

Australian/New Zealand Standard™

Geographic information — Ontology

Part 2: Rules for developing ontologies in the Web Ontology Language (OWL)



AS/NZS ISO 19150.2:2018

This Joint Australian/New Zealand Standard™ was prepared by Joint Technical Committee IT-004, Geographical Information/Geomatics. It was approved on behalf of the Council of Standards Australia on 5 June 2018 and by the New Zealand Standards Approval Board on 4 September 2019.

This Standard was published on 29 June 2018.

The following are represented on Committee IT-004:

- ANZLIC — the Spatial Information Council
- Australian Antarctic Division, Department of the Environment
(Australian Government)
- Australian Bureau of Statistics
- Australian Hydrographic Office
- Bureau of Meteorology (Australian Government)
- CSIRO
- Curtin University of Technology
- Department of Defence (Australian Government)
- Department of Human Services (Australian Government)
- Geoscience Australia
- InterGovernmental Committee on Surveying and Mapping
- Spatial Industries Business Association

This Standard was issued in draft form for comment as DR AS ISO 19150.2:2018.

Keeping Standards up-to-date

Ensure you have the latest versions of our publications and keep up-to-date about Amendments, Rulings, Withdrawals, and new projects by visiting:

www.standards.org.au

www.standards.govt.nz

ISBN 978 1 76072 113 8

Australian/New Zealand Standard™

Geographic information — Ontology

Part 2: Rules for developing ontologies in the Web Ontology Language (OWL)

First published as AS ISO 19150.2:2018.
Reissued and redesignated incorporating Amendment No 1 (October 2019) as AS/NZS
ISO 19150.2:2018.

COPYRIGHT

© ISO 2018 — All rights reserved
© Standards Australia Limited/the Crown in right of New Zealand, administered by the New
Zealand Standards Executive 2018

All rights are reserved. No part of this work may be reproduced or copied in any
form or by any means, electronic or mechanical, including photocopying, without the
written permission of the publisher, unless otherwise permitted under the Copyright
Act 1968 (Cth).

Preface

A1 This Standard was prepared by the Joint Standards Australia/Standards New Zealand Committee IT-004, Geographical Information/Geomatics.

After consultation with stakeholders in both countries, Standards Australia and Standards New Zealand decided to amend this Standard to an Australian/New Zealand Standard.

Amendment No. 1 to this Standard was prepared by the Joint Standards Australia/Standards New Zealand Committee IT-004, Geographical Information/Geomatics. As a consequence of Amendment No. 1, this Standard will be redesignated from AS ISO 19150.2 to AS/NZS ISO 19150.2.

The objective of this Standard is to define rules and guidelines for the development of ontologies to support better the interoperability of geographic information over the Semantic Web. The Web Ontology Language (OWL) is the language adopted for ontologies.

This part of AS/NZS ISO 19150 defines the conversion of the UML static view modelling elements used in the ISO geographic information standards into OWL. It further defines conversion rules for describing application schemas based on the General Feature Model defined in AS ISO 19109 into OWL.

This part of AS/NZS ISO 19150 does not define semantics operators, rules for service ontologies, and does not develop any ontology.

This Standard is identical with, and has been reproduced from, ISO 19150-2:2015, *Geographic information — Ontology — Part 2: Rules for developing ontologies in the Web Ontology Language (OWL)*.

As this document has been reproduced from an International Standard, the following applies:

- (a) In the source text 'this part of ISO 19150' should read 'this Australian/New Zealand Standard'.
- (b) A full point substitutes for a comma when referring to a decimal marker.

Australian or Australian/New Zealand Standards that are identical adoptions of international normative references may be used interchangeably. Refer to the online catalogue for information on specific Standards.

The terms 'normative' and 'informative' are used in Standards to define the application of the appendices or annexes to which they apply. A 'normative' appendix or annex is an integral part of a Standard, whereas an 'informative' appendix or annex is only for information and guidance.

Contents

Preface	ii
Foreword	v
Introduction	vi
1 Scope	1
2 Conformance	1
3 Normative references	1
4 Terms, definitions, abbreviations, and namespaces	2
4.1 Terms and definitions.....	2
4.2 Abbreviations.....	6
4.3 Namespaces.....	7
5 Namespace	7
6 Rules for mapping ISO geographic information UML models to OWL ontologies	8
6.1 General.....	8
6.2 Name.....	9
6.2.1 Scoping and namespaces.....	9
6.2.2 Ontology name.....	10
6.2.3 RDF namespace for ontology.....	10
6.2.4 Class name.....	11
6.2.5 Datatype name.....	11
6.2.6 Property name.....	11
6.2.7 Names for codelists and their members.....	12
6.3 Package.....	13
6.3.1 UML notation.....	13
6.3.2 OWL notation.....	13
6.3.3 Rules.....	13
6.4 Class.....	15
6.4.1 UML notation.....	15
6.4.2 OWL notation.....	15
6.4.3 Rules.....	15
6.5 Abstract class.....	16
6.5.1 UML notation.....	16
6.5.2 OWL notation.....	17
6.5.3 Rules.....	17
6.6 Attribute.....	18
6.6.1 UML Notation.....	18
6.6.2 OWL notation.....	19
6.6.3 Rules.....	21
6.7 Enumerated type.....	23
6.7.1 Enumeration.....	23
6.7.2 Code list.....	26
6.8 Union class.....	29
6.8.1 UML notation.....	29
6.8.2 OWL notation.....	30
6.8.3 Rules.....	30
6.9 Multiplicity.....	31
6.9.1 UML notation.....	31
6.9.2 OWL notation.....	31
6.9.3 Rules.....	31
6.10 Relationship.....	38
6.10.1 Generalization/inheritance.....	38
6.10.2 Association.....	40
6.10.3 Aggregation.....	41

6.11	Constraint	45
6.11.1	UML notation	45
6.11.2	OWL notation	46
6.11.3	Rules	46
7	Rules for formalizing an application schema in OWL	47
7.1	General	47
7.2	Rules for identification	49
7.3	Rules for ontology documentation	50
7.3.1	Ontology documentation	50
7.3.2	Ontology component documentation	50
7.4	Rules for integration	51
7.5	Rules for FeatureType	51
7.6	PropertyType	52
7.6.1	Attribute	52
7.6.2	Rules for Operation	58
7.6.3	Rules for FeatureAssociationRole	58
7.7	Rules for FeatureAssociationType	58
7.8	Rules for FeatureAggregationType	59
7.9	Rules for FeatureCompositionType	60
7.10	Rules for SpatialAssociationType	60
7.11	Rules for TemporalAssociationType	60
7.12	Rules for InheritanceRelation	61
7.13	Rules for constraints	61
7.14	Rules for ValueAssignment	61
7.14.1	Role of Association class	61
7.14.2	ValueAssignment property	61
7.14.3	RDF reification pattern	62
7.14.4	SPARQL named-graph pattern	64
7.14.5	Rules for ValueAssignment in OWL tern	65
Annex A	(normative) Abstract test suite	66
Annex B	(normative) Namespaces and component names for geographic information ontologies	85
Annex C	(informative) Augmented Backus Naur Form Notation	87
Annex D	(normative) "base" ontology	88
Annex E	(informative) Application ontology: The PropertyParcel example	90
Bibliography		101
AMENDMENT CONTROL SHEET		104

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2. www.iso.org/directives

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received. www.iso.org/patents

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT) see the following URL: [Foreword - Supplementary information](#).

The Committee responsible for this document is ISO/TC 211, *Geographic information/Geomatics*.

ISO 19150 consists of the following parts, under the general title *Geographic information — Ontology*:

- *Part 1: Framework*
- *Part 2: Rules for developing ontologies in the Web Ontology Language (OWL)*

Semantic operators, Service ontology, Domain ontology registry and Service ontology registry are planned to be covered in future Parts.

Introduction

Fundamentally, ontology comes from philosophy and refers to the study of the nature of the world itself. The information technology and artificial intelligence communities borrowed this term of ontology for the explicit specification of a conceptualization.^[2] Information technology and artificial intelligence consider that reality may be abstracted differently depending on the context from which “things” are perceived and, as such, recognize that multiple ontologies about the same part of reality may exist. In geographic information, ontology refers to a formal representation of phenomena of a universe of discourse with an underlying vocabulary including definitions and axioms that make the intended meaning explicit and describe phenomena and their interrelationships.^[1] An ontology can be formalized differently ranging from weak to strong semantics: taxonomy, thesaurus, conceptual model, logical theory.^[2]

Ontology is a fundamental notion for semantic interoperability on the Semantic Web since it defines the meaning of data and describes it in a format that machines and applications can read. As such, an application using data also has access to their inherent semantics through the ontology associated with it. Therefore, ontologies can support integration of heterogeneous data captured by different communities by relating them based on their semantic similarity. The W3C has proposed the Web Ontology Language (OWL) family of knowledge representation languages for authoring ontologies characterised by formal semantics on the Web.^[4,11]

Semantics is an important topic in the field of geographic information. The meaning of geographic information is essential for their discovery, sharing, integration, and use. Geographic information standards have recognized this fact in the standards on rules for application schema (ISO 19109) and the methodology for feature cataloguing (ISO 19110),^[7] which are both based on the General Feature Model (GFM). Basically, semantics relates phenomena and signs used to represent them (i.e. data) by the way of concepts. Typically, concepts are maintained in repositories called ontologies.

The ISO geographic information standards have chosen the conceptual modelling language UML^[10,12] for the formal representation of abstraction of the reality. Additionally as introduced in ISO/TS 19150-1:2012, there is a need to provide formal representation of abstraction of the reality in OWL to support the Semantic Web. Accordingly, this part of ISO 19150 defines rules to convert UML static views of geographic information and application schemas into OWL ontologies in order to benefit and support interoperability of geographic information over the Semantic Web. These rules are required for:

- ontology description completeness;
- consistency in the set of OWL ontologies for geographic information;
- consistency in conversion of UML diagrams to OWL ontologies; and
- cohesion and unity between UML models and ontologies.

These rules are based on but also extend the OMG's Ontology Definition Metamodel.^[11] OWL ontologies are complementary to UML static views and serve different purposes.

Australian/New Zealand Standard

Geographic information — Ontology

Part 2: Rules for developing ontologies in the Web Ontology Language (OWL)

1 Scope

This part of ISO 19150 defines rules and guidelines for the development of ontologies to support better the interoperability of geographic information over the Semantic Web. The Web Ontology Language (OWL) is the language adopted for ontologies.

This part of ISO 19150 defines the conversion of the UML static view modeling elements used in the ISO geographic information standards into OWL. It further defines conversion rules for describing application schemas based on the General Feature Model defined in ISO 19109 into OWL.

This part of ISO 19150 does not define semantics operators, rules for service ontologies, and does not develop any ontology.

2 Conformance

Any application ontology or profile claiming conformance with this part of ISO 19150 shall pass the requirements described in the abstract test suite, presented in [Annex A](#).

The abstract test suite is organized in two conformance classes that address the following purposes:

- Conversion of a UML package from the ISO/TC 211 Harmonized Model to OWL (conformance class 19150-2owl-conf); and
- Formalization of an application schema in OWL (conformance class 19150-2app-conf).

3 Normative references

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

ISO 19103:—¹⁾, *Geographic information — Conceptual schema language*

ISO 19107:2003, *Geographic information — Spatial schema*

ISO 19108:2002, *Geographic information — Temporal schema*

ISO 19109:—²⁾, *Geographic information — Rules for application schema*

ISO 19112:2003, *Geographic information — Spatial referencing by geographic identifiers*

ISO 19115-1:2014, *Geographic information — Metadata — Part 1: Fundamentals*

ISO 19123:2005, *Geographic information — Schema for coverage geometry and functions*

ISO 19156:2011, *Geographic information — Observations and measurements*

ISO 19157:2013, *Geographic information — Data quality*

W3C OWL 2, *OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax* (W3C Recommendation 27 October 2009)

1) To be published. (Revision of ISO/TS 19103:2005)

2) To be published. (Revision of ISO 19109:2005)

W3C OWL 2 RDF, *OWL 2 Web Ontology Language RDF-Based Semantics* (W3C Recommendation 27 October 2009)

W3C SKOS, *SKOS Simple Knowledge Organization System Reference* (W3C Recommendation 18 August 2009)

IETF RFC 5234, *Augmented BNF for Syntax Specifications: ABNF*

IETF RFC 3986, *Uniform Resource Identifier (URI): Generic Syntax*

4 Terms, definitions, abbreviations, and namespaces

4.1 Terms and definitions

4.1.1

aggregation

<UML> special form of *association* ([4.1.6](#)) that specifies a whole-part relationship between the aggregate (whole) and a component part

Note 1 to entry: See *composition* ([4.1.12](#)).

[SOURCE: ISO 19103:—1], 4.1]

4.1.2

annotation

<OWL> additional information associated to ontologies, entities, and axioms

[SOURCE: OWL]

4.1.3

annotation property

<OWL> element used to provide a textual *annotation* ([4.1.2](#)) for an *ontology* ([4.1.29](#)), axiom, or an IRI

[SOURCE: OWL]

4.1.4

application ontology

ontology ([4.1.29](#)) representing the concepts and relationships in an *application schema* ([4.1.5](#))

4.1.5

application schema

conceptual schema ([4.1.14](#)) for *data* ([4.1.16](#)) required by one or more applications

[SOURCE: ISO 19101-1:2014, 4.1.2]

4.1.6

association

<UML> semantic relationship that can occur between typed instances

Note 1 to entry: A binary association is an association among exactly two classifiers (including the possibility of an association from a classifier to itself).

[SOURCE: OMG UML, Superstructure, version 2.4.1, 7.3.3]

4.1.7

attribute

<UML> feature within a classifier that describes a range of values that instances of the classifier may hold

[SOURCE: ISO 19103:—1], 4.5]

4.1.8**cardinality**

<UML> number of elements in a set

Note 1 to entry: Contrast with *multiplicity* ([4.1.24](#)), which is the range of possible cardinalities a set may hold.

[SOURCE: ISO 19103:—1], 4.6]

4.1.9**class**

<OWL> set of *individuals* ([4.1.20](#))

[SOURCE: OWL]

4.1.10**class**

<UML> description of a set of objects that share the same *attributes* ([4.1.7](#)), *operations* ([4.1.30](#)), methods, relationships, and semantics

[SOURCE: ISO 19103:—1], 4.7]

4.1.11**codelist**

value domain including a code for each permissible value

[SOURCE: ISO 19136:2007, 4.1.7]

4.1.12**composition**

<UML> *aggregation* ([4.1.1](#)) where the composite object (whole) has responsibility for the existence and storage of the composed objects (parts)

[SOURCE: ISO 19103:—1], 4.10]

4.1.13**conceptual model**

model that defines concepts of a *universe of discourse* ([4.1.36](#))

[SOURCE: ISO 19101-1:2014, 4.1.5]

4.1.14**conceptual schema**

formal description of a *conceptual model* ([4.1.13](#))

[SOURCE: ISO 19101-1:2014, 4.1.6]

4.1.15**constraint**

<UML> condition or restriction expressed in natural language text or in a machine readable language for the purpose of declaring some of the semantics of an element

[SOURCE: ISO 19103:—1], 4.13]

4.1.16**data**

reinterpretable representation of information in a formalized manner suitable for communication, interpretation, or processing

[SOURCE: ISO/IEC 2382:2015, 2121272]

4.1.17**data property**

<OWL> semantic *association* ([4.1.6](#)) between an *individual* ([4.1.20](#)) and a typed *literal* ([4.1.21](#))

Note 1 to entry: Data properties were sometimes referred to as ‘concrete properties’ in Description Logic.

[SOURCE: OWL]

4.1.18**datatype**

<OWL> entities that refer to a set of concrete *data* ([4.1.16](#)) values

EXAMPLE xsd:string, xsd:integer, xsd:decimal

Note 1 to entry: Datatypes are distinct from *classes* ([4.1.9](#)) of *individuals* ([4.1.20](#)), the latter are denoted by URIs and may be used by reference.

[SOURCE: OWL]

4.1.19**generalization**

<UML> taxonomic relationship between a more general element and a more specific element of the same element type

Note 1 to entry: An instance of the more specific element can be used where the more general element is allowed.

[SOURCE: ISO 19103:—1], 4.18]

4.1.20**individual**

instance of a *class* ([4.1.9](#))

Note 1 to entry: It refers to a resource belonging to the extension of the class.

[SOURCE: Adapted from the OWL Web Ontology Language Guide]

4.1.21**literal value****literal**

constant, explicitly specified value

EXAMPLE "1"^^xsd:integer, "abc"^^xsd:string

Note 1 to entry: This contrasts with a value that is determined by resolving a chain of substitution (e.g. a variable).

[SOURCE: ISO 19143:2010, 4.15]

4.1.22**localName**

reference to a local object directly accessible from a *namespace* ([4.1.25](#))

[SOURCE: ISO 19103:—1], modified – Derived from 7.5.5.1]

4.1.23**metadata**

information about a resource

[SOURCE: ISO 19115-1:2014, 4.10]

4.1.24**multiplicity**

<UML> specification of the range of allowable cardinalities that a set may assume

[SOURCE: ISO 19103:—1), 4.24]

4.1.25**namespace**

<general> domain in which names, expressed by character strings, can be mapped to objects

Note 1 to entry: The names can be subject to local *constraints* ([4.1.15](#)) enforced by the namespace.

[SOURCE: ISO 19103:—1), modified – Derived from 7.5.2.1]

4.1.26**namespace**

<RDF> common URI prefix or stem used in identifiers for a set of related resources

Note 1 to entry: The RDF namespace is concatenated with the *localName* ([4.1.22](#)) to create the complete URI identifier for an RDF resource. Every RDF resource is identified by a URI. In contrast, an XML namespace URI scopes a local XML component name, but there is no rule for combining these into a single identifier string.

4.1.27**namespace**

<XML> collection of names, identified by a URI reference, that are used in XML documents as element names and *attribute* ([4.1.7](#)) names

[SOURCE: ISO/TS 19139:2007, 4.1]

4.1.28**object property**

<OWL> semantic *association* ([4.1.6](#)) between a pair of *individuals* ([4.1.20](#))

Note 1 to entry: Object properties have sometimes been referred to as ‘abstract properties’ in Description Logic.

[SOURCE: OWL]

4.1.29**ontology**

formal representation of phenomena of a *universe of discourse* ([4.1.36](#)) with an underlying vocabulary including definitions and axioms that make the intended meaning explicit and describe phenomena and their interrelationships

[SOURCE: ISO 19101-1:2014, 4.1.26]

4.1.30**operation**

<UML> behavioural <UML> feature of a classifier that specifies the name, type, parameters, and *constraints* ([4.1.15](#)) for invoking an associated behaviour

[SOURCE: ISO 19103:—1), 4.26]

4.1.31**package**

<UML> general purpose mechanism for organizing elements into groups

Note 1 to entry: A package provides a *namespace* ([4.1.27](#)) for the grouped elements.

[SOURCE: ISO 19103:—1), 4.27]

4.1.32

property restriction

<OWL> special kind of *class* ([4.1.9](#)) description through the definition of *constraints* ([4.1.15](#)) on values and cardinalities

[SOURCE: OWL]

4.1.33

qualified cardinality

<OWL> *cardinality* ([4.1.8](#)) restriction that applies to *literals* ([4.1.21](#)) or *individuals* ([4.1.20](#)) that are connected by a *data property* ([4.1.17](#)) or an *object property* ([4.1.28](#)) and are instance of the qualifying range [*datatype* ([4.1.18](#)) or *class* ([4.1.9](#))]

[SOURCE: OWL]

4.1.34

source document

document that contains the original definition of a resource

4.1.35

stereotype

<UML> extension of an existing metaclass that enables the use of platform or domain specific terminology or notation in place of, or in addition to, the ones used for the extended metaclass

[SOURCE: ISO 19103:—¹), 4.33]

4.1.36

universe of discourse

view of the real or hypothetical world that includes everything of interest

[SOURCE: ISO 19101-1:2014, 4.1.38]

4.1.37

unqualified cardinality

<OWL> *cardinality* ([4.1.7](#)) restriction that applies to all *literals* ([4.1.21](#)) or *individuals* ([4.1.20](#)) that are connected by a *data property* ([4.1.17](#)) or an *object property* ([4.1.28](#))

[SOURCE: OWL]

4.2 Abbreviations

DL	Description Logic
IRI	Internationalized Resource Identifier
MOF	MetaObject Facility
OMG	Object Management Group
OWL	Web Ontology Language (version 2)
RDF	Resource Description Framework
RDFS	RDF Schema
SKOS	Simplified Knowledge Organization System
UML	Unified Modeling Language
URI	Universal Resource Identifier

XML eXtensible Markup Language

XSD XML Schema Definition

4.3 Namespaces

dc	Dublin Core http://purl.org/dc/elements/1.1/ [5]
dct	Dublin Core http://purl.org/dc/terms/
owl	Web Ontology Language http://www.w3.org/2002/07/owl#
rdf	Resource Description Framework http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs	RDF Schema http://www.w3.org/2000/01/rdf-schema#
skos	Simple Knowledge Organization System http://www.w3.org/2004/02/skos/core#
19150-2owl	Requirements class for conversion of a UML package from the ISO/TC 211 Harmonized Model to OWL http://standards.iso.org/iso/19150-2/req/owl
19150-2owl-conf	Conformance class for conversion of a UML package from the ISO/TC 211 Harmonized Model to OWL http://standards.iso.org/iso/19150-2/conf/owl
19150-2app	Requirements class for formalization of an application schema in OWL http://standards.iso.org/iso/19150-2/req/applicationSchema
19150-2app-conf	Conformance class for formalization of an application schema in OWL http://standards.iso.org/iso/19150-2/conf/applicationSchema
iso19150-2	Base ontology elements required for formalization of UML models and application schemas in OWL http://def.isotc211.org/iso19150-2/2012/base#
exPk	Generic prefix for examples for UML and conceptual schema language rules http://def.isotc211.org/example/aPackage#
exPkCode	Generic prefix for examples of code-lists for UML and conceptual schema language rules http://def.isotc211.org/example/aPackage/code/
myapp	Dummy prefix for examples, representing an application schema namespace
xsd	XML Schema Definition http://www.w3.org/2001/XMLSchema#
gfm	ontology describing the General Feature Model (ISO 19109:—2)

5 Namespace

A namespace is a collection of names identified by a URI reference.[13] The definition of namespaces shall follow the rules for URI definition as documented in [Annex B](#). In RDF applications (including OWL) every resource, including definitions and datatypes as well as individuals, is identified by an IRI.

NOTE The prefix that references a namespace within an ontology in OWL is identified in the namespace declaration as part of the ontology header element (see [6.3](#)).

6 Rules for mapping ISO geographic information UML models to OWL ontologies

6.1 General

As introduced in ISO/TS 19150-1:2012, Clause 6 defines the conversion of UML static view modeling elements that are used in the ISO geographic information standards into OWL. These elements are name, package, class, stereotype, enumeration, code list, attribute, multiplicity, generalization/inheritance, association, aggregation, composition, and constraints.

ISO 19103:—¹) defines the profile of UML used in conceptual modelling in ISO geographic information standards. Among others, three important aspects of the UML profile are:

- a) every navigable association-end must have a role-name (ISO 19103:—¹), [6.8.2](#));
- b) class stereotypes «CodeList» and «Union» indicate a special behaviour different to normal classes (ISO 19103:—¹), [6.10](#));
- c) a set of primitive datatypes are provided.

Item a) means that the UML models under consideration map easily to the RDF model where all properties have semantic names. This allows for a more streamlined mapping from UML to OWL than, for example, the generic approach taken by OMG, which supports all the options implied by the MOF and UML meta model.

Item b) means that different UML-OWL transformation rules are required for classes with these stereotypes compared with standard classes. There are also standard stereotypes provided by UML, such as «enumeration».

Item c) requires that mappings from the UML classes representing datatypes to specific constructs using RDF, RDFS, OWL and the XSD datatypes accessible to OWL be defined.

The conversion rules are limited to OWL 2 RL, meaning that OWL RL shall be used for the definitions of the rules (see W3C OWL 2 and W3C OWL 2 RFD).

NOTE OWL2 RL profile corresponds approximately with OWL version 1 DL profile. This profile ensures a level of computability which is generally considered desirable for rigorous ontologies.

[Clause 6](#) uses Augmented Backus Naur Form notation (see IETF RFC 5234), which is summarized in [Annex C](#).

This part of ISO 19150 requires the use of standard HTTP URIs to identify resources in geographic information for the purpose of ontologies. The URI structures are defined in [Annex B](#).

This part of ISO 19150 requires an ontology that defines additional annotation properties, properties and classes to support the representation of ISO geographic information UML models into OWL ontologies. This ontology and its namespace are documented in [Annex D](#).

The requirements for representing a UML package from a standard in the series of ISO geographic information standards in OWL comprise a single requirements class ([Table 1](#)), identified as <http://standards.iso.org/iso/19150-2/req/2owl> and abbreviated as 19150-2owl.

Table 1 — Requirements class for representing a UML package from a standard in the series of ISO geographic information standards in OWL

Requirements class	
19150-2package = http://standards.iso.org/iso/19150-2/req/package	
Target type	Ontology
Dependency	http://www.w3.org/TR/owl2-syntax/ (OWL)
Dependency	http://tools.ietf.org/html/rfc3986 (URI Syntax)

Table 1 (continued)

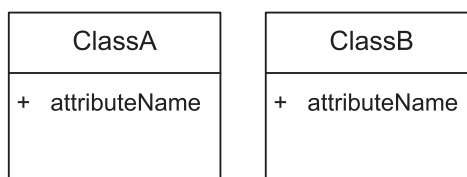
Requirements class	
19150-2package = http://standards.iso.org/iso/19150-2/req/package	
Dependency	http://standards.iso.org/iso/19103/ed-2/en/ (Conceptual schema language)
Requirement	19150-2package:name
Requirement	19150-2package:ontologyName
Requirement	19150-2package:rdNamespace
Requirement	19150-2package:className
Requirement	19150-2package:datatypeName
Requirement	19150-2package:propertyName
Requirement	19150-2package:codeName
Requirement	19150-2package:package
Requirement	19150-2package:class
Requirement	19150-2package:abstractClass
Requirement	19150-2package:attribute-dataProperty
Requirement	19150-2package:attribute-objectProperty
Requirement	19150-2package:enumeration
Requirement	19150-2package:codelist
Requirement	19150-2package:codelistextension
Requirement	19150-2package:union
Requirement	19150-2package:multiplicity
Requirement	19150-2package:relationship-generalization
Requirement	19150-2package:relationship-association
Requirement	19150-2package:relationship-aggregation
Requirement	19150-2package:constraint

6.2 Name

6.2.1 Scoping and namespaces

The first set of requirements deals with the construction of URIs used to identify ontology namespaces, classes, datatypes and properties, and is summarized in [Annex B](#).

UML sets a number of restrictions and conventions on element names. Key restrictions are that (a) each class shall have a unique name within the context of a package, and (b) each attribute and role has a unique name within the context of a class. Hence, a UML package provides the namespace for its classes, and a UML class provides the namespace for its attributes and association roles. Therefore, an attribute name *attributeName* that is used in a *ClassA* can also be used in a *ClassB* both having a specific semantics in the context of each class ([Figure 1](#)). The class is therefore a namespace for its attributes.

**Figure 1** — UML class and attribute names

Naming rules in OWL inherit from RDF and are different to those in UML. Each resource (Class, DataType, Property) is denoted by a URI. A set of related resources are usually denoted by URIs with

the same stem or base URI, known as the RDF Namespace. Within the context of an RDF namespace the localName for each class, datatype, and property must be unique and distinct from each other.

NOTE 1 OWL Ontologies are identified with a URI which is usually closely related to the namespace URI for the elements of the ontology. Under such a naming convention the resources are scoped to the ontology.

NOTE 2 The URIs used in an RDF document can be abbreviated using the QName syntax [XML Namespace], where the prefix stands for the URI of an RDF namespace.

NOTE 3 In OWL 2 the same URI can denote both a class and an individual. This is called 'punning'. However, use of punning limits reasoning performance and behaviour, so is generally avoided if possible. The profile of OWL used in this part of ISO 19150 does not allow punning.

The key resources that need naming within an OWL representation of a model are packages, classes, attributes and association roles.

[Table 2](#) sets the general requirements for name.

Table 2 — General name requirement

Requirement
19150-2package:name
URIs for all resources in geographic information shall the these representation restrictions:
— no space characters;
— dash and underscore characters allowed;
— other punctuation characters from the source are replaced by underscore characters.

6.2.2 Ontology name

The ontology name is constructed by appending an abbreviation of the package name to a base URI corresponding to the document or standard, with the "/" used as the separator. [Table 3](#) sets the requirement for ontology name.

Table 3 — Ontology name requirement

Requirement
19150-2package:ontologyName
The ontology name URI shall be based on the name of the corresponding UML package, according to the expression: ontologyName = URIBase "/" umlPackageName
— URIBase is a base URI in a domain owned by the organization that maintains the model or ontology (see Annex B),
— umlPackageName is an abbreviation for the name of the UML package corresponding to the ontology, which also follows the syntax rules for a segment as described in RFC 3986, 3.3.
umlPackageName shall follow these representation restrictions:
— lower case;
— only the semantic part of the package name is represented (see Annex B for examples).

6.2.3 RDF namespace for ontology

The ontology namespace is constructed by appending a "#" character at the end of the ontology name. [Table 4](#) sets the requirement for RDF namespace for ontology.

Table 4 — RDF namespace for ontology

Requirement
19150-2package:rdfNamespace
The RDF namespace for ontology items shall be based on the ontology name of the corresponding UML package, according to the expression: <code>rdfNamespace = ontologyName "#"</code>

NOTE The generic URI syntax uses the “#” style of URI to identify resources that are secondary to a primary resource (RFC 3986 §3.5). This is the most common pattern used to identify components of ontologies that describe classes and properties. As HTTP does not transmit the fragment after the “#” to the server, a request for a resource identified in this way results in the primary resource being retrieved, i.e. in this case a complete ontology.

6.2.4 Class name

Each class name is constructed by appending the UML class name to the RDF namespace. [Table 5](#) sets the requirement for class name in ontology.

Table 5 — Class name

Requirement
19150-2package:className
The class URI shall be based on the name of the corresponding UML class, according to the expression: <code>className = rdfNamespace umlClassName</code> — <code>umlClassName</code> is the name of the corresponding UML Class. <code>umlClassName</code> shall follow these representation restrictions: — upper camel case.

6.2.5 Datatype name

Datatype name follows a similar pattern to class name. The local name for a datatype is appended to the RDF namespace. [Table 6](#) sets the requirement for datatype name in ontology.

Table 6 — Datatype name

Requirement
19150-2package:datatypeName
The datatype URI shall be based on the name of the corresponding UML class, according to the expression: <code>datatypeName = rdfNamespace datatypeLocalName</code> — <code>datatypeLocalName</code> is the local name of the datatype. <code>datatypeLocalName</code> shall follow these representation restrictions: — upper camel case.

6.2.6 Property name

Attribute names are not required to be unique within a package in UML. However, it is uncommon to have multiple attributes with the same name with different semantics within a package. This part of ISO 19150 identifies a property by its `localName` scoped to the package where it is unique within the package, or scoped to its class where the same property name appears more than once in a package with different meanings. [Table 7](#) sets the requirement for property name in ontology.

Table 7 — Property name

Requirement
19150-2package:propertyName
<p>The property URI shall be based on the name of the corresponding UML attribute or association role name according to the expression:</p> <pre>propertyName = rdfNamespace [umlClassName "."] propertyLocalName propertyLocalName = umlAttributeName / umlRoleName</pre> <ul style="list-style-type: none"> — umlAttributeName means the name of a UML attribute, — umlRoleName means the name of a UML association role. <p>umlAttributeName and umlRoleName shall follow these representation restrictions:</p> <ul style="list-style-type: none"> — lower camel case. <p>NOTE In addition to its propertyLocalName, a propertyName can optionally include its umlClassName when the property is scoped to the class or if the same propertyLocalName may be used by another property of another class with a different semantics.</p>

6.2.7 Names for codelists and their members

Codelist classes and their members require specific treatment. As described in [6.8.2](#), a codelist class shall be implemented in OWL/RDF as both an OWL Class and as a SKOS ConceptScheme, and its members as SKOS Concepts.

SKOS (Simplified Knowledge Organization System) is a W3C standard that has been broadly adopted for vocabulary formalization. SKOS is formulated as an OWL ontology, and instances can be integrated with an OWL ontology. SKOS supports the codelist requirements of membership and extensibility. SKOS also supports the recording of basic thesaurus-like semantic relationships between members (broader, narrower, etc).

SKOS ConceptScheme and SKOS Concept resources are each individuals, as they have a specific *rdf:type* property that relates them to the class to which they belong. In the OWL context, the distinction between individuals and other ontology elements is important, and is often reflected in the use of different styles of URI to denote these different kinds of resources.

In particular, while classes and properties are normally used primarily in the context of the other classes and properties in an ontology, individuals are commonly accessed individually. Hence, while '#' URIs are commonly used to denote classes and properties in an ontology, it is common to denote individuals using '/' URIs. [Table 8](#) sets the requirement for code names in ontology.

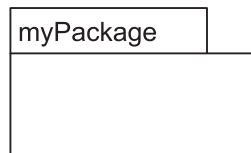
Table 8 — Code name

Requirement
19150-2package:codeName
<p>The URI for a codelist shall be based on the name of the corresponding UML class stereotyped «CodeList» according to the expression:</p> <pre>codeNamespace = ontologyName "/code/" conceptSchemeName = codeNamespace className collectionName = codeNamespace className "Collection"</pre> <p>— className means the name of the class stereotyped «CodeList».</p> <p>className shall follow these representation restrictions:</p> <p>— upper camel case.</p> <p>The URI for a codelist value shall be based on the name of the corresponding UML class stereotyped «CodeList» and the name of the codelist value (presented in UML similarly as a class attribute) according to the expression:</p> <pre>conceptName = conceptSchemeName "/" umlCodelistValueName</pre> <p>— umlCodelistValueName means the name of a codelist value.</p> <p>umlCodelistValueName shall follow these representation restrictions:</p> <p>— lower camel case.</p>

6.3 Package

6.3.1 UML notation

PACKAGE is the UML mechanism to group elements and to provide a namespace for the members of the group. A package may contain other packages. The UML notation of a PACKAGE is depicted as in [Figure 2](#).

**Figure 2 — UML PACKAGE notation**

PACKAGES may be stereotyped to extend their semantics.

6.3.2 OWL notation

In OWL, 'ontology' is the resource that provides for the grouping of elements. It may be uniquely identified by an IRI (see [Annex B](#)). It may also provide additional information with ontology annotation such as the version, the name or comments. It may also import other ontologies.

6.3.3 Rules

The UML model of each ISO geographic information standard corresponds to one or more Ontology <OWL> using an *owl:Ontology* declaration. An *rdfs:label* annotation property declaration provides the full name of the corresponding UML PACKAGE.

A *dct:source* annotation property declaration provides the title of the Ontology <OWL> reference document or standard and, when applicable, the hierarchy of PACKAGE names.

An *owl:versionInfo* declaration provides the date corresponding to the reference document or standard. If no document or standard corresponds to the Ontology <OWL>, the *owl:versionInfo* declaration provides the date of the official release of the Ontology <OWL>. The date is given according to ISO 8601. [6]

In the case where a UML PACKAGE depends on one or more other packages, each dependency package shall correspond to a specific Ontology <OWL>. The *owl:imports* declaration is used to set the dependency between Ontologies <OWL>.

Table 9 sets the requirement for the description of PACKAGES in Ontology <OWL>.

Table 9 — Package

Requirement
19150-2package:package
<p>The UML model of each ISO geographic information standard shall correspond to one or more ontologies.</p> <p>An Ontology <OWL> shall import the ontology denoted <code>http://def.isotc211.org/iso19150/-2/2012/base</code> to include the resources supporting this International Standard.</p> <p>An Ontology <OWL> shall be annotated with:</p> <ul style="list-style-type: none"> — the full name of the corresponding UML PACKAGE, using <i>rdfs:label</i>, — the source for the ontology, using <i>dct:source</i> for a citation for the standard containing the definition, according to the expression: <pre>title = [documentNumber " ,"] documentTitle [*[" : " packageName]]</pre> <ul style="list-style-type: none"> — documentNumber means the number of the document; — documentTitle means the title of the document; — packageName means the name of the package; — the version date of the reference document or the ontology, using <i>owl:versionInfo</i>. <p>NOTE Integration of both the resources of this part of ISO 19150, and other dependencies between ontologies, use the <i>owl:imports</i> mechanism.</p>

Additional annotation for the Ontology <OWL> may be provided such as:

- human-readable description, using an *rdfs:comment* declaration,
- version IRI, using an *owl:versionIRI* declaration to specify the prior version of the Ontology <OWL>,
- information on prior version, using an *owl:priorVersion* declaration,
- backward compatibility, using an *owl:backwardCompatibleWith* declaration to specify the previous version of the Ontology <OWL> that is compatible with the current version of the containing Ontology <OWL> and using an *owl:incompatibleWith* declaration to specify the prior version of the Ontology <OWL> that is incompatible with the current version of the containing Ontology <OWL>,
- deprecation, using an *owl:deprecated* declaration with the value equal to `"true"^^xsd:boolean` if the Ontology <OWL> is deprecated, and
- superseding ontology, using a *dct:replaces* declaration to identify the Ontology <OWL> it supersedes and using a *dct:isReplacedBy* declaration to identify the Ontology <OWL> that has superseded it.

The following example illustrates the specification of an Ontology <OWL>.

NOTE This example and all the following examples are not complete definitions. The prefix descriptions provided in 4.3 can help reading the examples.

EXAMPLE

RDF/Turtle serialization

```

</iso19107/2003/SpatialSchema> a owl:Ontology ;
    rdfs:label "ISO 19107:2003 Spatial Schema" ;
    dct:source "ISO 19107:2003, Geographic information – Spatial schema" ;
    owl:versionInfo "2003-05-01" ;
    owl:imports <http://www.isotc211.org/tc211ontologies/iso19107/2003/7/Geometry> .

```

RDF/XML serialization

```

<rdf:RDF xmlns="http://def.isotc211.org">
    <owl:Ontology rdf:about="/iso19107/2003/SpatialSchema">
        <rdfs:label>ISO 19107:2003 Spatial Schema</rdfs:label>
        <dct:source>ISO 19107:2003, Geographic information – Spatial schema</dct:source >
        <owl:versionInfo>2003-05-01</owl:versionInfo>
        <owl:imports rdf:resource="http://www.isotc211.org/tc211ontologies/iso19107/2003/7/Geometry"/>
    </owl:Ontology>
</rdf:RDF>

```

6.4 Class

6.4.1 UML notation

CLASS is the UML classifier that describes a set of objects sharing the same specifications of features, constraints, and semantics. CLASSES may be abstract or concrete. The CLASS classifier may contain attributes and operations. The UML notation of a CLASS is depicted as in [Figure 3](#).

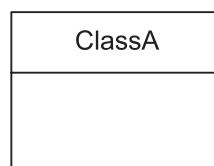


Figure 3 — UML CLASS notation

6.4.2 OWL notation

Class <OWL> is the resource corresponding to a set of individuals.

6.4.3 Rules

A UML CLASS corresponds to a Class <OWL> using an *owl:Class* declaration.

A Class <OWL> is annotated with a label, its definition, and the source of its definition. The label provides the name of the Class <OWL> as used in the UML CLASS and uses a *rdfs:label* declaration. The definition provides the semantic of the Class <OWL> and uses a *skos:definition* declaration. The source of the definition identifies the resource defining this Class <OWL>. It uses a *rdfs:isDefinedBy* declaration to specify the IRI of the source document.

[Table 10](#) sets the requirement for the description of CLASSES in Ontology <OWL>.

Table 10 — Class

Requirement
19150-2package:class
A UML CLASS shall correspond to a Class <OWL> using an <i>owl:Class</i> declaration. A Class <OWL> shall be annotated with:
— a label, using <i>rdfs:label</i> ,
— the source of the definition, using <i>rdfs:isDefinedBy</i> for the IRI of the source document.

Additional annotation for the Class <OWL> may be provided such as its definition, using a *skos:definition* declaration, and if the Class <OWL> is deprecated, using an *owl:deprecated* annotation property declaration with its value set to *"true"^^xsd:boolean*.

The following example illustrates the specification of a Class <OWL>.

EXAMPLE

RDF/Turtle serialization

```
exPk:ClassA a owl:Class ;
    rdfs:label "ClassA" ;
    skos:definition "ClassA definition" ;
    rdfs:isDefinedBy <http://sourceDefinitionIRI> .
```

RDF/XML serialization

```
<owl:Class rdf:about="&exPk;ClassA">
    <rdfs:label>ClassA</rdfs:label>
    <skos:definition>ClassA definition</skos:definition>
    <rdfs:isDefinedBy>http://sourceDefinitionIRI</rdfs:isDefinedBy>
</owl:Class>
```

6.5 Abstract class

6.5.1 UML notation

In UML, a CLASS can be specified as ABSTRACT. This means that such a CLASS cannot be instantiated and is usually subtyped by implementable or concrete CLASSES. Although they are not instantiable, ABSTRACT CLASSES are useful for classification purposes and, as such, provide important meaning. The UML notation of an ABSTRACT CLASS is depicted as in [Figure 4](#) with its name shown in italic characters.

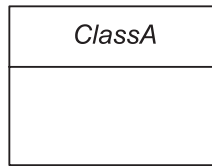


Figure 4 — UML ABSTRACT CLASS notation

6.5.2 OWL notation

OWL does not have a built-in mechanism to identify a Class <OWL> as abstract.

6.5.3 Rules

A UML ABSTRACT CLASS shall correspond to a Class <OWL>. The Class <OWL> is annotated to identify it as abstract.

To support identification of Class <OWL> as abstract, this part of ISO 19150 defines an annotation property *iso19150-2:isAbstract* formalized in the *base* ontology ([Annex D](#)):

OWL Definition 1

RDF/Turtle serialization

```

iso19150-2:isAbstract a owl:AnnotationProperty ;
rdfs:domain owl:Class;
rdfs:range xsd:Boolean .
  
```

RDF/XML serialization

```

<owl:AnnotationProperty rdf:about="&iso19150-2;isAbstract">
  <rdfs:range rdf:resource="&xsd:boolean"/>
  <rdfs:domain rdf:resource="&owl;Class"/>
</owl:AnnotationProperty>
  
```

This annotation property provides the mechanism for documenting if a Class <OWL> is abstract. *iso19150-2:isAbstract* is set to *"true"^^xsd:boolean* when a Class <OWL> is abstract. The default value is *false* when *iso19150-2:isAbstract* is not specified.

[Table 11](#) sets the requirement for the description of ABSTRACT CLASSES in Ontology <OWL>.

Table 11 — Abstract class

Requirement
19150-2package:abstractClass
A UML ABSTRACT CLASS shall correspond to a Class <OWL>. The Class <OWL> shall be annotated to identify it as abstract using an <i>iso19150-2:isAbstract</i> annotation property declaration.

The following example illustrates the identification of an abstract 'class.'

EXAMPLE

RDF/Turtle serialization

```
exPk:ClassA a owl:Class ;
  rdfs:label "ClassA" ;
  skos:definition "ClassA definition" ;
  rdfs:isDefinedBy <http://sourceDefinitionIRI> ;
  iso19150-2:isAbstract "true"^^xsd:boolean .
```

RDF/XML serialization

```
<owl:Class rdf:about="&exPk;ClassA">
  <rdfs:label>ClassA</rdfs:label>
  <skos:definition>ClassA definition</skos:definition>
  <rdfs:isDefinedBy>http://sourceDefinitionIRI</rdfs:isDefinedBy>
  <iso19150-2:isAbstract rdf:datatype="&xsd:boolean">true</iso19150-2:isAbstract>
</owl:Class>
```

6.6 Attribute

6.6.1 UML Notation

In UML, an **ATTRIBUTE** is a property related to a classifier (i.e., a class or a datatype) by the way of ownedAttribute. It associates an instance of a class to a value or set of values of the type specified in the attribute declaration. The general notation for an attribute is:

```
property = [visibility] ["/"] name [":"prop-type] ["[" multiplicity "]" ]
["=" default] [{" prop-modifier *["," prop-modifier] "}]
```

- visibility = "+" / "-" / "#" / "~"
- "+" means public;
- "-" means private;
- "#" means protected;
- "~" means package;
- "/" signifies that the property is derived;
- name is the name of the property;
- prop-type is the name of the type of the property;
- multiplicity is the multiplicity of the property. If this term is omitted, it implies a multiplicity of 1 (exactly one);
- default is an expression that evaluates to the default value or values of the property;

- `prop-modifier` = `"readOnly"` / `"union"` / `"subsets"` `property-name` / `"redefines"` `property-name` / `"ordered"` / `"unique"` / `"nonunique"` / `prop-constraint`
- `"readOnly"` means that the property is read only;
- `"union"` means that the property is a derived union of its subsets;
- `"subsets"` `property-name` means that the property is a proper subset of the property identified by `<property-name>`;
- `"redefines"` `property-name` means that the property redefines an inherited property identified by `<property-name>`;
- `"ordered"` means that the property is ordered;
- `"unique"` means that there are no duplicates in a multi-valued property;
- `prop-constraint` is an expression that specifies a constraint that applies to the property. It is usually shown in the compartment below the class name compartment of the class notation.

The UML notation of an ATTRIBUTE is depicted as in [Figure 5](#).

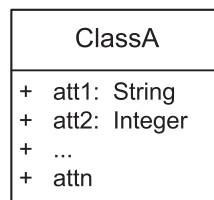


Figure 5 — UML ATTRIBUTE notation

6.6.2 OWL notation

OWL defines two kinds of resources for ATTRIBUTES: Data Property <OWL> and Object Property <OWL>.

A Data Property <OWL> connects a (literal) data value to the property of the Class <OWL>. It is defined with an *owl:DatatypeProperty* declaration. A Data Property <OWL> is connected to a Class <OWL> by specifying its domain with an *rdfs:domain* declaration. The target type of a Data Property <OWL> is specified by its range with an *rdfs:range* declaration. [Table 12](#) provides a mapping of the ISO 19103:—1) datatypes to OWL datatypes.

Table 12 — Datatype mapping

	ISO 19103 types	OWL datatypes	gcoDatatypes (GCOLiteral ^a)
Primitive types	CharacterString	xsd:string	gco:CharacterString
	Date	xsd:date (in conjunction with rdf:XMLLiteral)	gco:Date
	DateTime	xsd:dateTime	gco:DateTime
	Integer	xsd:integer	gco:Integer
	Decimal	xsd:decimal	gco:Decimal
	Real	xsd:double	gco:Real
	Boolean	xsd:boolean	gco:Boolean

Table 12 (continued)

	ISO 19103 types	OWL datatypes	gcoDatatypes (GCOLiteral ^a)
Name types	GenericName	<URI> xsd:Name	<i>gco:AbstractGenericName</i>
	LocalName	xsd:NCName	<i>gco:LocalName</i>
	ScopedName	<URI>	<i>gco:ScopedName</i>
	TypeName	xsd:NCName	<i>gco:TypeName</i>
	MemberName	xsd:Name	<i>gco:MemberName</i>
^a Allowed XML representations in conjunction with the use of <i>iso19150-2:GCOLiteral</i>			

Additionally, this part of ISO 19150 defines the *iso19150-2:GCOLiteral* datatype, which allows the representation of ISO 19103:—¹) datatypes in RDF based on the gco namespace as defined in ISO/TS 19139:2007^[9] (see [Table 12](#)).

OWL Definition 2

RDF/Turtle serialization

```
iso19150-2:GCOLiteral a rdfs:Datatype ;
    rdfs:label "GCOLiteral" ;
    rdfs:isDefinedBy <http://standards.iso.org/iso/ts/19139/ed-1/en/> ;
    owl:equivalentClass rdf:XMLLiteral .
```

RDF/XML serialization

```
<rdfs:Datatype rdf:about="&iso19150-2;GCOLiteral">
    <rdfs:label>GCOLiteral</rdfs:label>
    <rdfs:isDefinedBy>http://standards.iso.org/iso/ts/19139/ed-1/en/</rdfs:isDefinedBy>
    <owl:equivalentClass rdf:resource="&rdf;XMLLiteral"/>
</rdfs:Datatype>
```

The cardinality (i.e. the number of occurrences) for data properties can be from zero to multiple. How to set cardinality is specified in [6.10](#).

An Object Property <OWL> connects an instance or an individual of a Class <OWL> to the property of the Class <OWL>. It is defined with an *owl:ObjectProperty* declaration. An Object Property <OWL> is connected to a Class <OWL> by specifying its domain with an *rdfs:domain* declaration. The target type of an Object Property <OWL> is specified by its range with an *rdfs:range* declaration. The possible value types for the *rdfs:range* declaration are any valid Class <OWL>, The arity for Object Property <OWL> can be from zero to multiple. How to set arities is defined in [6.9](#).

6.6.3 Rules

6.6.3.1 OWL data property rules

A UML ATTRIBUTE described by a data value (e.g. number, character string, Boolean value, etc.) corresponds to a Data Property <OWL>. It is connected to the Class <OWL> it is associated with using an *rdfs:domain* declaration. The type of data values is specified using an *rdfs:range* declaration.

A Data Property <OWL> is annotated with a label and its source document. The label provides the name of the UML ATTRIBUTE and uses a *rdfs:label* declaration. The source document identifies the resource defining this UML ATTRIBUTE. It uses *rdfs:isDefinedBy* to declare the IRI of the source document.

[Table 13](#) sets the requirement for the description of ATTRIBUTES through Data Property <OWL> in Ontology <OWL>.

Table 13 — Attribute – OWL data property

Requirement
19150-2package:attribute-dataProperty
A UML ATTRIBUTE described by a data value shall correspond to a Data Property <OWL> and use an <i>owl:DatatypeProperty</i> declaration. It shall be connected to the Class <OWL> it is associated with using an <i>rdfs:domain</i> declaration. The type of data values shall be specified using an <i>rdfs:range</i> declaration.
A Data Property <OWL> shall be annotated with:
— a label for the name of the UML ATTRIBUTE, using <i>rdfs:label</i> ,
— a source document defining this UML ATTRIBUTE, using <i>rdfs:isDefinedBy</i> to declare the IRI of the resource.

Additional annotation for the Data Property <OWL> may be provided such as:

- definition, using a *skos:definition* declaration to provide the semantics of the UML ATTRIBUTE, and
- deprecation, using an *owl:deprecated* declaration, which is set to "true"^^*xsd:boolean* if the Data Property <OWL> is deprecated.

The following example illustrates the definition of a 'data property' for an attribute definition of a 'class'.

EXAMPLE

RDF/Turtle serialization

```
exPk:ClassA.att1 a owl:DatatypeProperty ;
    rdfs:label "att1" ;
    skos:definition "att1 definition" ;
    rdfs:isDefinedBy <http://sourceDefinitionIRI> ;
    rdfs:domain exPk:ClassA ;
    rdfs:range xsd:string .
```

RDF/XML serialization

```
<owl:DatatypeProperty rdf:about="&exPk;ClassA.att1">
    <rdfs:label>att1</rdfs:label>
    <rdfs:isDefinedBy>http://sourceDefinitionIRI</rdfs:isDefinedBy>
    <rdfs:domain rdf:resource="&exPk;ClassA"/>
```

```
<rdfs:range rdf:resource="&xsd:string"/>
</owl:DatatypeProperty>
```

NOTE As illustrated in this example, att1 is restricted to the domain of ClassA. This is consistent with the UML metamodel that bind attributes to classes. This UML to OWL conversion preserves this binding.

6.6.3.2 OWL object property rules

A UML ATTRIBUTE described by another UML CLASS corresponds to an Object Property <OWL>. It is connected to the Class <OWL> resources it is associated with using an *rdfs:domain* declaration. The types of the Object Property <OWL> resources are specified using an *rdfs:range* declaration.

A Object Property <OWL> is annotated with a label and its source document. The label provides the name of the UML ATTRIBUTE and uses a *rdfs:label* declaration. The source document identifies the resource defining this UML ATTRIBUTE. It uses *rdfs:isDefinedBy* to declare the IRI of the source document.

Table 14 sets the requirement for the description of ATTRIBUTES through Object Property <OWL> in Ontology <OWL>.

Table 14 — Attribute – OWL object property

Requirement
19150-2package:attribute-objectProperty
A UML ATTRIBUTE described by another UML CLASS shall correspond to a Object Property <OWL> and use an <i>owl:ObjectProperty</i> declaration. It shall be connected to the Class <OWL> resources it is associated with using an <i>rdfs:domain</i> declaration. The type of the Object Property <OWL> resources shall be specified using an <i>rdfs:range</i> declaration.
An Object Property <OWL> shall be annotated with:
— a label for the name of the UML ATTRIBUTE, using <i>rdfs:label</i> , and
— a source document defining this UML ATTRIBUTE, using <i>rdfs:isDefinedBy</i> to declare the IRI of the resource.

Additional annotation for the Object Property <OWL> may be provided such as:

- definition, using a *skos:definition* declaration to provide the semantics of the UML ATTRIBUTE, and
- deprecation, using an *owl:deprecated* declaration, which is set to "true"^^xsd:boolean if the Object Property <OWL> is deprecated.

The following example illustrates the definition of an 'object property' for an attribute definition of a 'class'.

EXAMPLE

RDF/Turtle serialization

```
exPk:ClassA.att1 a owl:ObjectProperty ;
  rdfs:label "att1" ;
  skos:definition "att1 definition" ;
  rdfs:isDefinedBy <http://sourceDefinitionIRI> ;
  rdfs:domain exPk:ClassA ;
  rdfs:range exPk:ClassB .
```

```

exPk:ClassB a owl:Class ;
    rdfs:label "ClassB" ;
    skos:definition "ClassB definition" ;
    rdfs:isDefinedBy <http://sourceDefinitionIRI> .

```

RDF/XML serialization

```

<owl:ObjectProperty rdf:about="&exPk;ClassA.att1">
    <rdfs:label>att1</rdfs:label>
    <rdfs:isDefinedBy>sourceDefinitionIRI</rdfs:isDefinedBy>
    <rdfs:domain rdf:resource="&exPk;ClassA"/>
    <rdfs:range rdf:resource="&exPk;ClassB"/>
</owl:ObjectProperty>
<owl:Class rdf:about="&exPk;ClassB">
    <rdfs:label>ClassB</rdfs:label>
    <skos:definition>ClassB definition</skos:definition>
    <rdfs:isDefinedBy>http://sourceDefinitionIRI</rdfs:isDefinedBy>
</owl:Class>

```

6.7 Enumerated type

6.7.1 Enumeration

6.7.1.1 UML notation

In UML, an ENUMERATION corresponds to a datatype defining a closed list of valid mnemonic identifiers. The list is part of the model as an enumeration literal. As such, the extension of an enumeration type implies a schema modification for most implementation mappings. An enumeration shall be used only when the membership is logically closed, so it is clear that there are no possible extensions. Attributes of a given enumeration type can take values only from this list. The notation for an ENUMERATION uses the STEREOTYPE «enumeration» and lists members as CLASS ATTRIBUTES, as shown in [Figure 6](#).

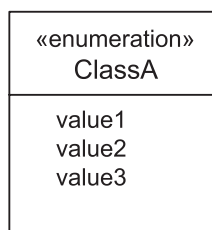


Figure 6 — UML ENUMERATION notation

6.7.1.2 OWL notation

OWL supports enumerations of literals directly using a Datatype <RDFS> specified with an *rdfs:Datatype* declaration and an *owl:oneOf* restriction declaration over a collection specification.

6.7.1.3 Rules

A UML ENUMERATION corresponds to a Datatype <RDFS> using an *rdfs:Datatype* declaration. The Datatype <RDFS> specifies the restricted list of literals using an *owl:oneOf* declaration over an *rdf:List* declaration of enumerated values.

The Datatype <RDFS> is annotated with a label and its source document. The label annotation provides the name of the UML ENUMERATION – i.e. the name of the UML ENUMERATION CLASS and uses a *rdfs:label* declaration. The source document identifies the resource defining this enumeration. It uses a *rdfs:isDefinedBy* declaration to specify the IRI of the resource.

Table 15 sets the requirement for the description of ENUMERATIONS in Ontology <OWL>.

Table 15 — Enumeration

Requirement
19150-2package:enumeration
A UML ENUMERATION shall correspond to a Datatype <RDFS> using an <i>rdfs:Datatype</i> declaration. The Datatype <RDFS> shall specify the restricted list of literals using an <i>owl:oneOf</i> declaration over an <i>rdf:List</i> declaration of enumerated values.
The Datatype <RDFS> shall be annotated with:
— a label for the name of the UML ENUMERATION, using <i>rdfs:label</i> , and
— a source document defining this UML ENUMERATION, using <i>rdfs:isDefinedBy</i> to declare the IRI of the resource.

Additional annotation for the Datatype <RDFS> may be provided such as a definition annotation, which provides the semantics of the UML ENUMERATION, using a *skos:definition*.

The following example illustrates the definition of an enumeration.

EXAMPLE 1

RDF/Turtle serialization

```
exPk:ClassA a rdfs:Datatype ;
  rdfs:label "ClassA" ;
  rdfs:isDefinedBy <http://sourceDefinitionIRI> ;
  owl:oneOf ("value1" "value2" "value3") .
```

RDF/XML serialization

```
<rdfs:Datatype rdf:about="&exPk;ClassA">
  <rdfs:label>ClassA</rdfs:label>
  <rdfs:isDefinedBy>http://sourceDefinitionIRI</rdfs:isDefinedBy>
  <owl:equivalentClass>
    <rdfs:Datatype>
```

```

<owl:oneOf>
  <rdf:List >
    <rdf:first rdf:datatype="&xsd:string">value1</rdf:first>
    <rdf:rest>
      <rdf:List >
        <rdf:first rdf:datatype="&xsd:string">value2</rdf:first>
        <rdf:rest>
          <rdf:List>
            <rdf:first rdf:datatype="&xsd:string">value3</rdf:first>
            <rdf:rest rdf:resource="&rdf:nil"/>
          </rdf:List>
        </rdf:rest>
      </rdf:List>
    </rdf:rest>
  </rdf:List>
</owl:oneOf>
<rdfs:Datatype>
  <owl:equivalentClass>
    </rdfs:Datatype>

```

This part of ISO 19150 connects a Data Property <OWL> of a Class <OWL> to a Datatype <RDFS> of enumerated values with the use of an *rdfs:range* declaration as shown in the following. In this example, a Data Property <OWL> *att1* is defined and associated with ClassB. Its range is specified by the *rdfs:Datatype* ClassA defined in Example 1.

The following example illustrates the definition of an enumeration.

EXAMPLE 2

RDF/Turtle serialization

```

exPk:ClassB.att1 a owl:DatatypeProperty ;
  rdfs:label "att1" ;
  skos:definition "att1 definition" ;
  rdfs:isDefinedBy <http://sourceDefinitionIRI> ;
  rdfs:domain exPk:ClassB ;
  rdfs:range exPk:ClassA .

```

RDF/XML serialization

```

<owl:DatatypeProperty rdf:about="&exPk;ClassB.att1">
  <rdfs:label>att1</rdfs:label>
  <skos:definition>att1 definition</skos:definition>
  <rdfs:isDefinedBy>http://sourceDefinitionIRI</rdfs:isDefinedBy>
  <rdfs:domain rdf:resource="&exPk;ClassB"/>
  <rdfs:range rdf:resource="&exPk;ClassA"/>
</owl:DatatypeProperty>

```

6.7.2 Code list

6.7.2.1 UML notation

ISO 19103:—¹) defines the UML extension CODELIST. A CODELIST corresponds to a datatype defining an open list of likely mnemonic identifiers. It expresses a list of potential values. Its UML notation is very similar to an ENUMERATION. A CODELIST is used only when use of other values apart from those in the initial list is allowed. The UML notation for a CODELIST uses the STEREOTYPE «CodeList» and lists the initial members as CLASS ATTRIBUTES, as shown in [Figure 7](#).

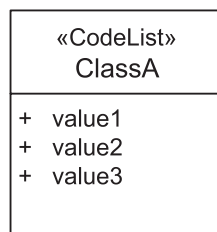


Figure 7 — UML CODELIST notation

6.7.2.2 OWL notation

SKOS has been broadly adopted for vocabulary formalization. SKOS supports the codelist requirements of membership and extensibility.

6.7.2.3 Rules

A CODELIST is implemented as a Class <OWL> and as a *skos:ConceptScheme*. The code list Class <OWL> is a subclass of *skos:Concept*. Since a concept scheme resource is an RDF individual, the concept scheme and OWL classes corresponding to the CODELIST are identified by different URIs for compliancy with OWL as used in this part of ISO 19150. The link between the OWL class and the concept scheme representing the CODELIST must therefore be formally asserted with an explicit property.

The members of the CODELIST are implemented as RDF resources that are members of the code list Class <OWL>, i.e. as individuals whose *rdf:type* is the code list Class <OWL>. Since the code list Class <OWL> is a subclass of *skos:Concept*, this entails that each member is itself an individual whose *rdf:type* is *skos:Concept*. The code list members are associated with the concept scheme using the standard *skos:inScheme* property. The concept scheme representing the code list is associated with its members using the standard *skos:hasTopConcept* property.

[Table 16](#) sets the requirement for the description of CODELISTs in Ontology <OWL>.

Table 16 — Code list

Requirement
19150-2package:codelist
<p>A CODELIST shall correspond to a Class <OWL>, a ConceptScheme <SKOS>, and a Collection <SKOS>. The Class <OWL> shall be a subclass of <i>skos:Concept</i>. The SKOS concept scheme shall be related to the Class <OWL> using a <i>dct:isFormatOf</i> property. Each member of the CODELIST shall correspond to an individual whose type is the Class <OWL> corresponding to the CODELIST, and with a <i>skos:inScheme</i> property whose value is the ConceptScheme <SKOS> corresponding to the CODELIST. Additionally, each member of the CODELIST shall also be member of the Collection <SKOS> using a <i>skos:members</i> declaration. Each of the resources shall be annotated with the following:</p> <ul style="list-style-type: none"> — a label, using <i>rdfs:label</i>, — a source for the definition, using <i>rdfs:isDefinedBy</i> for the IRI of the resource.

The following example illustrates the construction of a CodeList using the OWL and SKOS elements.

EXAMPLE

RDF/Turtle serialization

```

exPk:ClassA a owl:Class ;
    subclassOf skos:Concept ;
    rdfs:label "ClassA" ;
    rdfs:isDefinedBy <http://sourceDefinitionIRI> .

exPkCode:ClassA a skos:ConceptScheme ;
    skos:prefLabel "ClassA - ConceptScheme" ;
    rdfs:isDefinedBy <http://sourceDefinitionIRI> ;
    dct:isFormatOf exPk:ClassA .

exPkCode:ClassA/value1 a exPk:ClassA , skos:Concept ;
    skos:prefLabel "value1" ;
    skos:inScheme exPkCode:ClassA ;
    rdfs:isDefinedBy <http://sourceDefinitionIRI> .

exPkCode:ClassA/value2 a exPk:ClassA , skos:Concept ;
    skos:prefLabel "value2" ;
    skos:inScheme exPkCode:ClassA ;
    rdfs:isDefinedBy <http://sourceDefinitionIRI> .

exPkCode:ClassA/value3 a exPk:ClassA , skos:Concept ;
    skos:prefLabel "value3" ;
    skos:inScheme exPkCode:ClassA ;
    rdfs:isDefinedBy <http://sourceDefinitionIRI> .

exPkCode: ClassACollection a skos:Collection ;
    skos:prefLabel "ClassA - Concepts" ;
    skos:members exPkCode: ClassA/value1 , exPkCode: ClassA/value2 , exPkCode: ClassA/value3 .

```