE 2	*name* is included as an attribute of *Datatype* because the *Designation* associated with a *Designatable_Item* is optional, while *Datatype.name* is mandatory.

11.3.2.9.2.2 description

Attribute name:	*description*
Definition:	descriptive information to further clarify the *Datatype*
Obligation:	Mandatory

Multiplicity:	*1*
Datatype:	Text (6.2.12)
NOTE	*description* is included as an attribute of *Datatype* because the *Definition* associated with a *Designatable_Item* is optional, while *Datatype.description* is mandatory.

11.3.2.9.2.3 scheme_reference

Attribute name:	*scheme_reference*
Definition:	reference identifying the source of the *Datatype* specification
Obligation:	Mandatory
Multiplicity:	*1*
Datatype:	Reference_Document (6.3.7)
NOTE	In this edition of this part of ISO/IEC 11179, the manner of reference is specified by the *registration authority*.

11.3.2.9.2.4 annotation

Attribute name:	*annotation*
Definition:	specifying information to further define the *Datatype*
Obligation:	Optional
Multiplicity:	*0..1*
Datatype:	Text (6.2.12)

—— End of attributes of *Datatype* ——

11.3.2.9.3 Examples of Datatypes

Any applicable datatype may be specified, e.g.

EXAMPLE 1

name:	integer
description:	mathematical datatype comprising the exact integral values.
scheme_reference:	ISO/IEC 11404:2007

EXAMPLE 2

name:	BL
description:	BL stands for the values of two-valued logic. A BL value can be either true or false, or may have a nullFlavor.
scheme_reference:	ISO 21090:2010

11.3.3 Associations in the Conceptual and Value_Domain region

11.3.3.1 value_domain_meaning association

The association *value_domain_meaning* binds a *Value_Domain* (11.3.2.5) to a *Conceptual_Domain* (11.3.2.1) that provides the meanings for each of the values within the *Value_Domain*. The association has two roles:

— *meaning* (verb form: has meaning) which references a *Conceptual_Domain;*

— *representation* (verb form: has representation) which references a *Value_Domain.*

Each *representation* (*Value_Domain*) must have exactly one *meaning* (*Conceptual_Domain*). However, each *meaning* (*Conceptual_Domain*) may have zero or more *representation*s (*Value_Domains*).

Constraints on this association are specified in 11.3.4.2.

NOTE This version of the metamodel lacks any mechanism to specify the valid dates for the value_domain_meaning association.

11.3.3.2 value_meaning_set association

The association *value_meaning_set* binds a set of *Value_Meaning*s (11.3.2.3) to an *Enumerated_Conceptual_Domain* (11.3.2.2). The association has two roles:

— *containing_domain* (verb form: contained_in) which references an *Enumerated_Conceptual_Domain;*

— *member* (verb form: has_member) which references a *Value_Meaning*.

Each *member* (*Value_Meaning*) shall have one or more *containing_domain*s (*Enumerated_Conceptual_Domain*). Each *containing_domain* (*Enumerated_Conceptual_Domain*) shall have one or more *member*s (*Value_Meaning*s). 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 of the contained *Value_Meanings*, provided the *Value_Meaning* is shared with another *Enumerated_Conceptual_Domain*.

NOTE This version of the metamodel lacks any mechanism to specify the valid dates for the *value_meaning_set* association.

11.3.3.3 described_value_domain_meaning association

The association *described_value_domain_meaning* binds a *Described_Value_Domain* (11.3.2.8) to a *Described_Conceptual_Domain* (11.3.2.4). The association has two roles:

— *meaning* (verb form: has meaning) which references a *Described_Conceptual_Domain;*

— *representation* (verb form: has representation) which references a *Described_Value_Domain*.

Each *representation* (*Described_Value_Domain*) must have exactly one *meaning* (*Described_Conceptual_Domain*). However, each *meaning* (*Described Conceptual_Domain*) may have zero or more *representation*s (*Described_Value_Domains*).

NOTE This version of the metamodel lacks any mechanism to specify the valid dates for the described_value_meaning association.

11.3.3.4 permissible_value_meaning association

The association *permissible_value_meaning* binds one or more *Permissible_Values* (11.3.2.7) with a *Value_Meaning* (11.3.2.3). The association has two roles:

— *meaning* (verb form: has meaning) which references a *Value_Meaning;*

— *representation* (verb form: has representation) which references a *Permissible_Value*.

Each *representation* (*Permissible_Value*) must have exactly one *meaning* (*Value_Meaning*). However, each *meaning* (*Value_Meaning*) may have zero or more *representation*s (*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*.

11.3.3.5 permissible_value_set association

The association *permissible_value_set* binds a set of *Permissible_Value*s (11.3.2.7) to an *Enumerated_Value_Domain* (11.3.2.6). The association has two roles:

— *member* (verb form: has member) which references a *Permissible_Value*;

— *containing_domain* (verb form: contains_domain) which references an *Enumerated_Value_Domain*.

Each *member* (*Permissible_Value*) may have zero or more *containing_domain*s (*Enumerated_Value_Domains*). However, each *containing_domain* (*Enumerated_Value_Domain*) shall have one or more *member*s (*Permissible_Value*s). 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*). Thus it is possible to associate *permissible values* with *value meanings* without defining a complete *value domain*.

NOTE This version of the metamodel lacks any mechanism to specify the valid dates for the permissible_value_set association.

11.3.3.6 value_domain_subset association

The association *value_domain_subset* records a superset-subset relationship between two *Value_Domain*s (11.3.2.5). The association has two roles:

— *superdomain* (verb form: is superdomain) which references the *Value_Domain* that is the superset;

— *subdomain* (verb form: is subdomain) which references the *Value_Domain* that is the subset.

Constraints on this association are specified in 11.3.4.3.

11.3.4 Additional Constraints of the Conceptual and Value_Domain region

11.3.4.1 Overview

This sub-clause specifies additional constraints that are not included in the UML diagram.

11.3.4.2 value_domain_meaning association constraints

Constraint #1: Consistency of Enumeration, Description or combination for Conceptual and Value Domains

Suppose that r is an instance of the class *Value_Domain* (11.3.2.5) and s is an instance of the class *Conceptual_Domain* (11.3.2.1), such that s is the meaning of r according to the *value_domain_meaning* association (11.3.3.1). 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* (11.3.2.6) and s is an instance of *Enumerated_Conceptual_Domain* (11.3.2.2) or it is the case that r is an instance of *Described_Value_Domain* (11.3.2.8) and s is an instance of *Described Conceptual_Domain* (11.3.2.4). Since neither *Value_Domains*, nor *Conceptual_Domains* are disjoint with respect to the *Enumerated* and *Described* subclasses it may be that r and s are both Enumerated and Described 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* (11.3.2.8), such that the instance y is the meaning of x according to the *value_domain_meaning* association (11.3.3.1) (since every

instance of a *Described_Value_Domain* is also a *Value_Domain* (11.3.2.5)) where y is some instance of a *Conceptual_Domain* (11.3.2.1) (either a *Described_Conceptual_Domain* (11.3.2.4) or an *Enumerated_Conceptual_Domain* (11.3.2.2)). There must exist such an instance y according to the cardinality constraints on the *value_domain_meaning* association.

According to the cardinality constraints for the *described_value_domain_meaning* association (11.3.3.3) 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* (11.3.2.6), such that the instance *v* is the meaning of *u* according to the *value_domain_meaning* (11.3.3.1) association (since every instance of a *Enumerated_Value_Domain* is also a *Value_Domain* (11.3.2.5)) where v is some instance of a *Conceptual_Domain* (11.3.2.1) (either a *Described_Conceptual_Domain* (11.3.2.4) or an *Enumerated_Conceptual_Domain* (11.3.2.2)). There must exist such an instance *v* according to the cardinality constraints on the *value_domain_meaning* association.

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* (11.3.2.7) class according to the *permissible_value_set* (11.3.3.5) association. For each element w_i of W there is an exactly one instance m_i of the class *Value_Meaning* (11.3.2.3) such that m_i is the meaning of w_i according to the *permissible_value_meaning* (11.3.3.4) 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_Meaning*s 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*.

11.3.4.3 value_domain_subset association constraints

A *Value_Domain* (11.3.2.5) that particates in a *value_domain_subset* (11.3.3.6) association may not reference itself, either directly or indirectly through a chain of such associations.

11.3.4.4 Consistent dimensionalities

Conceptual_Domains (11.3.2.1) may have an attribute *dimensionality* (11.3.2.1.3.1). *Value_Domains* (11.3.2.5) may have an attribute *unit_of_measure* (11.3.2.5.2.4) of type *Unit_of_Measure* (11.4.2.1). 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* (11.3.4.2) association (or some equivalent path as above). Suppose that d (of type *Dimensionality* (11.4.2.3)) is the *dimensionaility* attribute of the instance c. Suppose that e is the *dimensionality* of the *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*.

11.4 Measurement metamodel region

11.4.1 Overview

The measurement region illustrated in Figure 14.

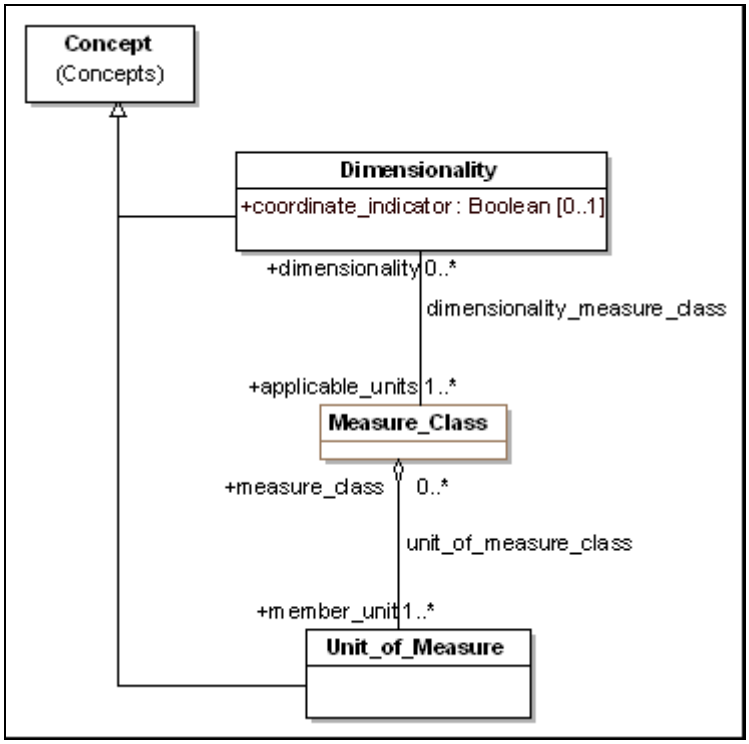

Figure 14 — Measurement metamodel region

11.4.2 Classes in the Measurement region

11.4.2.1 Unit_of_Measure class

11.4.2.1.1 Direct superclass

Concept (9.1.2.1)

11.4.2.1.2 Description of Unit_of_Measure

Unit_of_Measure is a class each instance of which models a **unit of measure** (3.2.138), the units in which associated values are measured. If appropriate, a *Value_Domain* (11.3.2.5) (not shown in Figure 14) may be associated with a *Unit_of_Measure* to specify the units in which any associated *Data_Element* (11.5.2.1) values are measured. *Unit_of_Measure* may be named and defined by making it a *Designatable_Item* (7.3.2.2).

Unit_of_Measure may participate in the following association:

— *unit_of_measure_class* (11.4.3.2) with zero or more *Measure_Class*es (11.4.2.2) which serves to group equivalent *Unit_of_Measure.*

11.4.2.2 Measure_Class class

11.4.2.2.1 Description of Measure_Class

Measure_Class is a class each instance of which models a **measure class** (3.2.72), a set of equivalent **units of measure** (3.2.138) that may be shared across multiple **dimensionalities** (3.2.58)*. Measure_Class* allows a grouping of units of measure to be specified once, and reused by multiple dimensionalities.

EXAMPLE: We could define the *Measure_Classes*: Metric Linear Distance, Imperial Linear Distance, each associated with the appropriate *Units_of_Measure*; and associate them with *Dimensionalities*: Height, Width, and Depth to model the three spatial dimensions.

Measure_Class shall participate in the following associations:

— *unit_of_measure_class* (11.4.3.2) with one or more *Units_of_Measure* which serves to group equivalent *Units_of_Measure.*

Measure_Class may participate in the following associations:

— *dimensionality_measure_class* (11.4.3.1) with zero or more *Dimensionality* (11.4.2.3), which use the *Measure_Class* to identify the applicable *Units_of_Measure* (11.4.2.1);

11.4.2.3 Dimensionality class

11.4.2.3.1 Direct superclass

Concept (9.1.2.1)

11.4.2.3.2 Description of Dimensionality

Dimensionality is a class each instance of which models a **dimensionality** (3.2.58), a set of **measure class**es (3.2.72) each of which in turn groups a set equivalent **units of measure** (3.2.138), 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.

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.

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.

EXAMPLE 1: 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 $C^o = (5/9)*(F^o - 32)$ and $F^o = (9/5)*(C^o) + 32$. (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.

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.

Dimensionalities which use the same units of measure may apply to very different concepts.

EXAMPLE 2: Angular velocity and frequency might be defined as two distinct Dimensionalities sharing a Measure Class grouping all Units of Measure that represent T^{-1} (where T represents Time).

Dimensionality shall participate in the following associations:

— *dimensionality_measure_class* (11.4.3.1) with one or more *Measure_Class*es (11.4.2.2), which identify the applicable *Units_of_Measure* (11.4.2.1);

Dimensionality has one attribute *coordinate_indicator* (11.4.2.3.3.1) of type *Boolean* (6.2.2).

11.4.2.3.3 Attributes of Dimensionality

11.4.2.3.3.1 coordinate_indicator

Attribute name:	*coordinate_indicator*
Definition:	predicate on a *Dimensionality* whose value is true if the *Dimensionality* is a *coordinate.*
Obligation:	Conditional
Multiplicity:	*0..1*
Datatype:	Boolean (6.2.2)
Condition:	The indicator must be specified for dimensionalities of physical units.
NOTE	If the *Dimensionality* refers to an interval measure, the value of the *coordinate_indicator* is false.
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).

—— End of attributes of *Dimensionality* ——

11.4.3 Associations in the Measurement region

11.4.3.1 dimensionality_measure_class association

dimensionality_measure_class associates zero or more *Dimensionalities* (11.4.2.3) with one or more *Measure_Classes* (11.4.2.2) which group together the *Units_of_Measure* (11.4.2.1) that apply to the *Dimensionality*.

dimensionality_measure_class has two roles:

— *dimensionality* (verb form: has_dimensionality), which references an instance of the *Dimensionality* class;

— *applicable_units* (verb form: has_applicable_units), which references an instance of the *Measure_Class* class.

NOTE While **units of measure** (3.2.138) are commonly physical units of measure, they might also be currency units (in which the corresponding **dimensionality** (3.2.58) might be Money).

11.4.3.2 unit_of_measure_class association

unit_of_measure_class associates one or more *Units_of_Measure* with zero or more *Measure_Classes*.

unit_of_measure_class has two roles:

— *measure_class* (verb form: is_member_of), which references an instance of the *Measure_Class* class;

— *member_unit* (verb form: has_member_unit), which references an instance of the *Unit_of_Measure* class.

NOTE While **units of measure** (3.2.138) are commonly physical units of measure, they might also be currency units (in which the corresponding **dimensionality** (3.2.58) might be Money).

11.5 Data_Element metamodel region

11.5.1 Overview

The Data_Element metamodel region, illustrated in Figure 15 below, 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*.

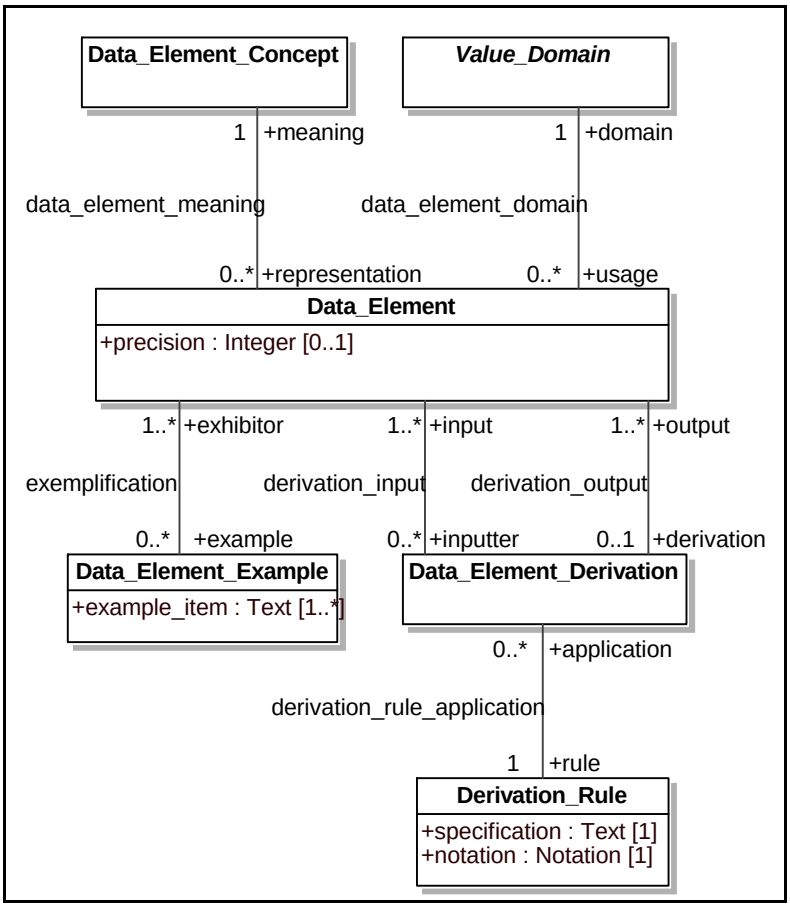

Figure 15 — Data_Element metamodel region

11.5.2 Classes in the Data_Element Region

11.5.2.1 Data_Element class

11.5.2.1.1 Description of Data_Element

Data_Element is a class each instance of which models a **data element** (3.2.28), a unit of **data** (3.2.27) that is considered in context to be indivisible.

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.

A *Data_Element* is formed when a *Data_Element_Concept* (11.5.2.2) is assigned a representation. One of the key components of a representation is the *Value_Domain* (11.5.2.3), i.e., restricted valid values.

A *Data_Element* shall participate in the following associations:

— *data_element_meaning* (11.5.3.2) with a *Data_Element_Concept*;

— *data_element_domain* (11.5.3.1) with a *Value_Domain*.

A *Data_Element* cannot be recorded in a **metadata registry** (3.2.78) without being associated with both a *Data_Element_Concept* and a *Value_Domain*.

A *Data_Element* may participate in the following associations:

— *deriviation_input* (11.5.3.4) as input to a *Data_Element_Derivation* (11.5.2.6)*;*

— *derivation_output* (11.5.3.5) as output from a *Data_Element_Derivation;*

— *exemplification* (11.5.3.3) with a *Data_Element_Example* (11.5.2.4) that exemplifies the *Data_Element*.

A *Data_Element* has one attribute *data_element_precision* (11.5.2.1.2.1) of type *Integer* (6.2.5).

11.5.2.1.2 Attributes of Data_Element

11.5.2.1.2.1 *data_element_precision*

Attribute name:	*data_element_precision*
Definition:	number of decimal places permitted in any associated data element values
Obligation:	Optional
Multiplicity:	*0..1*
Datatype:	*Integer* (6.2.5)

—— End of attributes of *Data_Element* ——

11.5.2.2 Data_Element_Concept class

Data_Element_Concept is described under the Data_Element_Concept region in 11.2.2.3. A *Data_Element_Concept* may be associated with several *Value_Domains* resulting in a different *Data_Element* for each association.

11.5.2.3 Value_Domain class

Value_Domain is described under the Conceptual_Domain and Value_Domain region in 11.3.2.5. 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*.

11.5.2.4 Data_Element_Example class

11.5.2.4.1 Description of Data_Element_Example

Data_Element_Example is a class each instance of which models a **data element example** (3.2.34), a representative illustration of a **data element** (3.2.28).

Every *Data_Element_Example* shall have an *exemplification* association (11.5.3.3) with one or more *exhibitor Data_Element*s (11.5.2.1), where the *Data_Element_Example* serves as an *example* for the *Data_Element*.

A *Data_Element_Example* shall have one or more *example_item* (11.5.2.4.2.1) attributes of type *Text* (6.2.12) that provide representative illustrations of instances of a *Data_Element*.

11.5.2.4.2 Attributes of Data_Element_Example

11.5.2.4.2.1 *example_item*

Attribute name:	*example_item*
Definition:	actual illustrative case of the *Data_Element*
Obligation:	Mandatory
Multiplicity:	*1..**
Datatype:	Text (6.2.12)

—— End of attributes of *Data_Element_Example* ——

11.5.2.5 Derivation_Rule class

11.5.2.5.1 Description of Derivation_Rule

Derivation_Rule is a class each instance of which models a **derivation rule** (3.2.45), logical, mathematical, and/or other operations specifying 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.

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*.

A *Derivation_Rule* may be registered as a *Registered_Item* without necessarily being associated with any *Data_Element_Derivation*. 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*.

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.*

11.5.2.5.2 Attributes of Derivation_Rule

11.5.2.5.2.1 *derivation_rule_specification*

Attribute name:	*derivation_rule_specification*
Definition:	text of a specification of a *data element Derivation_Rule*
Obligation:	Mandatory
Multiplicity:	*1*
Datatype:	Text (6.2.12)

11.5.2.5.2.2 *derivation_rule_notation*

Attribute name:	*derivation_rule_notation*
Definition:	notation used to specify the *Derivation_Rule*
Obligation:	Mandatory
Multiplicity:	*1*
Datatype:	Notation (6.2.7)

—— End of attributes of *Derivation_Rule* ——

11.5.2.5.3 Example of Derivation_Rule

One simple example of a data element derivation might be concatenation, where if *derivation_rule_notation* is 'EBNF', the *derivation_rule_specification* might be specified as:

— output := input[1] [input[2] [input[3] ...]]

11.5.2.6 Data_Element_Derivation class

Data_Element_Derivation is a class each instance of which models a **data element derivation** (3.2.33), the application of a **derivation rule** (3.2.45) to one or more input **data element**s (3.2.28) to derive one or more output data elements.

Data_Element_Derivation is a class that associates the *Data_Element(s)* (11.5.2.1) that serve as sources or inputs with a *Derivation_Rule* (11.5.2.5) and the *Data_Element(s)* that are the products or outputs of the *Derivation_Rule*.

Data_Element_Derivation shall participate in the following associations:

— *derivation_input* (11.5.3.4) with one or more *input Data_Element(s)*, where the *Data_Element_Derivation* serves as the *inputter* for the associated *Data_Element*;

— *derivation_rule_application* (11.5.3.6) with exactly one *Derivation_Rule* that provides the specification for the derivation, where the *Data_Element_Derivation* is the *application* of the *rule*;

— *derivation_output* (11.5.3.5) with one or more *output Data_Element(s)*, where the *Data_Element_Derivation* serves as the *derivation* for the associated *Data_Element*.

11.5.3 Associations in the Data_Element region

11.5.3.1 data_element_domain association

The *data_element_domain* association binds a *Data_Element* (11.5.2.1) and a *Value_Domain* (11.3.2.5) that describes a set of possible values that may be recorded in an instance of the *Data_Element*.

The association has two roles:

— *usage* (verb form: uses) which references a *Data_Element;*

— *domain* (verb form: is domain for) which references a *Value_Domain.*

11.5.3.2 data_element_meaning association

The *data_element_meaning* association binds a *Data_Element* (11.5.2.1) with a *Data_Element_Concept* (11.2.2.3) that provides the meaning for the *Data_Element*. The association has two roles:

— *meaning* (verb form: provides meaning for) which references a *Data_Element_Concept;*

— *representation* (verb form: represents) which references a *Data_Element.*

11.5.3.3 exemplification association

The *exemplification* association binds a *Data_Element* (11.5.2.1) with a *Data_Element_Example* (11.5.2.4) that provides an example instance or use of the *exhibitor Data_Element*. The association has two roles:

— *exhibitor* (verb form: exemplified by) which references a *Data_Element* to be exemplified;

— *example*.(verb form: exemplifies) which references a *Data_Element_Example* the exemplifies the *Data_Element.*

11.5.3.4 derivation_input association

The *derivation_input* association binds one or more *input Data_Element*(s) (11.5.2.1) with a *Data_Element_Derivation* (11.5.2.6). The association has two roles:

— *input* (verb form: is input to) which references a *Data_Element;*

— *inputter* (verb forms: inputs) which references a *Data_Element_Derivation*.

11.5.3.5 derivation_output association

The *derivation_output* association binds a *Data_Element_Derivation* (11.5.2.6) with one or more *output Data_Elements* (11.5.2.1) that are the result of the application of the *Data_Element_Derivation*. The association has two roles:

— *output* (verb form: is output) which references a *Data_Element;*

— *derivation* (verb form: derives) which references a *Data_Element_Derivation*.

11.5.3.6 derivation_rule_application association

The *derivation_rule_application* association binds a *Data_Element_Derivation* with a *Derivation_Rule* that specifies the rule to be used for the derivation. The association has two roles:

— application (verb form: applies) which references a *Data_Element_Derivation*;

— rule (verb form: provides rule for) which references a *Derivation_Rule*.

11.6 Consolidated Data Description Metamodel

A consolidated metamodel is shown in Figure 16. This combines the Data_Element_Concept, Data_Element, and Conceptual and Value_Domain regions of the model.

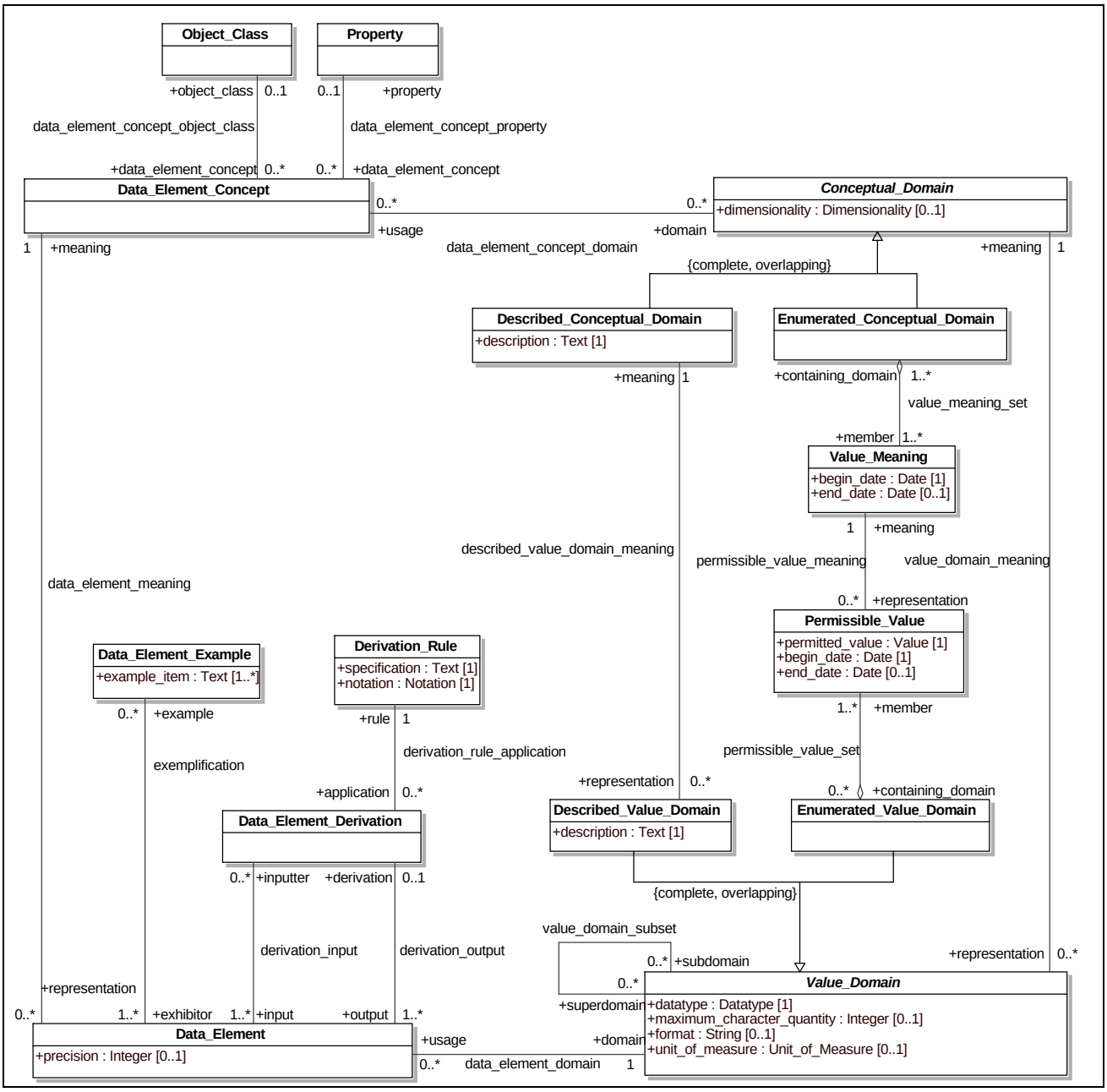

Figure 16 — Consolidated Data Description metamodel

11.7 Types of Concepts in the Data Description Metamodel

Figure 17 shows the classes in the Data Description package which are types of *Concepts* (i.e. they are sub-typed from the *Concept* class).

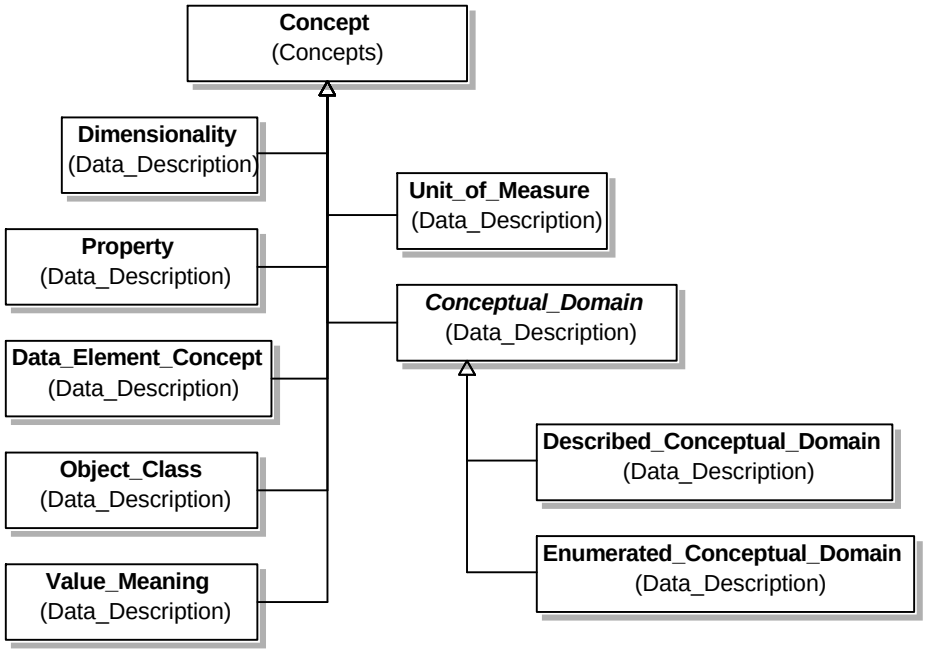

Figure 17 — Types of Concepts in the Data Description package

12 Basic attributes

12.1 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.

A mapping among the 1994 basic attributes, the 2011 basic attributes and the 2011 metamodel can be found in Annex C.

Clauses 6 through 11 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 might 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 Clauses 6 through 11.

12.2 Common attributes

The attributes listed in this subclause are common to all types of Administered_Item. These attributes are further categorized as: Identifying, Naming, Definitional, Administrative, and Relational.

12.2.1 Identifying

Attribute	**Obligation**
item identifier	Zero or more per metadata item. Required if *name* (see 12.2.2) is not unique within a given *context (see note 1).*
item identifier – identifier	One per *item identifier*. (The mandatory portion of an *item identifier*.)
item identifier – registration authority identifier	Zero or one per *item identifier*. (The optional portion of an *item identifier - see note 2.)*
version	Zero or one per metadata item *(see note 3).*

NOTE 1 While *item identifier* is mandatory within a registry (see 6.2.2.1), 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 2 While *item registration authority identifier* is mandatory within a registry (represented by Registration association between Administered_Item and Registration_Authority - see 8.1.2.2.2), it is optional in non-registry settings.

NOTE 3 Within a registry, *version* is an attribute of the *Scoped_Identifier* of an *Identified_Item* (see 7.2.2.2.2.2).

12.2.2 Naming

Attribute	Obligation
name	One or more per metadata item *(see note).*
designation language	Zero or one per name
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*.
designation acceptability	Zero or one per name.

NOTE If more than one *name* is specified within a given *context*, it is usual to nominate one name as "preferred", and the others (the synonyms) as "accepted".

12.2.3 Definitional

Attribute	Obligation
definition	One for each *context* in which the metadata item is used *(see note).*
definition language	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 might need to be different in different *contexts*, and thus separate *definitions* are permitted for each *context*.

12.2.4 Administrative

Administrative attributes are primarily associated with recording metadata items in a registry. They are therefore optional in non-registry settings.

Attribute	Obligation
comments	Zero or one per metadata item.
registration status	Zero or one per metadata item.
responsible organization	Zero or one per metadata item.
submitting organization	Zero or one per metadata item.

12.2.5 Relational

Attribute	Obligation
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 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).
type of relationship	One per *related metadata reference*.

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

12.3 Attributes specific to Data_Element_Concepts

The attributes listed in this subclause are specific to Data_Element_Concepts.

Attribute	Obligation
object class name	One per *data element concept.*
object class identifier	Zero or one per *data element concept.*
property name	One per *data element concept.*
property identifier	Zero or one per *data element concept.*

12.4 Attributes specific to Data_Elements

The attributes listed in this subclause are specific to Data_Elements.

Attribute	Obligation
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.*
maximum size	Zero or one per *data element.*
minimum size	Zero or one per *data element.*

NOTE This edition removes ‘representation class’ as a separate attribute, and views it as just an example of classification, which can be specified using the relational attributes. E.g. *classification scheme name* = ‘representation class’, and the *classification scheme item value* has the value that representation class would previously have had.

12.5 Attributes specific to Conceptual_Domains

The attributes listed in this subclause are specific to Conceptual_Domains.

Attribute	Obligation
dimensionality	Zero or one per *conceptual domain.*

12.6 Attributes specific to Value_Domains

The attributes listed in this subclause are specific to Value_Domains.

Attribute	Obligation
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.*

12.7 Attributes specific to Permissible_Values

The attributes listed in this subclause are specific to Permissible_Values.

Attribute	Obligation
permitted 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.*

12.8 Attributes specific to Value_Meanings

The attributes listed in this subclause are specific to Value_Meanings.

Attribute	Obligation
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.*

Annex A
(normative)

Alphabetical list of terms and designations

In the table below, if the definition comes from a clause 3.n.n, then the item being defined is a general term. If the definition comes from clause 5.n.n.n through 10.n.n.n.n.n, then the item begin defined is a construct within the metamodel.

Table A - 1 : Alphabetical List of Terms

Term / Designation	Defined in
abstract class	3.1.1
acceptability (attribute of *Definition_Context*)	7.3.3.1.2.1
acceptability (attribute of *Designation_Context)*	7.3.3.2.2.1
Acceptability (enumeration)	7.3.2.1
acceptability rating	3.2.1
administered item	3.2.2
Administered_Item (class)	8.1.2.2
administrative information	3.2.3
administrative_note (attribute of *Registration_State*)	8.1.2.6.2.4
administrative_status (attribute of *Registration_State*)	8.1.2.6.2.6
arity	3.2.4
arity (attribute of *Relation)*	9.1.2.4.3.1
assertion	3.2.5
Assertion (class)	9.1.2.3
assertion_concept (association)	9.1.3.6
assertion_inclusion (association)	9.1.3.5
assertion_predicate (association)	9.1.3.7
association	3.1.2
association class	3.1.3
attached item	3.2.6
Attached_Item (class)	8.1.2.3
attachment (association)	8.1.6.1
attribute	3.1.4
attribute instance	3.2.7
attribute value	3.2.8
authority_rule (attribute of *Naming_Convention*)	7.3.2.7.2.2
basic attribute	3.2.9
begin_date (attribute of *Permissible_Value*)	11.3.2.7.2.2
begin_date (attribute of *Value_Meaning*)	11.3.2.3.3.1
binary relation	3.2.10
Binary_Relation (class)	10.1.2.2
binding	3.2.11
boolean	3.2.12
Boolean (datatype)	6.2.2
cardinality	3.2.13
CD	3.3.1
change description (attribute of *Administered_Item*)	8.1.2.2.3.3

Term / Designation	Defined in
character_repertoire (attribute of *Registry_Specification*)	8.1.2.9.2.5
characteristic	3.2.14
CL	3.3.2
class	3.1.5
classifiable item	3.2.15
Classifiable_Item (class, type)	9.2.2.1
Classification (association)	9.2.3.1
classification scheme	3.2.16
classification_scheme (association)	9.2.4.1
CLIF	3.3.3
comment (attribute of *Registry_Specification*)	8.1.2.9.2.10
common attribute	3.2.17
composite attribute	3.1.6
composite datatype	3.1.7
concept	3.2.18
Concept (class)	9.1.2.1, 9.2.2.3
concept_ source (association)	9.1.3.2
concept system	3.2.19
Concept_System (class)	9.1.2.2, 9.2.2.2
concept_system_importation (association)	9.1.3.4
concept_system_membership (association)	9.1.3.1, 9.2.4.2
concept_system_reference (association)	9.1.3.3
conceptual data model	3.2.20
conceptual domain	3.2.21
Conceptual_Domain (class)	11.1.2.2, 11.2.2.4, 11.3.2.1
concrete class	3.1.8
conditional	3.2.22
conformance_level (attribute of *Registry_Specification*)	8.1.2.9.2.4
contact	3.2.23
contact (attribute of *Stewardship_Record)*	8.1.2.7.2.2
contact (attribute of *Submission_Record*)	8.1.2.8.2.2
Contact (class)	6.3.2, 8.1.3.1
contact information	3.2.24
context	3.2.25
context (attribute of *Registry_Specification*)	8.1.2.9.2.9

Term / Designation	Defined in
Context (class)	7.3.2.5
coordinate	3.2.26
coordinate_indicator (attribute of *Dimensionality)*	11.4.2.3.3.1
creation_date (attribute of (*Administered_Item*)	8.1.2.2.3.1
data	3.2.27
data element	3.2.28
Data_Element (class)	11.1.2.4,11.5.2.1
data element concept	3.2.29
Data_Element_Concept (class)	11.1.2.5,11.2.2.3
data element concept property	3.2.30
data_element_concept_property (association)	11.2.3.1
data element concept domain	3.2.31
data_element_concept_domain (association)	11.1.3.4, 11.2.3.2
data element concept object class	3.2.32
data_element_concept_object_class (association)	11.2.3.3
data element derivation	3.2.33
Data_Element_Derivation (class)	11.5.2.6
data_element_domain (association)	11.1.3.2, 11.5.3.1
data element example	3.2.34
Data_Element_Example (class)	11.5.2.4
data_element_meaning (association)	11.1.3.3, 11.5.3.2
data element precision	3.2.35
data_element_precision (attribute of *Data_Element*)	11.5.2.1.2.1
data model	3.2.36
datatype	3.1.8
datatype (attribute of *Value_Domain*)	11.3.2.5.2.1
Datatype (class)	11.3.2.9
datatype_annotation (attribute of *Datatype)*	11.3.2.9.2.4
datatype_description (attribute of *Datatype)*	11.3.2.9.2.2
datatype_name (attribute of *Datatype)*	11.3.2.9.2.1
datatype_scheme_reference (attribute of *Datatype)*	11.3.2.9.2.3
date	3.2.37
Date (datatype)	6.2.3
date time	3.2.38

Term / Designation	Defined in
Datetime (datatype)	6.2.4
DE	3.3.4
DEC	3.3.5
definition	3.2.39
definition <designatable item>	3.2.40
Definition (class)	7.3.2.4
definition context	3.2.41
Definition_Context (association class)	7.3.3.1
definition language	3.2.42
definition source reference	3.2.43
definition text	3.2.44
derivation_input (association)	11.5.3.4
derivation_output (association)	11.5.3.5
derivation rule	3.2.45
Derivation_Rule (class)	11.5.2.5
derivation_rule_application (association)	11.5.3.6
derivation rule notation	3.2.46
derivation rule specification	3.2.47
described conceptual domain	3.2.48
Described Conceptual_Domain (Class)	11.3.2.4
described value domain	3.2.49
Described_Value_Domain (class)	11.3.2.8
described_value_domain_meaning (association)	11.3.3.3
description (attribute of *Described_Conceptual_Domain)*	11.3.2.4.3.1
description (attribute of *Described_Value_Domain*)	11.3.2.8.3.1
designatable item	3.2.50
Designatable_Item (class, type)	7.3.2.2
designation	3.2.51
designation <designatable item>	3.2.52
Designation (class)	7.3.2.3
designation acceptability	3.2.53
designation context	3.2.54
Designation_Context (association class)	7.3.3.2
designation_definition_pairing (association)	7.3.4.1
designation language	3.2.55
designation namespace	3.2.56
designation_namespace (association)	7.3.4.2
designation sign	3.2.57

Term / Designation	Defined in
dimensionality	3.2.58
dimensionality (attribute of *Conceptual_Domain)*	11.3.2.1.3.1
Dimensionality (class)	11.4.2.3
disjoint	3.2.59
documentation_language_identifier (attribute of *Registration_Authority*)	8.1.2.5.3.2
effective_date (attribute of *Registration_State*)	8.1.2.6.2.2
email_address (attribute of *Individual*)	6.3.4.2.4
email_address (attribute of *Organization*)	6.3.6.2.3
end_date (attribute of *Permissible_Value*)	11.3.2.7.2.3
end_date (attribute of *Value_Meaning*)	11.3.2.3.3.2
enumerated conceptual domain	3.2.60
Enumerated_Conceptual_Domain (class)	11.3.2.2
enumerated value domain	3.2.61
Enumerated_Value_Domain (class)	11.3.2.6
example_item (attribute of *Data_Element_Example*)	11.5.2.4.2.1
exemplification (association)	11.5.3.3
explanatory_comment (attribute of *Administered_Item*)	8.1.2.2.3.4
extension <11179-3>	3.2.62
extension_identifier (attribute of *Language_Identification*)	6.3.5.2.5
format (attribute of *Value_Domain*)	11.3.2.5.2.2
formula (attribute of Assertion)	9.1.2.3.2.1
full_expansion (attribute of *Scoped_Identifier*)	7.2.2.2.2.3
generalization	3.1.10
identification (association)	7.2.3.1
identification scheme	3.2.63
identified item	3.2.64
Identified_Item (class, type)	7.2.2.1
identifier <metamodel>	3.1.11
identifier (attribute of *Reference_Document)*	6.3.7.2.1
identifier (attribute of *Scoped_Identifier*)	7.2.2.2.2.1
identifier_scope (association)	7.2.3.2
individual	3.2.65

Term / Designation	Defined in
individual (attribute of Contact)	6.3.2.2.1
Individual (class)	6.3.4
integer	3.2.66
Integer (datatype)	6.2.5
international code designator	3.2.67
international_code_designator (attribute of *Registration_Authority_Identifier)*	6.3.8.2.1
item_definition (association)	7.3.4.3
item_designation (association)	7.3.4.4
item_slot (association)	7.2.3.3
language	3.2.68
language (attribute of *Definition*)	7.3.2.4.2.2
language (attribute of *Designation)*	7.3.2.3.2.2
Language_Identification (class)	6.3.5
language_identifier (attribute of *Language_Identification)*	6.3.5.2.1
language_identifier (attribute of *Reference_Document)*	6.3.7.2.3
last_change_date (attribute of *Administered_Item*)	8.1.2.2.3.2
lexical_rule (attribute of *naming_Convention*)	7.3.2.7.2.5
link	3.2.69
Link (class)	9.1.2.6
link end	3.2.70
Link_End (class)	9.1.2.7
link_end_concept (association)	9.1.3.11
link_end_role (association)	9.1.3.12
link_has_link_end (association)	9.1.3.10
mail_address (attribute of *Individual*)	6.3.4.2.3
mail_address (attribute of *Organization*)	6.3.6.2.2
mail_address (attribute of *Rple*)	6.3.9.2.2
mandatory	3.2.71
mandatory_naming_convention_ indicator (attribute of *Namespace*)	7.2.2.3.2.4
maximum_character_quantity (attribute of *Value_Domain*)	11.3.2.5.2.3
MDR	3.2.78 , 3.3.6
measure class	3.2.72
Measure_Class (class)	11.4.2.2
mece	3.3.7

Term / Designation	Defined in
meronomy	3.2.73
metadata	3.2.74
metadata item	3.2.75
metadata object	3.2.76
metadata register	3.2.77
metadata registry	3.2.78
metadata registry product	3.2.79
metamodel	3.2.80
metamodel construct	3.2.81
metamodel region	3.1.12
multiplicity	3.2.82
multiplicity (attribute of *Relation_Role*)	9.1.2.5.3.1
name	3.2.83
name (attribute of *Individual*)	6.3.4.2.1
name (attribute of *Organization*)	6.3.6.2.1
name (attribute of *Registry_Specification*)	8.1.2.9.2.1.1
name (attribute of *Slot*)	7.2.2.4.2.1
namespace	3.2.84
Namespace (class)	7.2.2.3, 7.3.2.6, 8.1.4.1
naming_authority (attribute of *Namespace*)	7.2.2.3.2.1
naming convention	3.2.85
Naming_Convention (class)	7.3.2.7
naming_convention_conformance (association)	7.3.4.5
naming_convention_utilization (association)	7.3.4.6
Natural_Range (datatype)	6.2.6
notation	3.2.86
notation (attribute of *Concept_System)*	9.1.2.2.2.1
notation (attribute of *Derivation_Rule)*	11.5.2.5.2.2
notation (attribute of *Reference_Document)*	6.3.7.2.4
Notation (datatype)	6.2.7
object	3.2.87
object class	3.2.88
Object_Class (class)	11.2.2.1
one_item_per_name_indicator (attribute of *Namespace*)	7.2.2.3.2.3
one_name_per_item_indicator (attribute of *Namespace*)	7.2.2.3.2.2

Term / Designation	Defined in
opi	3.3.8
opi_source (attribute of *Registration_Authority_Identifier*)	6.3.8.2.4
optional	3.2.89
ordinal (attribute of *Relation_Role*)	9.1.2.5.3.2
organization	3.2.90
organization (attribute of Contact)	6.3.2.2.2
organization (attribute of *Stewardship_Record)*	8.1.2.7.2.1
organization (attribute of *Submission_Record*)	8.1.2.8.2.1
Organization (class)	6.3.6, 8.1.3.2
organization identification scheme	3.2.91
organization identifier	3.2.92
organization_identifier (attribute of *Registration_Authority_Identifier*)	6.3.8.2.2
organization part	3.2.93
organization part identifier (opi)	3.2.94
organization_part_identifier (attribute of *Registration_Authority_Identifier*)	6.3.8.2.3
organization Person	3.2.95
origin (attribute of *Administered_Item*)	8.1.2.2.3.5
ORM	3.3.9
OWL	3.3.10
OWL-DL	3.3.11
package	3.1.13
permissible value	3.2.96
Permissible_Value (class)	11.3.2.7
permissible_value_meaning (association)	11.3.3.4
permissible_value_set (association)	11.3.3.5
permitted_value (attribute of *Permissible_Value*)	11.3.2.7.2.1
Person	3.2.97
phone number	3.2.98
phone number (attribute of *Individual*)	6.3.4.2.5
phone number (attribute of *Organization*)	6.3.6.2.4
Phone_Number (datatype)	6.2.8
postal address	3.2.99
Postal_Address (datatype)	6.2.9
previous_state (attribute of *Registration_State*)	8.1.2.6.2.7

Term / Designation	Defined in
primary_language (attribute of *Registry_Specification*)	8.1.2.9.2.7
primitive datatype	3.2.10
private_use_qualifier (attribute of *Language_Identification*)	6.3.5.2.6
property	3.2.100
Property (class)	11.2.2.2
provider (attribute of *Reference_Document)*	6.3.7.2.6
quantity	3.2.101
RA	3.2.109 , 3.3.12
RDF	3.3.13
Reference (association class)	8.1.5.2
reference document	3.2.102
Reference_Document (class)	6.3.7, 8.1.3.3
reference_document_identifier_form (attribute of *Registry_Specification*)	8.1.2.9.2.6
reflexivity	3.2.103
reflexivity (attribute of *Binary_Relation)*	10.1.2.2.3.1
Reflexivity (enumeration)	10.1.2.3
geopolitical_territory_identifier (attribute of *Language_Identification)*	6.3.5.2.3
register	3.2.104
registered item	3.2.105
Registered_Item (class, type)	8.1.2.1
registrar	3.2.106
Registrar (class)	8.1.2.4
registrar identifier	3.2.107
registrar_identifier (attribute of *Registrar*)	8.1.2.4.3.1
registration	3.2.108
Registration (association class)	8.1.5.1
registration authority	3.2.109
Registration_Authority (class)	8.1.2.5
registration authority identifier	3.2.110
registration_authority_identifier (attribute of *Registration_Authority*)	8.1.2.5.3.1
Registration_Authority_Identifier (datatype, class)	6.3.8
registration_authority_namespace (association)	8.1.6.2
registration_authority_registrar (association)	8.1.6.3
registration state	3.2.111

Term / Designation	Defined in
registration_state (attribute of *Registration*)	8.1.5.1.2.1
Registration_State (class, datatype)	8.1.2.6
registration status	3.2.112
registration_status (attribute of *Registration_State*)	8.1.2.6.2.1
registry	3.2.113
registry instance	3.2.114
registry item	3.2.115
registry metamodel	3.2.116
registry product	3.2.117
Registry_Specification (class)	8.1.2.9
related metadata reference	3.2.118
relation	3.2.119
Relation (class)	9.1.2.4, 10.1.2.1
relation_link (association)	9.1.3.8
relation role	3.2.120
Relation_Role (class)	9.1.2.5
relation_role_set (association)	9.1.3.9
relationship <metamodel>	3.1.15
representation_class_scheme (attribute of *Registry_Specification*)	8.1.2.9.2.8
role	3.2.121
role (attribute of *Contact)*	6.3.2.2.3
role (attribute of *Individual*)	6.3.4.2.6
Role (class)	6.3.9
SBVR	3.3.14
scheme_reference (attribute of *Namespace*)	7.2.2.3.2.6
scope_rule (attribute of *Naming_Convention*)	7.3.2.7.2.1
scoped identifier	3.2.122
Scoped_Identifier (class)	7.2.2.2
script_identifier (attribute of *Language_Identification*)	6.3.5.2.2
semantic_rule (attribute of *Naming_Convention*)	7.3.2.7.2.3
shorthand_expansion (attribute of *Scoped_Identifier*)	7.2.2.2.2.4
shorthand_prefix (attribute of *Namespace*)	7.2.2.3.2.5
sign (attribute of *Designation)*	7.3.2.3.2.1
sign (noun)	3.2.123
Sign (datatype)	6.2.10

Term / Designation	Defined in
SKOS	3.3.15
slot	3.2.124
Slot (class)	7.2.2.4
source (attribute of *Definition*)	7.3.2.4.2.3
specialization	3.1.16
specification (attribute of *Derivation_Rule)*	11.5.2.5.2.1
standard (attribute of *Registry_Specification*)	8.1.2.9.2.3
stewardship (association)	8.1.6.4
stewardship <metadata>	3.2.125
stewardship contact	3.2.126
stewardship organization	3.2.127
stewardship record	3.2.127
Stewardship_Record (class)	8.1.2.7
string	3.2.129
String (datatype)	6.2.11
subclass	3.1.17
submission	3.2.130
submission (association)	8.1.6.5
submission contact	3.2.131
submission organization	3.2.132
submission record	3.2.133
Submission_Record (class)	8.1.2.8
superclass	3.1.18
symmetry	3.2.134
symmetry (attribute of *Binary_Relation*)	10.1.2.2.3.2
Symmetry (enumeration)	10.1.2.4
syntactic_rule (attribute of *Naming_Convention*)	7.3.2.7.2.4
taxonomy	3.2.135
text	3.2.136
text (attribute of *Definition*)	7.3.2.4.2.1
Text (datatype)	6.2.12
title (attribute if (*Individual*)	6.3.4.2.2
title (attribute of *Reference_Document)*	6.3.7.2.5
title (attribute of *Role*)	6.3.9.2.1
transitive relation	3.2.137
transitivity (attribute of *Binary_Relation*)	10.1.2.2.3.3
Transitivity (enumeration)	10.1.2.5

Term / Designation	Defined in
type (attribute of *Reference)*	8.1.5.2.2.1
type (attribute of *Slot*)	7.2.2.4.2.2
type_description (attribute of *Reference_Document)*	6.3.7.2.2
UML	3.3.17
unit of measure	3.2.138
unit_of_measure (attribute of *Value_Domain*)	11.3.2.5.2.4
Unit_of_Measure (class)	11.4.2.1
unit of measure dimensionality	3.2.139
unit_of_measure_dimensionality (association)	11.4.3.1
unresolved_issue (attribute of *Registration_State*)	8.1.2.6.2.5
until_date (attribute of *Registration_State*)	8.1.2.6.2.3
UPU	3.3.18
URI	3.3.19
uri (attribute of *Organization*)	6.3.6.2.5
uri (attribute of *Reference_Document)*	6.3.7.2.7
Value (datatype)	6.2.13
value (attribute of *Slot*)	7.2.2.4.2.3
value domain	3.2.140
Value_Domain (class)	11.1.2.3, 11.3.2.5
value_domain_meaning (association)	11.1.3.1, 11.3.3.1
value_domain_subset (association)	11.3.3.6
value meaning	3.2.141
Value_Meaning (class)	11.3.2.3
value_meaning_set (association)	11.3.3.2
variant_identifier (attribute of *Language_Identification*)	6.3.5.2.4
VD	3.2.140 , 3.3.20
version	3.2.142
version (attribute of *Scoped_Identifier)*	7.2.2.2.2.2
W3C	3.3.21
web_address (attribute of *Registry_Specification*)	8.1.2.9.2.2
XCL	3.3.22
XML	3.3.23
XTML	3.3.24

Annex B
(normative)

Consolidated Class Hierarchy

Figure 18 shows all classes that participate in a class hierarchy. Classes that do not participate in a hierarchy are not shown.

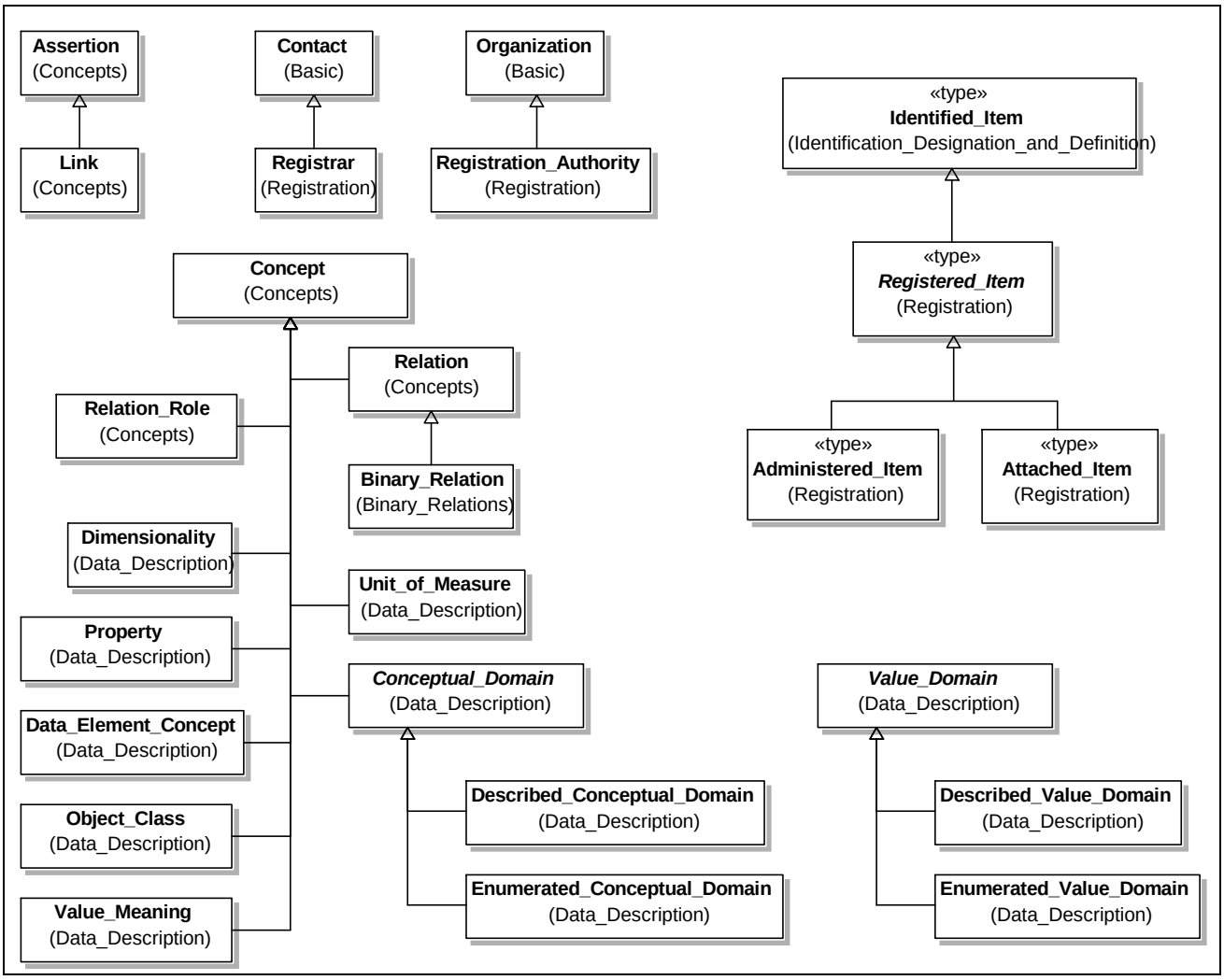

Figure 18 — Consolidated Class Hierarchy

Annex C
(informative)

Mapping the ISO/IEC 11179-3:1994 basic attributes to the ISO/IEC 11179-3:2011 metamodel and basic attributes

C.1 Introduction

C.1.1 Overview of Basic Attributes from ISO/IEC 11179-3:1994

ISO/IEC 11179-3:1994 lists 23 basic attributes of data elements, as shown in Figure 19.

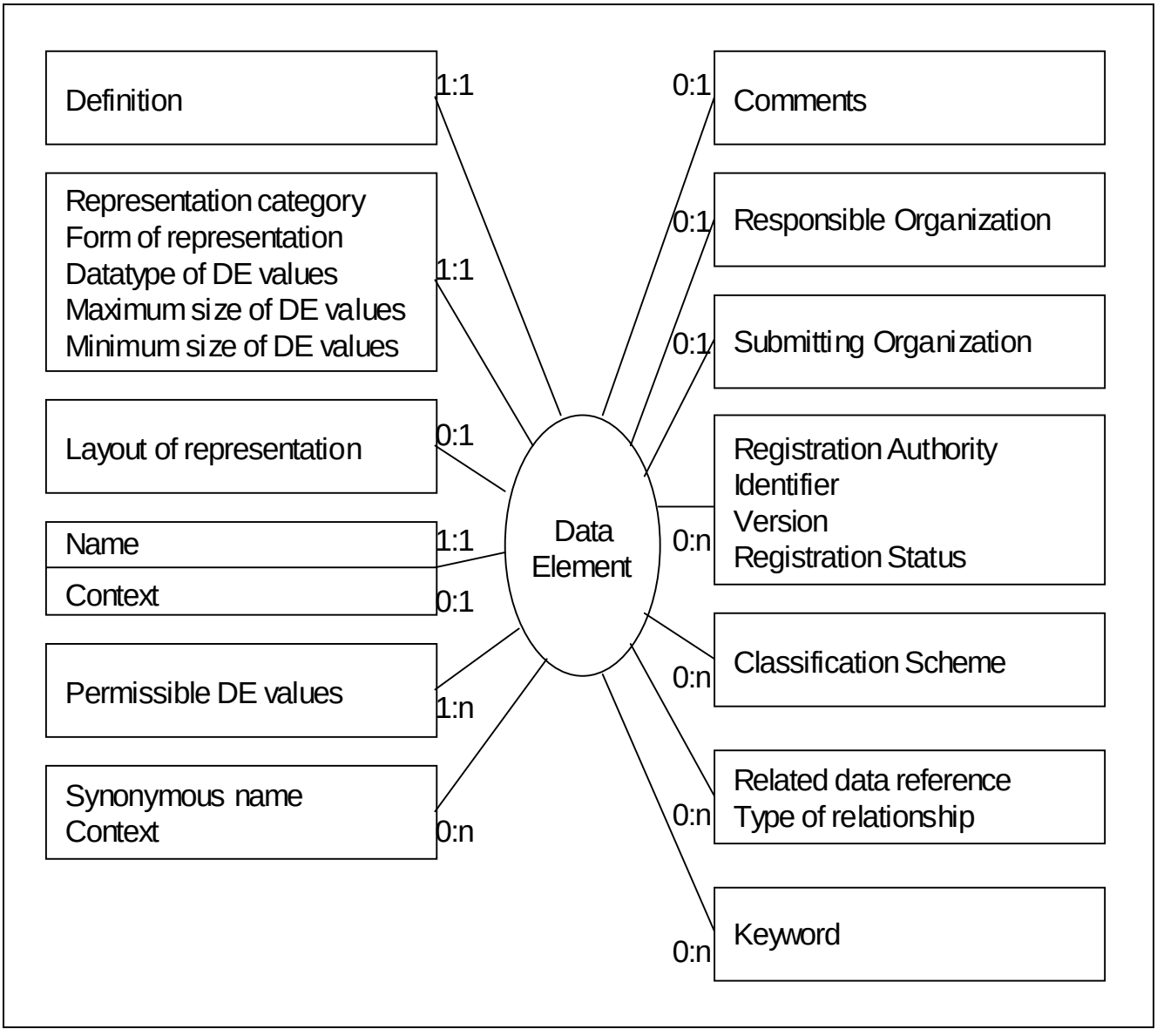

Figure 19 — Basic Attributes of Data elements

This third edition of the standard supports not only data elements, but also other metadata items associated with them, such as data element concepts, conceptual domains, value domains and concept systems.

This annex maps the 1994 basic attributes to the latest metamodel in Clauses 5 through 11, and the latest basic attributes in Clause 12.

C.1.2 Description of Table Structures in this Annex

C.1.2.1 Organization of the subclauses

Each attribute is described in this Annex by two subclauses:

The first subclause describes the attribute mapping using a table as illustrated by Table 5 below.

NOTE Table 5 is deliberately empty because it is just illustrating the layout used in the following clauses.

Table 5 – Template for attribute mapping

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:			
Definition:			
Obligation:			
Condition:			
Datatype:			
Comment:			

The second subclause describes the path to the attribute in the metamodel from *Identified_Item*, *Designatable_Item*, *Classificable_Item* or *Registered_Item*, as appropriate.

C.1.2.2 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 11179-3:2011 Metamodel (Clauses 5 through 11)

— Column 4: What is specified in ISO/IEC 11179-3:2011 Basic Attributes (Clause 12)

C.1.2.3 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.

C.1.2.4 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. Scoped_Identifier.identifier

C.1.2.5 Specification of Path to the named attribute

This information shows how the named attributed is related to an *Identified_Item*, *Designatable_Item*, *Classificable_Item* or *Registered_Item*, as appropriate,, 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. (See below for an explanation of the notation.)

In addition to designating the metamodel attribute in the context of a class (row 1 column 3), the “Path from xxx_Item” shows how the class is related to an *Identified_Item*, *Designatable_Item*, *Classificable_Item* or *Registered_Item*, as appropriate.. 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 one of the abstract superclasses), the starting point for navigation is the supeclass *Identified_Item*, *Designatable_Item*, *Classificable_Item* or *Registered_Item*, as appropriate.. For attributes specific to a particular subclass of the superclass, the starting point for navigation is that subclass (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 associations are lower case and italicised e.g. “*name entry*”

— the names of association classes have an initial capital letter and are italicised e.g. *Registration*

NOTE 2 The use of *italics* to indicate an association or association class 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 “*registration_status*”

In this example, the attribute is a Common Attribute (i.e. it can apply to any type of *registry item*), so the navigation starts from the appropriate superclass, in this case *Administered_Item*.

Administered_Item.*Registration*.registration_state.Registration_State.registration_status

specifies to follow the path in the model:

— from the class “Administered_Item” to the association class “*Registration*”, then via its attribute “registration_state” to the composite datatype “Registration_State”, then to its attribute “registration_status”

Example 2: Attribute "Datatype.name"

In this example, the attribute is specific to a Data_Element, so the navigation starts from the “Data_Element” class.

Data_Element.*data_element_domain*.Value_Domain.datatype.Datatype.name

specifies to follow the path in the model:

— from the class “Data_Element” via its relationship “*data_element_domain*” to the related class “Value_Domain”, then

— from the class “Value_Domain” via its attribute “datatype” to the composite datatype “Datatype” and its attribute “name”.

C.2 Mapping the Basic Attributes

The attributes are ordered in this Annex as in Clause 11 of this part of ISO/IEC 11179.

C.2.1 Common Identifying attributes

C.2.1.1 Item identifier – identifier

C.2.1.1.1 Attribute Mapping

Table 6 – Attribute mapping for ‘identifier’

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	Identifier	Scoped_Identifier.identifier	item identifier – identifier
Definition:	A language independent unique identifier of a data element within a Registration_Authority.	A unique identifier for an Identified_Item within a Registration_Authority.	A unique identifier for a metadata item within a specific namespace.
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.
Datatype:	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.

C.2.1.1.2 Path from Identified_Item

Identified_Item.*identification*.Scoped_Identifier.identifier.

C.2.1.2 Item identifier – registration authority identifier

C.2.1.2.1 Attribute Mapping

Table 7 – Attribute mapping for 'Registration Authority'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	Registration_Authority	Registration_Authority. registration_authority_identifier	item identifier – 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.
Datatype:	Character string	String	String

C.2.1.2.2 Path from Administered_Item

Administered_Item.*Registration*.Registration_Authority.registration_authority_identifier.

C.2.1.3 Version

C.2.1.3.1 Attribute Mapping

Table 8 – Attribute mapping for 'Version'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	Version	Scoped_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.	unique version identifier of the Scoped_Identifier used to identify an Identified_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_Authority.	N/A	N/A
Datatype:	Character	String	String

C.2.1.3.2 Path from Identified_Item

Identified_Item.*identification*.Scoped_Identifier.version

C.2.2 Common Naming attributes

C.2.2.1 Name

C.2.2.1.1 Attribute Mapping

Table 9 – Attribute mapping for 'Name'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	Name	Designation.sign	name
Definition:	Single or multi word designation assigned to a data element.	Representation of a Designatable_Item by a sign which denotes it.	A name by which a metadata item is known within a specific context.
Obligation:	Mandatory	Mandatory	Mandatory
Datatype:	Character string	String	String
Comment:		The attribute Designation_Context.acceptability may be used to specify the acceptability rating of the sign in a particular context.	

C.2.2.1.2 Path from Designatable_Item

Designatable_Item.*item_designation*.Designation.sign.

C.2.2.2 Synonymous name

C.2.2.2.1 Attribute Mapping

Table 10 – Attribute mapping for 'Synonymous name'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	Synonymous name	Designation.sign	name
Definition:	Single word or multi word designation that differs from the given name, but represents the same data element concept.	Representation of a Designatable_Item by a sign which denotes it.	A name by which a metadata item is known within a specific context.
Obligation:	Optional	Optional	Optional
Datatype:	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.	A Designatable_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 Designation_Context.acceptability, which should be set to "preferred" for the preferred name, and "accepted" 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 "designation acceptability", which should be set to "preferred" for the preferred name, and "accepted" for all synonyms.

C.2.2.2.2 Path from Designatable_Item

Designatable_Item.*item_designation*.Designation.sign.

C.2.2.3 designation language

C.2.2.3.1 Attribute Mapping

Table 11 – Attribute mapping for 'designation language'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	Not supported.	Designation.language	designation language
Definition:	N/a	Language or dialect in which the sign (usually a name) is expressed.	Language or dialect in which the sign (usually a name) is expressed.
Obligation:	N/a	Optional	Optional
Datatype:	N/a	String	String

C.2.2.3.2 Path from Designatable_Item

Designatable_Item.*item_designation*.Designation.language.

C.2.2.4 Context name

C.2.2.4.1 Attribute Mapping

Table 12 – Attribute mapping for 'Context name'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	Context	Designation.sign NOTE The "Designatable_Item" referred to here is the Context itself, not the Designatable_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. *Designation*: Representation of a Designatable_Item by a sign which denotes it.	*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 'Name' (6.1.1) occurs in an	N/A	Required if more than one *name* attribute exists for a particular metadata item.

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
	information exchange.		
Datatype:	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 Designatable_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.)	

C.2.2.4.2 Path from Designatable_Item

Designatable_Item(1).*item_designation*.Designation.*Designation_Context.*Context.
Designatable_Item(2).*item_designation*.Designation.sign.

NOTES (1) (2) The first Designatable_Item is the one to which context is being provided. The second Designatable_Item is that of the Context itself.

C.2.2.5 Context identifier

C.2.2.5.1 Attribute Mapping

Table 13 – Attribute mapping for 'Context identifier'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	Not supported.	Scoped_Identifier.identifier	context identifier
Definition:	N/A	*Context:* A universe of discourse in which a name or definition is used. *Scoped_Identifier.identifier*: *String* used to unambiguously denote an *Identified_Item* within the scope of a specified *Namespace.*	*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.
Datatype:	N/A	String	String
Comment:		A *Context* is subclassed from *Identified_Item* which has one or more *Scoped_Identifier*s, each providing a unique identifier for the *Context*.	

C.2.2.5.2 Path from Designatable_Item

Designatable_Item.*item_designation*.Designation.*Designation_Context.*Context.
Identified_Item.*identification*.Scoped_Identifier.identifier.

NOTE The Designatable_Item is the one to which context is being provided. The Identified_Item is that of the Context itself.

C.2.2.6 Context description

C.2.2.6.1 Attribute Mapping

Table 14 – Attribute mapping for 'Context description'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	Context	Definition.text	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. The Definition class provides the text for a Designatable_Item (7.3.2.2) as it applies in zero, one or more *Contexts* (7.3.2.5) *Definition.text:* text of the Definition.	*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.
Datatype:	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 Designatable_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.)	In this edition of this part of ISO/IEC 11179, *context description* and *context name* exist as two separate attributes.

C.2.2.6.2 Path from Designatable_Item

Designatable_Item(1).*item_designation*.Designation.*Designation_Context.*Context.
Designatable_Item(2).*item_definition*.Definition.text.

NOTES (1) (2) The first Designatable_Item is the one to which context is being provided. The second Designatable_Item is that of the Context itself.

C.2.3 Common Definitional attributes

C.2.3.1 Definition

C.2.3.1.1 Attribute Mapping

Table 15 – Attribute mapping for 'Definition'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	Definition	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* class provides the *text* for a *Designatable_Item* (7.3.2.2) as it applies in zero, one or more *Contexts* (7.3.2.5). *Definition.text:* text of the definition.	The definition of an metadata item within a context.
Obligation:	Mandatory	Mandatory	Mandatory
Datatype:	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 "Definition_Context. acceptability" to "preferred".	

C.2.3.1.2 Path from Designatable_Item

Designatable_Item.*item_definition*.Definition.text.

C.2.3.2 Definition language

C.2.3.2.1 Attribute Mapping

Table 16 – Attribute mapping for 'Definition language'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	Not supported.	Definition.language	definition language
Definition:	N/A	Language used to write the definition text.	Language used to write the definition text.
Obligation:	N/A	Optional	Optional
Datatype:	N/A	String	String

C.2.3.2.2 Path from Designatable_Item

Designatable_Item.*item_definition*.Definition.language

C.2.3.3 Definition source reference

C.2.3.3.1 Attribute Mapping

Table 17 – Attribute mapping for 'Definition source reference'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	Not supported.	Definition.source	definition source reference
Definition:	N/A	A reference to the source from which the definition text is taken.	A reference to the source from which the definition text is taken.
Obligation:	N/A	Optional	Optional
Datatype:	N/A	String	String

C.2.3.3.2 Path from Designatable_Item

Designatable_Item.*item_definition*.Definition.source

C.2.4 Common Administrative attributes

C.2.4.1 Comments

C.2.4.1.1 Attribute Mapping

Table 18 – Attribute mapping for 'Comments'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
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
Datatype:	Character string	String	String

C.2.4.1.2 Path from Administered_Item

Administered_Item.explanatory_comment.

C.2.4.2 Registration status

C.2.4.2.1 Attribute Mapping

Table 19 – Attribute mapping for 'Registration status'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	Registration status	Registration_State. 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.

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Obligation:	Conditional	Mandatory	Optional
Condition:	This attribute is mandatory during the data element life-cycle specified by any Registration_Authority.	N/A	N/A
Datatype:	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).		

C.2.4.2.2 Path from Administered_Item

Administered_Item.*Registration*.registration_state.Registration_State.registration_status.

C.2.4.3 Responsible organization

C.2.4.3.1 Attribute Mapping

Table 20 – Attribute mapping for 'Responsible organization'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	Responsible 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 association of an *Administered_Item* (8.1.2.2) to a *Stewardship_Record* (8.1.2.7), which records the stewardship *organization* (8.1.2.7.2.1) of type *Organization* (8.1.3.2) and *contact* (8.1.2.7.2.2) of type *Contact* (8.1.3.1) involved in the stewardship of the *Administered_Item*.	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
Datatype:	Character string	String	String
Comment:	The organization shall be considered as 'owner' of the data element.		

C.2.4.3.2 Path from Administered_Item

Administered_Item.*stewardship*.Stewardship_Record.steward.Organization.name.

C.2.4.4 Submitting organization

C.2.4.4.1 Attribute Mapping

Table 21 – Attribute mapping for 'Submitting organization'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	Submitting 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 association of a *Registered_Item* (8.1.2.1) with a *Submission_Record* (8.1.2.8), which records the submission *organization* (8.1.2.8.2.1) of type *Organization* (8.1.3.2) and *contact* (8.1.2.8.2.2) of type *Contact* (8.1.3.1) involved in the *submission* of the *Registered_Item*.	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
Datatype:	Character string	String	String

C.2.4.4.2 Path from Registered_Item

Registered_Item.*submission*.Submission_Record.submitter.Organization.name

C.2.5 Common Relational attributes

C.2.5.1 Classification scheme name

C.2.5.1.1 Attribute Mapping

Table 22 – Attribute mapping for 'Classification scheme name'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	Classification scheme	Designation.sign NOTE The Designatable_ Item referred to here is the Concept_System which models the Classification_ Scheme, not the Classifiable_ 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	*classification scheme:* The descriptive information for an arrangement or division of objects into groups based on	The name of a particular arrangement or division of objects into groups based on characteristics which the

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
	characteristics that the objects have in common, e.g. origin, composition, structure, application, function etc.	characteristics which the objects have in common. *item_designation*: Representation of a Designatable_Item by a sign which denotes it.	objects have in common.
Obligation:	Optional	Conditional	Conditional
Condition:	N/A	If a Concept_System is used to model a classification scheme, its name is mandatory.	If a classification scheme is used, its name is mandatory.
Datatype:	Character string	String	String
Comment	The definition does not specify whether the reference is by name or identifier.		

C.2.5.1.2 Path from Classifiable_Item

Classifiable_Item.*Classification*.*classification_scheme*.Concept_System.
Designatable_Item.*item_designation*.Designation.sign.

NOTE The Classifiable_Item is the one which is being classified. The Designatable_Item is that of the Concept_System which models the classification scheme.

C.2.5.2 Classification scheme identifier

C.2.5.2.1 Attribute Mapping

Table 23 – Attribute mapping for 'Classification scheme identifier'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	Classification scheme	Scoped_Identifier.identifier where the Identified_Item is the Concept_System that models the classification scheme.	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. *identifier: String* used to unambiguously denote an *Identified_Item* within the scope of a specified *Namespace.* NOTE The Identified_Item here is the Concept_System that models the classification scheme.	The identifier 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 Concept_System is used to model a classification scheme, its identifier is mandatory.	Required if *classification scheme name* is not unique within a

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
			context.
Datatype:	Character string	String	String
Comment	The definition does not specify whether the reference is by name or identifier.		

C.2.5.2.2 Path from Classifiable_Item

Classifiable_Item.*Classification*.*classification_scheme*.Concept_System.
Identified_Item.*identification*.Scoped_Identifier.identifier.

NOTE The Classifiable_Item is the one which is being classified. The Identified_Item is the superclass of the Concept_System which models the classification scheme.

C.2.5.3 Classification scheme item value

C.2.5.3.1 Attribute Mapping

Table 24 – Attribute mapping for 'Classification scheme item value'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	Keyword	Designation.sign NOTE The Designatable_ Item referred to here is the Concept which models the Classification_Scheme_Item, not the Classifiable_Item which is being classified.	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
Condition	N/a		*One for each* classification scheme item *by which a metadata item is classified.*
Datatype:	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 concept system, with individual keywords being represented as concepts.	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.

C.2.5.3.2 Path from Classified_Item:

Classifiable_Item.*Classification*.Concept.Designated_Item.*item_designation*.Designation.sign.

NOTE The Classifiable_Item is the one which is being classified. The Designatable_Item is that of the Concept which models the classification scheme item.

C.2.5.4 Related metadata reference

C.2.5.4.1 Attribute Mapping

Table 25 – Attribute mapping for 'Related metadata reference'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	Related data reference	Registration_State.administrative_note OR Administered_Item.explanatory_comment OR 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* *explanatory 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
Datatype:	Character string	String	String
Comment:	If this attribute occurs it shall be specified in pair with the attribute 'Type of relationship'		

C.2.5.4.2 Path from Administered_Item:

For "administrative_note":

Administered_Item.*Registration*.registration_state.administrative_note

For "explanatory_comment":

Administered_Item.explanatory_comment.

For "reference_document_identifier":

Administered_Item.Registered_Item.*Reference*.Reference_Document.dentifier

C.2.5.5 Type of relationship

C.2.5.5.1 Attribute Mapping

Table 26 – Attribute mapping for 'Type of relationship'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	Type of relationship	Reference_Document.type_description	type of relationship

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
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 and Optional	Conditional
Condition:	This attribute is mandatory if the attribute 'related data reference' occurs.	"type_description" is optional if Reference_Document is used, and not applicable otherwise.	This attribute is mandatory if the attribute 'related metadata reference' occurs.
Datatype:	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.2.5.4 Related metadata reference.	

C.2.5.5.2 Path from Administered_Item:

Administered_Item.Registered_Item.*Reference*.Reference_Document.type_description

C.2.6 Attributes specific to Data_Element_Concepts

C.2.6.1 Object class name

C.2.6.1.1 Attribute Mapping

Table 27 – Attribute mapping for 'Object class name'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	Not supported.	Designation.*sign* for the Object_Class.	object class name
Definition:	N/A	*Object_Class: 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. *designation:* representation of a *concept* by a *sign* which denotes it	The designation of an *object class* for a *data element concept*.
Obligation:	N/A	Optional	Optional
Datatype:	N/A	String	String

C.2.6.1.2 Path from Data_Element_Concept

Data_Element_Concept.*data_element_concept_object_class*.Object_Class.
Designatable_Item.*item_designation*.Designation.*sign*.

C.2.6.2 Object class identifier

C.2.6.2.1 Attribute Mapping

Table 28 – Attribute mapping for 'Object class identifier'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	Not supported.	Scoped_Identifier.identifier for the Object_Class.	object class identifier
Definition:	N/A	*Object_Class: 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. *Scoped_Identifier.identifier: String* used to unambiguously denote an *Identified_Item* within the scope of a specified *Namespace.*	The identifier of an *object class* for a *data element concept*.
Obligation:	N/A	Optional	Optional
Datatype:	N/A	String	String

C.2.6.2.2 Path from Data_Element_Concept

Data_Element_Concept.*data_element_concept_object_ class*.Object_Class.
Identified_Item.*identification*.Scoped_Identifier.identifier.

C.2.6.3 Property name

C.2.6.3.1 Attribute Mapping

Table 29 – Attribute mapping for 'Property name'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	Not supported.	Designation.*sign* for the Property.	property name
Definition:	N/A	*property*: quality common to all members of an *object class.* *designation:* representation of a *concept* by a *sign* which denotes it	The designation of a *property* for a *data element concept*.
Obligation:	N/A	Optional	Optional
Datatype:	N/A	String	String

C.2.6.3.2 Path from Data_Element_Concept

Data_Element_Concept.*data_element_concept_property*.Property.
Designatable_Item.*item_designation*.Designation.sign.

C.2.6.4 Property identifier

C.2.6.4.1 Attribute Mapping

Table 30 – Attribute mapping for 'Property identifier'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	Not supported.	Scoped_Identifier.identifier for the Property.	property identifier
Definition:	N/A	*property*: quality common to all members of an *object class*. *Scoped_Identifier.identifier: String* used to unambiguously denote an *Identified_Item* within the scope of a specified *Namespace.*	The identifier of a *property* for a *data element concept*.
Obligation:	N/A	Optional	Optional
Datatype:	N/A	String	String

C.2.6.4.2 Path from Data_Element_Concept

Data_Element_Concept.*data_element_concept_property*.Property.
Identified_Item.*identification*.Scoped_Identifier.identifier.

C.2.7 Attributes specific to Data_Elements

C.2.7.1 Representation category

C.2.7.1.1 Attribute Mapping

Table 31 – Attribute mapping for 'Representation category'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
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
Datatype:	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		

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
	registration no. 143) — bar coded representation (EIA-556) — graphical representation		

C.2.7.2 Representation class

C.2.7.2.1 Attribute Mapping

Table 32 – Attribute mapping for 'Representation class'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	Form of representation	Designation.sign for a Concept that represents a classification scheme item with a concept system that represent the Representation Class classification scheme.	Not supported.
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. *designation:* representation of a *concept* by a *sign* which denotes it	N/a
Obligation:	Mandatory	Optional	N/a
Datatype:	Character string	String	N/a
Comment:	1. See ISO/IEC 11179-5:1995 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'.	-	-

C.2.7.2.2 Path from Data_Element

Data_Element.Classifiable_Item.C*lassification*.Concept.
Designatable_Item.*item_designation*. Designation.sign.

C.2.7.3 Value domain name

C.2.7.3.1 Attribute Mapping

Table 33 – Attribute mapping for 'Value domain name'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	Not directly supported.	Designation.sign for 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. *designation:* representation of a *concept* by a *sign* which denotes it	The name of the value domain that provides representation for the data element.
Obligation:	N/A	Mandatory	Optional
Datatype:	N/A	String	String
Comment:	The closest equivalent is "permissible data element values" (see F.2.6.10), but this actually represents the values.	-	-

C.2.7.3.2 Path from Data_Element

Data_Element.*data_element_domain*.Value_Domain.
Designatable_Item.*item_designation*. Designation.sign.

C.2.7.4 Value domain identifier

C.2.7.4.1 Attribute Mapping

Table 34 – Attribute mapping for 'Value domain identifier'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	Not directly supported.	Scoped_Identifier.identifier for the Value_Domain.	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. *Scoped_Identifier.identifier: String* used to unambiguously denote an *Identified_Item* within the scope of a specified *Namespace.*	The identifier of the value domain that provides representation for the data element.
Obligation:	N/A	Mandatory	Optional
Datatype:	N/A	String	String
Comment:	The closest equivalent is		

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
	"permissible data element values" (see F.2.6.10), but this actually represents the values.		

C.2.7.4.2 Path from Data_Element

Data_Element. *data_element_domain*.Value_Domain.
Identified_Item.*identification*.Scoped_Identifier.identifier.

C.2.7.5 Datatype name

C.2.7.5.1 Attribute Mapping

Table 35 – Attribute mapping for 'Datatype name'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	Datatype of data element values	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.
Datatype:	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.	

C.2.7.5.2 Path from Data_Element

Data_Element. *data_element_domain*.Value_Domain.value_domain_datatype.Datatype.name

C.2.7.6 Datatype scheme reference

C.2.7.6.1 Attribute Mapping

Table 36 – Attribute mapping for 'Datatype scheme reference'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	Not supported.	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.
Datatype:	N/A	String	String
Comment:		In the metamodel, the datatype is an attribute of the value domain, not directly of the data element.	

C.2.7.6.2 Path from Data_Element

Data_Element.*data_element_domain*.Value_Domain.value_domain_datatype.Datatype.scheme_reference.

C.2.7.7 Maximum size

C.2.7.7.1 Attribute Mapping

Table 37 – Attribute mapping for 'Maximum size'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	Maximum size of data element values	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
Datatype:	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.	This is not exactly equivalent, because it applies only to character datatypes.	

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
	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).		

C.2.7.7.2 Path from Data_Element

Data_Element.*data_element_domain*.Value_Domain.maximum_character_quantity.

C.2.7.8 Minimum size

C.2.7.8.1 Attribute Mapping

Table 38 – Attribute mapping for 'Minimum size'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
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
Datatype:	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).		

C.2.7.9 Layout of representation

C.2.7.9.1 Attribute Mapping

Table 39 – Attribute mapping for 'Layout of representation'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	Layout of representation	Value_Domain.format	layout of representation
Definition:	The layout of characters in data element values expressed by a character string representation.	template for the structure of the presentation of the **value**(s)	The layout of characters in data element values expressed by a character string representation.
Obligation:	Conditional	Optional	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.	N/A	N/A
Datatype:	Character string	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'.	-	-

C.2.7.10 Permissible data element values

C.2.7.10.1 Attribute Mapping

Table 40 – Attribute mapping for 'Permissible data element values'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
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
Datatype:	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.		

C.2.8 Attributes specific to Conceptual_Domains

C.2.8.1 Dimensionality

C.2.8.1.1 Attribute Mapping

Table 41 – Attribute mapping for 'Dimensionality'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
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
Datatype:	N/A	String	String
Comment:		For example, length, mass, velocity, currency.	For example, length, mass, velocity, currency.

C.2.9 Attributes specific to Value_Domains

C.2.9.1 Datatype name

C.2.9.1.1 Attribute Mapping

Table 42 – Attribute mapping for 'Datatype name'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	See "Datatype of data element values" (F.2.6.5)	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
Datatype:		String	String

C.2.9.1.2 Path from Value_Domain

Value_Domain.value_domain_datatype.Datatype.name

C.2.9.2 Datatype scheme reference

C.2.9.2.1 Attribute Mapping

Table 43 – Attribute mapping for 'Datatype scheme reference'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	Not supported.	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
Datatype:		String	String

C.2.9.2.2 Path from Value_Domain

Value_Domain.value_domain_datatype.Datatype.scheme_reference

C.2.9.3 Unit of measure name

C.2.9.3.1 Attribute Mapping

Table 44 – Attribute mapping for 'Unit of measure name'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	Not supported.	Designation.sign for the Unit_of_Measure.	unit of measure name
Definition:		*Unit_of_Measure*: the units in which any associated *Data_Element* values are specified. *designation:* representation of a *concept* by a *sign* which denotes it	The name of a unit of measure.
Obligation:		Optional	Optional
Datatype:		String	String

C.2.9.3.2 Path from Value_Domain

Value_Domain.value_domain_unit_of_measure.Unit_of_Measure.
Designatable_Item.*item_designation*.Designation.sign

C.2.10 Attributes specific to Permissible_Values

C.2.10.1 Permitted value

C.2.10.1.1 Attribute Mapping

Table 45 – Attribute mapping for 'Value'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	See "permissible data element values" (F.2.6.10)	Permissible_Value. permitted_value	permitted 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
Datatype:	N/A	String	String
Comment:			

C.2.10.2 Permissible value begin date

C.2.10.2.1 Attribute Mapping

Table 46 – Attribute mapping for 'Permissible value begin date'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	Not supported.	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
Datatype:	N/A	Date	Date

C.2.10.3 Permissible value end date

C.2.10.3.1 Attribute Mapping

Table 47 – Attribute mapping for 'Permissible value end date'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	Not supported.	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
Datatype:	N/A	Date	Date

C.2.11 Attributes specific to Value_Meanings

C.2.11.1 Value meaning description

C.2.11.1.1 Attribute Mapping

Table 48 – Attribute mapping for 'Value meaning description'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	Not supported.	Definition.text for the Value_Meaning.	value meaning description
Definition:	N/A	A description of a value meaning.	A description of a value meaning.
Obligation:	N/A	Mandatory	Mandatory
Datatype:	N/A	String	String

C.2.11.1.2 Path from Value_Meaning

Value_Meaning.Designatable_Item.*item_definition*.Definition.text

C.2.11.2 Value meaning identifier

C.2.11.2.1 Attribute Mapping

Table 49 – Attribute mapping for 'Value meaning identifier'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	Not supported.	Scoped_Identifier.identifier for the Value_Meaning.	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
Datatype:	N/A	String	String

C.2.11.2.2 Path from Value_Meaning

Value_Meaning.Identified_Item.*identification*.Scoped_Identifier.identifier

C.2.11.3 Value meaning begin date

C.2.11.3.1 Attribute Mapping

Table 50 – Attribute mapping for 'Value meaning begin date'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	Not supported.	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
Datatype:	N/A	Date	Date

C.2.11.4 Value meaning end date

C.2.11.4.1 Attribute Mapping

Table 51 – Attribute mapping for 'Value meaning end date'

	1994 Clause 6	2011 Metamodel	2011 Basic Attributes
Attribute name:	Not supported.	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
Datatype:	N/A	Date	Date

Annex D
(informative)

Mapping the ISO/IEC 11179-3:2003 metamodel to the ISO/IEC 11179-3:2011 metamodel

D.1 Introduction

This Annex explains how the Edition 2 metamodel relates to the Edition 3 metamodel.

D.2 Mapping the Edition 2 Administration and Identification Region

D.2.1 Administered_Item

Table 52 – Mapping Edition 2 Administered_Item to Edition 3

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.8.1.1	Administered Item	8.1.2.2	Administered_Item
4.8.1.1	Administered Item. administration record	n/a	The former administration_record has been split between attributes directly on Administered_Item, and others moved to Registration_State, used by Registration.registration_state. See mapping of Administration_Record below.
Figure 4	reference association	8.1.5.2	Reference association class
Figure 4	registration association	8.1.5.1	Registration association class

D.2.2 Administration_Record

Table 53 – Mapping Edition 2 Administration_Record to Edition 3

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.8.1.2	Administration Record	n/a	The former Administration_Record has been split between attributes directly on Administered_Item, and others moved to Registration_State, used by Registration.registration_state, as listed below.
4.8.1.2	administered item identifier	n/a	The former administered_item_identifier has been split into its component parts. See mapping of Item_Identifier below.
4.8.1.2	administrative note	8.1.2.6.2.4	Registration_State.administrative_note
4.8.1.2	administrative status	8.1.2.6.2.6	Registration_State.administrative_status
4.8.1.2	change description	8.1.2.2.3.3	Administered_Item.change_description
4.8.1.2	creation date	8.1.2.2.3.1	Administered_Item.creation_date
4.8.1.2	effective date	8.1.2.6.2.2	Registration_State.effective_date
4.8.1.2	explanatory comment	8.1.2.2.3.4	Administered_Item.explanatory_comment
4.8.1.2	last change date	8.1.2.2.3.2	Administered_Item.last_change_date

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.8.1.2	origin	8.1.2.2.3.5	Administered_Item.origin
4.8.1.2	registration status	8.1.2.6.2.1	Registration_State.registration_status
4.8.1.2	unresolved issue	8.1.2.6.2.5	Registration_State.unresolved_issue
4.8.1.2	until date	8.1.2.6.2.3	Registration_State.until_date

D.2.3 Contact

Table 54 – Mapping Edition 2 Contact to Edition 3

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.8.1.3	Contact	6.3.2	Contact
4.8.1.3	contact information	6.3.2.2.1 6.3.2.2.2 6.3.2.2.3	Split into: - individual - organization - role
4.8.1.3	contact name	6.3.4.2.1	Contact.individual.Individual.name
4.8.1.3	contact title	6.3.4.2.2	Contact.individual.Individual.title

D.2.4 Item_Identifier

Table 55 – Mapping Edition 2 Item_Identifier to Edition 3

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.8.1.4	Item Identifier	7.2.2.2	Scoped_Identifier
4.8.1.4	item registration authority identifier	8.1.2.5.3.1	Registration_Authority. registration_authority_identifier accessed via the Registration association from Administered_Item.
4.8.1.4	data identifier	7.2.2.2.2.1	Scoped_Identifier.identifier
4.8.1.4	version	7.2.2.2.2.2	Scoped_Identifier.version

D.2.5 Language_Identification

Table 56 – Mapping Edition 2 Language_Identification to Edition 3

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.8.1.5	Language Identification	6.3.5	Language_Identification
4.8.1.5	language identifier	6.3.5.2.1	language_identifier
4.8.1.5	country identifier	6.3.5.2.3	geopolitical_territory_identifier

D.2.6 Organization

Table 57 – Mapping Edition 2 Organization to Edition 3

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.8.1.6	Organization	6.3.6	Organization
4.8.1.6	organization name	6.3.6.2.1	Organization.name
4.8.1.6	organization mail address	6.3.6.2.2	Organization.mail_address

D.2.7 Reference_Document

Table 58 – Mapping Edition 2 Reference_Document to Edition 3

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.8.1.7	Reference Document	6.3.7	Reference_Document
4.8.1.7	reference document identifier	6.3.7.2.1	Reference_Document.identifier
4.8.1.7	reference document language identifier	6.3.7.2.3	Reference_Document.language_identifier
4.8.1.7	reference document title	6.3.7.2.5	Reference_Document.title
4.8.1.7	reference document type description	6.3.7.2.2	Reference_Document.type_description

D.2.8 Registrar

Table 59 – Mapping Edition 2 Registrar to Edition 3

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.8.1.8	Registrar	8.1.2.4	Registrar
4.8.1.8	registrar identifier	8.1.2.4.3.1	Registrar.identifier
4.8.1.8	registrar contact	6.3.2	Contact. In Edition 3, Registrar is a subclass of Contact

D.2.9 Registration_Authority

Table 60 – Mapping Edition 2 Registration_Authority to Edition 3

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.8.1.9	Registration Authority	8.1.2.5	Registration_Authority
4.8.1.9	registration authority identifier	8.1.2.5.3.1	registration_authority_identifier
4.8.1.9	documentation language identifier	8.1.2.5.3.2	documentation_language_identifier
Figure 4	registration_authority_registrar association	8.1.6.3	registration_authority_registrar association

D.2.10 Registration_Authority_Identifier

Table 61 – Mapping Edition 2 Registration_Authority_Identifier to Edition 3

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.8.1.10	Registration Authority Identifier	6.3.8	Registration_Authority_Identifier
4.8.1.10	international code designator	6.3.8.2.1	international_code_designator
4.8.1.10	organization identifier	6.3.8.2.2	organization_identifier
4.8.1.10	organization part identifier (OPI)	6.3.8.2.3	organization_part_identifier (OPI)
4.8.1.10	OPI source	6.3.8.2.4	opi_source

D.2.11 Stewardship

Table 62 – Mapping Edition 2 Stewardship to Edition 3

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.8.1.11	Stewardship association class	8.1.2.7 8.1.6.4	Stewardship_Record class and stewardship association
4.8.1.11	stewardship contact	8.1.2.7.2.2	Stewardship_Record.contact

D.2.12 Submission

Table 63 – Mapping Edition 2 Submission to Edition 3

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.8.1.12	Submission association class	8.1.2.8 8.1.6.5	Submission_Record class and submission association
4.8.1.12	submission contact	8.1.2.8.2.2	Submission_Record.contact

D.3 Mapping the Edition 2 Naming and Definition Region

D.3.1 Context (for Administered_Item)

Table 64 – Mapping Edition 2 Context to Edition 3

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.9.1.2	Context (for Administered_Item)	7.3.2.5	Context
4.9.1.2	context description	n/a	Removed. Instead see the associated Definition.text
4.9.1.2	context description language identifier	n/a	Removed. Instead see the associated Definition.language
Figure 6	Administered_item_context association	7.3.3.1 7.3.3.2	Definition_Context association class and Designation_Context association class

D.3.2 Terminological_Entry

Table 65 – Mapping Edition 2 Terminological_Entry to Edition 3

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.9.1.3	Terminological Entry	n/a	Removed. It can be derived by selecting all Definitions (7.3.2.4) and Designations (7.3.2.3) for a particular Designatable_Item (7.3.2.2) and Context (7.3.2.5).
Figure 6	terminological_entry_languages association	7.3.2.4.2 7.3.2.3.2	Removed. Effectively replaced by Definition.language and Designation.language.

D.3.3 Language_Section

Table 66 – Mapping Edition 2 Language_Section to Edition 3

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.9.1.4	Language Section	n/a	Removed. It can be derived by selecting all Definitions (7.3.2.4) and Designations (7.3.2.3) for a particular Designatable_Item (6.2.2.2) and Context (7.3.2.5) for a particular language as specified by Definition.language (7.3.2.4.2.2) and Designation.language (7.3.2.3.2.2).
Figure 6	name_entry association	7.3.2.3.2	Removed. Effectively replaced by Designation.language.
Figure 6	definition_entry association	7.3.2.4.2	Removed. Effectively replaced by Definition.language.

D.3.4 Definition (of Administered_Item)

Table 67 – Mapping Edition 2 Definition (of Administered_Item) to Edition 3

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.9.1.5	Definition (of Administered_Item)	7.3.2.4	Definition
4.9.1.5	definition text	7.3.2.4.2.1	Definition.text
4.9.1.5	definition source reference	7.3.2.4.2.3	Definition.source
4.9.1.5	preferred definition (boolean)	7.3.3.1.2.1	Definition_Context.acceptability ('preferred' is just one of several possible ratings)

D.3.5 Designation (of Administered_Item)

Table 68 – Mapping Edition 2 Designation (of Administered_Item) to Edition 3

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.9.1.6	Designation (of Administered_Item)	7.3.2.3	Designation
4.9.1.6	name	7.3.2.3.2.1	Designation.sign
4.9.1.6	preferred designation (boolean)	7.3.3.2.2.1	Designation_Context.acceptability ('preferred' is just one of several possible ratings)
Figure 6	term_definition_pairing association	7.3.4.1	designation_definition_pairing association

D.4 Mapping the Edition 2 Classification Region

D.4.1 Classification_Scheme

Table 69 – Mapping Edition 2 Classification_Scheme to Edition 3

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.10.1.2	Classification Scheme	9.2.2.2	Use Concept_System to model a Classification_Scheme.
4.10.1.2	classification scheme type name	n/a	Not explicitly supported. If required, the Concept_System could itself be classified.
Figure 7	classification_scheme_membership association	9.1.3.1	concept_system_membership association

D.4.2 Classification_Scheme_Item

Table 70 – Mapping Edition 2 Classification_Scheme_Item to Edition 3

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.10.1.3	Classification Scheme Item	9.2.2.3	Use Concept to model a Classification_ Scheme_Item
4.10.1.3	classification scheme item type name	n/a	Not explicitly supported. If required, use Slot to extend the Concept.
4.10.1.3	classification scheme item value	n/a	Use the Designation associated with the Concept.
Figure 7	administered_item_classification association	9.2.3.1	Classification association class

D.4.3 Classification_Scheme_Item_Relationship

Table 71 – Mapping Edition 2 Classification_Scheme_Item_Relationship to Edition 3

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.10.1.4	Classification Scheme Item Relationship	9.1.2.4	Relation
4.10.1.4	classification scheme item relationship type description	n/a	Use the Definition of the associated Relation

D.5 Mapping the Edition 2 Data_Element_Concept Region

D.5.1 Object_Class

Table 72 – Mapping Edition 2 Object_Class to Edition 3

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.11.1.1	Object Class NOTE: In Edition 2, Object_Class is a superclass of Concept and Concept_Relationship	11.2.2.1	Object_Class NOTE: In Edition 3, Concept is a superclass of Object_Class and Relation (and others).
4.11.1.1	concept relationship type description	n/a	Use the Definition of the associated Relation

D.5.2 Property

Table 73 – Mapping Edition 2 Property to Edition 3

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.11.1.2	Property	11.2.2.2	Property

D.5.3 Data_Element_Concept

Table 74 – Mapping Edition 2 Property to Edition 3

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.11.1.3	Data Element Concept	11.2.2.3	Data_Element_Concept
4.11.1.3	data element concept object class	11.2.3.3	data_element_concept_object_class (association)
4.11.1.3	data element concept property	11.2.3.1	data_element_concept_property (association)
4.11.1.3	object class qualifier	n/a	Removed. The qualifier was a feature of naming. It should be part of the associated Designation
4.11.1.3	property qualifier	n/a	Removed. The qualifier was a feature of naming. It should be part of the associated Designation
Figure 8	Data_Element_Concept_Relationship association class	9.1.2.4	Relation. Data_Element_Concept is a subclass of Concept, and as such participates in the latter's associations.
Figure 8	data_element_concept_conceptual_domain_ relationship association	11.2.3.2	data_element_concept_domain association

D.5.4 Concept_Relationship

Table 75 – Mapping Edition 2 Property to Edition 3

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.11.1.1	Concept Relationship	9.1.2.4	Relation

D.6 Mapping the Edition 2 Conceptual and Value Domain Region

D.6.1 Conceptual_Domain

Table 76 – Mapping Edition 2 Conceptual_Domain to Edition 3

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.12.1.1	Conceptual Domain	11.3.2.1	Conceptual_Domain
4.12.1.1	dimensionality	11.3.2.1.3.1	dimensionality
Figure 9	Conceptual_Domain_Relationship association class	9.1.2.4	Relation. Conceptual_Domain is a subclass of Concept, and as such participates in the latter's associations.
Figure 9	conceptual_domain_representation association	11.3.3.1	value_domain_meaning association [Renamed because a Conceptual Domain can be defined without a Value Domain, but a Value Domain must always have an associated Conceptual Domain.]

D.6.2 Enumerated_Conceptual_Domain

Table 77 – Mapping Edition 2 Enumerated_Conceptual_Domain to Edition 3

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.12.1.2	Enumerated Conceptual Domain	11.3.2.2	Enumerated_Conceptual_Domain
Figure 9	value_meaning_set association	11.3.3.2	value_meaning_set association

D.6.3 Value_Meaning

Table 78 – Mapping Edition 2 Value_Meaning to Edition 3

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.12.1.3	Value Meaning	11.3.2.3	Value_Meaning
4.12.1.3	value meaning identifier	n/a	Use Identified_Item and Scoped_Identifier
4.12.1.3	value meaning begin date	11.3.2.3.3.1	Value_Meaning.begin_date
4.12.1.3	value meaning description	n/a	Use Designatable_Item and Definition
4.12.1.3	value meaning end date	11.3.2.3.3.2	Value_Meaning.end_date

D.6.4 Non-enumerated_Conceptual_Domain

Table 79 – Mapping Edition 2 Non-enumerated_Conceptual_Domain to Edition 3

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.12.1.4	Non-enumerated Conceptual Domain	11.3.2.4	Described_Conceptual_Domain [Renamed because a term should express what something is, not what it is not.]
4.12.1.4	non-enumerated conceptual domain description	11.3.2.4.3.1	Described_Conceptual_Domain.description
Figure 9	non-enumerated_conceptual_ domain_representation association	11.3.3.3	described_value_domain_meaning association [Renamed because a Conceptual Domain can be defined without a Value Domain, but a Value Domain must always have an associated Conceptual Domain.]

D.6.5 Value_Domain

Table 80 – Mapping Edition 2 Value_Domain to Edition 3

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.12.1.5	Value Domain	11.3.2.5	Value_Domain
4.12.1.5	value domain datatype	11.3.2.5.2.1	Value_Domain.datatype
4.12.1.5	value domain format	11.3.2.5.2.2	Value_Domain.format
4.12.1.5	value domain maximum character quantity	11.3.2.5.2.3	Value_Domain.maximum_character_quantity
4.12.1.5	value domain unit of measure	11.3.2.5.2.4	Value_Domain.unit_of_measure
Figure 9	value_domain_relationship association	11.3.3.6	value_domain_subset association
Figures 9 & 10	value_domain_representation_class association	9.2.3.1	Classification association class. Representation_class is now viewed as just a form of classification, which is represented as a Concept within a Concept_System.

D.6.6 Enumerated_Value_Domain

Table 81 – Mapping Edition 2 Enumerated_Value_Domain to Edition 3

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.12.1.6	Enumerated Value Domain	11.3.2.6	Enumerated_Value_Domain

D.6.7 Permissible_Value

Table 82 – Mapping Edition 2 Permissible_Value to Edition 3

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.12.1.7	Permissible Value	11.3.2.7	Permissible_Value
4.12.1.7	permissible value begin date	11.3.2.7.2.2	Permissible_Value.begin_date
4.12.1.7	permissible value end date	11.3.2.7.2.3	Permissible_Value.end_date
Figure 9	permissible_value_meaning association	11.3.3.4	permissible_value_meaning association
Figure 9	permissible_value_set association	11.3.3.5	permissible_value_set association
Figure 9	permitted_value association	11.3.2.7.2.1	Permissible_Value.permitted_value

D.6.8 Value

Table 83 – Mapping Edition 2 Value to Edition 3

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.12.1.8	Value	n/a	Collapsed into Permissible_Value
4.12.1.8	value item	11.3.2.7.2.1	Permissible_Value.permitted_value

D.6.9 Non-enumerated_Value_Domain

Table 84 – Mapping Edition 2 Non-enumerated_Value_Domain to Edition 3

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.12.1.9	Non-enumerated Value Domain	11.3.2.8	Described_Value_Domain
4.12.1.9	non-enumerated value domain description	11.3.2.8.3.1	Described_Value_Domain.description

D.6.10 Datatype

Table 85 – Mapping Edition 2 Datatype to Edition 3

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.12.1.10	Datatype	11.3.2.9	Datatype
4.12.1.10	datatype name	11.3.2.9.2.1	Datatype.name
4.12.1.10	datatype description	11.3.2.9.2.2	Datatype.description
4.12.1.10	datatype scheme reference	11.3.2.9.2.3	Datatype.scheme_reference
4.12.1.10	datatype annotation	11.3.2.9.2.4	Datatype.annotation

D.6.11 Unit_of_Measure

Table 86 – Mapping Edition 2 Unit_of_Measure to Edition 3

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.12.1.11	Unit of Measure	11.4.2.1	Unit_of_Measure
4.12.1.11	unit of measure name	n/a	Use Designatable_Item and Designation
4.12.1.11	unit of measure precision	n/a	Removed. Use Data_Element.precision

D.7 Mapping the Edition 2 Data_Element Region

D.7.1 Data_Element

Table 87 – Mapping Edition 2 Data_Element to Edition 3

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.13.1.1	Data_Element	11.5.2.1	Data_Element
4.13.1.1	data element precision	11.5.2.1.2.1	Data_Element.precision
4.13.1.1	representation class qualifier	n/a	Removed. The qualifier is a feature of naming, and the Representation_Class is now a classification_scheme_item modelled as a Concept. Use the Designation of the Designatable_Item that is the Concept.
Figure 3 and Figure 10	data_element_concept_expression association	11.5.3.2	data_element_meaning association Renamed because a Data Element Concept can be registered without an associated Data Element, but a Data Element must always have a Data Element Concept.
Figure 3 and Figure 10	data_element_representation association	11.5.3.1	data_element_domain association
Figure 10	data_element_representation_ class association	9.2.3.1	Classification association class. Representation_class is now viewed as just a form of classification, which is represented as a Concept within a Concept_System.
Figure 10	derivation_input association	11.5.3.4	derivation_input association
Figure 10	derivation_output association	11.5.3.5	derivation_output association
Figure 10	derivation_rule_application association	11.5.3.6	derivation_rule_application association
Figure 10	Exemplification association	11.5.3.3	exemplification association

D.7.2 Representation_Class

Table 88 – Mapping Edition 2 Representation_Class to Edition 3

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.13.1.4	Representation Class	n/a	Representation_Class is now viewed as a Classification_Scheme (modelled as a Concept_System) with each classification_scheme_item modelled as a Concept.

See also Annex F.

D.7.3 Data_Element_Example

Table 89 – Mapping Edition 2 Data_Element_Example to Edition 3

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.13.1.5	Data Element Example	11.5.2.4	Data_Element_Example
4.13.1.5	data element example item	11.5.2.4.2.1	Data_Element_Example.example_item

D.7.4 Derivation_Rule

Table 90 – Mapping Edition 2 Derivation_Rule to Edition 3

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.13.1.6	Derivation Rule	11.5.2.5	Derivation_Rule
4.13.1.6	derivation rule specification	11.5.2.5.2.1	Derivation_Rule.specification
n/a	n/a	11.5.2.5.2.2	Derivation_Rule.notation

D.7.5 Data_Element_Derivation

Table 91 – Mapping Edition 2 Data_Element_ Derivation to Edition 3

Edn. 2 clause #	Edition 2 metadata object	Edn. 3 clause #	Edition 3 metadata object
4.13.1.7	Data Element Derivation	11.5.2.6	Data_Element_Derivation

Annex E
(informative)

Concept System Examples

This Annex illustrates the use of the Concepts region (see 9.1).

E.1 Concept System Metamodels

The concept system metamodel specified in 9.1 is very generic, so that it may be used to register concept systems that are defined in a wide range of formalisms. Most formalisms will have some built-in constructs which are not built-in to the metamodel specified in 9.1. The concept system metamodel is generic enough to also support registration of such built-in constructs, in a "notation" concept system. For example, an OWL concept system can be used to define OWL ontological relations such as rdf:type, rdfs:range, and owl:disjointWith.

By registering a full metamodel of each such formalism also as a concept system, the same facility can also be used to describe mappings between them, as well as to the concept system metamodel specified in 9.1. The table below summarizes some suggested primary mappings between this part of ISO/IEC 11179 and a selection of notations. Note that in the RDF-based notations (SKOS and OWL) it is suggested to treat inverse properties as roles of an implicit binary relation. Some further details are called out below.

Table 92 – Correspondences of ISO/IEC 11179-3 concept system metamodel to selected notations

Notation	Concept	Relation	Relation_Role	Link	Assertion
SKOS	Concept	N/a	*semantic relations*	N/a	Statement
OWL	Class *or* Thing	N/a	ObjectProperty	N/a	Statement
UML	Class *or* Object	Association	AssociationEnd	Link	N/a
ORM	non-lexical object *or* non-lexical object type	idea type	role	idea	N/a
XTM	Topic	Association Type	Association Role	Association	N/a
SBVR	concept *or* characteristic	(non-unary) fact type	fact type role	N/a	fact
CLIF	N/a	=	N/a	N/a	Statement

NOTE 1 N/a = Not applicable

NOTE 2 *Relation* and *Relation_Role* are shown separately, even though they are subclasses on *Concept*.

SKOS

There is no metaclass in SKOS for semantic relations, instead there are only the foundational broader, narrower and related properties, and a semantic relation is one defined by those properties or by subproperties of those properties.

OWL

Some OWL built-in constructs are most naturally described as ternary relations, and others as variable arity relations with two roles. It is also reasonable to describe object properties as relations rather than relation roles. See OWL Example below.

UML

Because the metamodel specified in this part of ISO/IEC 11179 does not impose a meta-level hierarchy, the Generalization metaclass in UML can be interpreted as a built-in relation, and generalization relationships thus as links (in the ISO/IEC 11179 sense) between UML Classes.

ORM

There is no official standard for ORM, but in this appendix the ORM metamodel provided in ISO TR 9007 (1987) is taken as normative.

XTM

An XML syntax for Topic Maps (ISO/IEC 13250).

SBVR

It is most natural to treat SBVR characteristics (unary fact types) as concepts in this part of ISO/IEC 11179, rather than as relations, because links in this part of ISO/IEC 11179 are required to have at least two ends. See SBVR Example below.

E.2 SKOS Example

This is a very simple example using SKOS (Simple Knowledge Organization System)[1].

E.2.1 SKOS Metamodel

The core of the SKOS (meta)model[2] contains only two semantic relations[3], defined by three RDF properties. Describing these two semantic relations in terms of the ISO/IEC 11179 metamodel is very straightforward:

Table 93 – SKOS-CORE as an ISO/IEC 11179 Concept System

<Concept_System>			
	notation	**referencedConceptSystem**	**importedConceptSystem**
SKOS-CORE			

[1] See http://www.w3.org/TR/skos-primer/

[2] See http://www.w3.org/TR/skos-primer/#secsimple

[3] See http://www.w3.org/TR/skos-primer/#secrel

Table 94 – SKOS relations as ISO/IEC 11179 Binary Relations

<Binary_Relation>					
	source	**role**	**reflexivity**	**symmetry**	**transitivity**
generalization	SKOS-CORE	skos:broader		antisymmetric	transitive
		skos:narrower			
association	SKOS-CORE	skos:related		symmetric	intransitive

E.2.2 SKOS Example Thesaurus

Our SKOS example is a simple thesaurus of marital statuses. Here is its expression in Turtle[4]:

```
:MaritalStatus rdf:type skos:ConceptScheme .

:Married rdf:type skos:Concept ;
  skos:inScheme :MaritalStatus ;
  skos:related :Single .

:Single rdf:type skos:Concept ;
  skos:inScheme :MaritalStatus ;
  skos:narrower :NeverMarried .

:NeverMarried rdf:type skos:Concept ;
  skos:inScheme :MaritalStatus ;

:Widowed rdf:type skos:Concept ;
  skos:inScheme :MaritalStatus ;
  skos:broader :Single .

:Divorced rdf:type skos:Concept ;
  skos:inScheme :MaritalStatus ;
  skos:broader :Single .
```

Here is a description of the above thesaurus in terms of the draft 11179-3 metamodel:

Table 95 – SKOS Thesaurus Example – ISO/IEC 11179 Concept System

<Concept_System>			
	notation	**referencedConceptSystem**	**importedConceptSystem**
MaritalStatus	SKOS/Turtle	SKOS-CORE	

[4] See http://www.w3.org/TeamSubmission/2008/SUBM-turtle-20080114/

Table 96 – SKOS Thesaurus Example – ISO/IEC 11179 Concepts

<Concept>	
	source
ms:Married	MaritalStatus
ms:Single	MaritalStatus
ms:NeverMarried	MaritalStatus
ms:Widowed	MaritalStatus
ms:Divorced	MaritalStatus

Note In Table 95 and in the following tables, the concepts are represented by their designations.

Table 97 – SKOS Thesaurus Example – ISO/IEC 11179 Links

<Link>			
source	**relation**	**link_end**	
		end_role	**end**
MaritalStatus	association	skos:related	ms:Married
		skos:related	ms:Single
MaritalStatus	generalization	skos:broader	ms:Single
		skos:narrower	ms:NeverMarried
MaritalStatus	generalization	skos:broader	ms:Single
		skos:narrower	ms:Widowed
MaritalStatus	generalization	skos:broader	ms:Single
		skos:narrower	ms:Divorced

E.2.3 Example Value Domain References

To illustrate one possible connection to data description, here is an example pair of enumerated value domains described with references to the above SKOS thesaurus.

NOTE Unused attributes have been omitted from these descriptions.

Table 98 – SKOS Thesaurus Example – ISO/IEC 11179 Conceptual Domains

<Enumerated_Conceptual_Domain>	
	member
binary_ms_CD	ms:Single
	ms:Married
specific_ms_CD	ms:NeverMarried
	ms:Married
	ms:Widowed
	ms:Divorced

Table 99 – SKOS Thesaurus Example – ISO/IEC 11179 Value Domains

<Enumerated_Value_Domain>				
	datatype	**meaning**	**member**	
			value	**meaning**
binary_marital_status	Bit	binary_ms_CD	'0'	ms:Single
			'1'	ms:Married
marital_status_code	Character	specific_ms_CD	'S'	ms:NeverMarried
			'M'	ms:Married
			'W'	ms:Widowed
			'D'	ms:Divorced

E.3 ORM Example

This example uses the Object Role Modelling[5] (ORM) model from ISO TR 9007:1987 Appendix E.

E.3.1 ORM Metamodel

There are two relations built-into ORM which need to be registered first. They are described in TR 9007 using the PASCAL syntax defined in TR 9007 Appendix C. The subtype relation is defined by the declaration:

```
(R5) nolot-subtype = "NOLOT called"
                          (nolot-name-1 | nolot-name-1-list)
                     "is subtype-of NOLOT called" nolot-name-2 ";".
```

The other relation is implicitly defined within the idea-declaration definition:

```
(R7) idea-declaration = "IDEA (with-first ROLE (called" role-name-1
                              "and on NOLOT called" nolot-name-1 ")"
                        "and with-second ROLE (called" role-name-2
                              "and on NOLOT called" nolot-name-2 "))"
                        "is called" idea-name ";".
```

For clarity, we will first rewrite the idea-declaration rule as follows:

```
(R7a) idea-declaration = "IDEA (with-first" idea-role-1
                         "and with-second" idea-role-2 ")"
                         "is called " idea-name ";".
(R7b) idea-role        = "ROLE (called" role-name
                         "and on NOLOT called" nolot-name ")".
(R7c) idea-role-1      = idea-role.
(R7d) idea-role-2      = idea-role.
```

Rule (R7a) corresponds to relation_role_set in the 11179-3 metamodel; rule (R7b) defines a relation which is not built-in to the 11179-3 metamodel. The ORM relations defined by rules (R5) and (R7b) thus may be described in terms of the 11179-3 metamodel as follows:

Table 100 – ORM as an ISO/IEC 11179 Concept System

<Concept_System>			
	notation	**referencedConceptSystem**	**importedConceptSystem**
ORM	PASCAL		

Table 101 – ORM Relations as ISO/IEC 11179 Binary Relations

<Binary_Relation>					
	source	**role**	**reflexivity**	**symmetry**	**transitivity**
subtype	ORM	subtype-of		antisymmetric	transitive
		supertype-of			

[5] see http://www.orm.net/

<Binary_Relation>					
	source	role	reflexivity	symmetry	transitivity
role-on	ORM	on			
		role			

Table 102 – ORM Roles as ISO/IEC 11179 Relation Roles

<Relation_Role>		
	multiplicity	ordinal
subtype-of		1
supertype-of		2
on	1	1
role		2

E.3.2 Car Registration Model

The example model is depicted graphically in Figure 20, reproduced below:

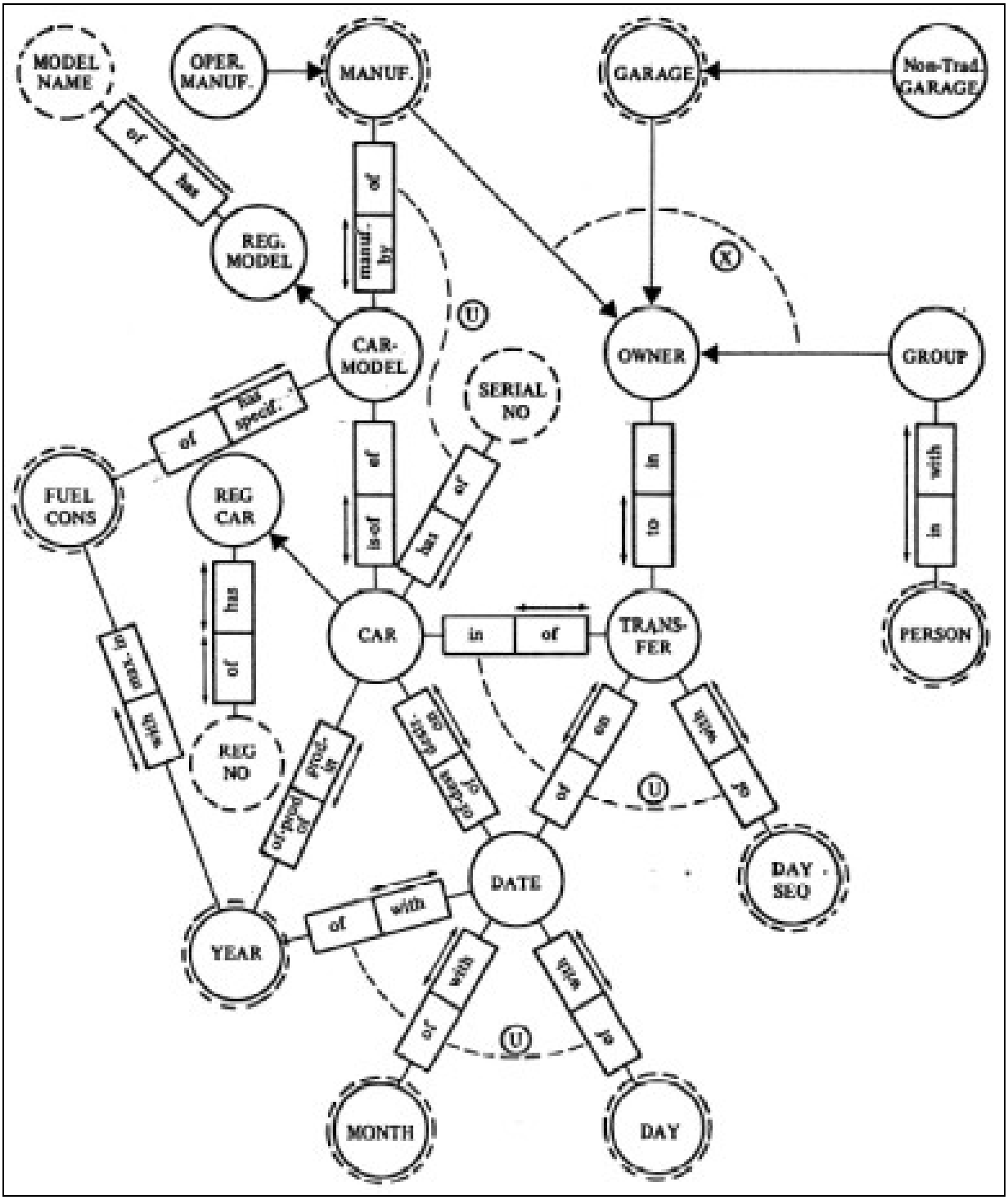

Figure 20— Car Registration Model in ORM

The textual definition of the Car Registration Model is reproduced below:

```
begin
add CONCEPTUAL-SCHEMA called 'CAR-REGISTRATION' ;
add NOLOT called ('MANUFACTURER' 'OPERATING-MANUFACTURER' 'REG-CAR'
                  'CAR' 'REG-MODEL' 'CAR-MODEL' 'FUEL-CONSUMPTION'
                  'DATE' 'YEAR' 'MONTH' 'DAY' 'TRANSFER' 'DAY-SEQ'
                  'OWNER' 'GARAGE' 'NON-TRADING-GARAGE' 'GROUP'
                  'PERSON');
add LOT called {'MANUFACTURER-NAME' 'REG-NO' 'SERIAL-NO' 'MODEL-NAME'
                'FUEL-CONSUMPTION-AMOUNT' 'YEAR-NO' 'MONTH-NO'
                'DAY-NO' 'SEQ-NO' 'GARAGE-NAME' 'PERSON-NAME');
add NOLOT called 'OPERATING-MANUFACTURER'
          is subtype-of NOLOT called 'MANUFACTURER';
add NOLOT called ('MANUFACTURER' 'GARAGE' 'GROUP')
          is subtype-of NOLOT called 'OWNER';

    Note: three other subtype declarations omitted here.

add IDEA (with-first ROLE (called 'manuf-by'
                           and on NOLOT called 'CAR-MODEL')
          and with-second ROLE (called 'of'
                           and on NOLOT called 'MANUFACTURER'))
          is called 'builds';

    Note: thirteen other idea declarations omitted here.

add BRIDGE (with-first ROLE (called 'called'
                             and on NOLOT called 'REG-CAR')
            and with-second ROLE (called 'of'
                             and on LOT called 'REG-NO'))
            is called 'registration';

    Note: two other explicit bridge declarations omitted here.

add BRIDGE (with-first ROLE (called 'called'
                             and on NOLOT called 'MANUFACTURER')
            and with-second ROLE (called 'of'
                             and on LOT called 'MANUFACTURER-NAME'))
            is called 'naming-of-model';

    Note: seven other implicit bridge declarations omitted here.
    Note: the list of constraints is given on the next pages

end.
```

It has been chosen to regard only the non-lexical object types as concepts, and thus only the subtype declarations as links, and the idea types as relations. For the omitted subtype declarations, we will assume:

```
add NOLOT called 'NON-TRADING-GARAGE'
          is subtype-of NOLOT called 'GARAGE';
add NOLOT called 'CAR-MODEL'
          is subtype-of NOLOT called 'REG-MODEL';
add NOLOT called 'CAR'
          is subtype-of NOLOT called 'REG-CAR';
```

For the omitted idea type declarations, we will not go through all of them, but will assume:

```
add IDEA (with-first ROLE (called 'is-of'
                           and on NOLOT called 'CAR-MODEL')
          and with-second ROLE (called 'of'
```

```
                        and on NOLOT called 'CAR'))
          is called 'model';

add IDEA (with-first ROLE (called 'to'
                        and on NOLOT called 'OWNER')
          and with-second ROLE (called 'in'
                        and on NOLOT called 'TRANSFER'))
          is called 'transfer-owner';

add IDEA (with-first ROLE (called 'of'
                        and on NOLOT called 'CAR')
          and with-second ROLE (called 'in'
                        and on NOLOT called 'TRANSFER'))
          is called 'transfer-car';

add IDEA (with-first ROLE (called 'in'
                        and on NOLOT called 'GROUP')
          and with-second ROLE (called 'with'
                        and on NOLOT called 'PERSON'))
          is called 'group-member';
```

The following description in terms of the 11179-3 Concept System metamodel results:

Table 103 – Car Registration Model in ORM – ISO/IEC 11179 Concept System

<Concept_System>			
	notation	referencedConceptSystem	importedConceptSystem
CAR-REGISTRATION	ISO9007-E3	ORM	

Table 104 – Car Registration Model in ORM – ISO/IEC 11179 Concepts

<Concept>	
	source
MANUFACTURER	CAR-REGISTRATION
OPERATING-MANUFACTURER	CAR-REGISTRATION
REG-CAR	CAR-REGISTRATION
CAR	CAR-REGISTRATION
REG-MODEL	CAR-REGISTRATION
CAR-MODEL	CAR-REGISTRATION
FUEL-CONSUMPTION	CAR-REGISTRATION
DATE	CAR-REGISTRATION
YEAR	CAR-REGISTRATION
MONTH	CAR-REGISTRATION
DAY	CAR-REGISTRATION

<Concept>	
	source
TRANSFER	CAR-REGISTRATION
DAY-SEQ	CAR-REGISTRATION
OWNER	CAR-REGISTRATION
GARAGE	CAR-REGISTRATION
NON-TRADING-GARAGE	CAR-REGISTRATION
GROUP	CAR-REGISTRATION
PERSON	CAR-REGISTRATION

Note In Table 104 and in the following tables, the concepts are represented by their designations.

Table 105 – Car Registration Model in ORM – ISO/IEC 11179 Binary Relations

<Binary_Relation>					
	source	**role**	**reflexivity**	**symmetry**	**transitivity**
builds	CAR-REGISTRATION	manuf-by			
		of			
model	CAR-REGISTRATION	is-of			
		of			
transfer-owner	CAR-REGISTRATION	to			
		in			
transfer-car	CAR-REGISTRATION	of			
		in			
group-member	CAR-REGISTRATION	in			
		with			

Table 106 – Car Registration Model in ORM – ISO/IEC 11179 Links

<Link>			
source	**relation**	**link_end**	
		end_role	**end**
CAR-REGISTRATION	subtype	supertype-of	OPERATING_MANUFACTURER
		subtype-of	MANUFACTURER
CAR-REGISTRATION	subtype	supertype-of	MANUFACTURER
		subtype-of	OWNER
CAR-REGISTRATION	subtype	supertype-of	GARAGE
		subtype-of	OWNER
CAR-REGISTRATION	subtype	supertype-of	GROUP
		subtype-of	OWNER
CAR-REGISTRATION	subtype	supertype-of	NON-TRADING-GARAGE
		subtype-of	GARAGE
CAR-REGISTRATION	subtype	supertype-of	REG-CAR
		subtype-of	CAR
CAR-REGISTRATION	subtype	supertype-of	REG-MODEL
		subtype-of	CAR-MODEL
CAR-REGISTRATION	role-on	role	builds.manuf-by
		on	MANUFACTURER
CAR-REGISTRATION	role-on	role	builds.of
		on	CAR-MODEL
CAR-REGISTRATION	role-on	role	model.is-of
		on	CAR-MODEL
CAR-REGISTRATION	role-on	role	model.of
		on	CAR
CAR-REGISTRATION	role-on	role	transfer-owner.to
		on	OWNER
CAR-REGISTRATION	role-on	role	transfer-owner.in
		on	TRANSFER

<Link>			
source	relation	link_end	
		end_role	end
CAR-REGISTRATION	role-on	role	transfer-car.of
		on	CAR
CAR-REGISTRATION	role-on	role	transfer-car.in
		on	TRANSFER
CAR-REGISTRATION	role-on	role	group-member.in
		on	GROUP
CAR-REGISTRATION	role-on	role	group-member.of
		on	PERSON

E.4 OWL Example

Because OWL (Web Ontology Language[6]) is founded upon the very simple binary predicate model of RDF, there is more than one reasonable way to map OWL into the 11179-3 Concept System metamodel. Perhaps the more obvious is to treat each ObjectProperty as a relation, each with relation roles rdf:subject and rdf:object. However, an analogy to Properties in UML and MOF will instead suggest treating each ObjectProperty as representing a relation role, of an underlying binary relation taken to be implicit in the OWL representation. Either approach is workable, and indeed one might even mix the two approaches, treating some ObjectProperties as relations and others as relation roles, based on some case-by-case evaluation of the relative merits of each treatment.

E.4.1 OWL Metamodel

A convenient synopsis of OWL built-in constructs is provided in the OWL Web Ontology Language Overview. Some of the OWL built-in constructs correspond directly to elements of the 11179-3 Concept System metamodel. These are:

Table 107 – OWL constructs with directly corresponding ISO/IEC 11179-3 metamodel elements

OWL Constructs	11179-3 metamodel description type
Ontology	Concept_System
imports	Importation
minCardinality, maxCardinality, cardinality, FunctionalProperty, InverseFunctionalProperty	multiplicity
TransitiveProperty	transitivity

We also may capture OWL *inverseOf* assertions using only built-in 11179-3 metamodel constructs, as will be shown below. (We thus may also describe all *domain* assertions as *range* assertions on the opposing role, and therefore omit *domain* from our OWL metamodel as well.) And since assertions of membership in *SymmetricProperty* can also be regarded as simply an alternate syntax for expressing reflexive *inverseOf* assertions, they too may be captured in the same way.

Many other OWL built-in constructs do not have corresponding elements built-in to the 11179-3 Concept System metamodel, as summarized in the following table:

[6] see http://www.w3.org/TR/owl-features/

Table 108 – OWL built-in constructs described in OWL metamodel

Group of OWL Constructs	11179-3 metamodel description type
metaclasses Class, Property, ObjectProperty and DatatypeProperty	Concept
classes Thing and Nothing	Concept
datatypes	Concept
equivalentClass, equivalentProperty, sameAs	Binary_Relation (symmetric, transitive)
differentFrom, complementOf, disjointWith	Binary_Relation (symmetric, intransitive)
subClassOf, subPropertyOf	Binary_Relation (asymmetric, transitive)
type, range	Binary_Relation (asymmetric, intransitive)
AllDifferent (and distinctMembers)	Relation (variable arity, 1 role)
intersectionOf, oneOf, unionOf	Relation (variable arity, 2 roles)
allValuesFrom, someValuesFrom, hasValue	Relation (arity=3, 3 roles)

Some of these constructs are actually reused from RDFS, which in turn is defined on top of RDF, and most of the OWL datatypes are taken from XML Schema. To describe this explicitly, we actually describe four interrelated metamodels to support OWL: one each for RDF, RDFS, and the subset of XML Schema used in OWL; and one for OWL proper. The following description results:

Table 109 – OWL as an ISO/IEC 11179 Concept System

<Concept_System>			
	notation	referencedConceptSystem	importedConceptSystem
RDF			
RDFS			RDF
XSD		RDFS	
OWL		RDFS	XSD

Table 110 – OWL Concepts as ISO/IEC 11179 Concepts

<Concept> (excluding Relations)	
	source
rdf:Property	RDF
rdfs:Resource	RDFS
rdfs:Class	RDFS
rdfs:Datatype	RDFS
rdfs:Literal	RDFS
xsd:string	XSD
xsd:decimal	XSD
xsd:integer	XSD
xsd:boolean	XSD
xsd:date	XSD
...other XSD datatypes...	
owl:Class	OWL
owl:Thing	OWL
owl:Nothing	OWL
owl:ObjectProperty	OWL
owl:DatatypeProperty	OWL

Note In Table 109 and in the following tables, the concepts are represented by their designations.

Table 111 – OWL Binary Relations as ISO/IEC 11179 Binary Relations

<Binary_Relation>					
	source	**role**	**reflexivity**	**symmetry**	**transitivity**
instance-type	RDF	instance rdf:type		asymmetric	intransitive
role-range	RDFS	role rdfs:range		asymmetric	intransitive
class-subsumption	RDFS	subclass rdfs:subclassOf		asymmetric	transitive
property-subsumption	RDFS	subproperty rdfs:subpropertyOf		asymmetric	transitive
class-equivalence	OWL	owl:equivalentClass		symmetric	transitive
property-equivalence	OWL	owl:equivalentProperty		symmetric	transitive
individual-equivalence	OWL	owl:sameAs		symmetric	transitive
inequality	OWL	owl:differentFrom		symmetric	intransitive
disjointness	OWL	owl:disjointFrom		symmetric	intransitive
complementarity	OWL	owl:complementOf		symmetric	intransitive

Table 112 – OWL Relations (except Binary Relations) as ISO/IEC 11179 Relations

<Relation> (excluding Binary_Relations)			
	source	arity	role
owl:AllDifferent	OWL		operand
owl:allValuesFrom	OWL	3	class
			role
			range
owl:someValuesFrom	OWL	3	class
			role
			range
owl:hasValue	OWL	3	class
			role
			value
owl:intersectionOf	OWL		intersection
			operand
owl:unionOf	OWL		union
			operand
owl:oneOf	OWL		enumeration
			member

Table 113 – OWL Constructs as ISO/IEC 11179 Relation Roles

<Relation_Role>		
	multiplicity	ordinal
instance		1
rdf:type		2
subclass		1
rdfs:subclassOf		2
subproperty		1

<Relation_Role>		
	multiplicity	**ordinal**
rdfs:subpropertyOf		2
role		1
rdfs:range		2
owl:equivalentClass		
owl:equivalentProperty		
owl:sameAs		
owl:differentFrom		
owl:disjointFrom		
owl:complementOf		
owl:AllDifferent.operand		
owl:allValuesFrom.class		1
owl:allValuesFrom.role		2
owl:allValuesFrom.range		3
owl:someValuesFrom.class		1
owl:someValuesFrom.role		2
owl:someValuesFrom.range		3
owl:hasValue.class		1
owl:hasValue.role		2
owl:hasValue.value		3
owl:intersectionOf.intersection		1
owl:intersectionOf.operand		2
owl:unionOf.union		1
owl:unionOf.operand		2
owl:oneOf.enumeration		1
owl:oneOf.member		2

Table 114 – OWL Constructs as ISO/IEC 11179 Links

<Link>			
source	relation	link_end	
		end_role	end
RDF	instance-type	instance	rdf:type
		rdf:type	rdf:Property
RDFS	instance-type	instance	rdfs:Class
		rdf:type	rdfs:Class
RDFS	instance-type	instance	rdfs:range
		rdf:type	rdf:Property
RDFS	role-range	role	rdfs:range
		rdf:range	rdf:Class
RDFS	role-range	role	role
		rdf:range	rdf:Property
RDFS	role-range	role	rdf:type
		rdf:range	rdfs:Class
RDFS	instance-type	instance	rdfs:subclassOf
		rdf:type	rdf:Property
RDFS	role-range	role	rdfs:subclassOf
		rdf:range	rdfs:Class
RDFS	role-range	role	subclass
		rdf:range	rdfs:Class
RDFS	instance-type	instance	rdfs:subpropertyOf
		rdf:type	rdf:Property
RDFS	role-range	role	rdfs:subpropertyOf
		rdf:range	rdf:Property
RDFS	role-range	role	subproperty
		rdf:range	rdf:Property

<Link>			
source	relation	link_end	
		end_role	end
RDFS	instance-type	instance	rdfs:Resource
		rdf:type	rdfs:Class
RDFS	class-subsumption	subclass	rdfs:Class
		rdfs:subclassOf	rdfs:Resource
RDFS	instance-type	instance	rdf:Property
		rdf:type	rdfs:Class
RDFS	class-subsumption	subclass	rdf:Property
		rdfs:subclassOf	rdfs:Resource
RDFS	instance-type	instance	rdfs:Datatype
		rdf:type	rdfs:Class
RDFS	class-subsumption	subclass	rdfs:Datatype
		rdfs:subclassOf	rdfs:Class
RDFS	instance-type	instance	rdfs:Literal
		rdf:type	rdfs:Class
RDFS	class-subsumption	subclass	rdfs:Literal
		rdfs:subclassOf	rdfs:Resource
XSD	instance-type	instance	xsd:string
		rdf:type	rdfs:Datatype
XSD	instance-type	instance	xsd:decimal
		rdf:type	rdfs:Datatype
XSD	instance-type	instance	xsd:integer
		rdf:type	rdfs:Datatype
XSD	instance-type	instance	xsd:boolean
		rdf:type	rdfs:Datatype
XSD	instance-type	instance	xsd:date
		rdf:type	rdfs:Datatype

<Link>			
source	relation	link_end	
		end_role	end
...other XSD datatype links...			
OWL	instance-type	instance	owl:Thing
		rdf:type	owl:Class
OWL	instance-type	instance	owl:Nothing
		rdf:type	owl:Class
...other OWL metamodel links...			

E.4.2 Car Registration Ontology

An OWL ontology has been developed for illustration, for the application described in ISO TR 9007 Appendix B. Below is a graphical depiction of the ontology, as rendered with OntoViz.

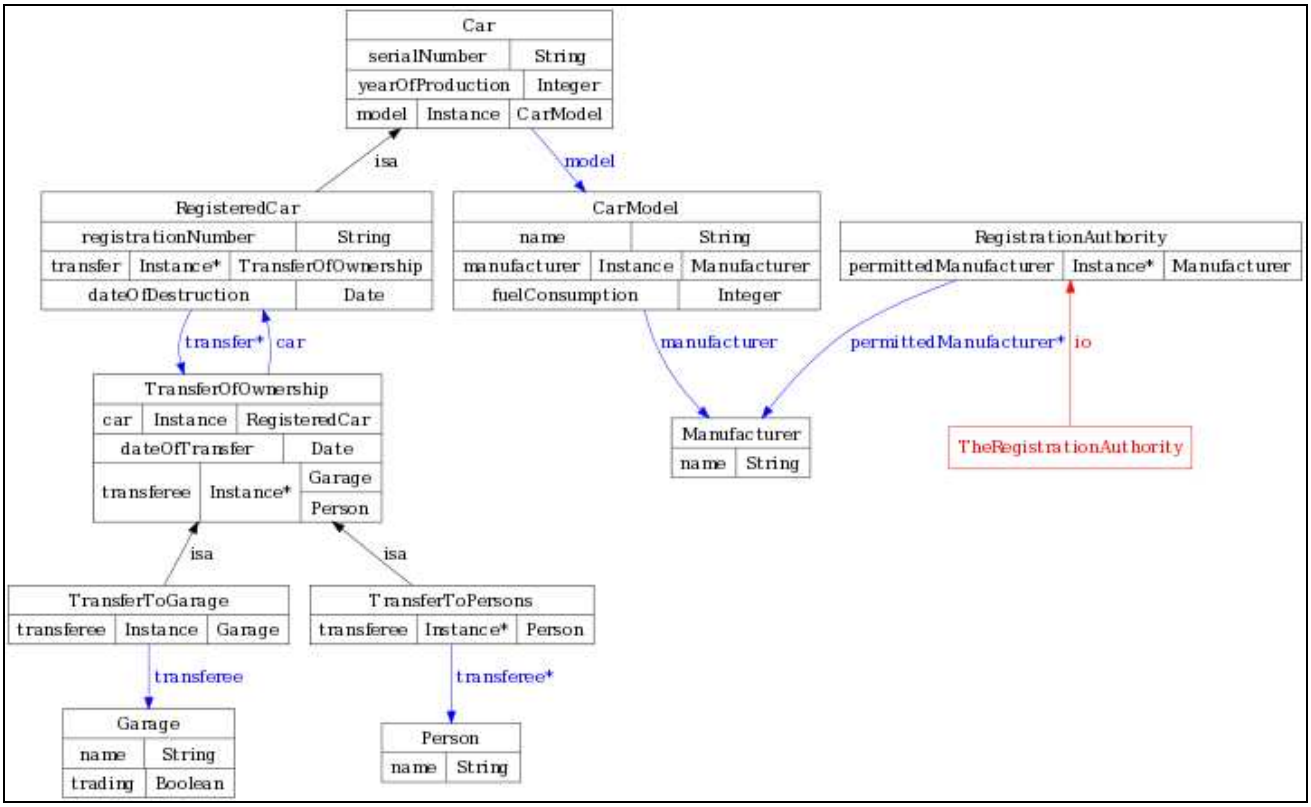

Figure 21 — Car Registration Ontology

Below is the complete text of the ontology, in Turtle syntax:

```
@prefix :  <http://xmdr.org/ont/ISO9007.owl#> .
```

```
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl:     <http://www.w3.org/2002/07/owl#> .
@prefix xsd:     <http://www.w3.org/2001/XMLSchema#> .

<http://xmdr.org/ont/ISO9007.owl>
      rdf:type owl:Ontology .

:Car
      rdf:type owl:Class ;
      rdfs:subClassOf owl:Thing ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:cardinality "1"^^xsd:int ;
                owl:onProperty :yearOfProduction
              ] ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:cardinality "1"^^xsd:int ;
                owl:onProperty :model
              ] ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:cardinality "1"^^xsd:int ;
                owl:onProperty :serialNumber
              ] ;
      owl:disjointWith :TransferOfOwnership , :Person , :Garage ,
               :Manufacturer , :RegistrationAuthority , :CarModel .

:CarModel
      rdf:type owl:Class ;
      rdfs:subClassOf owl:Thing ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:cardinality "1"^^xsd:int ;
                owl:onProperty :name
              ] ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:cardinality "1"^^xsd:int ;
                owl:onProperty :fuelConsumption
              ] ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:cardinality "1"^^xsd:int ;
                owl:onProperty :manufacturer
              ] ;
      owl:disjointWith :TransferOfOwnership , :Person , :Garage ,
               :Manufacturer , :Car , :RegistrationAuthority .

:Manufacturer
      rdf:type owl:Class ;
      rdfs:subClassOf owl:Thing ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:cardinality "1"^^xsd:int ;
                owl:onProperty :name
              ] ;
      owl:disjointWith :TransferOfOwnership , :Person , :Garage ,
               :Car , :RegistrationAuthority , :CarModel .
```

```
:RegistrationAuthority
      rdf:type owl:Class ;
      rdfs:subClassOf owl:Thing ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:maxCardinality "5"^^xsd:int ;
                owl:onProperty :permittedManufacturer
              ] ;
      owl:disjointWith :TransferOfOwnership , :Person , :Garage ,
                :Manufacturer , :Car , :CarModel ;
      owl:equivalentClass
              [ rdf:type owl:Class ;
                owl:oneOf (:TheRegistrationAuthority)
              ] .

:TheRegistrationAuthority
      rdf:type :RegistrationAuthority .

:RegisteredCar
      rdf:type owl:Class ;
      rdfs:subClassOf :Car ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:cardinality "1"^^xsd:int ;
                owl:onProperty :registrationNumber
              ] .

:TransferOfOwnership
      rdf:type owl:Class ;
      rdfs:subClassOf owl:Thing ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:minCardinality "1"^^xsd:int ;
                owl:onProperty :transferee
              ] ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:cardinality "1"^^xsd:int ;
                owl:onProperty :dateOfTransfer
              ] ;
      owl:disjointWith :Person , :Garage , :Manufacturer , :Car ,
                :RegistrationAuthority , :CarModel ;
      owl:equivalentClass
              [ rdf:type owl:Class ;
                owl:unionOf (:TransferToGarage :TransferToPersons)
              ] .

:TransferToGarage
      rdf:type owl:Class ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:cardinality "1"^^xsd:int ;
                owl:onProperty :transferee
              ] ;
      owl:disjointWith :TransferToPersons ;
      owl:equivalentClass
              [ rdf:type owl:Class ;
                owl:intersectionOf ([ rdf:type owl:Restriction ;
                            owl:allValuesFrom :Garage ;
                            owl:onProperty :transferee
                          ] :TransferOfOwnership)
```

```
            ] .

:Garage
      rdf:type owl:Class ;
      rdfs:subClassOf owl:Thing ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:cardinality "1"^^xsd:int ;
                owl:onProperty :name
              ] ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:cardinality "1"^^xsd:int ;
                owl:onProperty :trading
              ] ;
      owl:disjointWith :TransferOfOwnership , :Person , :Manufacturer ,
               :Car , :RegistrationAuthority , :CarModel .

:TransferToPersons
      rdf:type owl:Class ;
      owl:disjointWith :TransferToGarage ;
      owl:equivalentClass
              [ rdf:type owl:Class ;
                owl:intersectionOf      (:TransferOfOwnership      [      rdf:type
owl:Restriction ;
                            owl:allValuesFrom :Person ;
                            owl:onProperty :transferee
                          ])
              ] .

:Person
      rdf:type owl:Class ;
      rdfs:subClassOf owl:Thing ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:cardinality "1"^^xsd:int ;
                owl:onProperty :name
              ] ;
      owl:disjointWith :TransferOfOwnership , :Garage , :Manufacturer ,
               :Car , :RegistrationAuthority , :CarModel .

:yearOfProduction
      rdf:type owl:DatatypeProperty , owl:FunctionalProperty ;
      rdfs:domain :Car ;
      rdfs:range xsd:int .

:model
      rdf:type owl:ObjectProperty , owl:FunctionalProperty ;
      rdfs:domain :Car ;
      rdfs:range :CarModel .

:serialNumber
      rdf:type owl:DatatypeProperty , owl:FunctionalProperty ;
      rdfs:domain :Car ;
      rdfs:range xsd:string .

:name
      rdf:type owl:DatatypeProperty , owl:FunctionalProperty ;
      rdfs:domain
              [ rdf:type owl:Class ;
                owl:unionOf (:CarModel :Manufacturer :Garage :Person)
```

```
              ] ;
      rdfs:range xsd:string .

:fuelConsumption
      rdf:type owl:DatatypeProperty , owl:FunctionalProperty ;
      rdfs:comment
            "number of litres of hydrocarbon fuel per 100 kilometres. Ranges from
4 to 25."^^xsd:string ;
      rdfs:domain :CarModel ;
      rdfs:range xsd:int .

:manufacturer
      rdf:type owl:ObjectProperty , owl:FunctionalProperty ;
      rdfs:domain :CarModel ;
      rdfs:range :Manufacturer .

:permittedManufacturer
      rdf:type owl:ObjectProperty ;
      rdfs:domain :RegistrationAuthority ;
      rdfs:range :Manufacturer .

:registrationNumber
      rdf:type owl:DatatypeProperty , owl:FunctionalProperty ;
      rdfs:domain :RegisteredCar ;
      rdfs:range xsd:string .

:dateOfDestruction
      rdf:type owl:DatatypeProperty , owl:FunctionalProperty ;
      rdfs:domain :RegisteredCar ;
      rdfs:range xsd:date .

:transfer
      rdf:type owl:ObjectProperty , owl:InverseFunctionalProperty ;
      rdfs:domain :RegisteredCar ;
      rdfs:range :TransferOfOwnership ;
      owl:inverseOf :car .

:car
      rdf:type owl:ObjectProperty , owl:FunctionalProperty ;
      rdfs:domain :TransferOfOwnership ;
      rdfs:range :RegisteredCar ;
      owl:inverseOf :transfer .

:transferee
      rdf:type owl:ObjectProperty ;
      rdfs:domain :TransferOfOwnership ;
      rdfs:range
              [ rdf:type owl:Class ;
                owl:unionOf (:Garage :Person)
              ] .

:dateOfTransfer
      rdf:type owl:DatatypeProperty , owl:FunctionalProperty ;
      rdfs:domain :TransferOfOwnership ;
      rdfs:range xsd:date .

:trading
      rdf:type owl:DatatypeProperty , owl:FunctionalProperty ;
      rdfs:domain :Garage ;
      rdfs:range xsd:boolean .
```

Using the metamodels above, the following description in terms of the 11179-3 Concept System metamodel results:

Table 115 – Car Registration Model in OWL – ISO/IEC 11179 Concept System

<Concept_System>			
	notation	**referencedConceptSystem**	**importedConceptSystem**
ISO9007.B	OWL/Turtle	OWL	

Table 116 – Car Registration Model in OWL – ISO/IEC 11179 Concepts

<Concept> *(excluding Relations and Relation_Roles)*	
	source
:Car	ISO9007.B
:CarModel	ISO9007.B
:Manufacturer	ISO9007.B
:RegistrationAuthority	ISO9007.B
:TheRegistrationAuthority	ISO9007.B
:RegisteredCar	ISO9007.B
:TransferOfOwnership	ISO9007.B
:TransferToGarage	ISO9007.B
:Garage	ISO9007.B
:TransferToPersons	ISO9007.B
:Person	ISO9007.B
:yearOfProduction	ISO9007.B
:serialNumber	ISO9007.B
:name	ISO9007.B
:fuelConsumption	ISO9007.B
:registrationNumber	ISO9007.B
:dateOfDestruction	ISO9007.B
:dateOfTransfer	ISO9007.B
:trading	ISO9007.B

Table 117 – Car Registration Model in OWL – ISO/IEC 11179 Binary Relations

<Binary_Relation>					
	source	**role**	**reflexivity**	**symmetry**	**transitivity**
relation1	ISO9007.B	:model		asymmetric	instransitive
		inverse of :model			
relation2	ISO9007.B	:manufacturer		asymmetric	instransitive
		inverse of :manufacturer			
relation3	ISO9007.B	:permittedManufacturer		asymmetric	instransitive
		inverse of :permittedManufacturer			
relation4	ISO9007.B	:transfer		asymmetric	instransitive
		:car			
relation5	ISO9007.B	:transferee		asymmetric	instransitive
		inverse of :transferee			

Table 118 – Car Registration Model in OWL – ISO/IEC 11179 Relation Roles

<Relation_Role>		
	multiplicity	**ordinal**
:model	1	
inverse of :model		
:manufacturer	1	
inverse of :manufacturer		
:permittedManufacturer		
inverse of :permittedManufacturer		
:transfer		
:car	1	
:transferee		
inverseOf :transferee		

Table 119 – Car Registration Model in OWL – ISO/IEC 11179 Links

<Link>			
assertor	**relation**	**link_end**	
		end_role	**end**
ISO9007.B	instance-type	instance	:Car
		rdf:type	owl:Class
ISO9007.B	instance-type	instance	:CarModel
		rdf:type	owl:Class
ISO9007.B	instance-type	instance	:Manufacturer
		rdf:type	owl:Class
ISO9007.B	instance-type	instance	:RegistrationAuthority
		rdf:type	owl:Class
ISO9007.B	instance-type	instance	:TheRegistrationAuthority
		rdf:type	:RegistrationAuthority
ISO9007.B	instance-type	instance	:RegisteredCar
		rdf:type	owl:Class
ISO9007.B	instance-type	instance	:TransferOfOwnership
		rdf:type	owl:Class
ISO9007.B	instance-type	instance	:TransferToGarage
		rdf:type	owl:Class
ISO9007.B	instance-type	instance	:Garage
		rdf:type	owl:Class
ISO9007.B	instance-type	instance	:TransferToPersons
		rdf:type	owl:Class
ISO9007.B	instance-type	instance	:Person
		rdf:type	owl:Class
ISO9007.B	instance-type	instance	:yearOfProduction
		rdf:type	owl:DatatypeProperty
ISO9007.B	instance-type	instance	:serialNumber
		rdf:type	owl:DatatypeProperty
ISO9007.B	instance-type	instance	:name
		rdf:type	owl:DatatypeProperty
ISO9007.B	instance-type	instance	:fuelConsumption

<Link>			
		rdf:type	owl:DatatypeProperty
ISO9007.B	instance-type	instance	:registrationNumber
		rdf:type	owl:DatatypeProperty
ISO9007.B	instance-type	instance	:dateOfDestruction
		rdf:type	owl:DatatypeProperty
ISO9007.B	instance-type	instance	:dateOfTransfer
		rdf:type	owl:DatatypeProperty
ISO9007.B	instance-type	instance	:trading
		rdf:type	owl:DatatypeProperty
ISO9007.B	instance-type	instance	:model
		rdf:type	owl:ObjectProperty
ISO9007.B	instance-type	instance	:manufacturer
		rdf:type	owl:ObjectProperty
ISO9007.B	instance-type	instance	:permittedManufacturer
		rdf:type	owl:ObjectProperty
ISO9007.B	instance-type	instance	:transfer
		rdf:type	owl:ObjectProperty
ISO9007.B	instance-type	instance	:car
		rdf:type	owl:ObjectProperty
ISO9007.B	instance-type	instance	:transferee
		rdf:type	owl:ObjectProperty
ISO9007.B	property-domain	domainOf	:yearOfProduction
		rdfs:domain	:Car
ISO9007.B	property-range	rangeOf	:yearOfProduction
		rdfs:range	xsd:int
ISO9007.B	property-domain	domainOf	:model
		rdfs:domain	:Car
ISO9007.B	property-range	rangeOf	:model
		rdfs:range	:CarModel
ISO9007.B	property-domain	domainOf	:serialNumber
		rdfs:domain	:Car
ISO9007.B	property-range	rangeOf	:serialNumber
		rdfs:range	xsd:string
ISO9007.B	property-range	rangeOf	:name

<Link>			
		rdfs:range	xsd:string
ISO9007.B	property-domain	domainOf	:fuelConsumption
		rdfs:domain	:CarModel
ISO9007.B	property-range	rangeOf	:fuelConsumption
		rdfs:range	xsd:int
ISO9007.B	property-domain	domainOf	:manufacturer
		rdfs:domain	:CarModel
ISO9007.B	property-range	rangeOf	:manufacturer
		rdfs:range	:Manufacturer
ISO9007.B	property-domain	domainOf	:permittedManufacturer
		rdfs:domain	:RegistrationAuthority
ISO9007.B	property-range	rangeOf	:permittedManufacturer
		rdfs:range	:Manufacturer
ISO9007.B	property-domain	domainOf	:registrationNumber
		rdfs:domain	:RegisteredCar
ISO9007.B	property-range	rangeOf	:registrationNumber
		rdfs:range	xsd:string
ISO9007.B	property-domain	domainOf	:dateOfDestruction
		rdfs:domain	:RegisteredCar
ISO9007.B	property-range	rangeOf	:dateOfDestruction
		rdfs:range	xsd:date
ISO9007.B	property-domain	domainOf	:transfer
		rdfs:domain	:RegisteredCar
ISO9007.B	property-range	rangeOf	:transfer
		rdfs:range	:TransferOfOwnership
ISO9007.B	property-domain	domainOf	:car
		rdfs:domain	:TransferOfOwnership
ISO9007.B	property-range	rangeOf	:car
		rdfs:range	:RegisteredCar
ISO9007.B	property-domain	domainOf	:transferee
		rdfs:domain	:TransferOfOwnership
ISO9007.B	property-domain	domainOf	:dateOfTransfer
		rdfs:domain	:TransferOfOwnership
ISO9007.B	property-range	rangeOf	:dateOfTransfer

<Link>			
		rdfs:range	xsd:date
ISO9007.B	property-domain	domainOf	:trading
		rdfs:domain	:Garage
ISO9007.B	property-range	rangeOf	:trading
		rdfs:range	xsd:boolean
ISO9007.B	class-subsumption	subclass	:Car
		rdfs:subClassOf	owl:Thing
ISO9007.B	class-subsumption	subclass	:CarModel
		rdfs:subClassOf	owl:Thing
ISO9007.B	class-subsumption	subclass	:Manufacturer
		rdfs:subClassOf	owl:Thing
ISO9007.B	class-subsumption	subclass	:RegistrationAuthority
		rdfs:subClassOf	owl:Thing
ISO9007.B	class-subsumption	subclass	:RegisteredCar
		rdfs:subClassOf	:Car
ISO9007.B	class-subsumption	subclass	:TransferOfOwnership
		rdfs:subClassOf	owl:Thing
ISO9007.B	class-subsumption	subclass	:Garage
		rdfs:subClassOf	owl:Thing
ISO9007.B	class-subsumption	subclass	:Person
		rdfs:subClassOf	owl:Thing
ISO9007.B	disjointness	owl:disjointWith	:Car
		owl:disjointWith	:TransferOfOwnership
ISO9007.B	disjointness	owl:disjointWith	:Car
		owl:disjointWith	:Person
ISO9007.B	disjointness	owl:disjointWith	:Car
		owl:disjointWith	:Garage
ISO9007.B	disjointness	owl:disjointWith	:Car
		owl:disjointWith	:Manufacturer
ISO9007.B	disjointness	owl:disjointWith	:Car
		owl:disjointWith	:RegistrationAuthority
ISO9007.B	disjointness	owl:disjointWith	:Car
		owl:disjointWith	:CarModel
ISO9007.B	disjointness	owl:disjointWith	:CarModel

<Link>			
		owl:disjointWith	:TransferOfOwnership
ISO9007.B	disjointness	owl:disjointWith	:CarModel
		owl:disjointWith	:Person
ISO9007.B	disjointness	owl:disjointWith	:CarModel
		owl:disjointWith	:Garage
ISO9007.B	disjointness	owl:disjointWith	:CarModel
		owl:disjointWith	:Manufacturer
ISO9007.B	disjointness	owl:disjointWith	:CarModel
		owl:disjointWith	:RegistrationAuthority
ISO9007.B	disjointness	owl:disjointWith	:Manufacturer
		owl:disjointWith	:TransferOfOwnership
ISO9007.B	disjointness	owl:disjointWith	:Manufacturer
		owl:disjointWith	:Person
ISO9007.B	disjointness	owl:disjointWith	:Manufacturer
		owl:disjointWith	:Garage
ISO9007.B	disjointness	owl:disjointWith	:Manufacturer
		owl:disjointWith	:RegistrationAuthority
ISO9007.B	disjointness	owl:disjointWith	:RegistrationAuthority
		owl:disjointWith	:TransferOfOwnership
ISO9007.B	disjointness	owl:disjointWith	:RegistrationAuthority
		owl:disjointWith	:Person
ISO9007.B	disjointness	owl:disjointWith	:RegistrationAuthority
		owl:disjointWith	:Garage
ISO9007.B	disjointness	owl:disjointWith	:TransferOfOwnership
		owl:disjointWith	:Person
ISO9007.B	disjointness	owl:disjointWith	:TransferOfOwnership
		owl:disjointWith	:Garage
ISO9007.B	disjointness	owl:disjointWith	:TransferToGarage
		owl:disjointWith	:TransferToPersons
ISO9007.B	disjointness	owl:disjointWith	:Garage
		owl:disjointWith	:Person
ISO9007.B	owl:unionOf	union	:TransferOfOwnership
		operand	:TransferToGarage
		operand	:TransferToPersons

<Link>			
ISO9007.B	owl:oneOf	enumeration	:RegistrationAuthority
		member	:TheRegistrationAuthority

Table 120 – Car Registration Model in OWL – ISO/IEC 11179 Assertions

<Assertion> (*excluding Links*)		
assertor	**assertion_formula**	**term**
ISO9007.B	`:Car rdfs:subClassOf [ rdf:type owl:Restriction ; owl:cardinality "1"^^xsd:int ; owl:onProperty :yearOfProduction ] .`	:Car :yearOfProduction
ISO9007.B	`:Car rdfs:subClassOf [ rdf:type owl:Restriction ; owl:cardinality "1"^^xsd:int ; owl:onProperty :model ] .`	:Car :model
ISO9007.B	`:Car rdfs:subClassOf [ rdf:type owl:Restriction ; owl:cardinality "1"^^xsd:int ; owl:onProperty :serialNumber ] .`	:Car :serialNumber
ISO9007.B	`:CarModel rdfs:subClassOf [ rdf:type owl:Restriction ; owl:cardinality "1"^^xsd:int ; owl:onProperty :name ] .`	:CarModel :name
ISO9007.B	`:CarModel rdfs:subClassOf [ rdf:type owl:Restriction ; owl:cardinality "1"^^xsd:int ; owl:onProperty :fuelConsumption ] .`	:CarModel :fuelConsumption
ISO9007.B	`:CarModel rdfs:subClassOf [ rdf:type owl:Restriction ; owl:cardinality "1"^^xsd:int ; owl:onProperty :manufacturer ] .`	:CarModel :manufacturer
ISO9007.B	`:Manufacturer rdfs:subClassOf [ rdf:type owl:Restriction ; owl:cardinality "1"^^xsd:int ; owl:onProperty :name ] .`	:Manufacturer :name
ISO9007.B	`:RegistrationAuthority rdfs:subClassOf [ rdf:type owl:Restriction ; owl:maxCardinality "5"^^xsd:int ; owl:onProperty :permittedManufacturer ] .`	:RegistrationAuthority :permittedManufacturer
ISO9007.B	`:RegisteredCar rdfs:subClassOf [ rdf:type owl:Restriction ; owl:cardinality "1"^^xsd:int ; owl:onProperty :registrationNumber ] .`	:RegisteredCar :registrationNumber
ISO9007.B	`:TransferOfOwnership rdfs:subClassOf [ rdf:type owl:Restriction ; owl:minCardinality "1"^^xsd:int ; owl:onProperty :transferee ] .`	:TransferOfOwnership :transferee
ISO9007.B	`:TransferOfOwnership rdfs:subClassOf`	:TransferOfOwnership

<Assertion> (*excluding Links*)		

	`[ rdf:type owl:Restriction ;`
	`owl:cardinality "1"^^xsd:int ;`
	`owl:onProperty :dateOfTransfer ] .`
		:dateOfTransfer

ISO9007.B	`:TransferToGarage rdfs:subClassOf`
	`[ rdf:type owl:Restriction ;`
	`owl:cardinality "1"^^xsd:int ;`
	`owl:onProperty :transferee ] .`
		:TransferToGarage
		:transferee

ISO9007.B	`:TransferToGarage owl:equivalentClass`
	`[ rdf:type owl:Class ;`
	`owl:intersectionOf ([ rdf:type owl:Restriction ;`
	`owl:allValuesFrom :Garage ;`
	`owl:onProperty :transferee ]`
	`:TransferOfOwnership) ] .`
		:TransferToGarage
		:Garage
		:transferee
		:TransferOfOwnership

ISO9007.B	`:Garage rdfs:subClassOf`
	`[ rdf:type owl:Restriction ;`
	`owl:cardinality "1"^^xsd:int ;`
	`owl:onProperty :name ] .`
		:Garage
		:name

ISO9007.B	`:Garage rdfs:subClassOf`
	`[ rdf:type owl:Restriction ;`
	`owl:cardinality "1"^^xsd:int ;`
	`owl:onProperty :trading ] .`
		:Garage
		:trading

ISO9007.B	`:TransferToPersons owl:equivalentClass`
	`[ rdf:type owl:Class ;`
	`owl:intersectionOf ([ rdf:type owl:Restriction ;`
	`owl:allValuesFrom :Person ;`
	`owl:onProperty :transferee ]`
	`:TransferOfOwnership) ] .`
		:TransferToPersons
		:Person
		:transferee
		:TransferOfOwnership

ISO9007.B	`:Person rdfs:subClassOf`
	`[ rdf:type owl:Restriction ;`
	`owl:cardinality "1"^^xsd:int ;`
	`owl:onProperty :name ] .`
		:Person
		:name

ISO9007.B	`:name rdfs:domain`
	`[ rdf:type owl:Class ;`
	`owl:unionOf (:CarModel :Manufacturer :Garage :Person) ] .`
		:name
		:CarModel
		:Manufacturer
		:Garage
		:Person

ISO9007.B	`:transferee rdfs:domain`
	`[ rdf:type owl:Class ;`
	`owl:unionOf (:Garage :Person) ] .`
		:transferee
		:Garage
		:Person

E.5 CLIF Example

Common Logic Interchange Format (CLIF) is a dialect of ISO/IEC 24707 Common Logic (CL) which closely resembles the older (but never standardized) Knowledge Interchange Format (KIF).

E.5.1 CL Metamodel

Common Logic (and thus CLIF) defines several logical connectives and quantifiers, but only one built-in relation between concepts: equality (designated as '=' in CLIF). The description of this one built-in Relation is very simple (see below).

However, it is also very useful to include in a CL metamodel a second relation which is typically implicit in CL ontologies: the "isa" relation between an individual and a class (typically represented simply by a "unary relation" in KIF or CLIF) of which it is a member. Describing this implicit relation in the CL metamodel will allow us to then describe unary atomic sentence assertions as links of that implicit binary relation (see examples below).

Table 121 – CL Metamodel – ISO/IEC 11179 Concept System

<Concept_System>			
	notation	**referencedConceptSystem**	**importedConceptSystem**
CL			

Table 122 – CL Metamodel – ISO/IEC 11179 Binary Relations

<Binary_Relation>					
	source	**role**	**reflexivity**	**symmetry**	**transitivity**
=	CL	*(unnamed)*	reflexive	symmetric	transitive
isa	CL	class	irreflexive	asymmetric	intransitive
		instance			

E.5.2 CLIF Units Example from ISO/IEC 19763-3

In Annex A of ISO/IEC 19763-3 an example of a KIF ontology is given (in clause A.3) containing the following three sentences:

```
(and (KernelUnit metre) (Dimensionality length) (dimensionality metre length))

(and (PrefixedlUnit micron) (MetricPrefix micro) (KernelUnit metre)
(prefix micron micro) (kernel micron metre))

((forall ?Unit1 ?Unit2 ?Dimensionality1 ?Dimensionality2)
(implies (and (PrefixedUnit ?Unit1) (kernel ?Unit1 ?Unit2)
(dimensionality ?Unit1 ?Dimensionality1)
(dimensionality ?Unit2 ?Dimensionality2))
(equal ?Dimensionality1 ?Dimensionality2)))
```

The differences between KIF and CLIF are small enough that the same three sentences are also valid CLIF sentences with essentially equivalent meaning, modulo only some very minor changes such as substitution of the token 'if' in place of 'implies'. However, the first two sentences are oddly constructed (I believe primarily for making the example smaller in terms of number of sentences), because semantically there is no reason for the enclosing logical conjunctions—the assertion of the conjunction is precisely equivalent to the assertion of each of its constituents separately. Since it is more illustrative for our purposes to describe them as decomposed, we are going to take the liberty of doing so here. The equivalent CLIF sentences we are going to use as an example here are thus:

```
(KernelUnit metre)
(Dimensionality length)
(dimensionality metre length)
(PrefixedlUnit micron)
(MetricPrefix micro)
(prefix micron micro)
(kernel micron metre)
(forall (?Unit1 ?Unit2 ?Dimensionality1 ?Dimensionality2)
        (if (and (PrefixedUnit ?Unit1) (kernel ?Unit1 ?Unit2)
                 (dimensionality ?Unit1 ?Dimensionality1)
(dimensionality ?Unit2 ?Dimensionality2))
            (equal ?Dimensionality1 ?Dimensionality2)))
```

Using the CL metamodel above, the following description in terms of the 11179-3 Concept System metamodel results:

Table 123 – CLIF Units Example – ISO/IEC 11179 Concept System

<Concept_System>			
	notation	**referencedConceptSystem**	**importedConceptSystem**
CLIF-UNITS	CLIF	CL	

Table 124 – CLIF Units Example – ISO/IEC 11179 Concepts

<Concept> (excluding Relations and Relation_Roles)	
	source
Dimensionality	CLIF-UNITS
KernelUnit	CLIF-UNITS
MetricPrefix	CLIF-UNITS
PrefixedlUnit	CLIF-UNITS
length	CLIF-UNITS
metre	CLIF-UNITS
micro	CLIF-UNITS
micron	CLIF-UNITS

Table 125 – CLIF Units Example – ISO/IEC 11179 Binary Relations

<Binary_Relation>					
	source	role	reflexivity	symmetry	transitivity
dimensionality	CLIF-UNITS	dimensionality.role1		asymmetric	intransitive
		dimensionality.role2			
kernel	CLIF-UNITS	kernel.role1		asymmetric	intransitive
		kernel.role2			
prefix	CLIF-UNITS	prefix.role1		asymmetric	intransitive
		prefix.role2			

Table 126 – CLIF Units Example – ISO/IEC 11179 Relations Roles

<Relation_Role>		
	multiplicity	ordinal
dimensionality.role1		1
dimensionality.role2		2
kernel.role1		1
kernel.role2		2
prefix.role1		1
prefix.role2		2

Table 127 – CLIF Units Example – ISO/IEC 11179 Links

<Link>				
assertor	assertion_formula	relation	link_end	
			end_role	end
CLIF-UNITS	(KernelUnit metre)	isa	instance	metre
			class	KernelUnit
CLIF-UNITS	(Dimensionality length)	isa	instance	length
			class	Dimensionality

<Link>				
CLIF-UNITS	(PrefixedlUnit micron)	isa	instance	micron
			class	PrefixedlUnit
CLIF-UNITS	(MetricPrefix micro)	isa	instance	micro
			class	MetricPrefix
CLIF-UNITS	(dimensionality metre length)	dimensionality	dimensionality.role1	metre
			dimensionality.role2	length
CLIF-UNITS	(kernel micron metre)	kernel	kernel.role1	micron
			kernel.role2	metre
CLIF-UNITS	(prefix micron micro)	prefix	prefix.role1	micron
			prefix.role2	micro

Table 128 – CLIF Units Example – ISO/IEC 11179 Assertions

<Assertion> (*excluding Links*)		
assertor	**assertion_formula**	**term**

CLIF-UNITS	`(forall (?Unit1 ?Unit2 ?Dimensionality1 ?Dimensionality2)`
	`(if (and (PrefixedUnit ?Unit1)`
	`(kernel ?Unit1 ?Unit2)`
	`(dimensionality ?Unit1 ?Dimensionality1)`
	`(dimensionality ?Unit2 ?Dimensionality2))`
	`(= ?Dimensionality1 ?Dimensionality2)))`
		PrefixedUnit
		kernel
		dimensionality
		=

Annex F
(informative)

Representation Class as a Concept System

F.1 Introduction

This Annex illustrates the use of a *Concept System* to implement a Representation Class classification scheme.

In Edition 2 of this standard, *Representation Class* was specified as a distinct component of the metamodel, even though it was really just a *Classification Scheme* for representation. In this Edition, Representation Class is considered to be an instance of a Classification Scheme, which in turn is modelled as a Concept System (see 9.2).

F.2 Description of Representation Class

The major intent of *Representation class* is to provide a discrete and complete set of high-level (coarse granularity) definitions for data element/value domain categorization. This is an aid to the user in terms of application of business rules.

Representation Class is a mechanism by which the functional and/or presentational category of an item may be conveyed to a user.

An informational list of representation class terms is provided in ISO/IEC 11179-5. The list below has been expanded to provide a more comprehensive list of examples.

- Code -– A system of valid symbols that substitute for specified values e.g. alpha, numeric, symbols and/or combinations.
- Count –- Non-monetary numeric value arrived at by counting.
- Currency –- Monetary representation
- Date –- Calendar representation e.g. YYYY-MM-DD
- Graphic –- Diagrams, graphs, mathematical curves, or the like – usually a vector image.
- Icon –- A sign or representation that stands for its object by virtue of a resemblance or analogy to it
- Picture –- A visual representation of a person, object, or scene – usually a raster image.
- Quantity –- A continuous number such as the linear dimensions, capacity/amount (non-monetary) of an object
- Text – A text field that is usually unformatted.
- Time –- Time of day or duration eg HH:MM:SS.SSSS.

None of the terms in this list is required in any specific implementation of representation class.

By using representation class, enhanced semantic control over the contents of value domains can be maintained. Rules can be drawn against representation classes that allow enforcement of content within and among value domains. For example:

- “A date-class data element must be in the format YYYY-MM-DD.”
- “A relationship must exist between a code representation and the specific form of the value meanings which the code represents.”
- “Currency data elements give other Currency data elements when added or subtracted, but not when multiplied or divided.”
- “Dividing one Currency data element by another yields a ratio (Count or Quantity), not another Currency data element.”

The set of classes make it easy to distinguish among the elements in the registry. For instance, a data element categorized with the representation class 'Currency' is different from an element categorized as 'Count' or ‘Quantity’. It probably won't make sense to compare the contents of these elements, or perform additions or subtractions using them together (though multiplication or division may be meaningful).

F.3 Implementation of Representation Class as a Concept_System

A *Concept_System* is registered with the *designation* ‘Representation class’. Within the *Concept_System* individual *Concept*s are registered with *designation*s that correspond to each of the desired class terms.

The *Classifiable_Item* class is used to classify the *Data_Element*s by associating them with the appropriate *Concept* in the *Concept_System*.

Annex G
(informative)

Comparison for Conformance Levels across Editions of this part of ISO/IEC 11179

G.1 Introduction

This Annex compares the range of registry conformance for Edition 3, to the conformance requirements found in prior Editions of the standard.

Table 129 – Comparison for Conformance Levels across Editions of this part of ISO/IEC 11179

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
Concepts package	Mandatory	Mandatory	N/a	N/a
Binary_Relations package	Optional	N/a	N/a	N/a
Data Description package	Optional	Mandatory	N/a	N/a
Basic attributes	Optional	N/a	Mandatory	Mandatory

G.2 Conformance Levels for Edition 2 Level 2

The closest profile to Edition 2 Level 2 is the **Metadata Registry** profile (see H.4).

G.3 Conformance Levels for Edition 2 Level 1 and Edition 1

Only those metadata elements, relationships and properties specified in Clause 12 are supported and used.

Annex H
(Normative)

Standard Conformance Profiles for this part of ISO/IEC 11179

H.1 Introduction

This Annex provides some standard conformance profiles to assist implementations claiming conformance to this standard.

H.2 Profile for Concept Systems Registry

Implements clauses: 7, 9, and also satisfies the following additional provisions:

1. All Concepts and Concept_Systems must be Registered_Items.
2. All Registered_Items must also be Designatable_Items and Classifiable_Items.

H.3 Profile for Extended Concept Systems Registry

Implements clause 10, in addition to all provisions of the Concept Systems Registry profile.

H.4 Profile for Metadata Registry

Implements clause 11, in addition to all provisions of the Concept Systems Registry profile, and also satisfies the following additional provision:

— All Data_Elements, Value_Domains, and Derivation_Rules must be Registered_Items.

H.5 Profile for Extended Metadata Registry

Implements all of clauses 7-11, and also satisfies all provisions of the Metadata Registry profile.

Bibliography

[1] IETF RFC 5646, *Tags for Identifying Languages*

http://www.rfc-editor.org/rfc/rfc5646.txt

[2] ISO 31-0:1992, Quantities and units — Part 0: General principles

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

[4] ISO 1087-1:2000, Terminology work — Vocabulary — Part 1: Theory and application

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

[6] ISO/IEC 2382-17:1999, Information technology — Vocabulary — Part 17: Databases

[7] ISO 3166-1:2006, Codes for the representation of names of countries and their subdivisions — Part 1: Country codes

[8] ISO 5127:2001, Information and documentation — Vocabulary

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

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

[11] ISO 8601:2004, Data elements and interchange formats — Information interchange — Representation of dates and times

[12] 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.

[13] ISO/IEC 10027:1990, Information technology — Information Resource Dictionary System (IRDS) framework

ISO/IEC 10027 describes the concept of levels of modelling.

[14] ISO/IEC TR 10032:2003, Information technology — Reference model for data management

ISO/IEC 10032 describes the concept of levels of modelling.

[15] ISO 10241:1992, International terminology standards — Preparation and layout

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

[17] ISO/IEC 11179-3:1994 (Edition 1), Information technology — Specification and standardization of data elements — Part 3: Basic attributes of data elements

[18] ISO/IEC 11179-3:2003 (Edition 2), Information technology — Metadata registries (MDR) — Part 3: Registry metamodel and basic attributes

[19] ISO/IEC 11179-3 Technical Corrigendum 1 for ISO/IEC 11179-3:2003 Information technology — Metadata registries (MDR) — Part 3: Registry metamodel and basic attributes

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

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

[22] ISO/IEC 11404:2007, Information technology — General Purpose Datatypes (GPD)

[23] ISO 12620:2009, Terminology and other language and content resources -- Specification of data categories and management of a Data Category Registry for language resources

[24] ISO 15924:2004 Information and documentation — Codes for the representation of names of scripts

[25] ISO/IEC 15944-1:2011, Information technology — Business Operational View — Part 1: Operational aspects of Open-edi for implementation

[26] ISO/IEC 19501:2005, Information technology — Open Distributed Processing — Unified Modeling Language (UML) Version 1.4.2.

[27] ISO/IEC 19505-1:2012, Information technology — Object Management Group Unified Modeling Language (OMG UML)— Part 1: Infrastructure
(See http://www.omg.org/spec/UML/ISO/19505-1/PDF)

[28] ISO/IEC 19505-2:2012 Information technology — Object Management Group Unified Modeling Language (OMG UML)— Part 2: Superstructure
(See http://www.omg.org/spec/UML/ISO/19505-2/PDF)

[29] ISO/IEC 19773:2011, Information technology — Metadata registries (MDR) Modules

[30] 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 an ISO/IEC 11179-3 metadata registry.

[31] 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 an ISO/IEC 11179-3 metadata registry.

[32] ISO 21090:2011 Health Informatics – Harmonized datatypes for information interchange

ISO 21090 specifies datatypes for representing and exchanging basic concepts that are commonly encountered in health care environments.

[33] ISO/IEC 24707:2007 Common Logic

Includes a specification of XCL.

[34] ISO 80000-1:2009 Quantities and Units – Part 1: General (supersedes ISO 31-0:1992)

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

http://www.itu.int/rec/T-REC-E.164-200502-I/en

[36] OASIS ebXML Registry Information Model (RIM) version 3.0.1

Defines the types of metadata and content that can be stored in an ebXML Registry

[37] OASIS ebXML Registry Services and Protocols (RS) version 3.0.1

Defines the services and protocols for an ebXML Registry

[38] OMG SBVR (Semantics of Business Vocabulary and Business Rules)

http://www.omg.org/spec/SBVR/1.0/

[39] OMG Unified Modeling Language (OMG UML) Version 2.4.1 — Part 1: Infrastructure

http://www.omg.org/spec/UML/2.4.1/Infrastructure/PDF

[40] OMG Unified Modeling Language (OMG UML) Version 2.4.1 — Part 2: Superstructure

http://www.omg.org/spec/UML/2.4.1/Superstructure/PDF

[41] OWL Web Ontology Language Guide

http://www.w3.org/TR/owl-guide/

[42] RDF – Resource Description Framework

http://www.w3.org/RDF/

[43] SKOS – Simple Knowledge Organization System

http://www.w3.org/TR/skos-reference/skos.html

[44] Turtle – Terse RDF Triple Language

http://www.w3.org/TeamSubmission/2008/SUBM-turtle-20080114/

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

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.