*L*ABORATOIRE D'*I*NFORMATIQUE *S*CIENTIFIQUE ET *I*NDUSTRIELLE

Rapport de recherche
N° 01-2007

Proposal for an XML representation of the PLIB ontology model: OntoML

Guy PIERRA, Eric SARDET

Ecole Nationale Supérieure de Mécanique et d'Aérotechnique
Site du Futuroscope – BP 40109 – 86961 Futuroscope Cedex – FRANCE
Tel.: +33 5 49 49 80 63 - Fax: +33 5 49 49 80 64
Web: http://www.lisi.ensma.fr

Introduction

This research report was developed with three goals.

1 – To provide a synthetic view of the current PLIB ontology model, more precisely the one defined in ISO 13584-25; this model is currently described through a set of documents with internal references (ISO 13584-42, ISO 13584-24, ISO 13584-25).

2 – To describe this view in UML, more broadly understood than EXPRESS.

3 - To define an XML exchange format for PLIB-driven data.

The model presented in this document results from a number of transformations over the original PLIB model. These transformations were performed using model driven engineering techniques to simplify the model structure. The resulting model is much simpler than the initial one while having the capability to represent the complete content of a PLIB EXPRESS file.

It should be noticed that neither UML nor XML support a constraint language powerful enough for expressing the complete PLIB semantics. Thus, in this document, constraints are informally stated. Constraint checking may only be done when the XML file content is converted into an ISO 10303-21 file using EXPRESS based checkers.

After experimentation in LISI/ENSMA, it is our intent to submit this report to ISO TC184/SC4/WG2 for consideration for standardization.

G. Pierra and E. Sardet

2007 April 04

Contents

Page

1 Scope ... 1

2 Normative references ... 1

3 Terms, definitions, and abbreviations ... 2

4 OntoML implementation levels ... 6

5 Overview of OntoML ontology representation ... 6
- 5.1 CIIM ontology concepts ... 6
- 5.2 OntoML structure of a CIIM ontology concept ... 7
- 5.3 UML-like graphical representation of OntoML constructs ... 7
  - 5.3.1 Graphical notations ... 8
    - 5.3.1.1 Representation of an XML complex type ... 8
    - 5.3.1.2 Representation of references to external information elements ... 8
    - 5.3.1.3 Representation of XML attributes and XML elements whose content model are XML simple types ... 9
    - 5.3.1.4 Representation of an XML element whose content model is an XML complex type ... 9
    - 5.3.1.5 Representation of the cardinality of embedded XML elements ... 10
    - 5.3.1.6 Representation of XML complex type extensions ... 10
  - 5.3.2 Reference mechanism between ontology concepts ... 11
    - 5.3.2.1 Identification of an ontology concept ... 11
    - 5.3.2.2 OntoML representation of a reference to a CIIM ontology concept ... 12
    - 5.3.2.3 OntoML representation of simple and multi-valued references between CIIM ontology concepts ... 13
    - 5.3.2.4 Simplified graphical representation of references between CIIM ontology concepts ... 13
  - 5.3.3 UML diagrams color conventions ... 15
  - 5.3.4 Description of the structure of all OntoML complex types ... 15
    - 5.3.4.1 Graphical presentation ... 15
    - 5.3.4.2 Internal item definition ... 16
    - 5.3.4.3 Internal type definition ... 16
    - 5.3.4.4 External type definition ... 16
- 5.4 OntoML general structure ... 17
- 5.5 OntoML header ... 17
- 5.6 Root class of an ontology ... 19
- 5.7 OntoML representation of CIIM ontology concepts ... 21
  - 5.7.1 Supplier ... 21
  - 5.7.2 Simple-level ontology class ... 23
    - 5.7.2.1 Item class and categorization class ... 23
    - 5.7.2.2 Item class case-of ... 26
    - 5.7.2.3 Class valued property ... 28
  - 5.7.3 Advanced-level ontology class ... 29
    - 5.7.3.1 Functional view class ... 29
    - 5.7.3.2 Functional model class ... 31
    - 5.7.3.3 Functional model class *view-of* ... 33
    - 5.7.3.4 View control variable range ... 34
  - 5.7.4 Simple-level ontology property ... 35
  - 5.7.5 Advanced-level ontology property ... 38
  - 5.7.6 Named data types ... 38
  - 5.7.7 Document ... 41

6 Overview of OntoML libraries representation ... 43

6.1 Root class of a library 44
6.2 Class extension general structure 45
6.2.1 Property classification 46
6.2.2 Properties presentation 47
6.3 Simple-level library: content of classes of products 47
6.4 Advanced-level library: content of classes of product representations 49

7 Other structured information elements 51
7.1 Translations 51
7.1.1 Language specification 51
7.1.2 Translation of string valued elements 51
7.1.3 Translation management 53
7.2 External content 53
7.2.1 Simple-level ontology external resources 54
7.2.1.1 HTTP file 54
7.2.1.2 Illustration 55
7.2.1.3 Message 56
7.2.1.4 External files 57
7.2.2 Advanced-level ontology external resources 57
7.2.2.1 Program reference 58
7.2.2.2 Representation reference 58
7.2.3 Source document and graphics 59
7.2.3.1 Source document 59
7.2.3.1.1 Identified document 60
7.2.3.1.2 Referenced document 60
7.2.3.2 Graphics 60
7.2.3.2.1 Documented graphics 61
7.2.3.2.2 External graphics 61
7.3 Data type system 61
7.3.1 Boolean type 63
7.3.2 String types 63
7.3.3 Enumeration of string codes type 64
7.3.4 Numeric types 66
7.3.5 Numeric currency types 67
7.3.6 Numeric measure types 68
7.3.7 Enumeration of integer codes type 70
7.3.8 Collection types 71
7.3.8.1 Bag type 71
7.3.8.2 Set type 72
7.3.8.3 List type 72
7.3.8.4 Array type 73
7.3.8.5 Set with subset constraint type 74
7.3.9 Class instance type 75
7.3.10 Level type 75
7.3.11 Named data type 76
7.3.12 Advanced-level data types 77
7.4 Units 78
7.4.1 Unit structure 79
7.4.2 Named unit 79
7.4.2.1 Dimensional exponent 80
7.4.2.2 Internationally standardized unit 81
7.4.2.3 Non internationally standardized unit 82
7.4.2.4 Conversion based unit 82
7.4.2.5 Context dependent unit 83
7.4.3 Derived unit 83
7.5 Constraints 84
7.5.1 Subtypes of domain constraint type 85
7.5.2 Subclass constraint 85
7.5.3 String pattern constraint 86

7.5.4 Context restriction constraint 87
7.5.5 String size constraint 87
7.5.6 Range constraint 88
7.5.7 Subset constraint 89
7.6 A posteriori semantics relationship 90
7.6.1 A posteriori mapping in a *case-of* relationship 91
7.6.2 A posteriori mapping in a *view-of* relationship 92
7.7 Data exchange specification identification 93
7.7.1 Simple-level ontology data exchange specification: library integrated information model identification 93
7.7.2 Advanced-level ontology data exchange specification: view exchange protocol identification 94
7.8 Other structured information elements 95
7.8.1 Organization representation 95
7.8.2 Mathematical string 95

8 OntoML exchange structure 96
8.1 Identifiers of ontology concepts 96
8.1.1 Registration Authority Identifier (RAI) structure 96
8.1.2 Version Identifier (VI) structure 98
8.1.3 Data Identifier (DI) structure 98
8.1.3.1 DI for classes 99
8.1.3.2 DI for properties, named types and documents 99
8.1.3.3 DI for ontologies and libraries 99
8.1.4 OntoML identifier representation 100
8.2 OntoML namespace 100
8.2.1 OntoML URN 100
8.2.2 OntoML URI 100
8.3 Modular structure 100
8.4 Levels of exchange and conformance classes 102
8.5 Conformance class requirements 103
8.5.1 Conformance class 1 103
8.5.2 Conformance class 2 108
8.5.3 Conformance class 3 109
8.5.4 Conformance class 4 110

Annex A (normative) Information object registration 112

Annex B (normative) The OntoML XML Schema 113

Annex C (normative) Standard data requirements for OntoML 144

Annex D (normative) Structural transformation of the CIIM model from OntoML XML Schema to EXPRESS 145

Annex E (informative) XML file example 178

Bibliography 186

Index 187

Figures

Figure 1 – CIIM ontology concepts description 7
Figure 2 – UML-like representation of an XML complex type 8
Figure 3 – UML-like representation of a reference to an XML complex type 8
Figure 4 – UML-like representation of an external reference to an XML complex type 8

Figure 5 – UML-like representation of XML attributes and simple type elements 9
Figure 6 – XML representation of XML attributes and simple type elements 9
Figure 7 – UML-like representation of an XML complex type element 9
Figure 8 – XML representation of an XML complex type element 10
Figure 9 – UML-like representation of XML elements cardinality 10
Figure 10 – XML representation of XML elements cardinality 10
Figure 11 – UML-like representation of XML complex type extensions 11
Figure 12 – XML representation of XML complex type extensions 11
Figure 13 – Identification of a CIIM ontology concept 12
Figure 14 – CIIM ontology concept reference 13
Figure 15 – Reference between ontology concepts 13
Figure 16 – UML-like representation of a simple reference between CIIM ontology concepts 14
Figure 17 – XML representation of a simple reference between CIIM ontology concepts 14
Figure 18 – UML-like representation of a multi-valued reference between CIIM ontology concepts 14
Figure 19 – XML representation of a multi-valued reference between CIIM ontology concepts 15
Figure 20 – Ontology structure UML diagram 17
Figure 21 – Ontology header structure 18
Figure 22 – Root class of an ontology 20
Figure 23 – Supplier ontology concept UML diagram 22
Figure 24 – Simple class ontology concept UML diagram 24
Figure 25 – Item class case-of relationship 27
Figure 26 – Class value assignment structure 28
Figure 27 – Advanced-level ontology class concept UML diagram: functional view class 30
Figure 28 – Advanced class ontology concept UML diagram: functional model class 32
Figure 29 – Advanced class ontology concept UML diagram: functional model class view of 33
Figure 30 – Mathematical string structure 35
Figure 31 – Simple property ontology concept UML diagram 36
Figure 32 – Advanced property ontology concept UML diagram 38
Figure 33 – Named data type UML diagram 39
Figure 34 – Simple-level document UML diagram 41
Figure 35 – Root class of library 44
Figure 36 – General class extension structure 45
Figure 37 – Properties classification 46
Figure 38 – Properties presentation 47
Figure 39 – Products representation structure 48
Figure 40 – Engineering models structure UML diagram 50
Figure 41 – Language specification 51
Figure 42 –Translation resources 52
Figure 43 – Translation data structure 53
Figure 44 – Simple-level ontology external resources 54
Figure 45 – Simple-level ontology external resources: HTTP file structure 55
Figure 46 – Simple-level ontology external resources: illustration 56
Figure 47 – Simple-level ontology external resources: message 57
Figure 48 – Simple-level ontology external resources: external files 57
Figure 49 – Advanced-level ontology external resources 58
Figure 50 – Advanced-level ontology external resources: program reference 58
Figure 51 – Advanced-level ontology external resources: representation reference 59
Figure 52 – External resources: source document 60
Figure 53 – External resources: graphics 60
Figure 54 – OntoML datatype system 62
Figure 55 – Boolean type structure 63
Figure 56 – String types structure 64
Figure 57 – Enumeration of string codes type structure 65
Figure 58 – Numeric types structure 67
Figure 59 – Numeric currency types structure 67
Figure 60 – Numeric measure types structure 69
Figure 61 – Enumeration of integer codes type structure 70
Figure 62 – Bag type structure 71
Figure 63 – Set type structure 72

Figure 64 – List type structure 73
Figure 65 – Array type structure 73
Figure 66 – Set with a subset constraint type structure 74
Figure 67 – Instance value domain structure 75
Figure 68 – Levels value domain structure 76
Figure 69 – Named value domain structure 77
Figure 70 – Advanced value domain structures 77
Figure 71 – General measure property unit structure 78
Figure 72 – Basic unit structures 79
Figure 73 – Named unit general structure 80
Figure 74 – Dimensional exponent structure 80
Figure 75 – International standardized unit structure 81
Figure 76 – Non international standardized unit structure 82
Figure 77 – Conversion based unit structure 82
Figure 78 – Context dependent unit structure 83
Figure 79 – Derived unit structure 84
Figure 80 – General constraints structure 84
Figure 81 – Constraints representation 85
Figure 82 – Subclass constraint representation 86
Figure 83 – String pattern constraint representation 86
Figure 84 – Context restriction constraint representation 87
Figure 85 – String size constraint representation 88
Figure 86 – Range constraint representation 89
Figure 87 – Subset constraint representation 90
Figure 88 – A posteriori relationship general structure representation 90
Figure 89 – A posteriori *case-of* relationship representation 91
Figure 90 – A posteriori semantic relationships structure 92
Figure 91 – Library integrated information model identification structure 93
Figure 92 – View exchange protocol identification structure 94
Figure 93 – Organization structure 95
Figure 94 – Mathematical string structure 95
Figure D.1 – A UML information model example 146
Figure D.2 – An UML-like representation of the information model 147
Figure D.3 – An XML Schema example 147
Figure D.4 – Mapping representation in OntoML 149
Figure D.5 – XML source Path 150
Figure D.6 – Global Vs local XML elements 150
Figure D.7 – Local EXPRESS target path structure 153
Figure D.8 – Complete EXPRESS target path structure 155
Figure E.1 – General model example: ontology definition 178
Figure E.2 – General model example: product specification 179

Tables

Table 1 – OntoML modules cross-references 102
Table 2 – Conformance options of OntoML 103
Table C.1 – ISO 13584 LIIM 32 conformance class specification 144
Table D.1 – XML and corresponding ISO 10303 instances 148
Table D.2– SELF meaning in its use context 151
Table D.3 – OntoML identifiers mapping 160
Table D.4 – OntoML list of class identifiers mapping 162
Table D.5 – OntoML ontology identifier mapping 162
Table D.6 – OntoML label and translated label mapping 163
Table D.7 – OntoML text and translated text mapping 165
Table D.8 – OntoML synonymous and translated synonymous mapping 166
Table D.9 – OntoML HTTP proptocol mapping 168
Table D.10 – OntoML translated and not translated files mapping 169
Table D.11 – OntoML external resource mapping 170

Table D.12 – OntoML a posteriori case-of relationship mapping 172
Table D.13 – OntoML a posteriori view-of relationship mapping 172
Table D.14 – OntoML global language mapping 174
Table D.15 – OntoML complex types / CIIM entity datatypes correspondence 174

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.

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

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

Attention is drawn to the possibility that some of the elements of this part of ISO 13584 may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

International Standard ISO 13584-32 was prepared by Technical Committee ISO/TC184, *Industrial automation system and integration,* Subcommittee SC4, *Industrial data.*

ISO 13584 consists of the following parts under the general title *Industrial automation systems and integration – Parts library:*

— Part 1, Overview and fundamental principles;

— Part 20, Logical resource: Logical model of expressions;

— Part 24, Logical resource: Logical model of supplier library;

— Part 25, Logical resource: : Logical model of supplier library with aggregate values and explicit content;

— Part 26, Logical resource: Supplier identification;

— Part 31, Implementation resource: Geometric programming interface;

— Part 32, Implementation resource: OntoML: Ontology Markup Language;

— Part 35, Implementation resource: Spreadsheet data interface for parts library;

— Part 42, Description methodology: Methodology for structuring part families;

— Part 101, View exchange protocol: Geometric view exchange protocol by parametric program;

— Part 102, View exchange protocol: View exchange protocol by ISO 10303 conforming specification;

— Part 501, Reference Dictionaries: Reference dictionary for measuring instruments — Registration procedure;

— Part 511, Reference Dictionaries: Mechanical systems and components for general use — Reference dictionary for fasteners.

The structure of this International Standard is described in ISO 13584-1. The numbering of the parts of this International Standard reflects its structure:

— Parts 10 to 19 specify the conceptual descriptions,

— Parts 20 to 29 specify the logical resources,

— Parts 30 to 39 specify the implementation resources,

— Parts 40 to 49 specify the description methodology,

— Parts 100 to 199 specify the view exchange protocols,

— Parts 500 to 599 specify the standardized content.

A complete list of parts of ISO 13584 is available from the following URL:

<http://www.tc184-sc4.org/titles/PLIB_Titles.htm>

Should further parts of ISO 13584 be published, they will follow the same numbering pattern.

— Annexes A, B, C and D form an integral part of this part of ISO 13584.

— Annex E is for information only.

Introduction

ISO 13584 is an International Standard for the computer-interpretable representation and exchange of parts library data. The objective is to provide a neutral mechanism capable of transferring parts library data, independent of any application that is using a parts library data system. The nature of this description makes it suitable not only for the exchange of files containing parts, but also as a basis for implementing and sharing databases of parts library data.

This International Standard is organized as a series of parts, each published separately. The parts of ISO 13854 fall into one of the following series: conceptual descriptions, logical resources, implementation resources, description methodology, conformance testing, view exchange protocol, and standardized content. The series are described in ISO 13584-1. This part of ISO 13584 is a member of the implementation resources series.

This part of ISO 13584 specifies an XML based exchange structure for ISO 13584 / IEC 61360 compliant data. It provides a set of constructs allowing to represent both an ontology, possibly together with its external resources, and a description of a set of products that reference ontologies and that constitute a library, or catalogue. This exchange structure is called OntoML. The knowledge presented in ISO/IEC Guide 77-2 is recommended for understanding this part.

Industrial automation systems and integration – Parts library – Part 32 : Implementation resources : OntoML: Ontology Markup Language

1 Scope

This part of ISO 13584 specifies use of the Extensible Markup Language (XML) and the XML Schema specification to represent data according to the ISO 13584 data model.

The following are within the scope of this part of ISO 13584:

— specification of XML markup declarations that enable both simple ontologies and advanced ontologies compliant with the common ISO13584/IEC61360 model to be exchanged using XML;

— definition of two levels of implementation of the common ISO13584/IEC61360 model, respectively called the simple level and the advanced level;

— specification of XML markup declarations that enable to exchange together both ontologies compliant with the common ISO13584/IEC61360 model and families of products whose characterizations are defined by means of these ontologies;

NOTE 1 In this part of ISO 13584, such an exchange context is called an OntoML library.

NOTE 2 The information model for exchanging families of products whose characterizations are defined by means of the common ISO13584/IEC61360 model compliant ontologies is defined in ISO13584-25:2003.

— specification of a formal mapping allowing to associate each OntoML elements and attributes of the corresponding entities and attributes of the CIIM EXPRESS data model.

The following are outside the scope of this part of ISO 13584:

— rules used to build OntoML from the common ISO13584/IEC61360 model;

— the specification of the program intended to interpret all the mapping operators defined in OntoML for building the corresponding CIIM compliant ISO 10303-21 instances;

— the exchange of individual products, or of sets of individual products, whose characterizations are defined by means of ontologies compliant with the common ISO13584/IEC61360 model.

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 13584-1:2000*, Industrial automation systems and integration — Parts library — Part 1: Overview and fundamental principles.*

ISO 13584-24:2002*, Industrial automation systems and integration — Parts library — Part 24: Logical resource: Logical model of supplier library.*

ISO 13584-25:2003*, Industrial automation systems and integration — Parts library — Part 25: Logical resource: : Logical model of supplier library with aggregate values and explicit content.*

ISO 13584-26:2000*, Industrial automation systems and integration — Parts library — Part 26: Logical resource: Supplier identification.*

ISO 13584-42:2003*, Industrial automation systems and integration — Parts library — Part 42: Description methodology: Methodology for structuring parts families.*

Multipurpose Internet Mail Extensions (MIME) Part One*: Format of Internet Message Bodies. Internet Engineering Task Force RFC 2045 November 1996 [cited 2000-08-15]. Available from World Wide Web: <http://www.ietf.org/rfc/rfc2045.txt>.*

Uniform Resource Identifiers (URI)*: Generic Syntax. Internet Engineering Task Force RFC 2396 August 1998 [cited 2000-08-07]. Available from World Wide Web: <http://www.ietf.org/rfc/rfc2396.txt>.*

Extensible Markup Language (XML) 1.0*. Fourth Edition. World Wide Web Consortium Recommendation 14-June-2006. Available from World Wide Web: <http://www.w3.org/TR/2006/PER-xml-20060614>*

XML Schema Part 1 *: Structures. Second Edition. World Wide Web Consortium Recommendation 28-October-2004 Available from World Wide Web: <http://www.w3.org/TR/2004/REC-xmlschema-1-20041028>*

XML Schema Part 2 *: Datatypes. Second Edition. World Wide Web Consortium Recommendation 28-October-2004 Available from World Wide Web: http://www.w3.org/TR/2004/REC-xmlschema-2-20041028*

XML Path Language (XPath) 1.0*. World Wide Web Consortium Recommendation 16-November-1999 Available from World Wide Web: <http://www.w3.org/TR/1999/REC-xpath-19991116>*

Namespaces in XML 1.0. Second Edition. World Wide Web Consortium Recommendation 14 June 2006. Available from World Wide Web: <http://www.w3.org/TR/2006/PER-xml-names-20060614>.

3 Terms, definitions, and abbreviations

For the purpose of this part of ISO 13584, the following terms and definitions apply. Some of these terms and definitions are repeated for convenience from:

— ISO 10303-1:1994;

— ISO 10303-11:1994;

— ISO 13584-1:2000;

— ISO 13584-24:2002;

— ISO 13584-25:2003;

— ISO 13584-42:2003;

— *ISO/IEC CD Guide 77-2;*

— XML 1.0 Recommendation;

3.1

class
abstraction of a set of similar objects

3.2

common ISO13584/IEC61360 model
CIIM
the information model for product ontology, using the information modeling language EXPRESS, resulting from a joint effort between ISO TC 184/SC4/WG2 and IEC SC3D

NOTE The current version of the common IS13584O/IEC61360 ontology model is published in both IEC 61360-5, and ISO 13584-25:2004.

3.3

CIIM ontology concept
basic unit of knowledge represented in an ontology based on the common IS13584O/IEC61360 ontology model, CIIM ontology concepts are information source (supplier), class, named data type and document

NOTE 1 Each CIIM ontology concept is associated with a global identifier allowing to reference it externally to an exchange file.

NOTE 2 The same CIIM ontology concept may be referenced several times in the same exchange file. Therefore, a referencing mechanism is defined in OntoML.

3.4

EXPRESS attribute
data element for the computer-sensible description of a property, a relation or a class.

NOTE An attribute describe only one single detail of a property, of a class or of a relation.

EXAMPLE The name of a property, the code of a class, the measure unit in which values of a property are provided are examples of attributes.

3.5

EXPRESS entity
class of information defined by common properties

[ISO 10303-11:1994, definition 3.2.5]

3.6

global identifier
GId
code providing an unambiguous and universally unique identification of some concepts or objects

NOTE All CIIM ontology concepts are associated with a GId.

3.7

library
representation of a set of products by their product characterizations possibly associated with the ontology where product characterizations classes and properties are defined

NOTE A library is also called a catalogue.

3.8

OntoML document instance
XML document that complies with the OntoML XML Schema

3.9

product characterization
description of a product by means of a product characterization class, to which it belongs, and a set of property value pairs

EXAMPLE *Hexagon_head_bolts_ISO_4014* (*Product grades* = A, *thread type* = M, *length* = 50, *Diameter* = 8) is an example of product characterization.

NOTE In the example, *Hexagon_head_bolts_ISO_4014* stands for the identifier of the “Hexagon head bolts" product characterization class defined by ISO 4014. All the names in italics between parenthesis stand for the identifier of the bolt properties defined in ISO 4014.

3.10

product ontology
ontology
model of product knowledge, done by a formal and consensual representation of the concepts of a product domain in terms of characterization classes, of class relations and of properties

NOTE 1 Ontologies are not concerned with words but with concepts, independent of any particular language.

NOTE 2 Consensual means that the conceptualization is agreed upon in some community.

EXAMPLE The product ontology defined in IEC 61360 is agreed upon by all member bodies of IEC SC3D. A corporate ontology is agreed upon by experts designated by management on behalf of the company.

NOTE 3 Formal means that the ontology should be machine interpretable. Some level of machine reasoning shall be possible over ontology, e.g., consistency checking.

NOTE 4 In OntoML, advanced ontologies are those ontologies that use all the modeling mechanisms defined in the common ISO13584/IEC61360 ontology model. OntoML also defines a simple functional subset of this model allowing to define simple ontologies.

3.11

property
defined parameter suitable for the description and differentiation of objects

NOTE 1 A property describes one aspect of a given object.

NOTE 2 The property as such is constituted by the totality of all its associated attributes. Types and number of attributes that describe a property with highest possible accuracy are defined in this guide.

NOTE 3 The term “property” used in this document and the term “data element type” used in IEC 61360 are synonyms.

3.12

reference dictionary
ontology that is based on the common ISO13584-IEC61360 model

NOTE A reference dictionary contains only CIIM ontology concepts. It does not contain any class member.

3.13

XML attribute
XML construct included in an element and defined by a name and a simple value pair

NOTE 1 Adapted from XML 1.0 Recommendation.

NOTE 2 In this part of ISO 13584, the name of an XML attribute will be prefixed by “@” to stipulate that the corresponding piece of information is represented as an attribute and not as an XML embedded element.

3.14

XML complex type
a set of XML element and/or attribute declarations describing an XML element content model

3.15

XML element
XML structure including a start tag, an end tag, information between these tags, and, possibly, a set of XML attributes

NOTE 1 Adapted from XML 1.0 Recommendation.

NOTE 2 The information structure between start tag and end tag is defined by either by an XML simple type or an XML complex type.

NOTE 3 An XML element may contain other XML elements defined by either an XML simple type or an XML complex type.

3.16

XML simple type
a set of constraints applicable to the value·of an XML attribute, or to the value of an XML element without any XML child element

NOTE An XML simple type applies to the values of attributes and the text-only content of elements.

3.17 Abbreviated terms

For the purpose of this part of ISO 13584, the following abbreviations apply.

— CIIM: common ISO13584/IEC61360 model;

— URI: Uniform Resource Identifier;

— URN: Uniform Resource Name;

— XML: Extensible Markup Language.

4 OntoML implementation levels

The CIIM includes a number of concepts and of modelling mechanisms allowing to characterize not only items, such that products, but also (1) multi point of view disciplines specific representations of items, and (2) characterization of the various possible point of views. Such advanced concepts may not be necessary in a number of applications.

Therefore, this part of ISO 13584 identifies a subset of all the modelling mechanisms defined in the CIIM that should prove useful in most application contexts. This subset defines allowed levels of implementation of OntoML. These levels are denoted as “simple” in the document.

All the modelling mechanisms that do not belong to the simple level are referenced as “advanced” when they are presented. Clause 8.5 summarizes those OntoML constructs that belong to simple levels and those that belong to advanced level.

NOTE 1 Simple levels being defined as a consistent functional subset, reading and understanding advanced mechanisms is not needed to understand and to use the simple levels.

Moreover, OntoML allows to model two kind of information:

— ontologies,

— libraries, that are set of instances data possibly associated with their ontology definitions.

Depending on the application context, not all those kinds of information may prove useful. Therefore, four subsets of OntoML are defined as allowed levels of implementation:

— simple ontology,

— advanced ontology,

— simple library,

— advanced library.

A simple instance data level does not exist, because this kind of information exchange will be addressed by ISO 22709-10.

NOTE 2 ISO 22709 is developed as a joint effort of several standardization committees to promote interoperability between the various standards that require product characterization.

NOTE 3 Only simple ontology and advanced ontology subsets comply with the CIIM. Representation of libraries complies with extensions of the CIIM defined in ISO 13584-24:2002 and ISO 13584-25:2003.

5 Overview of OntoML ontology representation

In this clause, CIIM ontology concepts are defined and their underlying structure is presented. Additionally, graphical notations used to illustrate every ontology concept structure are introduced.

5.1 CIIM ontology concepts

According to the CIIM, an ontology consists of five kinds of main concepts:

— supplier,

— class,

— property,

— named domain of values,

— document.

Each of these CIIM ontology concepts carries two kinds of information:

— its identification that is a global identifier. This identifier allows to reference this concept from within or outside the OntoML document that defines the concept. The structure of CIIM ontology concept global identifiers is defined in clause 8.1.

NOTE The OntoML global identifier of a CIIM ontology concept contains the same information than the ones defined in the other parts of ISO 13584, and known as basic semantic unit.

— its definition that consists of a set of pieces of information specified in the CIIM.

5.2 OntoML structure of a CIIM ontology concept

Each CIIM ontology concept definition includes a number of pieces of information and of relations with other CIIM ontology concepts. In the XML representation, each CIIM ontology concept is represented by one XML element. Then, the pieces of information that contribute to the definition of the CIIM ontology concept are represented either externally or internally to its associated XML element, depending on the relationships involved:

— external representation are used to reference any piece of information that represents another CIIM ontology concept, through its own identifier;

— internal representation are used to reference any other piece of information.

Figure 1 Illustrates these two kinds of representation.

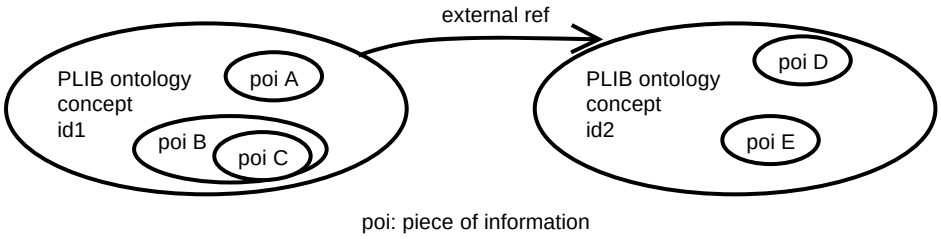

Figure 1 – CIIM ontology concepts description

Assumes that two CIIM ontology concepts are defined. They are both unambiguously identified (*id1* and *id2* identifiers). Additionally, both are defined by several pieces of information (*poi*) that are embedded within the XML representation of the CIIM ontology concepts. In turn, these pieces of information may themselves embed some others pieces of information. Finally, the CIIM ontology concept identified by *id1* references the CIIM ontology concept identified by *id2*.

NOTE Internal representation that consists in embedding the referenced piece of information within the XML element that represents a CIIM ontology concept may result in a duplication of some pieces of information. Anyway, it does not change the semantics of the underlying CIIM EXPRESS data model.

5.3 UML-like graphical representation of OntoML constructs

This clause presents the UML-like graphical representation of OntoML constructs. It also describes the mechanism used for representing in OntoML references between ontology concepts together with its graphical representation.

After presenting the referencing mechanism used to provide for external reference between CIIM ontology concepts, this clause presents the structure of the various CIIM ontology concepts defined in OntoML using UML diagrams.

5.3.1 Graphical notations

In the remaining part of the document, the following conventions will be used to represent the OntoML structure using UML-like diagrams.

5.3.1.1 Representation of an XML complex type

An XML complex type is represented as a box split in two parts: the XML complex type name at the top, the XML attribute(s) and/or the embedded XML element(s) below. It is illustrated in Figure 2.

EXAMPLE 1 In Figure 2, an XML complex type called **PROPERTY_Type** is represented.

Figure 2 – UML-like representation of an XML complex type

If the XML complex type is abstract, its name is represented using italic font style.

A complex type may also be represented as a rounded thin line box: it means that the XML attribute(s) and/or the embedded XML element(s) specifying its content model are defines elsewhere.

EXAMPLE 2 Figure 3 specifies a reference to a **PROPERTY_Type** XML complex type.

Figure 3 – UML-like representation of a reference to an XML complex type

5.3.1.2 Representation of references to external information elements

OntoML is using external XML Schema resources for defining its own content. For that purpose, a graphical notation has been introduced. Because it is a reference to an XML complex type, it is firstly represented as a rounded thin line box. Secondly, because it is an external reference, this box is filled with a light gray color. It is illustrated in Figure 4 below.

EXAMPLE In Figure 4, a complex type called **Detail** is referenced from another XML schema identified by the **cat** prefix.

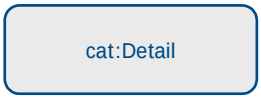

Figure 4 – UML-like representation of an external reference to an XML complex type

NOTE The prefix is defined according to the XML namespaces mechanism and allows to recognizes XML definitions specified in some other external XML Schema vocabularies.

5.3.1.3 Representation of XML attributes and XML elements whose content model are XML simple types

Both XML attributes and XML elements whose content are an XML simple type embedded within an XML complex type are represented by their name and their type. To distinguish XML attributes and XML embedded elements, the name of the former is prefixed by the "@" character.

In Figure 5 below, a **PROPERTY_Type** is an XML complex type that embeds a **revision** element whose type is the **REVISION_TYPE_Type** XML simple type, and an **id** XML attribute whose type is the **PropertId** XML simple type.

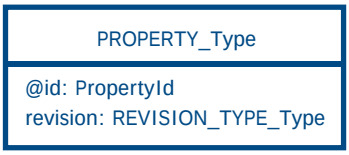

Figure 5 – UML-like representation of XML attributes and simple type elements

EXAMPLE Figure 6 below shows the XML document instance corresponding to Figure 5.

```
<xs:complexType name="PROPERTY_Type" abstract="true">
    <xs:sequence>
        <xs:element name="revision" type="REVISION_TYPE_Type"/>
    </xs:sequence>
    <xs:attribute name="id" type="PropertyId" use="required"/>
</xs:complexType>
```

Figure 6 – XML representation of XML attributes and simple type elements

5.3.1.4 Representation of an XML element whose content model is an XML complex type

An XML element whose content model is an XML complex type is represented as a relationship between the complex type XML element, and the complex type that embeds the content model of this XML element. This relationship is represented by a filled diamond followed by a plain line whose end is an arrow. The label associated to the relationship represents the XML element name.

NOTE 1 The filled diamond is used to denote a composition relationship.

NOTE 2 The default direct relationship cardinality is exactly one.

EXAMPLE In Figure 7, an XML element called **domain** is defined: it represents an embedded element of the **PROPERTY_Type** XML complex type, and its own content model is the abstract **DATA_TYPE_Type** XML complex type.

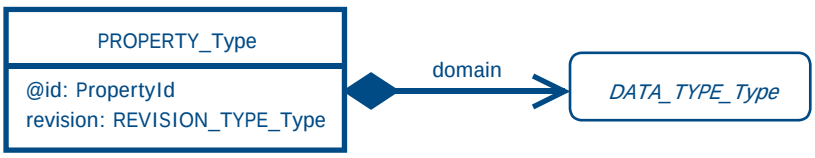

Figure 7 – UML-like representation of an XML complex type element

Figure 8 below shows the XML document instance corresponding to Figure 7.

```
<xs:complexType name="PROPERTY_Type" abstract="true">
    <xs:sequence>
        <xs:element name="revision" type="REVISION_TYPE_Type"/>
        <xs:element name="domain" type="DATA_TYPE_Type"/>
    </xs:sequence>
    <xs:attribute name="id" type="PropertyId" use="required"/>
</xs:complexType>
```

Figure 8 – XML representation of an XML complex type element

5.3.1.5 Representation of the cardinality of embedded XML elements

XML elements cardinality is specified using the UML-like notation: *minimum cardinality .. maximum cardinality*.

EXAMPLE 1 In Figure 9, a minimum cardinality equal to 0 and a maximum cardinality equal to 1 are assigned both to the **is_deprecated** and **icon** XML elements.

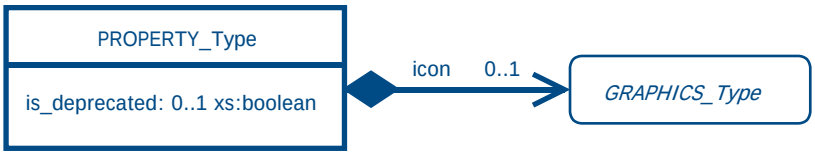

Figure 9 – UML-like representation of XML elements cardinality

NOTE The cardinality relationships expressed in Figure 7 stand for optimality.

EXAMPLE 2 Figure 8 below shows the XML document instance corresponding to Figure 7.

```
<xs:complexType name="PROPERTY_Type" abstract="true">
    <xs:sequence>
        <xs:element name="is_deprecated" type="xs:boolean" minOccurs="0"/>
        <xs:element name="icon" type="GRAPHICS_Type" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>
```

Figure 10 – XML representation of XML elements cardinality

5.3.1.6 Representation of XML complex type extensions

XML complex type extensions are represented using the usual triangle UML inheritance symbol.

EXAMPLE 1 In Figure 11 below, the **NON_DEPENDENT_P_DET_Type** XML complex type is defined as an extension of the **PROPERTY_Type** abstract XML complex type.

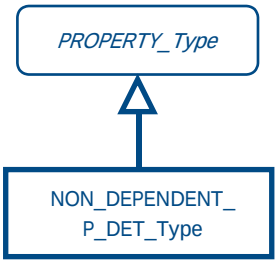

Figure 11 – UML-like representation of XML complex type extensions

EXAMPLE 2 Figure 10 below shows the XML document instance corresponding to Figure 9.

```
<xs:complexType name="NON_DEPENDENT_P_DET_Type">
    <xs:complexContent>
        <xs:extension base="PROPERTY_Type"/>
    </xs:complexContent>
</xs:complexType>
```

Figure 12 – XML representation of XML complex type extensions

5.3.2 Reference mechanism between ontology concepts

This clause presents the graphical notations that are used to represent the identification of an ontology concept and its reference from another ontology concept. Moreover, it introduces graphical notations for representing multi-references from one ontology concepts to a set of other ontology concepts.

5.3.2.1 Identification of an ontology concept

The CIIM specifies how to associate a global identifier with any ontology concept.

In OntoML, for identifying each particular type of ontology concept, a particular XML complex type is defined:

— the name of these XML complex types reflect the name of their target data type,

— each reference element contains an attribute whose value is the global identifier of the particular ontology concept it identifies,

— the name of this attribute also reflects the name of the target type.

Such elements are intended to be embedded within CIIM ontology concepts.

The name of these XML complex types is given according to the following rule:

— supplier: the type of the identifier is **SupplierId**; It belongs to a **SUPPLIER_Type** XML complex type;

— class: the type of the identifier is **ClassId**; It belongs to a **CLASS_Type** XML complex type;

— property: the type of the identifier is **PropertyId**; It belongs to a **PROPERTY_Type** XML complex type;

— named data type: the type of the identifier is **DatatypeId**; It belongs to a **DATATYPE_Type** XML complex type;

— document: the type of the identifier is **DocumentId**; It belongs to a **DOCUMENT_Type** XML complex type.

NOTE The structures of these identifier types are given in clause 8.1.

The name of this identification attribute is “id”. This name will be represented as **@id** throughout this document to specify that it is an XML attribute and not an XML embedded element.

EXAMPLE In Figure 13, the global identifier of a class ontology concept, represented by the **CLASS_Type** XML complex type, contains an **@id** attribute whose data type is **ClassId**.

Figure 13 – Identification of a CIIM ontology concept

5.3.2.2 OntoML representation of a reference to a CIIM ontology concept

For referencing each particular type of CIIM ontology concept, a particular XML complex type is defined:

— the name of these XML complex types reflect the name of their target data type,

— each reference element contains an attribute whose value is the global identifier of the CIIM ontology concept it references,

— the name of this attribute also reflects the name of the target type.

The XML reference complex type and the XML reference attribute name, prefixed by “@” to stipulate that it is an XML attribute, are defined according to the CIIM ontology concepts referenced:

— **SUPPLIER_REFERENCE_Type** XML complex type and **supplier_ref** XML attribute (whose data type is **SupplierId**) : reference to a supplier;

— **CLASS_REFERENCE_Type** XML complex type and **class_ref** XML attribute (whose data type is **ClassId**): reference to a class;

— **PROPERTY_REFERENCE_Type** XML complex type and **property_ref** XML attribute (whose data type is **PropertyId**): reference to a property;

— **DATATYPE_REFERENCE_Type** XML complex type and **datatype_ref** XML attribute (whose data type is **DatatypeId**): reference to a named data type;

— **DOCUMENT_REFERENCE_Type** XML complex type and **document_ref** XML attribute (whose data type is **DocumentId**): reference to a document;

NOTE The reference attribute type is defined to match the type of the referenced ontology concept.

EXAMPLE The reference to a class ontology concept is performed using a **CLASS_REFERENCE_Type** XML complex type and an XML reference attribute called **@class_ref** (whose data type is **ClassId**), as illustrated in Figure 14.

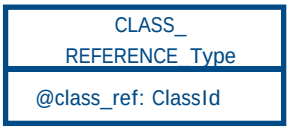

Figure 14 – CIIM ontology concept reference

5.3.2.3 OntoML representation of simple and multi-valued references between CIIM ontology concepts

When the reference from a CIIM ontology concept to another CIIM ontology concept is multi-valued, an additional XML complex type is created as shown in Figure 15.

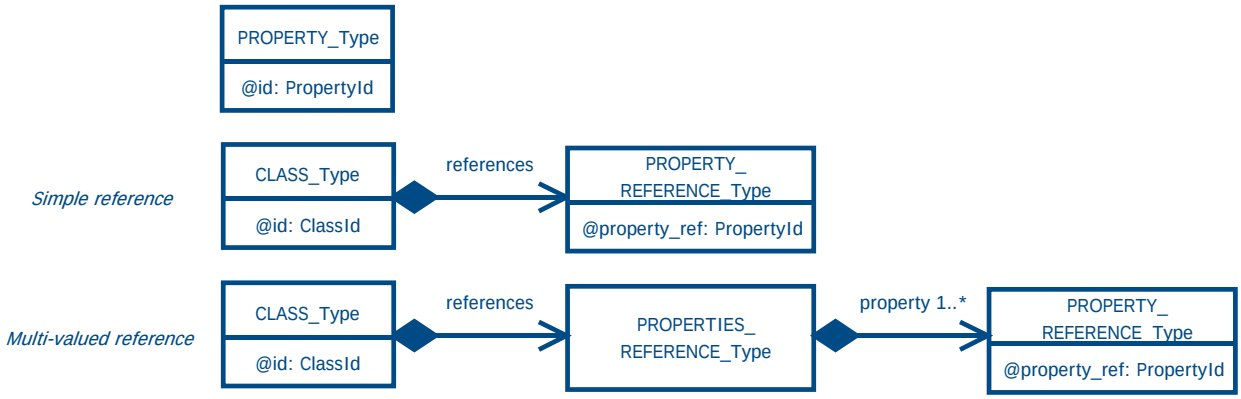

Figure 15 – Reference between ontology concepts

In this figure:

— a filled diamond is used to represent the composition relationship that denotes the underlying XML nested structure,

— the arrow specifies the corresponding relationship orientation.

Moreover, in this figure, a property and a class ontology concepts, respectively represented by a **PROPERTY_Type** and a **CLASS_Type** XML complex types are defined. Both are identified by an **id** XML attribute whose type depends on the identified ontology concept. Two cases of relationships are presented:

— *simple reference*: it is a one to one relationship from a class to a property. The relationship is represented by an XML element (*references*) whose content definition is a **PROPERTY_REFERENCE_Type** XML complex type;

— *multi-valued reference*: it is a one to many relationship from a class to a set of properties. In this case, the relationship is represented by an XML element (*references*) that acts as a container, and whose content definition is a **PROPERTIES_REFERENCE_Type** XML complex type.

5.3.2.4 Simplified graphical representation of references between CIIM ontology concepts

As explained in clause 5.3.2.3, references between CIIM ontology concepts involve a complex chain of one or two composition relationships followed by an identifier / reference matching. To simplify its graphical representation, a particular graphical notation is introduced. This representation is as follows. A reference between CIIM ontology concepts will be represented by a filled diamond followed by a dashed line that joins the referencing XML complex type to the referenced CIIM ontology concept.

The target CIIM ontology concept is represented in a dashed box by its corresponding name, in capital letters, as follows:

— supplier: **SUPPLIER**;

— class: **CLASS**;

— property: **PROPERTY**;

— named data type: **DATATYPE**;

— document: **DOCUMENT**.

When different from a one to one relationship, the relationship cardinality is represented like in UML.

EXAMPLE 1 Figure 16 represents a simple reference between two CIIM ontology concepts together with its meaning using the previous notation.

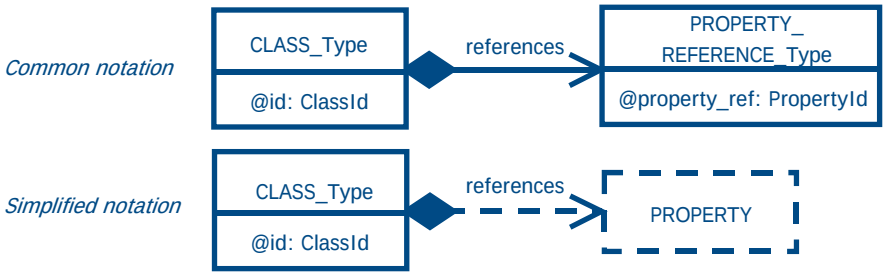

Figure 16 – UML-like representation of a simple reference between CIIM ontology concepts

EXAMPLE 2 Figure 17 below shows the XML document instance corresponding to Figure 16.

```
<xs:complexType name="CLASS_Type">
      <xs:sequence>
            <xs:element name="references" type="PROPERTY_REFERENCE_Type"/>
      </xs: sequence >
</xs:complexType>

<xs:complexType name="PROPERTY_REFERENCE_Type">
      <xs:attribute name="property_ref" type="PropertyId" use="required"/>
</xs:complexType>
```

Figure 17 – XML representation of a simple reference between CIIM ontology concepts

EXAMPLE 3 Figure 18 represents a multi-valued reference between CIIM ontology concepts.

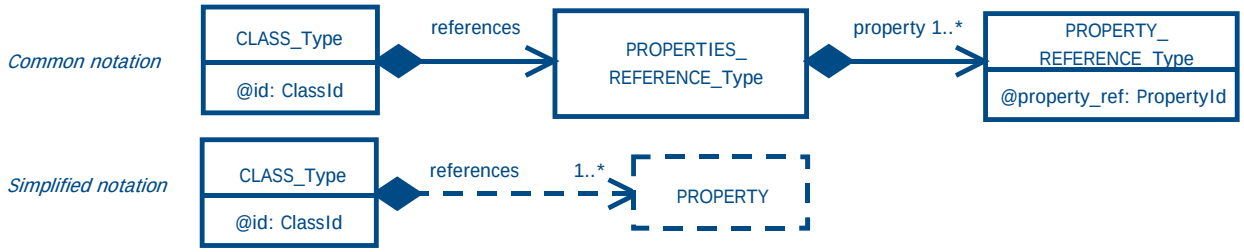

Figure 18 – UML-like representation of a multi-valued reference between CIIM ontology concepts

EXAMPLE 4 Figure 19 below shows the XML document instance corresponding to Figure 18.

```
<xs:complexType name="CLASS_Type">
    <xs:sequence>
        <xs:element name="references" type=" PROPERTIES_REFERENCE_Type"/>
    </xs: sequence >
</xs:complexType>
<xs:complexType name="PROPERTIES_REFERENCE_Type">
    <xs:sequence>
        <xs:element name="property"
            type="PROPERTY_REFERENCE_Type" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="PROPERTY_REFERENCE_Type">
    <xs:attribute name="property_ref" type="PropertyId" use="required"/>
</xs:complexType>
```

Figure 19 – XML representation of a multi-valued reference between CIIM ontology concepts

5.3.3 UML diagrams color conventions

In UML diagrams, a color convention is used to highlight those XML attributes and XML element that are mandatory for the description of any information elements (CIIM ontology concepts or pieces of information). The convention is:

— black line / text when the information element is mandatory,

— gray line / text when the information element is optional.

5.3.4 Description of the structure of all OntoML complex types

In the following, the structure and content of all the OntoML complex type is defined through a set of clauses. Each clause focuses on one particular XML complex type or possibly a small number of related XML complex types not embedded within a single XML type. This XML type or these XML types are called the clause main type or types. The clause main type or types are clearly identified by the name and the header of the clause.

EXAMPLE Clause 5.6 is titled “Root class of an ontology" and the header says “In OntoML, every ontology pieces of information are gathered into a general structure that is a **DICTIONARY_TYPE** XML complex type”. **DICTIONARY_TYPE** is then clause main type.

To make the description more synthetic, the same clause also defines the content and structure of a number of other OntoML complex types that are connected with the clause main type or types, either by inheritance or by composition.

The description of these set of complex type is as follows

5.3.4.1 Graphical presentation

In each clause, the complete structure of the clause main type or types is defined graphically using the UML-like notations presented above. The clause main type or types may include embedded XML elements whose content models are defined by complex types.

Some of these XML complex types are represented as squared-boxes. This means that the complete structure and content of these complex types are also defined in the current clause. Their structure are defined in the same figure as the one that defines their embedding complex-type.

EXAMPLE 1 In Figure 21, the **HEADER_Type** embeds a **ontoml_information** XML element, whose content model is defined by an **INFORMATION_Type** XML complex type. This type is represented as a squared-box.

Thus, its complete structure is defined in Figure 22: it embeds different XML elements: **synonnymous_names**, **preferred_name**, **short_name**, **icon**, **remark** and **note** XML elements.

The content of such an XML complex type is defined under the "Internal type definition" header of the same clause.

EXAMPLE 2 In clause 5.5 the content of the **INFORMATION_Type** is defined under the "Internal type definition" header as follows: "the list of class descriptions contained in the dictionary".

Some other of these XML complex types are represented as rounded boxes. Their structure and content are defined in another clause of the document, whose number is defined under the "External type definition" header of the current clause.

EXAMPLE 3 In clause 5.5, the content model of the **icon** embedded XML element is a **GRAPHICS_Type** XML complex type. The corresponding box is rounded. This means that this complex type is defined in another clause. Under the "External type definition" header of clause 5.5, it is specified that **GRAPHICS_TYPE** is defined in Clause 7.2.3.2.

5.3.4.2 Internal item definition

Under the "Internal item definition" header of each clause, all the XML attributes, and all the XML elements that are embedded within all the XML complex type defined in this clause are defined. Those items, XML attributes or embedded XML elements, that belong to the clause main type are listed by their names. Those items that belong to an embedded XML element whose complex type is represented by a squared box are identified using a path notation starting from the main element. The path separator is stash ('/')

EXMPLE 1 Under the "Internal item definition" header of clause 5.6, the definition of the **class** XML element that is embedded within the **contained_classes** element of **DICTIONARY_Type** is associated with the following identifier: **contained_classes/class.**

When the clause addresses several clause main types connected by inheritance relationship, those items that belong to the root of the inheritance hierarchy are not qualified, those items that belong to children classes are qualified by the name of the class between parenthesis.

EXAMPLE 2 Clause 5.7.3 defines the simple ontology-level properties that include **PROPERTY_TYPE**, **NON_DEPENDENT_P_DET_Type**, **CONDITION_DET_type** and **DEPENDENT_P_DET_Type**. Under the "Internal item definition" header of this clause, the definition of the **depends_on** XML element that is embedded within the **DEPENDENT_P_DET_Type** subtype of **PROPERTY_Type** is associated with the following identifier: **depends_on** (**DEPENDENT_P_DET_Type**).

5.3.4.3 Internal type definition

Under the "Internal type definition" header of the clause, the content of the following type are defined:

— types of the attributes of all the XML complex types defined in the clause;

— types of all the XML elements that are embedded within one of the XML complex types defined in the clause and whose XML type are simple types;

— types of those XML elements that are embedded within one of the XML complex types defined in the clause and whose XML type are XML complex types represented as squared boxes in the figure.

5.3.4.4 External type definition

Under the "External type definition" header of the clause, each XML complex type represented in the figure as a rounded box are associated with the clause number of the clause where they are defined.

5.4 OntoML general structure

An OntoML compliant XML document instance allows to represent data describing an ontology, instances, or both. The upper level of a OntoML document instance is defined through the **ONTOML_Type** XML complex type, as is illustrated in Figure 20.

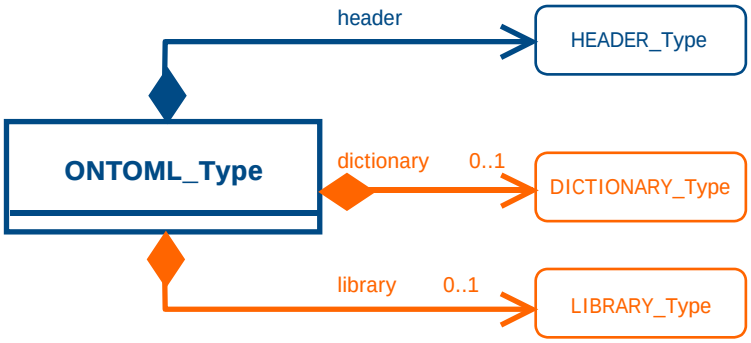

Figure 20 – Ontology structure UML diagram

Internal item definitions:
header: general information about the file that is exchanged.

dictionary: the CIIM ontology concepts that constitute the exchanged ontology.

library: the set of product descriptions that constitute the content of the exchanged library.

NOTE Either a dictionary XML element, or a library XML element or both shall be defined.

External type definitions:
HEADER_Type: the specification of the OntoML XML document header. See Clause 5.5.

DICTIONARY_Type: the specification of the OntoML ontology. See Clause 5.4.

LIBRARY_Type: the specification of the OntoML library. See Clause 6.

5.5 OntoML header

The OntoML header provides:the version of this part of ISO 13584 used to create the OntoML XML document instance the human readable information about the XML document instance. Additionally, it gives information about the general structure of the OntoML document instance. It is represented by the **HEADER_Type** XML complex type as illustrated in Figure 21.

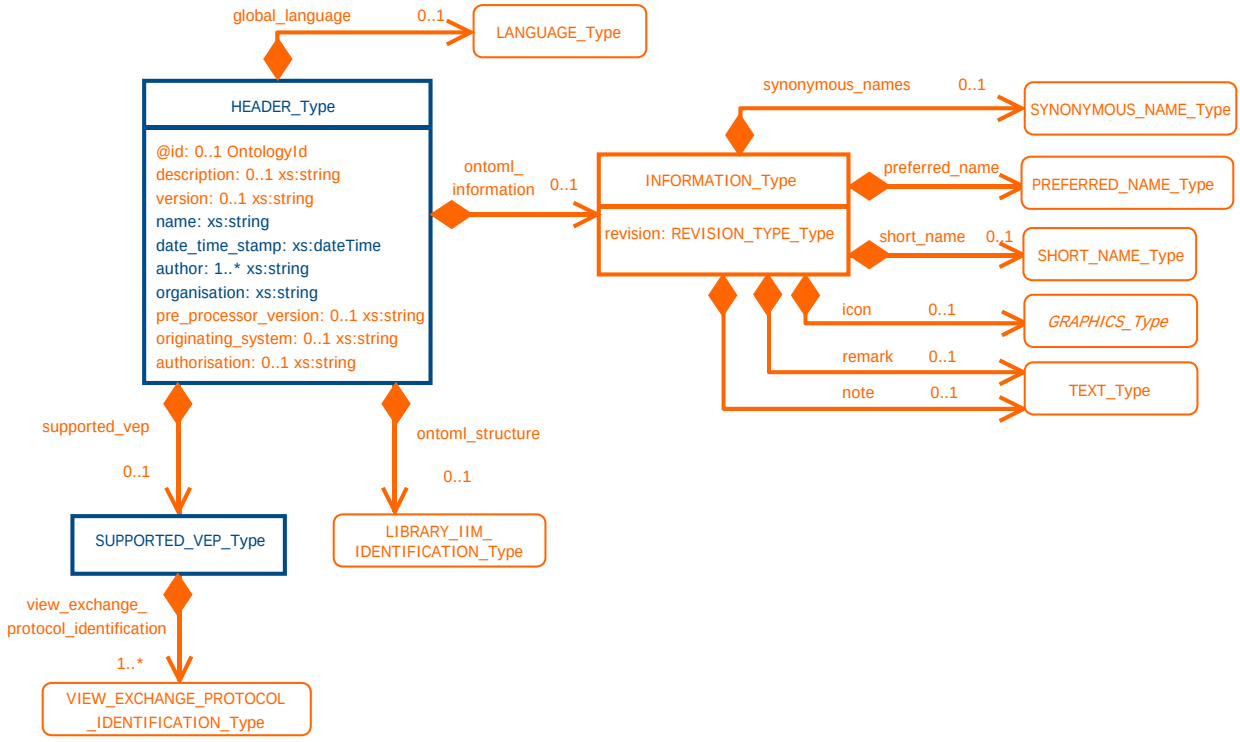

Figure 21 – Ontology header structure

Internal item definitions:
@id: the possible dictionary and / or library identifier.

global_language: the possible global language used to describe non translated information associated to any ontology concept.

description: an informal description of the content of the OntoML document instance.

version: the version of the OntoML schema to which the exchange structure conforms.

name: the string used to name this particular OntoML document instance.

date_time_stamp: the date and time specifying when the exchange structure was created.

author: the name and mailing address of the person responsible for creating the exchange structure.

organisation: the group or organisation with which is responsible for the ontology exchange structure / ontology document instance.

pre_processor_version: the system used to create the exchange structure, including the system product name and version.

originating_system: the system from which the data in this exchange structure originated.

authorisation: the name and mailing address of the person who authorized the sending of the exchange structure.

supported_vep: the list of view exchange protocols supported by the dictionary and / or library.

supported_vep/view_exchange_protocol_identification: a view exchange protocol supported by the dictionary and / or library.

ontoml_structure: the library integrated information model that the OntoML dictionary and / or library realizes.

ontoml_information/revision: the dictionary and / or library revision number.

ontoml_information/preferred_name: the name of the dictionary and / or library that is preferred for use.

ontoml_information/short_name: the abbreviation of the preferred name.

ontoml_information/synonymous_names: the set of synonymous names.

ontoml_information/icon: an optional graphics which represents the description associated with the different names.

ontoml_information/note: further information on any part of the dictionary and / or library, which is essential to the understanding.

ontoml_information/remark: explanatory text further clarifying the meaning of this dictionary and / or library.

Internal type definitions:
INFORMATION_Type: clear text information, possibly translated, of the delivered dictionary and / or library.

REVISION_TYPE_Type: a string (**xs:string** XML Schema datatype) that represents the values allowed for a revision. Its value length shall not exceed 3 characters.

SUPPORTED_VEP_Type: the specification of the view exchange protocols supported by the dictionary and / or library.

External type definitions:
OntologyId: see Clause 8.1.

GRAPHICS_Type: See Clause 7.2.3.2.

LANGUAGE_Type: see Clause 7.1.1.

LIBRARY_IIM_IDENTIFICATION_Type: See Clause 7.7.1.

PREFERRED_NAME_Type: See Clause 7.1.2.

SHORT_NAME_Type: See Clause 7.1.2.

SYNONYMOUS_NAME_Type: See Clause 7.1.2.

TEXT_Type: See Clause 7.1.2.

VIEW_EXCHANGE_PROTOCOL_IDENTIFICATION_Type: See Clause 7.7.2.

5.6 Root class of an ontology

In OntoML, every ontology pieces of information are gathered into a general structure that is a **DICTIONARY_TYPE** XML complex type. It is illustrated in Figure 22.

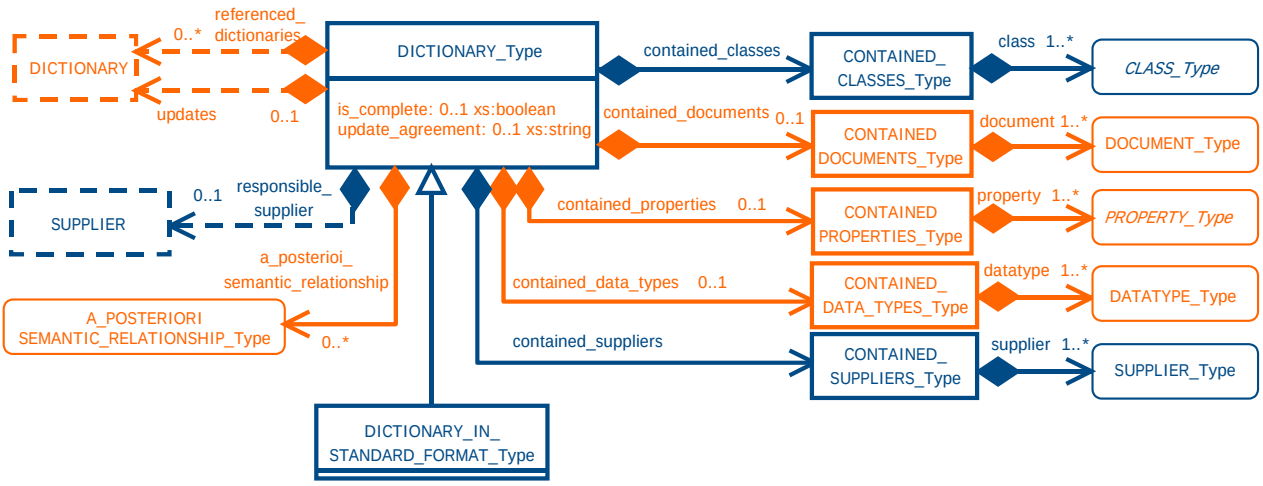

Figure 22 – Root class of an ontology

Internal item definitions:

is_complete: specifies whether the dictionary describes completely the exchanged ontology or only its changes.

NOTE 1 The **is_complete** XML element is only used when the dictionary is identified through its **@id** XML attribute.

update_agreement: the identifier, if any, that identifies the process to be used for creating the dictionary on the receiving system from the list of dictionary defined in the **updates** XML element. The **update_agreement** may only be used when the **updates** XML element is itself used.

responsible_supplier: the possible data supplier responsible for the dictionary elements.

NOTE The supplier of all or part of the dictionary content is referenced as the **responsible_supplier** only when she is the responsible of the OntoML document instance. Else, she should be referenced in the **contained_supplier** XML element.

contained_classes: the list of class descriptions contained in the dictionary

contained_classes/class: a class description contained in the dictionary

contained_documents: the list of document descriptions contained in the dictionary.

contained_document/documents: a document description contained in the dictionary.

contained_properties: the list of property descriptions contained in the dictionary.

contained_properties/property: a property description contained in the dictionary.

contained_data_types: the list of value domain descriptions contained in the dictionary.

contained_data_types/datatype: a value domain description contained in the dictionary.

contained_suppliers: the list of supplier descriptions contained in the dictionary.

contained_supplier/supplier: a supplier description contained in the dictionary.

a_posteriori_semantic_relationship: the list of a posteriori relationships that are specified in the dictionary.

referenced_dictionaries: the dictionary identifiers, if any, referencing the other dictionaries for which some classes are referenced in this dictionary.

updates: dictionary identification, if any, of the dictionary that should be already available on the receiving system to be able to create the complete content of this dictionary.

NOTE 2 The **updates** XML element may only exist when the **identified_by** XML element exists, and when the **is_compelete** XML element is valued to false.

Internal type definitions:
CONTAINED_CLASSES_Type: sequence of classes descriptions.

CONTAINED_DATATYPES_Type: sequence of named data types descriptions.

CONTAINED_DOCUMENTS_Type: sequence of documents descriptions.

CONTAINED_PROPERTIES_Type: sequence of properties descriptions.

CONTAINED_SUPPLIERS_Type: sequence of suppliers descriptions.

DATE_TYPE_Type: the values allowed for a date (the specific **xs:date** XML Schema datatype).

DICTIONARY_IN_STANDARD_FORMAT_Type: a dictionary that only uses external file protocols that are allowed either by the library integrated information model indicated by the **library_structure** XML element or the view exchange protocols referenced in the **supported_vep** XML element, both defined in the **HEADER_Type** XML complex type.

External type definitions:
A_POSTERIORI_SEMANTIC_RELATIONSHIP_Type: See Clause 7.6.

CLASS_Type: dictionary class description. See Clause 5.7.2.

DATATYPE_Type: dictionary datatype description. See Clause 5.7.6.

DOCUMENT_Type: dictionary document description. See Clause 5.7.7.

PROPERTY_Type: dictionary property description. See Clause 5.7.4.

SUPPLIER_Type: supplier description. See Clause 5.7.1.

5.7 OntoML representation of CIIM ontology concepts

In this clause, the OntoML representation of the various CIIM ontology concepts is defined.

5.7.1 Supplier

The supplier ontology concept stands for the description of an organization responsible of some information identified in an OntoML document instance. It is represented as illustrated in the UML diagram of Figure 23.

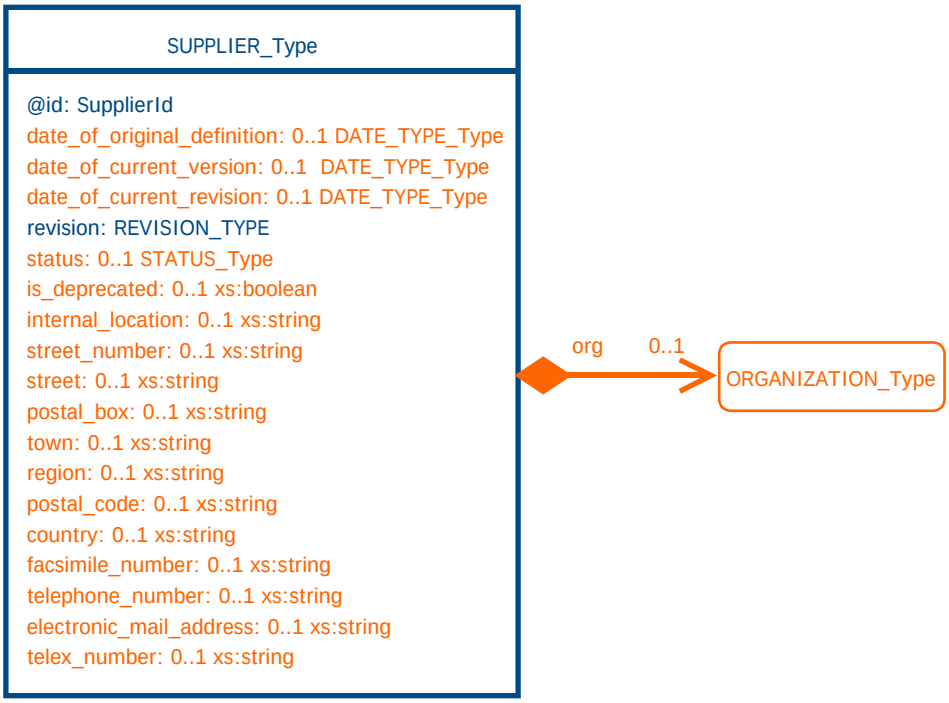

Figure 23 – Supplier ontology concept UML diagram

Internal item definitions:
@**id**: the supplier identifier.

date_of_original_definition: the date associated to the first stable version of the supplier definition.

date_of_current_version: the date associated to the present version of the supplier definition.

date_of_current_revision: the date associated to the present revision of the supplier definition.

revision: the revision number of the present supplier definition.

status: defines the life cycle state of the supplier definition.

NOTE 1 Allowed **status** values are defined by private agreement between the dictionary supplier and dictionary users.

NOTE 2 If the **status** XML element is not provided, and if this supplier definition is not deprecated as denoted by a possible **is_deprecated** XML element, then the supplier definition has the same standardization status as the whole ontology into which it is used. In particular, if the ontology is standardized, this supplier definition is part of the current edition of the standard.

is_deprecated: a Boolean that specifies, when true, that the supplier definition shall no longer be used.

internal_location: organization-defined address for internal mail delivery.

street_number: the number of a building in a street.

street: the name of a street.

postal_box: the number of a postal box.

town: the name of a town.

region: the name of a region.

postal_code: the code that is used by the country's postal service.

country: the name of a country.

facsimile_number: the number at which facsimiles may be received.

telephone_number: the number at which telephone calls may be received.

electronic_mail_address: the electronic address at which electronic mail may be received.

telex_number: the number at which telex messages may be received.

org: organizational data of this supplier.

Internal type definitions:
SupplierId: see Clause 8.1.

DATE_TYPE_Type: identifies the values allowed for a date (the specific **xs:date** XML Schema datatype).

REVISION_TYPE_Type: a string (**xs:string** XML Schema datatype) that represents the values allowed for a revision. Its value length shall not exceed 3 characters.

STATUS_Type: a string (**xs:string** XML Schema datatype) that represents the values allowed for a status. This string shall not contain any hyphen « - » or space characters.

External type definitions:
ORGANIZATION_Type: See Clause 7.8.1.

TRANSLATION_Type : see Clause 7.1.3.

5.7.2 Simple-level ontology class

OntoML defines three subtypes of the generic and abstract concept of class as simple classes:

— *item class*: allows to characterize any kind of items, and in particular products, by a class belonging and a set of property value pairs. Item classes belong to a single *is-a* hierarchy associated with inheritance.

— *categorization class*: allows to classify an item characterized as an item class in various classification system. Such a classification does not imply any additional properties.

— *item class case-of*: a special kind of item class that, beside inheriting properties from its possible is-a parent, borrows some properties from some other existing classes that encompass its own scope.

NOTE Product characterization class and categorization class are defined in Clause 6 of the draft ISO/IEC CD Guide 77-2.

5.7.2.1 Item class and categorization class

Figure 24 describes the structure of item class and categorization class.

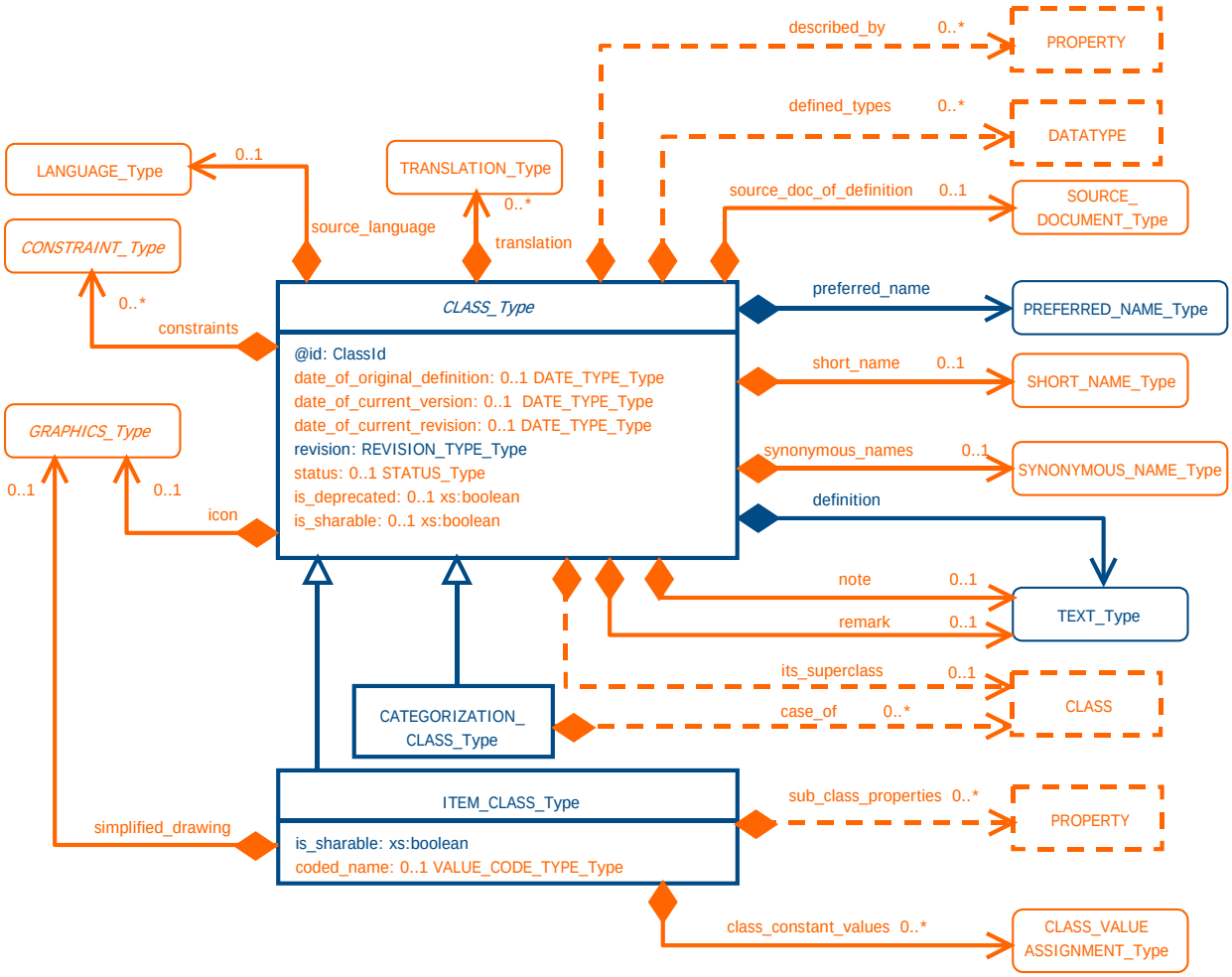

Figure 24 – Simple class ontology concept UML diagram

Both item class (represented by the **ITEM_CLASS_Type** XML complex type) and categorization class (represented by the **CATEGORIZATION_CLASS_Type** XML complex type) inherits the XML content description defined in the abstract **CLASS_Type** XML complex type.

NOTE 1 The most basic representation of classes only requires to define an identifier (**@id**), a **revision** number, a **preferred_name** and a **definition**.

Internal item definitions:
@id: the class identifier.

date_of_original_definition: the date associated to the first stable version of the class definition.

date_of_current_version: the date associated to the present version of the class definition.

date_of_current_revision: the date associated to the present revision of the class definition.

revision: the revision number of the present class definition.

status: defines the life cycle state of the class definition.

NOTE 2 Allowed **status** values are defined by private agreement between the dictionary supplier and dictionary users.

NOTE 3 If the **status** XML element is not provided, and if this class definition is not deprecated as denoted by a possible **is_deprecated** XML element, then the class definition has the same standardization status as the whole ontology into which it is used. In particular, if the ontology is standardized, this class definition is part of the current edition of the standard.

source_language: the language in which the class definition was initially defined and that provides the reference meaning in case of translation discrepancy.

is_deprecated: a Boolean that specifies, when true, that the class definition shall no longer be used.

preferred_name: the name of the class that is preferred for use, possibly translated.

short_name: the abbreviation of the preferred name, possibly translated.

synonymous_names: the set of synonymous names of the preferred name, possibly translated.

icon: a graphics representing the description associated with the names.

definition: the text describing this class, possibly translated.

source_doc_of_definition: the possible source document from which the definition comes.

note: further information on any part of the class, which is essential to its understanding, possibly translated.

remark: explanatory text further clarifying the meaning of this class, possibly translated.

translation: the possible set of translations information provided for the translatable items.

its_superclass: reference to the class the current one is a subclass of.

described_by: the list of references to the additional properties available for use in the description of the instances within this class, and any of its subclasses.

NOTE 4 Every property referenced in the **described_by** collection is said applicable to the class.

defined_types: the set of references to the additional named types that can be used for various properties throughout the inheritance tree descending from this class.

NOTE 5 Every named data type referenced in the **defined_types** collection is said applicable to the class.

constraints: the set of constraints that restrict the target domains of values of some visible properties of the class to some subsets of their inherited domains of values.

simplified_drawing: drawing that can be associated to the described class.

sub_class_properties (**ITEM_CLASS_Type**): declares properties as class-valued, i.e. in subclasses one single value will be assigned per class.

class_constant_values (**ITEM_CLASS_Type**): assignments in the current class for class-valued properties declared in superclasses.

NOTE 6 **sub_class_properties** and **class_constant_values** define class selectors, as specified in Clause 5.5 of the draft ISO/IEC CD Guide 77-2.

coded_name (**ITEM_CLASS_Type**): a possible coded name of the class.

is_sharable (**ITEM_CLASS_Type**): when false, it specifies that instances of the described class may only exist when referenced by another instance they belong to.; when true, it specifies that the instances of the described class may referenced by several instances.

case_of (**CATEGORIZATION_CLASS_Type**): the categorization classes that are one step above the categorization class in a case-of class inclusion hierarchy.

Internal type definitions:
ClassId: see Clause 8.1.

DATE_TYPE_Type: identifies the values allowed for a date (the specific **xs:date** XML Schema datatype).

REVISION_TYPE_Type: a string (**xs:string** XML Schema datatype) that represents the values allowed for a revision. Its value length shall not exceed 3 characters.

STATUS_Type: a string (**xs:string** XML Schema datatype) that represents the values allowed for a status.

VALUE_CODE_TYPE_Type: a string (**xs:string** XML Schema datatype) that represents the values allowed for a value code. Its value length shall not exceed 18 characters.

Internal type definitions:
CLASS_VALUE_ASSIGNMENT_Type: See Clause 5.7.2.3.

CONSTRAINT_Type: See Clause 7.5.

LANGUAGE_Type: see Clause 7.1.1.

GRAPHICS_Type: See Clause 7.2.3.2.

PREFERRED_NAME_Type: See Clause 7.1.2.

SHORT_NAME_Type: See Clause 7.1.2.

SOURCE_DOCUMENT_Type: See Clause 7.2.3.1.

SYNONYMOUS_NAME_Type: See Clause 7.1.2.

TEXT_Type: See Clause 7.1.2.

TRANSLATION_Type : see Clause 7.1.3.

5.7.2.2 Item class case-of

Standard product ontologies are designed to represent precisely and formally the product classes and the product properties on the definition for which experts of some product domain have agreed.

In each particular organization:

— only a few of all the standard product characterization classes may be useful;

— these standard product characterization classes may be defined in different standard product ontologies;

— some organization-specific classes may exist;

— only a few of the standard properties may be considered as needed, and

— some organization specific properties may be in use.

Thus, for representing such a domain using directly the standard product ontology within each organization would be not efficient.

The case-of relationship allows each organization to define its own class hierarchy while allowing to import all needed standard properties, thus providing for data integration and for data exchange with other organizations.

The case-of relationship may also be used within standard dictionaries.

In OntoML, the **ITEM_CLASS_CASE_OF_Type** XML complex type is used for such a purpose. It is illustrated in Figure 25.

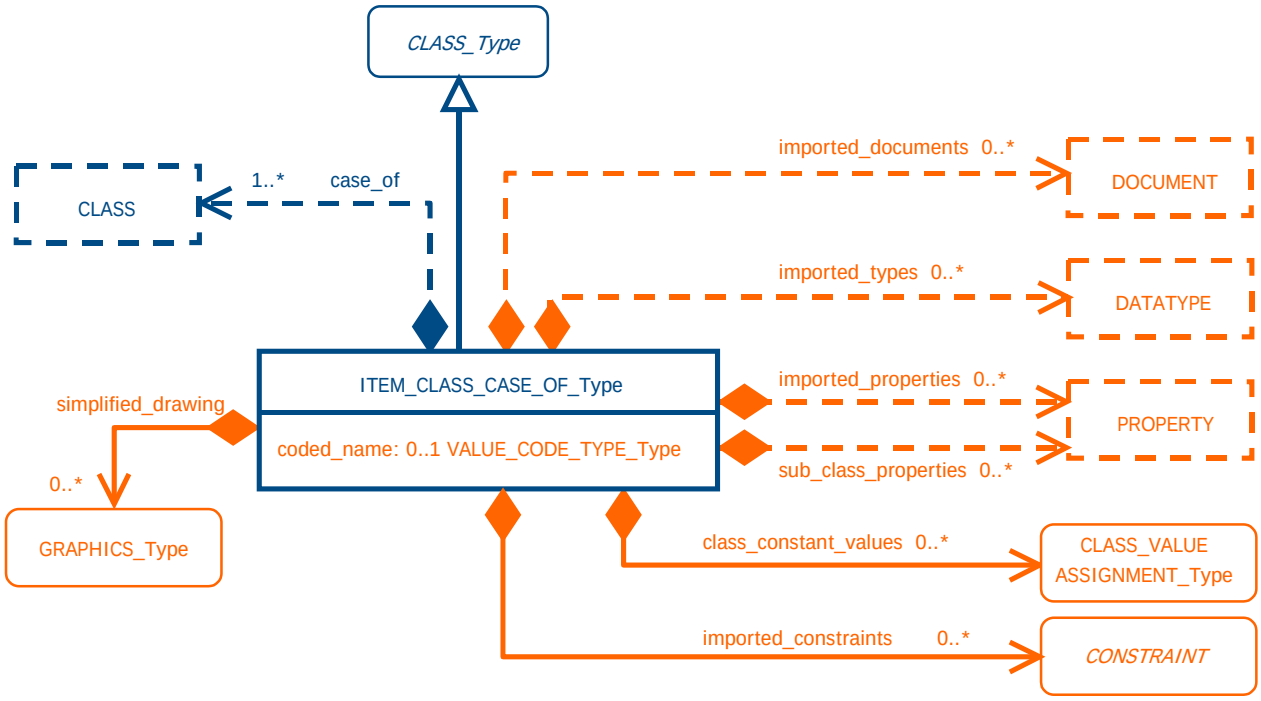

Figure 25 – Item class case-of relationship

Internal item definitions:
case_of: the class(es) for which the described class is case-of; it is represented by a reference to the corresponding class(es) ontology concept identifier(s).

NOTE 1 The case-of relationship and its use are described in Clause 7 of the ISO/IEC JWG1 Guide for specification of product properties and classes.

NOTE 2 The **case_of** XML element shall reference either item classes (**ITEM_CLASS_Type**) or item class case-of classes (**ITEM_CLASS_CASE_OF_Type**)

imported_properties: the imported property(ies) from the **case_of** class(es) that are required to describe the current class; it is represented by a reference to the corresponding property(ies) ontology concept identifier(s).

imported_types: the imported type(s) from the **case_of** class(es) that are required to describe the current class; it is represented by a reference to the corresponding type(s) ontology concept identifier(s).

imported_documents: the imported document(s) from the **case_of** class(es) that are required to describe the current class; it is represented by a reference to the corresponding document(s) ontology concept identifier(s).

imported_constraints: the imported constraint(s) from the **case_of** class(es) that applies on properties referenced in the **imported_properties** collection.

sub_class_properties (**ITEM_CLASS_CASE_OF_Type**): declares properties as class-valued, i.e. in subclasses one single value will be assigned per class.

class_constant_values (**ITEM_CLASS_CASE_OF_Type**): assignments in the current class for class-valued properties declared in superclasses.

NOTE 3 **sub_class_properties** and **class_constant_values** define class selectors, as specified in Clause 5.5 of the draft ISO/IEC CD Guide 77-2.

simplified_drawing: drawing that can be associated to the described class.

Internal type definitions:
CLASS_VALUE_ASSIGNMENT_Type: See Clause 5.7.2.3.

GRAPHICS_Type: See Clause 7.2.3.2.

VALUE_CODE_TYPE_Type: a string (**xs:string** XML Schema datatype) that represents the values allowed for a value code. Its value length shall not exceed 18 characters.

5.7.2.3 Class valued property

A property may be specified as taking a single value for a given class. Such a property is called a class valued property. The definition, of such a property is twofold:

— it is declared in a given class;

NOTE 1 The property is described as any other property, but with two restrictions. Firstly, the property value domain shall be an enumeration of string codes (see Clause 7.3.3), and the property shall be referenced in the **sub_class_properties** XML element of an **ITEM_CLASS_Type** (see Clause 5.7.2.1) or **ITEM_CLASS_CASE_OF_Type** (see Clause 5.7.2.2).

— a value is assigned in every subclasses of this given class.

NOTE 2 The value assignment is inherited by all the subclasses of the class where it is defined.

The assignment of a class valued property value to a class is represented by the **CLASS_CONSTANT_VALUE_Type** XML complex type as illustrated in Figure 26.

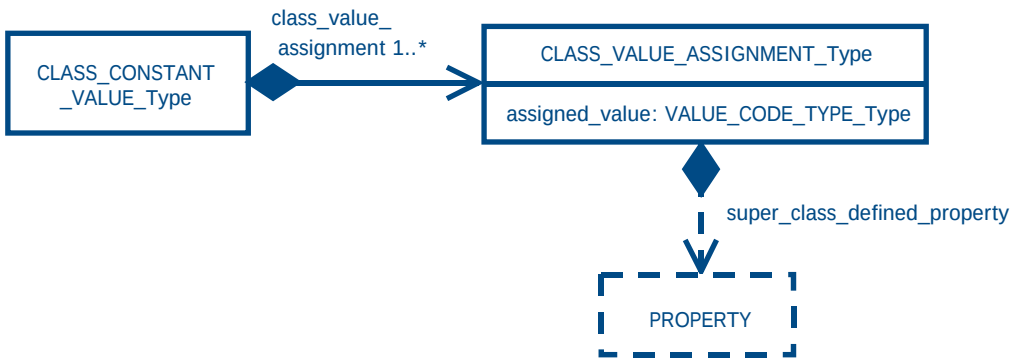

Figure 26 – Class value assignment structure

Internal item definitions:
class_value_assignment: the specification of all the class value assignments.

class_value_assignment/assigned_value: value assigned to the property, valid for the whole class referring this class value assignment.

NOTE 3 The **assigned_value** shall belong to referenced subclass property value domain.

class_value_assignment/super_class_defined_property: reference to the property (defined in a superclass as being a subclass property) to which a certain value is assigned.

NOTE 4 A property is defined as being a subclass property when it appear in the **sub_class_properties** XML element defined in an **ITEM_CLASS_Type** or **ITEM_CLASS_CASE_OF_Type** XML complex type.

Internal type definitions:
CLASS_VALUE_ASSIGNMENT_Type: the specification of a class value assignment.

VALUE_CODE_TYPE_Type: a string (**xs:string** XML Schema datatype) that represents the values allowed for a value code. Its value length shall not exceed 18 characters.

5.7.3 Advanced-level ontology class

This clause defines modelling constructs that may only be used within the advanced subset of OntoML.

NOTE Details of these modelling constructs are defined in ISO 13584-24:2003.

In some engineering contexts, product characterization classes are not sufficient to describe the engineering knowledge. Indeed, in addition to characterization of products, it is often required to associate other representations such that engineering models, also called functional models.

EXAMPLE Geometry model and procurement model are example of functional models.

For that purpose, OntoML provides two main kinds of classes:

— functional view classes;

— functional model classes.

5.7.3.1 Functional view class

A functional view class allows to characterize a particular representation category, for instance corresponding to a particular discipline, that may prove useful for various products. If needed, a functional view class may contain properties called view control variables to further specify the representation category.

EXAMPLE 1 A *geometry* functional view class specifies that the view is a geometrical representation. This maybe further defined by a view control variable *geometry_level* whose values would be *2D* or *3D*.

The representation itself is intended to be represented for each particular class of products by functional model classes. A functional view class is represented either by a **FUNCTIONAL_VIEW_CLASS_Type** or a **NON_INSTANCIABLE_FUNCTIONAL_VIEW_CLASS_Type** XML complex type. It is illustrated in Figure 27.

A non instanciable functional view class defines a representation category that only consists of property-value pairs that must be explicitly contained in the functional models that reference this non instanciable functional view class.

EXAMPLE 2 A non instanciable functional view class may be used for representing *inventory status* of products. This non instanciable functional view class might be defined without any view control variable because the concept of *inventory status* defines a unique point of view that does not need to be further refined. Then for each item class describing a class of products of a company, a functional model class view-of this item class in the *inventory status* point of view would need to be defined. This functional model class would contain for example three properties: the *part number* (imported from the item class), the *inventory size*, i. e., number of each parts that are currently available in the inventory, the *quantity of order* of this part. The *part number* might be used by the system to make a natural joint and to display the *inventory status* properties together with the characterization properties defined in the item class each time this item class is displayed.

NOTE 1 ISO 13584 methodology recommends that only properties that are essential for a product should be defined in item classes. Properties that are functional point of view-dependent should be defined in functional model class.

A functional view class defines representation categories that contains beside property-value pairs, a particular construct, called an EXPRESS representation, that need to be generated dynamically by the functional model class.

NOTE 2 The information model of an EXPRESS representation is defined in ISO 10303-41.

EXAMPLE 3 ISO 13584-102 defines the *basic_geometry* functional view class together with the *geometry_level*, *detail_level*, *side*, *variant*, and *unreg_variant* view control variables. Its content must be a **geometrical_representation** as defined in ISO 10303-42. ISO 13584-102 also defines how such a representation should be created from a functional model class that contains a program based on the ISO 13584-31 geometric programming interface.

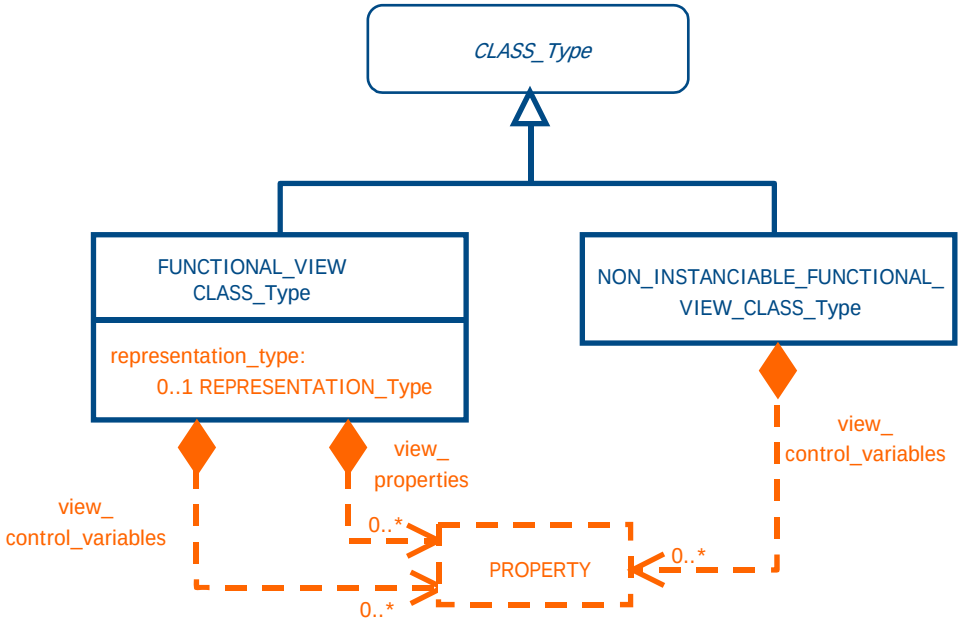

Figure 27 – Advanced-level ontology class concept UML diagram: functional view class

The inherited (from the **CLASS_Type** XML complex type) **described_by** XML element shall not be used, neither when specifying a class as a **FUNCTIONAL_VIEW_CLASS_Type** or a **NON_INSTANCIABLE_FUNCTIONAL_VIEW_CLASS_Type**.

Internal item definitions:
view_control_variables: the list of properties that further defines the point of view specified the functional view class.

NOTE 3 Properties referenced in the **view_control_variables** list shall be representation properties (**REPRESENTATION_P_DET_Type** XML complex type, see Clause 5.7.5).

NOTE 4 Each view control variable data type is a **NON_QUANTITATIVE_INT_TYPE_Type** (see Clause 7.3.7) whose values are successive integers.

representation_type (**FUNCTIONAL_VIEW_CLASS**): the specification, in the format of a string, of the ISO 10303 representation subtype the functional view content is a subtype of.

view_properties (**FUNCTIONAL_VIEW_CLASS**): the list of properties that must be generated in the view by a functional model class.

NOTE 5 Properties referenced in the **view_properties** list shall be representation properties (**REPRESENTATION_P_DET_Type** XML complex type, see Clause 5.7.5).

Internal type definitions:
REPRESENTATION_Type: a string (**xs:string** XML Schema datatype) that identifies the values allowed for the representation specification. It shall end with the "_SCHEMA" substring.

5.7.3.2 Functional model class

A functional model class is intended to describe the representation, specified by a functional view class, that may be associated to instances belonging to a product characterization class. This representation is supposed to be contained in the functional model class if a non instanciable functional view class is referenced. It is supposed to be generated by the functional model class if a functional view class is referenced. In order to be able to describe representations, the functional model class may need to import:

— properties and/or types and/or documents from the functional view class that specifies the representation category;

— properties and/or types and/or documents from the functional model class(es) for which the current functional model class could be case-of.

NOTE 1 Engineering models are introduced in Clause 6.8 of the draft ISO/IEC Guide 77-2.

NOTE 2 Functional model classes describe representations, e.g., geometry, independently of any particular product. Functional model class view-of describe representations that are directly associated with particular products.

Such a functional model class, represented by a **FUNCTIONAL_MODEL_CLASS_Type** XML complex type is depicted in Figure 28.

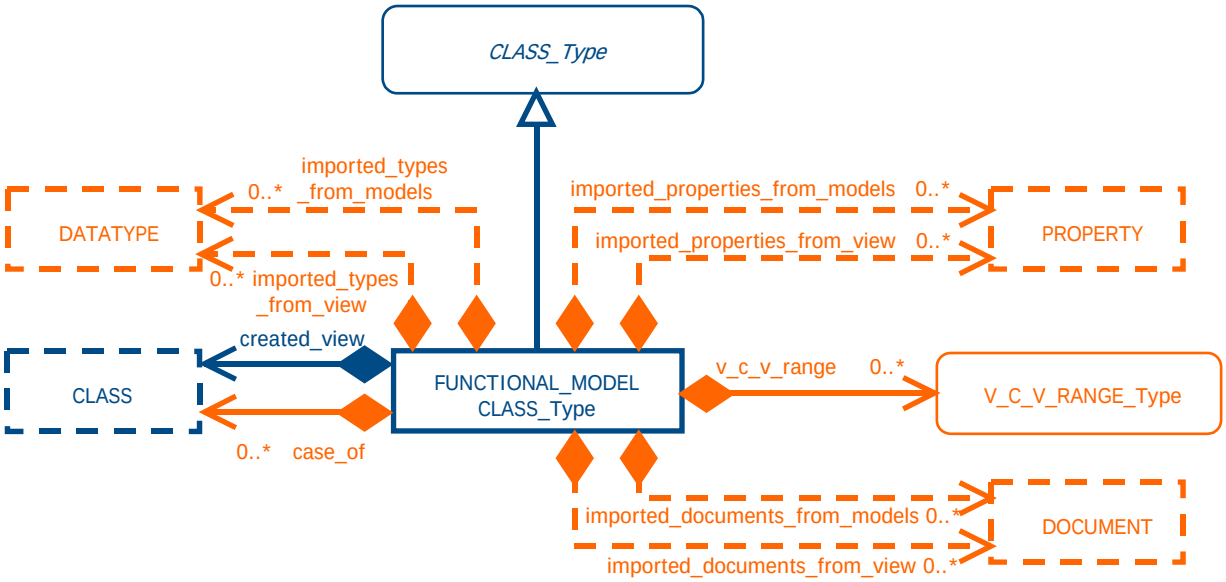

Figure 28 – Advanced class ontology concept UML diagram: functional model class

Internal item definitions:
created_view: the functional view class that characterizes the user point of view addressed by the functional model class, and, in the case of an instanciable functional view class, the views that may be generated by the functional model class.

NOTE 2 The **created_view** XML element shall reference a functional view class (**FUNCTIONAL_VIEW_CLASS_Type** XML complex type) or a non instanciable functional view class (**NON_INSTANCIABLE_FUNCTIONAL_VIEW_CLASS_Type** XML complex type).

case_of: the other functional model classes the current functional model class is a *case-of*.

v_c_v_range: the list of the view control variable ranges that specify the various views the functional model class is able to create. Each of them shall correspond to a view control variable of the functional view that is referenced by the **created_view** XML element. When a view control variable of the functional view defined by the **created_view** element is not represented in the **v_c_v_range** element, its range is its complete value domain.

imported_properties_from_model: the properties that are imported from the *case-of* functional model classes.

imported_properties_from_view: the properties that are imported from the created view. Each view control variable used shall belong to this list.

imported_types_from_model: the types that are imported from the *case-of* functional model classes.

imported_types_from_view: the types that are imported from the created view.

imported_documents_from_model: the documents that are imported from the *case-of* functional model classes.

imported_documents_from_view: the documents that are imported from the created view.

External type definitions:
V_C_V_RANGE_Type: see Clause 5.7.3.4.

5.7.3.3 Functional model class *view-of*

A functional model class whose representations described are directly associated with the product of a particular item class. In such a case, and in addition to the properties and/or types and/or documents possibly imported from the referenced functional view and *case-of* functional model(s), properties and/or types and/or documents may also be imported from the product characterization class for which the current functional model class is specified.

NOTE 1 In particular, the part number should be imported to connect each representation defined by the functional model class view-of with each product of the item class.

NOTE 2 Engineering models are introduced in Clause 6.8 of the draft ISO/IEC Guide 77-2.

Such a functional model class is represented by the **FM_CLASS_VIEW_OF_Type** XML complex type as depicted in Figure 29.

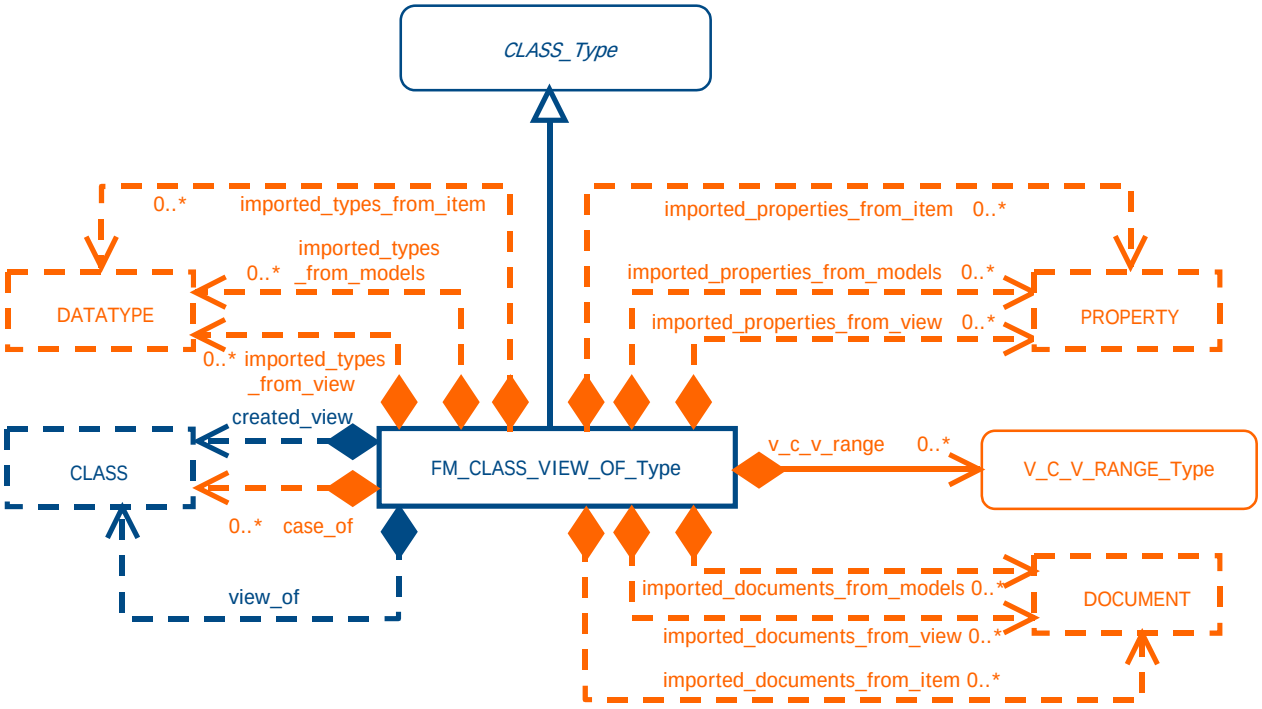

Figure 29 – Advanced class ontology concept UML diagram: functional model class view of

Internal item definitions:

created_view: the functional view class that characterizes the user point of view addressed by the functional model class, and, in the case of an instanciable functional view class, the views that may be generated by the functional model class.

NOTE 2 The **created_view** XML element shall reference a functional view class (**FUNCTIONAL_VIEW_CLASS_Type** XML complex type) or a non instanciable functional view class (**NON_INSTANCIABLE_FUNCTIONAL_VIEW_CLASS_Type** XML complex type).

case_of: the other functional model classes the current functional model class is a *case-of*.

view_of: the product characterization class whose the described functional model class is able to define representations.

NOTE 3 The **view_of** XML element shall reference an item class (**ITEM_CLASS_Type** XML complex type) or an item class *case-of* (**ITEM_CLASS_CASE_OF_Type** XML complex type).

v_c_v_range: the view control variable ranges that specify the various views the functional model class is able to create. Each of them shall correspond to a view control variable of the functional view that is referenced by the **created_view** XML element When a view control variable of the functional view defined by the **created_view** element is not represented in the **v_c_v_range** element, its range is its complete value domain.

imported_properties_from_model: the properties that are imported from the *case-of* functional model classes

imported_properties_from_view: the properties that are imported from the created view. Each view control variable used shall belong to this list.

imported_properties_from_item: the properties that are imported from the item class for which the present class is able to generate the view.

imported_types_from_model: the types that are imported from the *case-of* functional model classes.

imported_types_from_view: the types that are imported from the created view.

imported_types_from_item: the types that are imported from the item class for which the present class is able to generate the view.

imported_documents_from_model: the documents that are imported from the *case-of* functional model classes.

imported_documents_from_view: the documents that are imported from the created view.

imported_documents_from_item: the documents that are imported from the item class for which the present class is able to generate the view.

External type definitions:
V_C_V_RANGE_Type: see Clause 5.7.3.4.

5.7.3.4 View control variable range

Functional views may be further specified using view control variables. Associated to a functional model that references a functional view, each view control variable range specifies which particular view are described by that model.

EXAMPLE Assume that a *geometry* functional view defines a view control variable *geometry_level* whose domain of value is {2D, 3D}. A particular functional model may describe only *2D* views. In this case, the range would be [2D;2D].

NOTE When a view control variable of a functional view referenced by the functional model is not represented using a view control variable range, its range is its complete value domain.

A view control variable is represented by the **VIEW_CONTROL_VARAIBLE_RANGE_Type** XML complex type. All the view control variable ranges are gathered in a container specified by the V_C_V_RANGE_Type. It is illustrated in Figure 30.

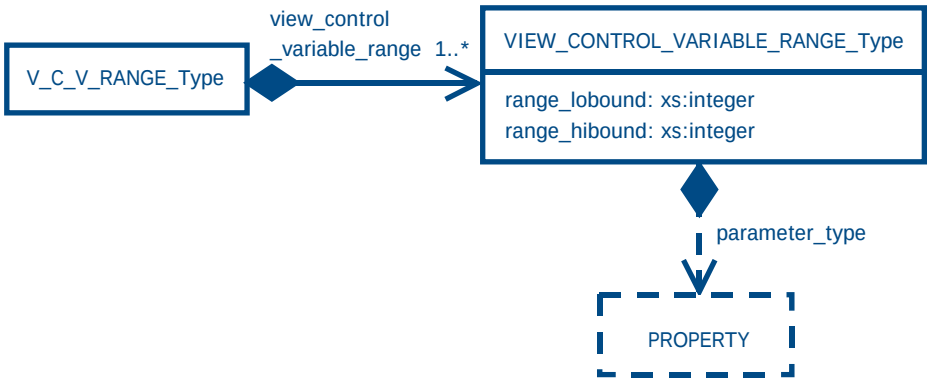

Figure 30 – Mathematical string structure

Internal item definitions:
view_control_variable_range: the specification of the functional model view control variable ranges.

view_control_variable_range/range_lobound: the integer that describes the low bound of the specified range.

view_control_variable_range/range_hibound: the integer that describes the high bound of the specified range.

view_control_variable_range/parameter_type: the reference to the property that plays the role of a view control variable for which the range is described.

Internal item definitions:
VIEW_CONTROL_VARIABLE_RANGE_Type: the specification of a view control variable range.

5.7.4 Simple-level ontology property

In the CIIM, values of a property are either simple values like integer or strings, or other class items. Moreover, the CIIM distinguishes:

— properties, that may be used for characterizing an item, and

— properties that are used only in functional model or functional view classes.

This clause defines the simple-level ontology property that are the properties used to characterize items.

The most commonly used properties in any ontology are the characterization properties that associates an item either with values or with other items. Such a property is represented by a **NON_DEPENDENT_P_DET_Type** XML complex type as illustrated in Figure 31.

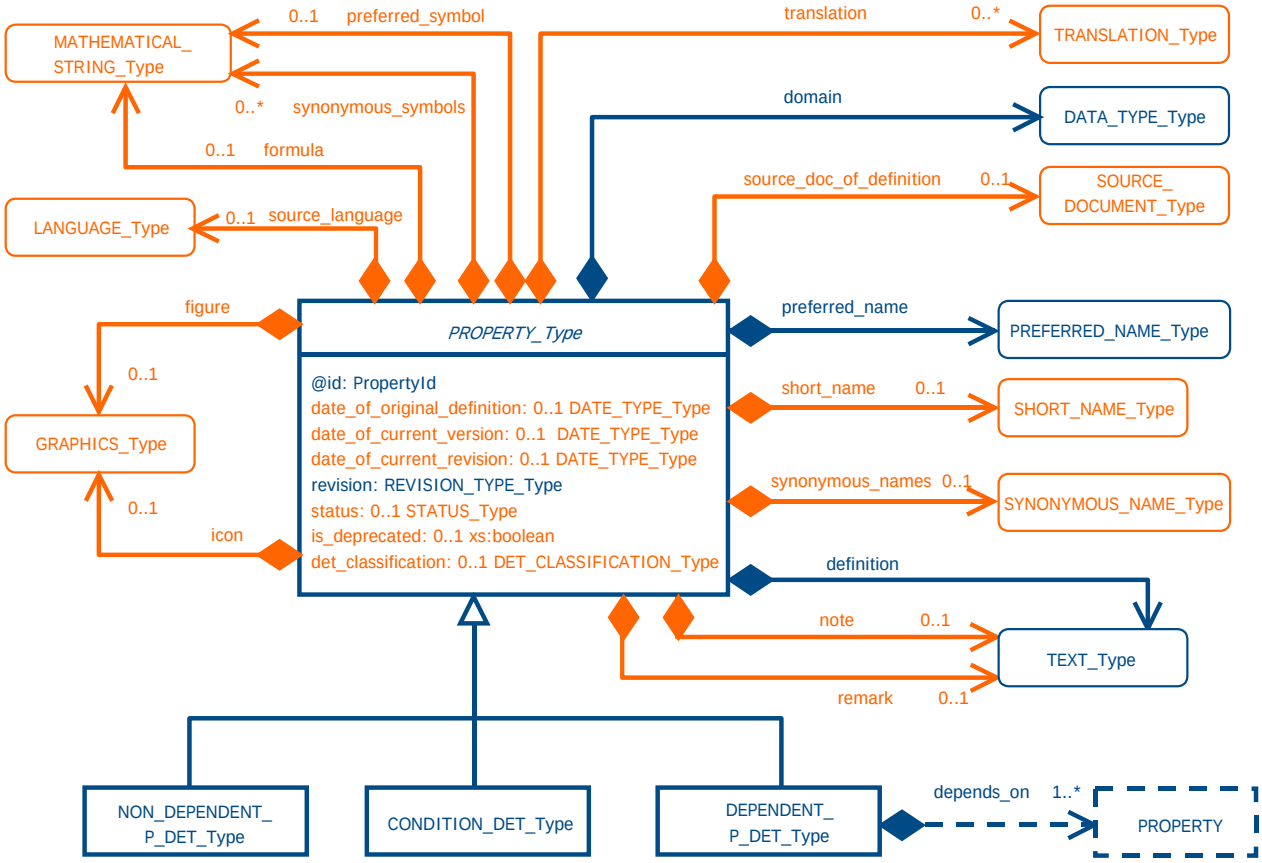

Figure 31 – Simple property ontology concept UML diagram

NOTE 1 Characterization property is defined in Clause 5 of the draft ISO/IEC Guide 77-2.

But, in the real world, no object can be considered as isolated from its environment. Quantitative properties that are measured, therefore, have to be related to conditions under which their values were obtained.

NOTE 2 Strictly speaking, any instance of a dimension should be accompanied by quoting the temperature at which the measurement was made. However, in practice, either this measuring context may be specified in the definition of a property, or it may be considered as not significant. But, for example, the *resistance* of an *electric thermistor* may strongly depend upon *ambient temperature*. Hence this information should always be supplied.

Thus, in addition to the usual characterization properties, that may be considered as context independent properties, it is possible to define context parameters, through the **CONDITION_DET_Type** XML complex type, and context dependent properties, through the **DEPENDENT_P_DET_Type** XML complex type. For this latter kind of property, its context is defined (**depends_on** XML element) by reference to the context parameter(s) global identifier(s).

NOTE 3 Properties and context of evaluation is documented in Clause 5.4 of the ISO/IEC JWG1 Guide for specification of product properties and classes.

NOTE 4 The most basic representation of such a property requires only to define an identifier a **revision** number, a **preferred_name**, a **definition** and its domain of values (**domain** XML element).

Internal item definitions:
@id: the property identifier.

date_of_original_definition: the date associated to the first stable version of the property definition.

date_of_current_version: the date associated to the present version of the property definition.

date_of_current_revision: the date associated to the present revision of the property definition.

revision: the revision number of the present property definition.

status: defines the life cycle state of the property definition.

NOTE 5 Allowed **status** values are defined by private agreement between the dictionary supplier and dictionary users.

NOTE 6 If the **status** XML element is not provided, and if this property definition is not deprecated as denoted by a possible **is_deprecated** XML element, then the property definition has the same standardization status as the whole ontology into which it is used. In particular, if the ontology is standardized, this property definition is part of the current edition of the standard.

source_language: the language in which the property definition was initially defined and that provides the reference meaning in case of translation discrepancy.

is_deprecated: a Boolean that specifies, when true, that the property definition shall no longer be used.

det_classification: the ISO 31 class for this property.

preferred_name: the name of the property that is preferred for use, possibly translated.

short_name: the abbreviation of the preferred name, possibly translated.

synonymous_names: the set of synonymous names of the preferred name, possibly translated.

icon: a graphics representing the description associated with the names.

definition: the text describing this property, possibly translated.

source_doc_of_definition: the possible source document from which the definition comes.

note: further information on any part of the property, which is essential to its understanding, possibly translated.

remark: explanatory text further clarifying the meaning of this property, possibly translated.

translation: the possible set of translations information provided for the translatable items.

domain: the data type (the value domain) associated to the property.

preferred_symbol: a shorter description of this property.

synonymous_symbols: the set of synonymous names of the preferred symbol of the property..

figure: a possible graphics that describes the property..

formula: a mathematical expression for explaining the property.

depends_on (**DEPENDENT_P_DET_type**): set of references identifying the properties on which this property depends on.

Internal type definitions:
DATE_TYPE_Type: identifies the values allowed for a date (the specific **xs:date** XML Schema datatype).

REVISION_TYPE_Type: a string (**xs:string** XML Schema datatype) that represents the values allowed for a revision. Its value length shall not exceed 3 characters.

STATUS_Type: a string (**xs:string** XML Schema datatype) that represents the values allowed for a status.

External type definitions:
PropertyId: see Clause 8.1.

GRAPHICS_Type: See Clause 7.2.3.2.

LANGUAGE_Type: see Clause 7.1.1.

MATHEMATICAL_STRING: See Clause 7.8.2

PREFERRED_NAME_Type: See Clause 7.1.2.

SHORT_NAME_Type: See Clause 7.1.2.

SOURCE_DOCUMENT_Type: See Clause 7.2.3.1.

SYNONYMOUS_NAME_Type: See Clause 7.1.2.

TEXT_Type: See Clause 7.1.2.

TRANSLATION_Type : see Clause 7.1.3.

5.7.5 Advanced-level ontology property

In OntoML, those properties that are defined for describing advanced-level ontology classes, i.e., functional view classes and functional model classes, shall be represented as the **representation_P_DET_Type**. It is shown in Figure 32. Such properties correspond to representation properties that are intended to be used only as functional view class (see clause 5.7.3.1) or functional model class properties (see clause 5.7.3.2 and 5.7.3.3).

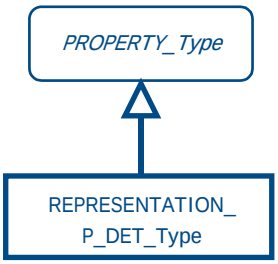

Figure 32 – Advanced property ontology concept UML diagram

NOTE The representation property concept is specified in ISO 13584-24:2002, clause 11.15.1.

5.7.6 Named data types

In some context, it proves useful to define domain of values that are associated with a global identifier and that could be re-used for several properties possibly in several ontologies.

EXAMPLE An ontology defining domain-specific units could be based on the use of named data types.

For that purpose, OntoML proposes a **DATATYPE_Type** XML complex type as illustrated in Figure 33.

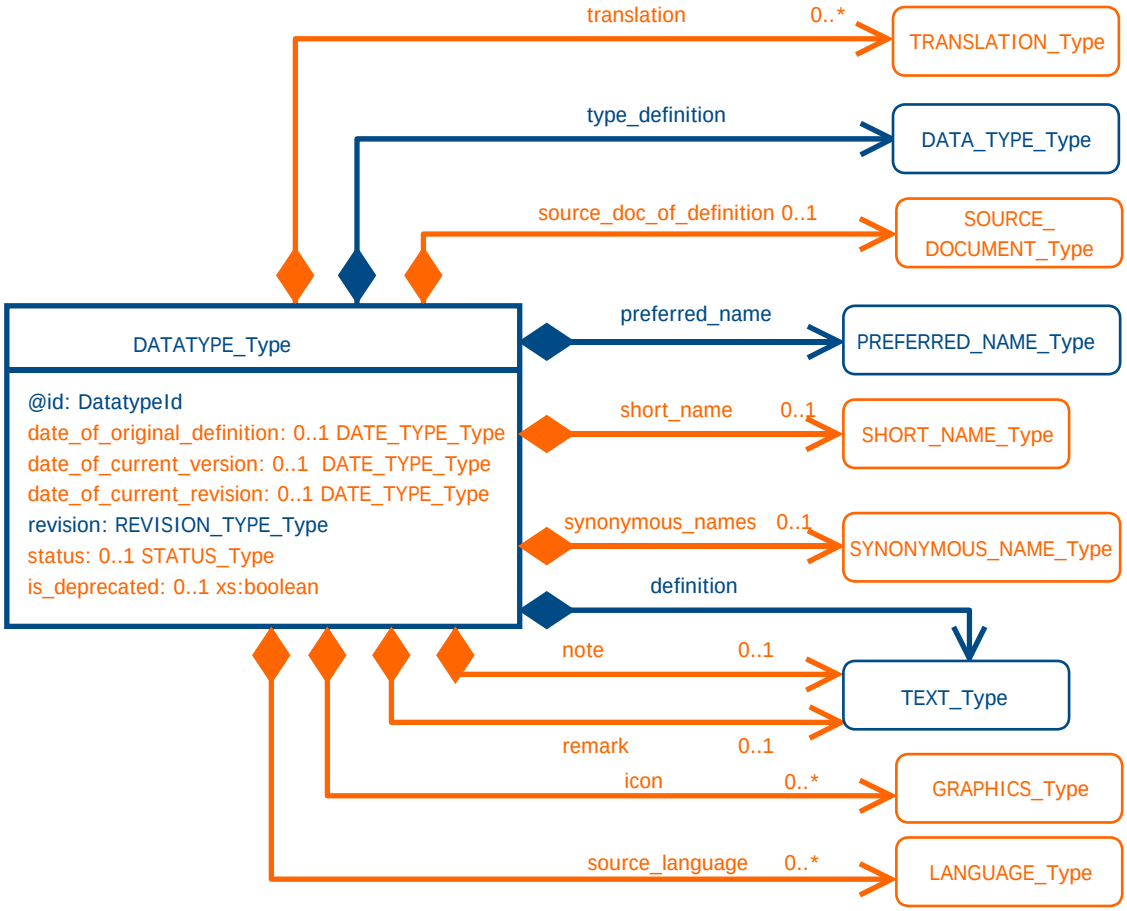

Figure 33 – Named data type UML diagram

Internal item definitions:
@id: the datatype identifier.

date_of_original_definition: the date associated to the first stable version of the datatype definition.

date_of_current_version: the date associated to the present version of the datatype definition.

date_of_current_revision: the date associated to the present revision of the datatype definition.

revision: the revision number of the present datatype definition.

status: defines the life cycle state of the datatype definition.

NOTE 1 Allowed **status** values are defined by private agreement between the dictionary supplier and dictionary users.

NOTE 2 If the **status** XML element is not provided, and if this datatype definition is not deprecated as denoted by a possible **is_deprecated** XML element, then the datatype definition has the same standardization status as the whole ontology into which it is used. In particular, if the ontology is standardized, this datatype definition is part of the current edition of the standard.

source_language: the language in which the datatype definition was initially defined and that provides the reference meaning in case of translation discrepancy.

is_deprecated: a Boolean that specifies, when true, that the datatype definition shall no longer be used.

preferred_name: the name of the datatype that is preferred for use, possibly translated.

short_name: the abbreviation of the preferred name, possibly translated.

synonymous_names: the set of synonymous names of the preferred name, possibly translated.

icon: a graphics representing the description associated with the names.

definition: the text describing this datatype, possibly translated.

source_doc_of_definition: the possible source document from which the definition comes.

note: further information on any part of the datatype, which is essential to its understanding, possibly translated.

remark: explanatory text further clarifying the meaning of this datatype, possibly translated.

translation: the possible set of translations information provided for the translatable items.

type_definition: the description of the type carried by the data types.

Internal type definitions:
DATE_TYPE_Type: identifies the values allowed for a date (the specific **xs:date** XML Schema datatype).

REVISION_TYPE_Type: a string (**xs:string** XML Schema datatype) that represents the values allowed for a revision. Its value length shall not exceed 3 characters.

STATUS_Type: a string (**xs:string** XML Schema datatype) that represents the values allowed for a status.

External type definitions:
DatatypeId: see Clause 8.1.

DATA_TYPE_Type: See Clause 7.3.

GRAPHICS_Type: See Clause 7.2.3.2.

LANGUAGE_Type: see Clause 7.1.1.

PREFERRED_NAME_Type: See Clause 7.1.2.

SHORT_NAME_Type: See Clause 7.1.2.

SOURCE_DOCUMENT_Type: See Clause 7.2.3.1.

SYNONYMOUS_NAME_Type: See Clause 7.1.2.

TEXT_Type: See Clause 7.1.2.

TRANSLATION_Type : see Clause 7.1.3.

5.7.7 Document

In OntoML, a document may be associated with a global identifier and considered as an ontology concept. For that purpose, OntoML proposes a **DOCUMENT_Type** XML complex type as illustrated in Figure 34.

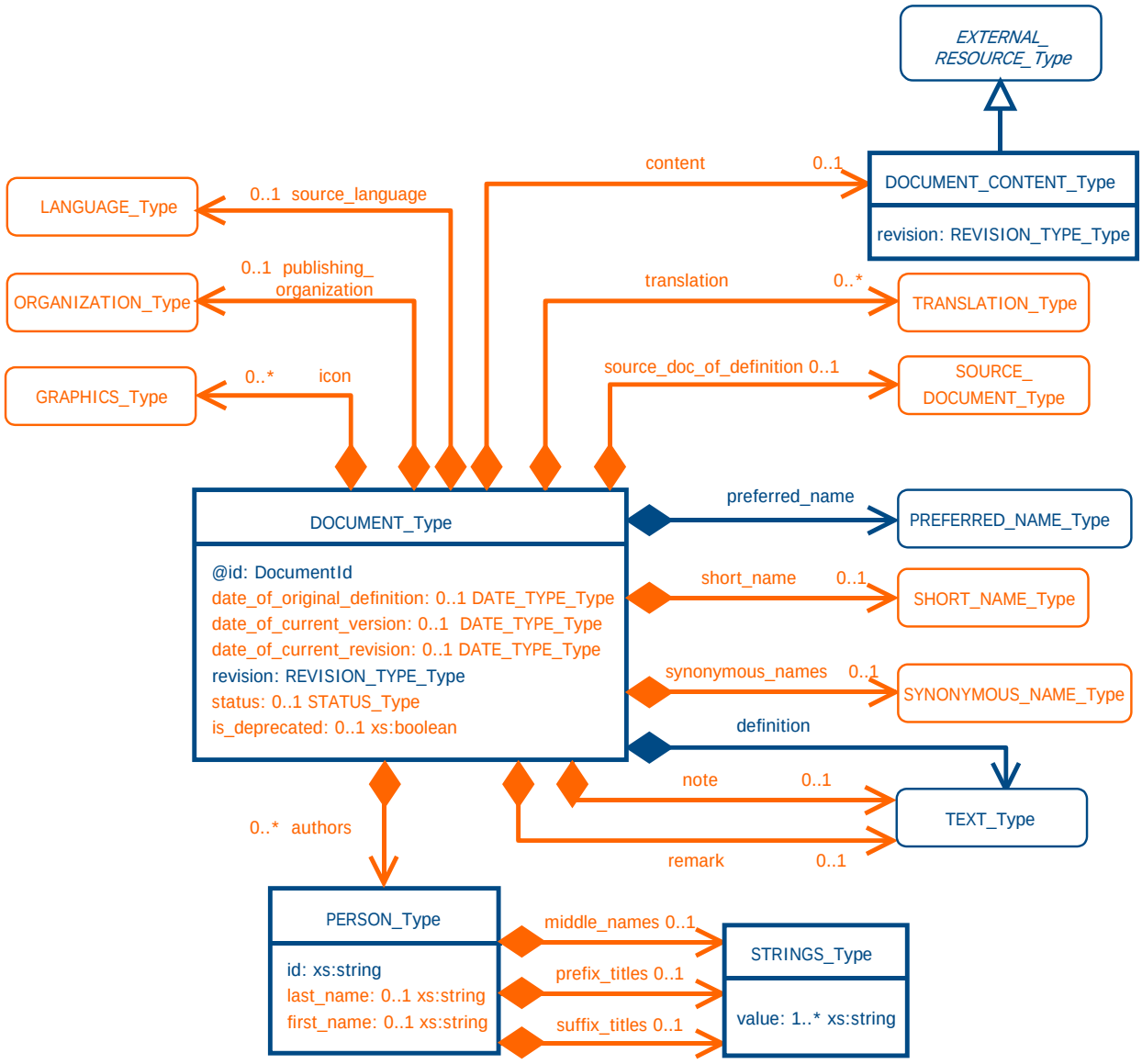

Figure 34 – Simple-level document UML diagram

Internal item definitions:
@**id**: the document identifier.

NOTE 1 A document is applicable in the class where it is defined visible, i.e., in the class that defines the context in which the document identification code is defined. Such a context is defined in Clause 8.1.3.

date_of_original_definition: the date associated to the first stable version of the document definition.

date_of_current_version: the date associated to the present version of the document definition.

date_of_current_revision: the date associated to the present revision of the document definition.

revision: the revision number of the present document definition.

status: defines the life cycle state of the document definition.

NOTE 2 Allowed **status** values are defined by private agreement between the dictionary supplier and dictionary users.

NOTE 3 If the **status** XML element is not provided, and if this document definition is not deprecated as denoted by a possible **is_deprecated** XML element, then the document definition has the same standardization status as the whole ontology into which it is used. In particular, if the ontology is standardized, this document definition is part of the current edition of the standard.

source_language: the language in which the document definition was initially defined and that provides the reference meaning in case of translation discrepancy.

is_deprecated: a Boolean that specifies, when true, that the document definition shall no longer be used.

preferred_name: the name of the document that is preferred for use, possibly translated.

short_name: the abbreviation of the preferred name, possibly translated.

synonymous_names: the set of synonymous names of the preferred name, possibly translated.

icon: a graphics representing the description associated with the names.

definition: the text describing this document, possibly translated.

source_doc_of_definition: the possible source document from which the definition comes.

note: further information on any part of the document, which is essential to its understanding, possibly translated.

remark: explanatory text further clarifying the meaning of this document, possibly translated.

translation: the possible set of translations information provided for the translatable items.

publishing_organization: the organisation that publishes the document.

authors: the author(s) of the document.

content: the physical document for which the **DOCUMENT_Type** XML complex type gives a description.

NOTE 4 The content of the document is represented by the **DOCUMENT_CONTENT_Type** XML complex type. It is defined as a subtype of the **EXTERNAL_RESOURCE_Type** XML complex type defined in Clause 7.2. Thus, the inherited **file** XML element allows to reference, using a URI, the target document.

NOTE 5 The physical document is optional, and may, or not, be delivered in the same OntoML document instance.

content/revision: characterization of the updating of the document.

authors/id: a means by which the person may be identified.

authors/last_name: the person's surname.

authors/first_name: the first element of the person's list of forenames.

authors/middle_names: the person's other forenames, if there are any.

authors/prefix_titles: the word, or group of words, which specify the person's social and/or professional standing and appear before his or her names.

authors/suffix_titles: the word, or group of words, which specify the person's social and/or professional standing and appear after his or her names.

authors/*/value: a string element of a string collection.

Internal type definitions:
DATE_TYPE_Type: identifies the values allowed for a date (the specific **xs:date** XML Schema datatype).

REVISION_TYPE_Type: a string (**xs:string** XML Schema datatype) that represents the values allowed for a revision. Its value length shall not exceed 3 characters.

STATUS_Type: a string (**xs:string** XML Schema datatype) that represents the values allowed for a status.

External type definitions:
DatatypeId: see Clause 8.1.

GRAPHICS_Type: See Clause 7.2.3.2.

LANGUAGE_Type: see Clause 7.1.1.

ORGANIZATION_Type: See Clause 7.8.1.

PREFERRED_NAME_Type: See Clause 7.1.2.

SHORT_NAME_Type: See Clause 7.1.2.

SOURCE_DOCUMENT_Type: See Clause 7.2.3.1.

SYNONYMOUS_NAME_Type: See Clause 7.1.2.

TEXT_Type: See Clause 7.1.2.

TRANSLATION_Type : see Clause 7.1.3.

6 Overview of OntoML libraries representation

The library content part of OntoML provides resources for representing instances belonging to the classes defined in a given domain ontology. Such instances may be product characterizations if they belong to a product characterization class, or product representations if they belong to functional model class.

NOTE 1 Product is understood in a very generic sense that covers all items that may be characterized by OntoML item classes.

NOTE 2 A library content may, or not, be associated with an ontology describes in an OntoML document instance.

NOTE 3 A library content provides in particular for exchanging electronic catalogues.

6.1 Root class of a library

In OntoML, every library pieces of information are gathered into a general structure that is a **LIBRARY_TYPE** XML complex type. It is illustrated in Figure 35.

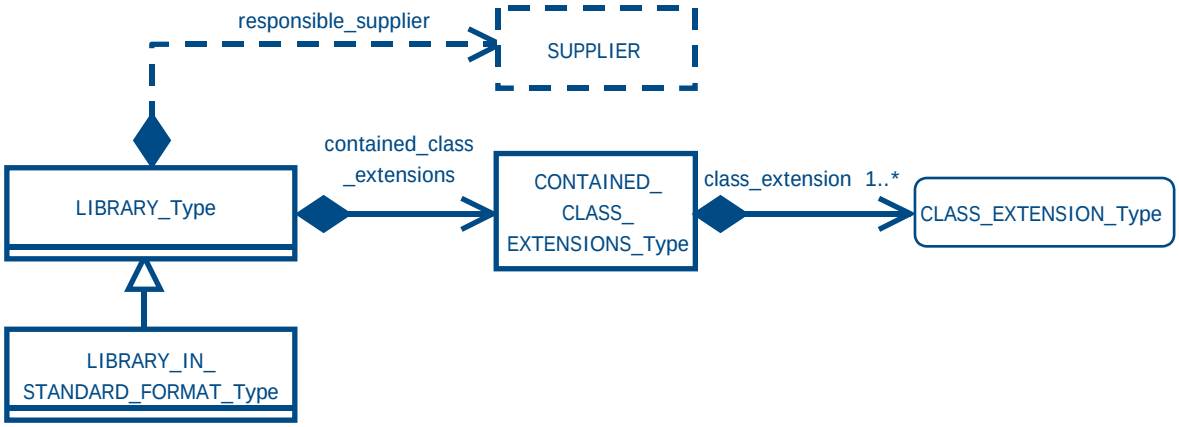

Figure 35 – Root class of library

Internal item definitions:
contained_class_extensions: the set of class extensions of the set of ontology classes.

contained_class_extensions/class_extension: a class extension contained in the dictionary

responsible_supplier: the data supplier responsible for the library content.

Internal type definitions:
CONTAINED_CLASS_EXTENSIONS_Type: sequence of class extensions descriptions.

LIBRARY_IN_STANDARD_FORMAT_Type: a library that only uses external file protocols that are allowed either by the library integrated information model indicated by the **library_structure** XML element or the view exchange protocols referenced in the **supported_vep** XML element, both defined in the **HEADER_Type** XML complex type.

External type definitions:
CLASS_EXTENSION_Type: see Clause 6.2.

GRAPHICS_Type: See Clause 7.2.3.2

PREFERRED_NAME_Type: See Clause 7.1.2.

PROPERTY_Type: dictionary property description. See Clause 5.7.4.

SHORT_NAME_Type: See Clause 7.1.2.

SUPPLIER_Type: supplier description. See Clause 5.7.1.

SYNONYMOUS_NAME_Type: See Clause 7.1.2.

TEXT_Type: See Clause 7.1.2.

6.2 Class extension general structure

A class extension is represented by the **CLASS_EXTENSION_Type** abstract XML complex type, as illustrated in Figure 36.

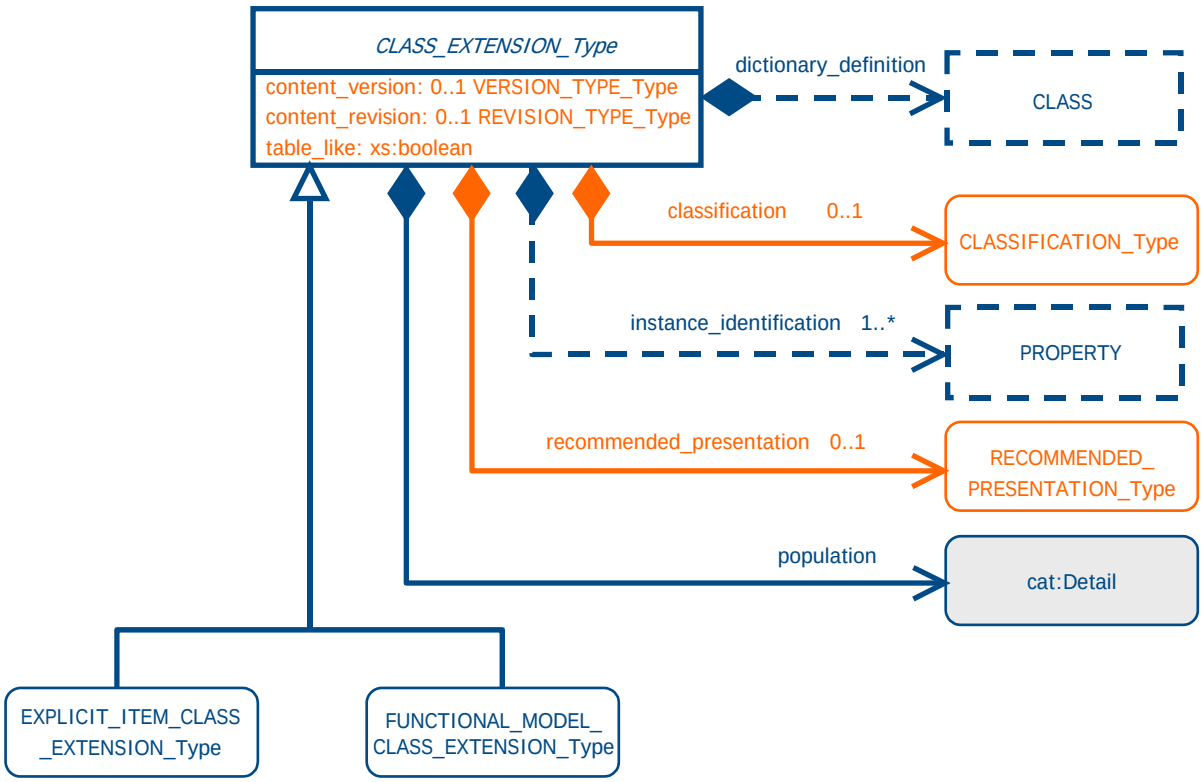

Figure 36 – General class extension structure

Internal item definitions:
content_version: the version number that characterizes the extension of a class, i.e., the set of all allowed instances.

content_revision: the revision number that corresponds to the current description of the **content_version** version of a class extension.

table_like: a Boolean value that specifies whether all the class instances are characterized by the same properties in the same order, or not.

dictionary_definition: the reference to the class extension dictionary definition.

classification: the possible reference to a classification of the properties used for describing class instances.

instance_identification: the references to the properties that allow to identify unambiguously each instance belonging to a class.

recommended_presentation: the recommended scaling factor, presentation units and value formats to be used when displaying the values of some referenced properties in the context of the referencing class.

population: the list of instances that describe the class population.

Internal type definitions:
CLASS_EXTENSION_Type: the abstract XML complex type, supertype of the various class extensions.

REVISION_TYPE_Type: a string (**xs:string** XML Schema datatype) that represents the values allowed for a revision. Its value length shall not exceed 3 characters.

VERSION_TYPE_Type: a string (**xs:string** XML Schema datatype) that represents the values allowed for a version. It shall contain only digits, and its value length shall not exceed 9 characters.

External type definitions:
CLASSIFICATION_Type: see Clause 6.2.1.

EXPLICIT_ITEM_CLASS_EXTENSION_Type: see Clause 6.3.

EXPLICIT_FUNCTIONAL_MODEL_CLASS_EXTENSION_Type: see Clause 6.4.

RECOMMENDED_PRESENTATION_Type: see Clause 6.2.2.

cat:Detail: specification of instances as a set of property reference and value couples.
NOTE cat:Detail is defined in the common product exchange format intended to be jointly defined by ISO 13584 and ISO 22745 as ISO 29002-10.

6.2.1 Property classification

Some properties participating to instance descriptions may be grouped using some classification resources. Each group is identified by an integer. The intent is to allow to process differently (on the receiving system) properties belonging to a given group.

NOTE 1 A property may belong to different groups.

NOTE 2 A property which is not associated with a classification value is not associated with any particular processing.

NOTE 3 This part of ISO 13584 does not make any assumption on how each classification value should be interpreted on a receiving system. It may result from private agreement between the sender and the receiver or from latter standardization.

Properties classification is represented using the **CLASSIFICATION_Type** XML complex type as illustrated in Figure 37.

Figure 37 – Properties classification

Internal item definitions:
property_classification: the specification of the properties classifications.

property_classification/its_value: the classification value associated with the referenced **prop_def** property.

property_classification/prop_def: the reference to the property that is classified by means of the **its_value** value.

Internal type definitions:
PROPERTY_CLASSIFICATION_Type: an association that assigns a classification group to a property.

6.2.2 Properties presentation

When a particular property is used for describing instances of a particular class, it might prove useful to use a particular scaling factor, display unit and / or value format. Such a kind of recommendation is represented using **RECOMMENDED_PRESENTATION_Type** XML complex type as illustrated in Figure 38.

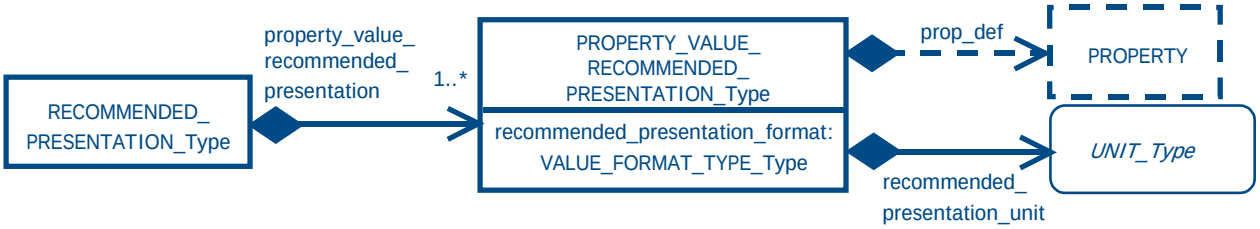

Figure 38 – Properties presentation

Internal item definitions:
property_value_recommended_representation: the specification of the properties recommended representations.

property_value_recommended_representation/prop_def: the reference to the property whose the library data supplier recommends to convert value for presentation purpose.

property_value_recommended_representation/recommended_presentation_format: the presentation format recommended by the library data supplier for presenting the values of the referenced **prop_def** property, if and only if these values are converted into the **recommended_presentation_unit** unit.

property_value_recommended_representation/recommended_presentation_unit: the particular unit in which the library data supplier recommends to convert data for presentation purpose.

Internal type definitions:
PROPERTY_VALUE_RECOMMENDED_PRESENTATION_Type: the recommended property presentation format together with its presentation unit.

External type definitions:
UNIT_Type: specification of a unit. See Clause 7.4.

6.3 Simple-level library: content of classes of products

The description of products belonging to a given product characterization class is performed using the **EXPLICIT_ITEM_CLASS_EXTENSION_Type** XML complex type as illustrated in Figure 39. It provides also optional information that may be used on a receiving system to display the content of a product characterization class.

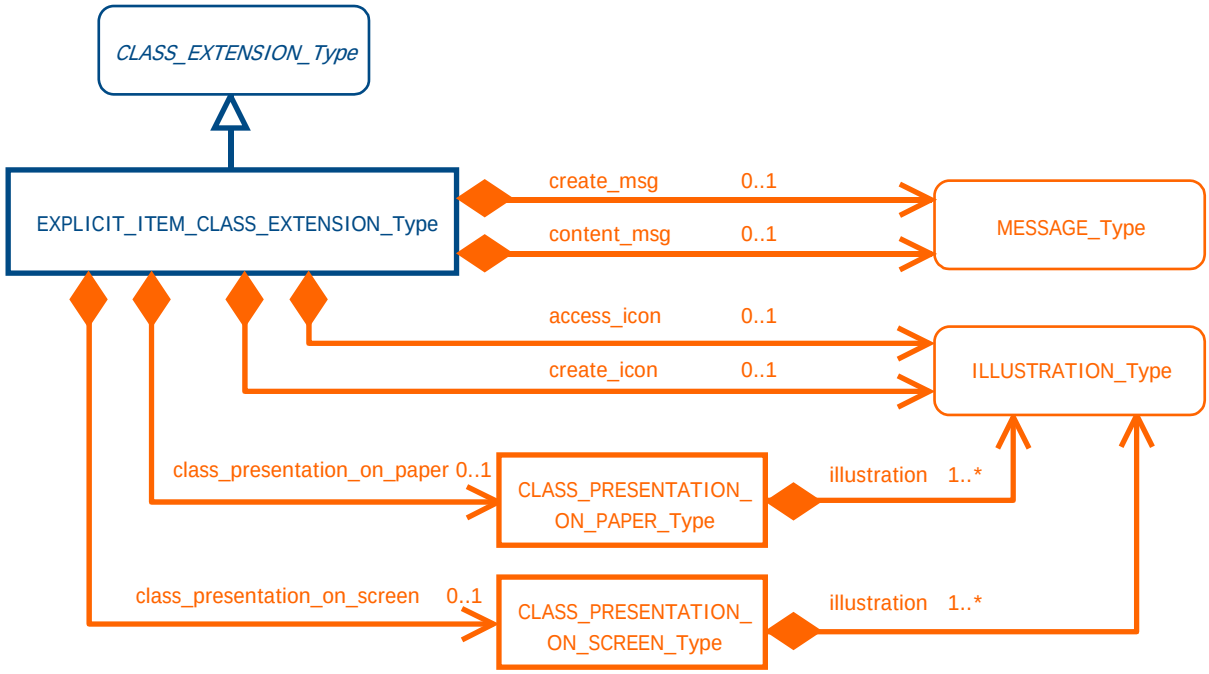

Figure 39 – Products representation structure

Internal item definitions:
create_msg: the optional message that describes the properties of a class items and their reference coordinate system.

content_msg: the message that describes the content of the class.

access_icon: the optional icon that enables class presentation in a menu.

NOTE 1 The Icon shall be defined as A9 standardized size icon.

create_icon: the optional icon(s) that enable(s) visual presentation of the properties of class items and reference coordinate system.

NOTE 2 Icons shall be defined as A6 standardized size icons.

class_presentation_on_paper: the ordered set of illustrations that are recommended by the library data supplier to be presented to the user when the content of the class is presented on paper, for instance for printing the class content in a booklet.

class_presentation_on_paper/illustration: the class illustrations to be presented on paper, for instance for printing the class content in a booklet.

class_presentation_on_screen: the ordered set of illustrations that are recommended by the library data supplier to be presented to the user when the content of the class is presented on screen.

class_presentation_on_screen/illustration: the class illustrations to be displayed on screen.

Internal type definitions:
CLASS_PRESENTATION_ON_PAPER_Type: specifies the structure of a paper illustration.

CLASS_PRESENTATION_ON_SCREEN_Type: specifies the structure of a screen illustration.

6.4 Advanced-level library: content of classes of product representations

The representation of engineering models as an extension of a functional model class requires:

— to define what are the product properties that are required to process the representation,

— to specify which property is intended to contain, as a value, the product representation, if and only if the functional view referenced by the functional model data dictionary definition is instanciable.

Each instance of a functional model class is a product representation that consists of a list of property-value pairs. The subset of these properties that is included in the **instance_identification** inherited XML element of the functional model class constitutes the key of these instances, this means the necessary information for identifying one instance, and discriminating it from other instances of the same class. This set of instances of the class is recorded in the **population** inherited XML element.

When the **instance_identification** XML element of the functional model class are all imported from the product characterization class, this means that once the product is selected within its product characterization class, the corresponding instance defined by the functional model class is already identified, and thus it may be computed automatically by a system.

NOTE 1 Properties are imported by a functional model class by means of the **imported_properties_from_item** XML element of the **FM_CLASS_VIEW_OF_Type** XML complex type corresponding to this functional model.

EXAMPLE 1 Assume that a functional model class view-of is defined for each item class of a library. Assume that this functional model class defines the inventory status of each product of this class. The functional model class contains three properties: *the part number* (imported from the item class), *stock availability*, and *quantity of order*. The **instance_identification** XML element contains only the *part_number* property. Thus, when a product is selected, its inventory status may be automatically computed. Moreover, the inventory status of each product may be computed also automatically by a system just by a natural joint between each explicit item class extension and the functional model class extension that is view-of this item class.

When the **instance_identification** XML element contains properties that do not belong to the **imported_property_from_item** XML element, then these properties must be assigned a value by a user to select the right product representation corresponding to each product.

EXAMPLE 2 When a functional model class provides representations corresponding to several view control variables values, for each product, a particular value needs to be selected by the user for each view control variable to select the right corresponding product representation.

Functional model class extensions are represented using the **FUNCTIONAL_MODEL_CLASS_EXTENSION_Type** XML complex type as illustrated in Figure 40.

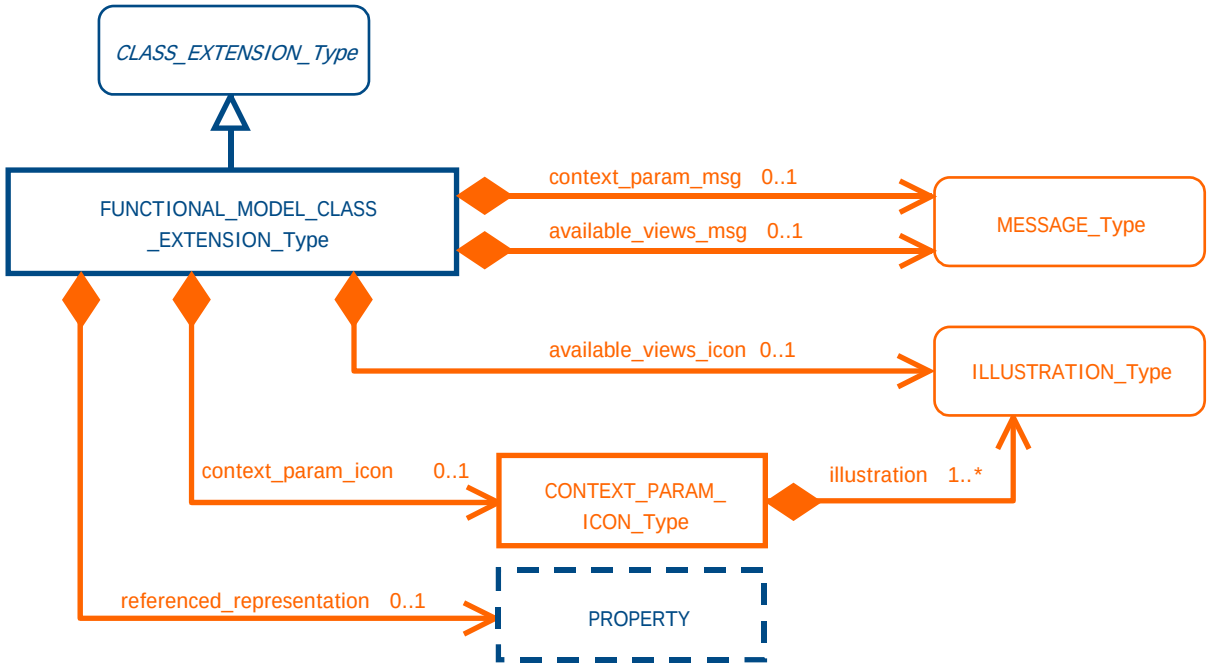

Figure 40 – Engineering models structure UML diagram

Internal item definitions:
context_param_msg: the message that explains the model properties that shall be assigned a value for selecting a particular view.

NOTE 2 These properties are the ones that belong to the instance identification inherited attribute.

available_views_msg: message that describes the various views that may be created by the functional model class

available_views_icon: the icon that enable visual presentation of the various views that may be created by the functional model class.

NOTE 3 The Icon shall be defined as a A6 standardized size icon.

context_param_icon: the icon(s) that enable(s) visual presentation of the properties that shall be assigned a value for selecting a particular view.

NOTE 4 Icons shall be defined as A6 standardized size icons.

referenced_representation: if and only if the functional model class refers to an (instanciable) functional view class, the property that carries the representation generation mechanism in the functional model class description.

NOTE 5 This mechanism may be either a program to be triggered or an EXPRESS representation. See ISO 13584 for details.

Internal type definitions:
CONTEXT_PARAM_ICON_Type: specifies the structure of context parameter icon.

7 Other structured information elements

This clause specifies the other structured information element constructs that were referenced in clause 6.

7.1 Translations

OntoML provides resources for translating clear text information and for managing translations.

7.1.1 Language specification

The specification of a language is twofold: the language specification, and possibly the associted country that further specify the language. It is illustrated in Figure 41.

Figure 41 – Language specification

Internal item definitions:
@**language**: the code of the language.

@**country**: the possible country code that further specifies the language code.

Internal type definitions:
COUNTRY_CODE_Type: the type of a language code. It is a string which contains 2 characters, and that defines a country according to ISO 3166.

LANGUAGE_CODE_Type: the type of a language code. It is a string which contains either 2 or 3 characters, and that defines a language according to ISO 639-1 or ISO 639-2.

7.1.2 Translation of string valued elements

Every ontology concept is described using clear text information that may or not be translated. The corresponding information elements are :

— **preferred_names**;

— **short_names**;

— **synonymous_names**;

— **definitions, notes and remarks**.

OntoML provides constructs for representing these information elements as illustrated in Figure 42.

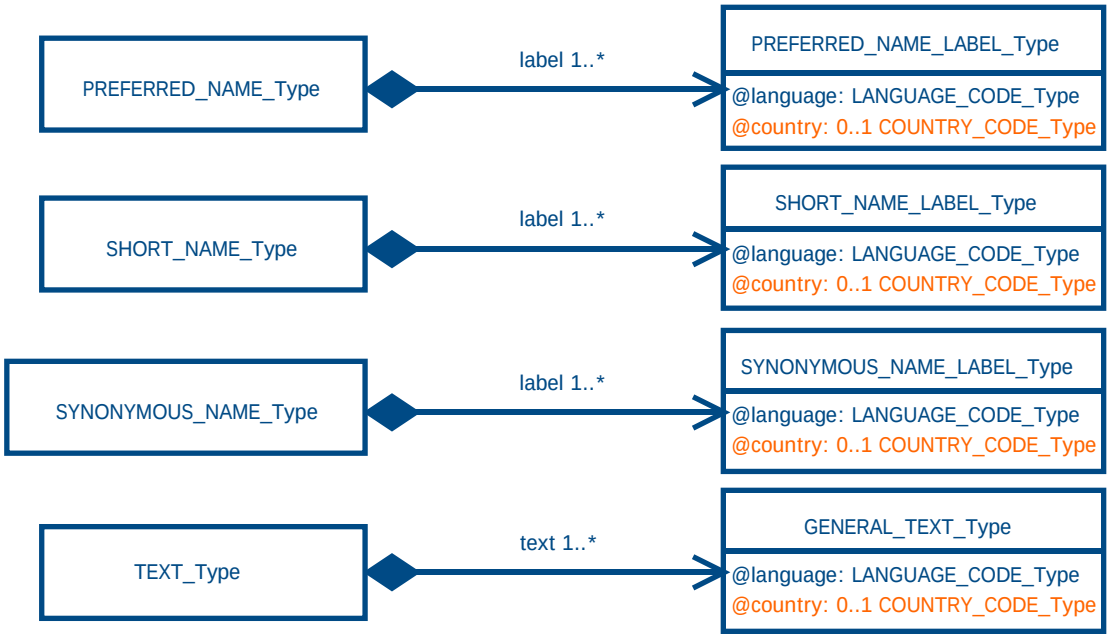

Figure 42 –Translation resources

Internal item definitions:
label (**PREFERRED_NAME_Type**): defines the preferred names labels that are translated, together with their corresponding translation.

label (**SHORT_NAME_Type**): defines the short names labels that are possibly translated, together with their corresponding translation.

label (**SYNONYMOUS_NAME_Type**): defines the synonymous names labels that are possibly translated, together with their corresponding translation.

text (**TEXT_Type**): defines the definitions, notes and remarks texts that are possibly translated, together with their corresponding translation.

Internal type definitions:
COUNTRY_CODE_Type: the type of a language code. It is a string which contains 2 characters, and that defines a country according to ISO 3166.

GENERAL_TEXT_Type: a string (**xs:string** XML Schema datatype) that represents the values allowed for a definition, a note or a remark together with its language (**@language** and possible **@country** XML attributes). Its value length is not constrained.

LANGUAGE_CODE_Type: the type of a language code. It is a string which contains either 2 or 3 characters, and that defines a language according to ISO 639-1 or ISO 639-2.

PREFERRED_NAME_LABEL_Type: a string (**xs:string** XML Schema datatype) that represents the values allowed for a preferred name with its language (**@language** and possible **@country** XML attributes). Its value length shall not exceed 70 characters.

SHORT_NAME_LABEL_Type: a string (**xs:string** XML Schema datatype) that represents the values allowed for a short name with its language (**@language** and possible **@country** XML attributes). Its value length shall not exceed 30 characters.

SYNONYMOUS_NAME_LABEL_Type: a string (**xs:string** XML Schema datatype) that represents the values allowed for a synonymous name with its language (**@language** and possible **@country** XML attributes). Its value length shall not exceed 70 characters.

7.1.3 Translation management

Management information about the translation performed at the level of an ontology concept may be represented. For that purpose, OntoML provides the **TRANSLATION_Type** XML complex type. It is illustrated in Figure 43.

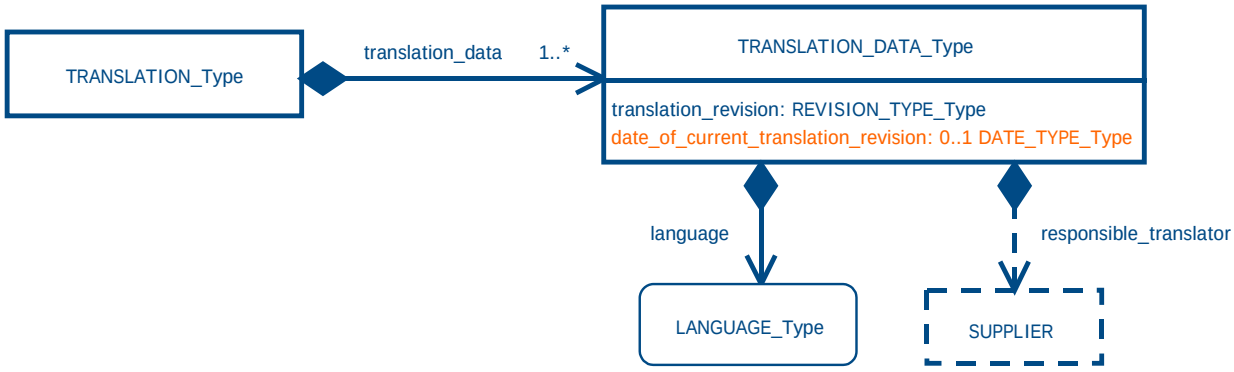

Figure 43 – Translation data structure

Internal item definitions:
translation_data: the set of translation information associated to an ontology concept.

language (**TRANSLATION_DATA_Type**): the language for which the translation information is given.

translation_revision (**TRANSLATION_DATA_Type**): the revision status of the translation in the **@language** language.

date_of_current_translation_revision (**TRANSLATION_DATA_Type**): the date corresponding to the current revision of the translation in the **@language** language.

responsible_translator: a reference to the identifier of the organization who performed the translation in the **@language** language.

Internal type definitions:
REVISION_TYPE_Type: a string (**xs:string** XML Schema datatype) that represents the values allowed for a revision. Its value length shall not exceed 3 characters.

DATE_TYPE_Type: identifies the values allowed for a date (the specific **xs:date** XML Schema datatype).

External type definitions:
LANGUAGE_Type: see Clause 7.1.1.

7.2 External content

External contents allow to reference externally defined information from ontology concepts or from information elements. This reference may be performed in different ways:

— by specifying a URI (local or global) that references an external resource;;

— by specifying a code identifying a document;

— by specifying a reference to a document or a documented graphics that is described and identified by a document ontology concept.

7.2.1 Simple-level ontology external resources

A piece of information may be provided as an external resource. This resource may be

— either a file (or a set of files) associated with the OntoML document instance and identified by a local URI,

— or an Internet resource, identified by a global URI.

Figure 44 illustrates the external resource root type, represented by the **EXTERNAL_RESOURCE_Type** abstract XML complex type.

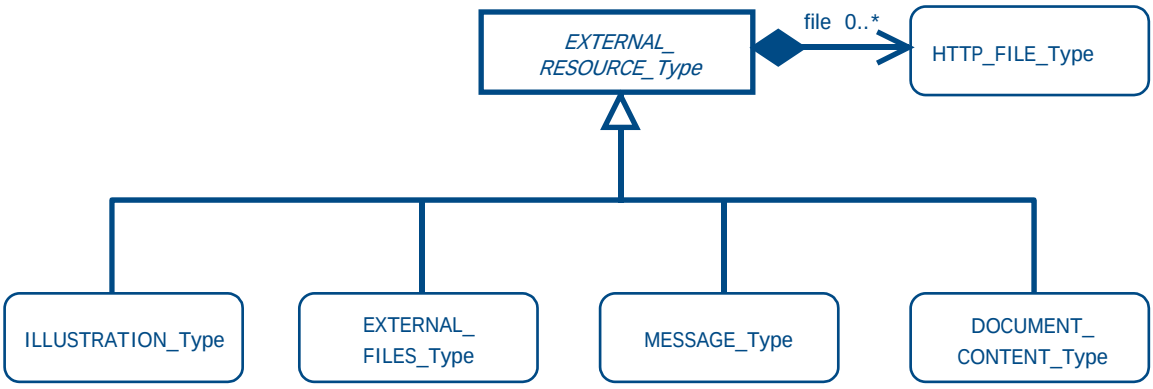

Figure 44 – Simple-level ontology external resources

Internal item definition:
file: a set of XML elements that describe and identify external resources represented by HTTP files.

External type definitions:
DOCUMENT_CONTENT_Type: see Clause 5.7.7.

EXTERNAL_FILES_Type: see Clause 7.2.1.4.

ILLUSTRATION_Type: see Clause 7.2.1.2.

HTTP_FILE_Type: see Clause 7.2.1.1.

MESSAGE_Type: see Clause 7.2.1.3.

7.2.1.1 HTTP file

An HTTP file is the basic OntoML construct for referencing external information from an OntoML document instance.

An HTTP file is defined according to the **HTTP_FILE_Type** XML complex Type as presented in Figure 45.

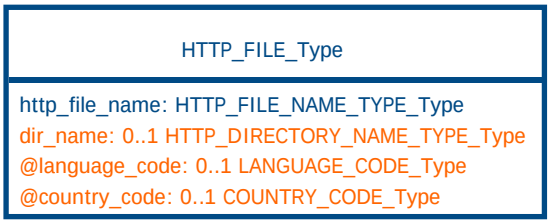

Figure 45 – Simple-level ontology external resources: HTTP file structure

Internal item definitions:
http_file_name: the name of the HTTP file. Depending on the fact that the file is exchanged together with the OntoML file or globally referenced, it is represented respectively as a local URI or as a global URI.

NOTE 1 The MIME protocol interpretation completely defines the protocol to be used for processing any referenced external information.

dir_name: the possible target directory in which the HTTP file shall be stored on a receiving system if the http files include reference to each others.

NOTE 2 When dir_name exists, all directories of a same OntoML document instance shall be defined under a common root.

@language_code: the possible language in which the information contained in the HTTP file is expressed.

@country_code: the possible country code that further specifies the language.

Internal type definitions:
COUNTRY_CODE_Type: the type of a language code. It is a string which contains 2 characters, and that defines a country according to ISO 3166.

HTTP_FILE_NAME_Type: the type of an **http_file_name**. Its representation fulfills constraints defined for representing URIs.

NOTE 3 URI is defined by [RFC 2396], and is amended by [RFC 2732].

HTTP_DIRECTORY_NAME_Type: the type of the name of an http file directory (**dir_name**). The length of the directory name shall not be greater than 128.

LANGUAGE_CODE_Type: the type of a language code. It is a string which contains either 2 or 3 characters, and that defines a language according to ISO 639-1 or ISO 639-2.

NOTE 4 Allowed language codes are defined in ISO 639-1 and in ISO 639-2.

7.2.1.2 Illustration

An illustration is an informative item that may be a schematic drawing, a realistic picture, or any other informative element that is not a static picture, whose content is defined by an http file possibly translated. An illustration may also be specified as being an *A6* or an *A9* standard format illustration. If such a standard size is not provided, the window size recommended to display the illustration may be specified. An illustration is represented by an **ILLUSTRATION_Type** XML complex type (see Figure 46).

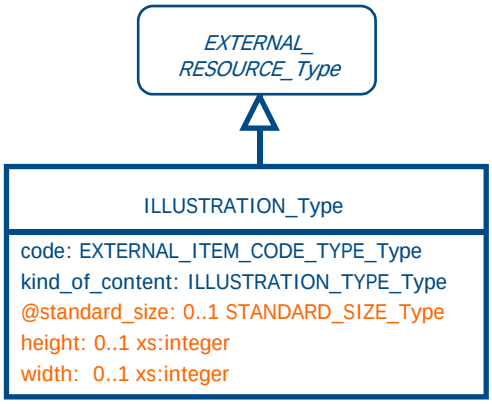

Figure 46 – Simple-level ontology external resources: illustration

Internal item definitions:
code: a code that identifies the illustration.

NOTE 1 The **code** must be unique in the class where it is used to define the illustration.

kind_of_content: the categorization of the illustration content, the value must be schematic drawing, realistic picture or not static picture.

@standard_size: if provided, specifies that the illustration is either an *A6* or an *A9* standard format illustration.

height: in case of a non standardized format illustration, specifies the height of the window recommended by the library data supplier for viewing the illustration.

width: in case of a non standardized format illustration, specifies the width of the window recommended by the library data supplier for viewing the illustration.

NOTE 2 Either both **height** and **width** are defined or none is defined.

NOTE 3 The default **height** and **width** unit is millimeter.

Internal type definitions:
EXTERNAL_ITEM_CODE_TYPE_Type: an XML simple type specifying a string restriction : the string shall be less or equal to 18 and shall not contain any space.

ILLUSTRATION_TYPE_Type: an XML simple type specifying a string restriction : the string shall be equal either to « SCHEMATIC_DRAWING » or « REALISTIC_PICTURE » or « NOT_STATIC_PICTURE ».

STANDARD_SIZE_Type: an XML simple type specifying a string restriction : the string shall be equal either to « a6_illustration » or « a9_illustration ».

7.2.1.3 Message

A message is a text that shall be short, typically up to 256 characters, and that is supposed to be automatically displayed on the screen of a user library user within a clearly defined work context. A message is not associated with any particular window dimension. It may be provided in different languages, and it is stored in an http file (one file for each language). A message is represented by a **MESSAGE_Type** XML complex type (see Figure 47).

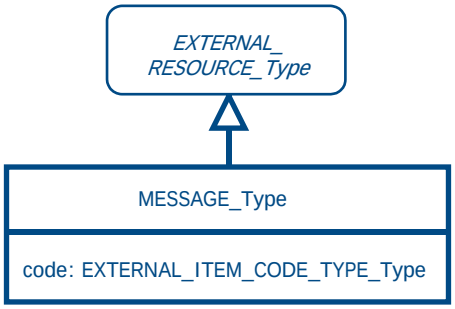

Figure 47 – Simple-level ontology external resources: message

Internal item definitions:
code: a code that identifies the message.

NOTE The **code** must be unique in the class where the message is defined.

Internal type definitions:
EXTERNAL_ITEM_CODE_TYPE_Type: an XML simple type specifying a string restriction : the string shall be less or equal to 18 and shall not contain any space.

7.2.1.4 External files

An external file provides for exchanging graphical data by means of reference to external resources. It is represented by an **EXTERNAL_FILES_Type** XML complex type (see Figure 48)

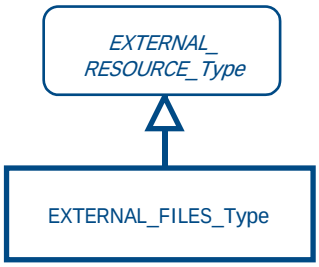

Figure 48 – Simple-level ontology external resources: external files

7.2.2 Advanced-level ontology external resources

For one category of representation, the set of the representations of all the products of an item class may be modeled either explicitly or implicitly. If it is modeled explicitly (**REPRESENTATION_REFERENCE_Type** XML complex type), each item representation is described as a set of data. If it is modeled implicitly (**PROGRAM_REFERENCE_Type** XML complex type), there is an algorithm that shall be triggered and provided with parameter values to generate each item representation. It is illustrated in Figure 49.

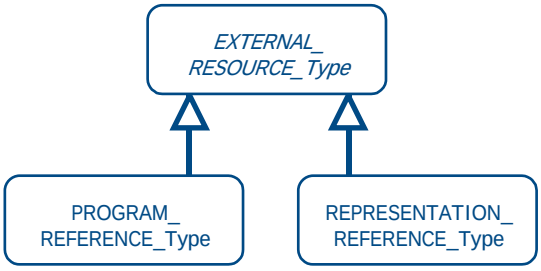

Figure 49 – Advanced-level ontology external resources

External type definitions:
PROGRAM_REFERENCE_Type: see Clause 7.2.2.1.

REPRESENTATION_REFERENCE_Type: see Clause 7.2.2.2.

7.2.2.1 Program reference

Functional model class associated with instanciable functional view class must generate representations of library products.

EXAMPLE A 3D geometric representation category may be generated from a functional model class.

If the set of representations of all the instances of a given product characterization class is modeled implicitly by an algorithm, it should be possible to exchange a program stored in an http file, that shall be triggered and provided with parameter values to generate each item representation. Such a program is specified by a program reference. A program reference is represented by an **PROGRAM_REFERENCE_Type** XML complex type (see Figure 50).

NOTE 1 A program reference may only be assigned to the property referenced by the **referenced_representation** XML element of a **FUNCTIONAL_MODEL_CLASS_EXTENSION_Type** XML complex type.

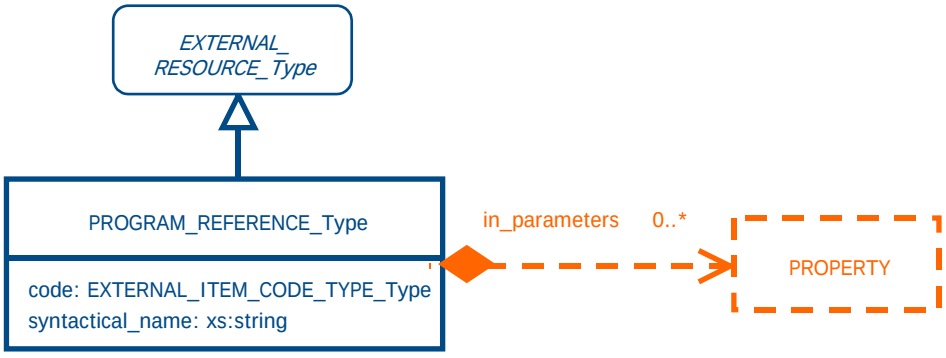

Figure 50 – Advanced-level ontology external resources: program reference

Internal item definitions:
code: a code that identifies the program.

NOTE 2 The **code** must be unique in the class where it is used to define the message.

syntactical_name: the name by which the program shall be triggered.

in_parameters: the list of references to properties or data types that specifies the types of the input parameters of the referenced program.

NOTE 3 The **in_paramters** properties must belong to the property that are applicable to the functional model class.

7.2.2.2 Representation reference

When representation(s) intended to be assigned to product characterization classes are modeled explicitly, each item representation is described as a set of data stored in an http file: it is specified by a representation reference. A representation reference is represented by an **REPRESENTATION_REFERENCE_Type** XML complex type (see Figure 51).

NOTE 1 A representation reference may only be assigned to the property referenced by the **referenced_representation** XML element of a **FUNCTIONAL_MODEL_CLASS_EXTENSION_Type** XML complex type.

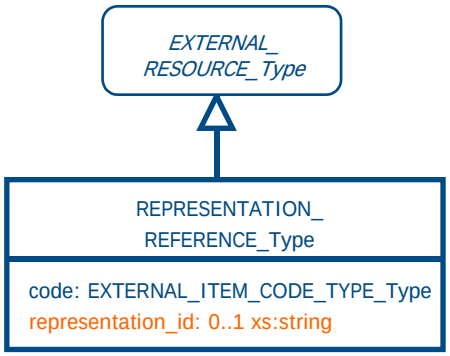

Figure 51 – Advanced-level ontology external resources: representation reference

Internal item definitions:
code: a code that identifies the representation.

NOTE The **code** must be unique in the class where it is used to define the message.

representation_id: the possible label that corresponds to the referenced representation.

Internal type definitions:
EXTERNAL_ITEM_CODE_TYPE_Type: an XML simple type specifying a string restriction : the string shall be less or equal to 18 and shall not contain any space.

7.2.3 Source document and graphics

Documentation that may be associated to an ontology concept is twofold:

— source documents: general construct for referencing document-like documentation;

— graphics: general construct for referencing graphics-like documentation.

7.2.3.1 Source document

External resources that represent a source document are represented by a **SOURCE_DOCUMENT_Type** XML abstract complex type. It is illustrated in Figure 52.

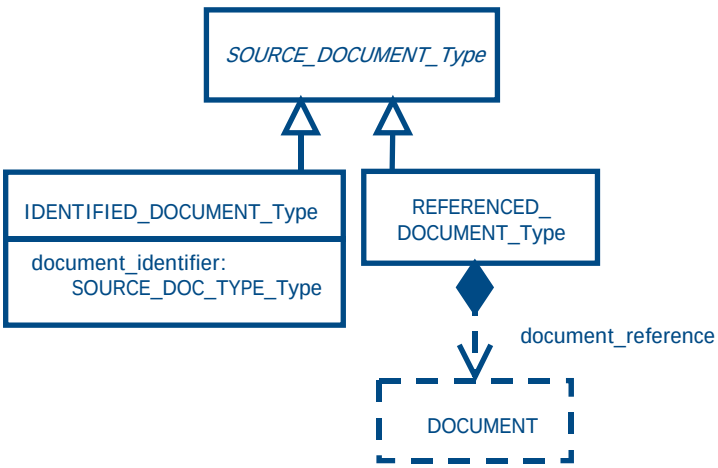

Figure 52 – External resources: source document

External type definitions:
IDENTIFIED_DOCUMENT_Type: see Clause 7.2.3.1.1.

REFERENCED_DOCUMENT_Type: see Clause 7.2.3.1.2.

7.2.3.1.1 Identified document

An identified document describes a document identified by a code. It is represented by an **IDENTIFIED_DOCUMENT_Type** XML complex type (see Figure 52). The inherited **file** XML element shall not be used.

Internal item definitions:
document_identifier: the code of the described document.

Internal type definitions:
SOURCE_DOC_TYPE_Type: identifies the values allowed for a document code. The code length shall be less or equal to 80 characters.

7.2.3.1.2 Referenced document

A referenced document enables to reference a document that is described and identified by a document ontology concept. It is represented by a **REFERENCED_DOCUMENT_Type** XML complex type (see Figure 52).

Internal item definitions:
document_reference: a reference to a document ontology concept identifier.

7.2.3.2 Graphics

External resources that represent a graphics are represented by a **GRAPHICS_Type** XML abstract complex type. It is illustrated in Figure 53.

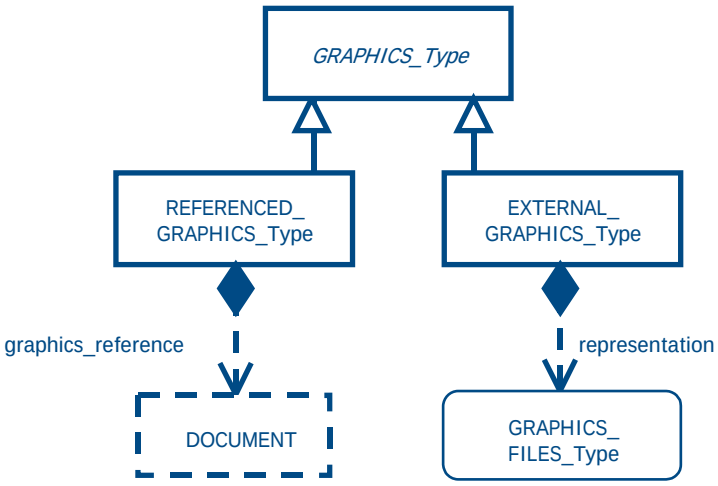

Figure 53 – External resources: graphics

External type definitions:
REFERENCED_GRAPHICS_Type: see Clause 7.2.3.2.1.

EXTERNAL_GRAPHICS_Type: see Clause 7.2.3.2.2.

7.2.3.2.1 Documented graphics

A documented graphics enables to identify and to document a graphics as a described and identified by a document ontology concept. It is represented by a **REFERENCED_GRAPHICS_Type** XML complex type (see Figure 53).

Internal item definitions:
graphics_reference: a reference to a document ontology concept identifier.

7.2.3.2.2 External graphics

An external graphics provides for exchanging graphical data by means of external files. It is represented by an **EXTERNAL_GRAPHICS_Type** XML complex type (see Figure 53). The inherited **file** XML element shall be provided.

Internal item definitions:
representation: the external file(s) specifying the external graphics.

7.3 Data type system

OntoML provides resources for describing data types that allow to constrain domain of values assigned to properties. Simple (Boolean, real, string, …) to complex data type (named type, aggregates, classes, …) are available. Every datatype is defined as a subtype of the **DATA_TYPE_Type** XML complex type. Figure 54 gives an overview of the main data types available in OntoML.

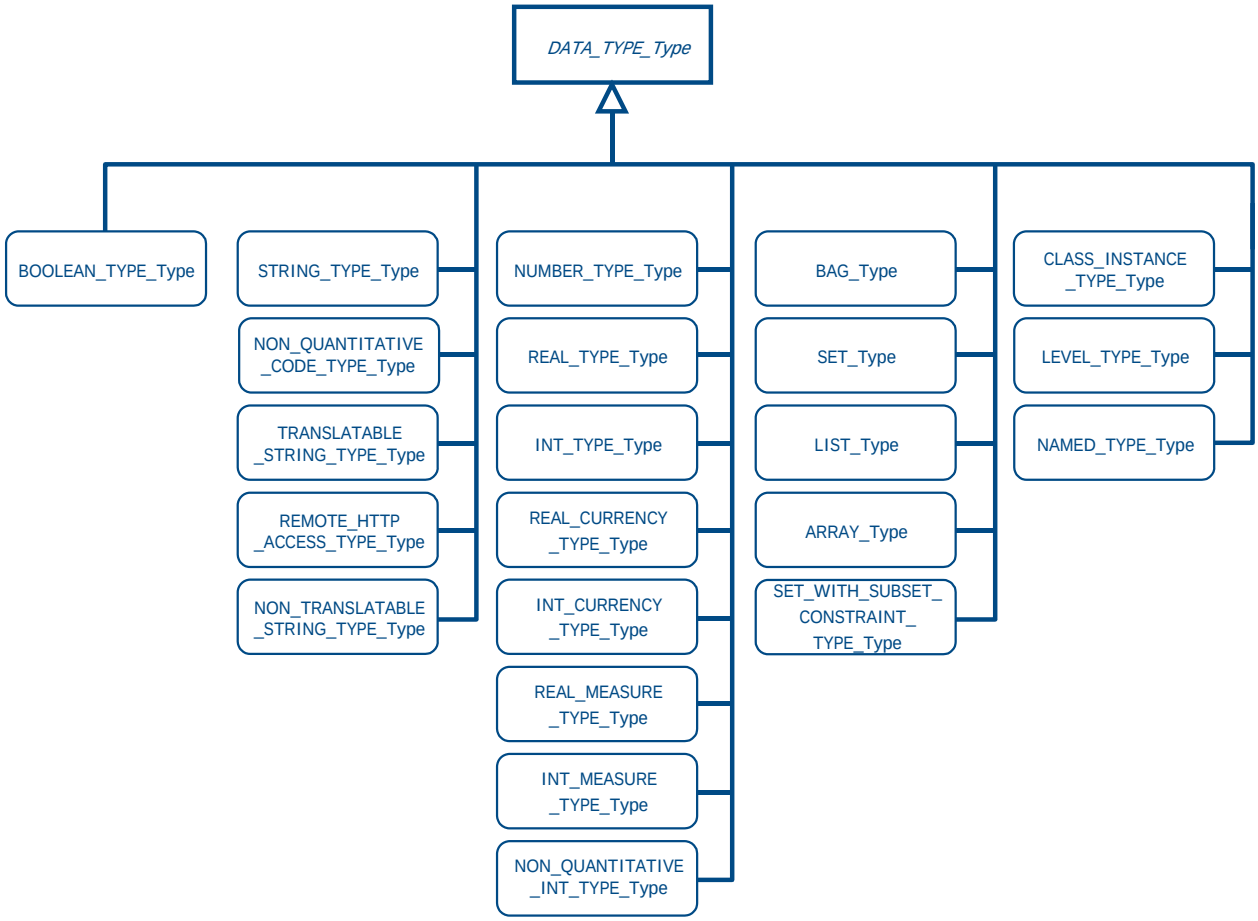

Figure 54 – OntoML datatype system

External type definitions:
ARRAY_TYPE_Type: array collection. See Clause 7.3.8.

BAG_TYPE_Type: bag collection. See Clause 7.3.8.

BOOLEAN_TYPE_Type: Boolean type. See Clause 7.3.1.

CLASS_INSTANCE_TYPE_Type: reference to an identified class type. See Clause 7.3.9.

INT_CURRENCY_TYPE_Type: integer currency type. See Clause 7.3.5.

INT_MEASURE_TYPE_Type: integer without unit type. See Clause 7.3.6.

INT_TYPE_Type: integer without unit type. see Clause 7.3.4.

LEVEL_TYPE_Type: level type. See Clause 7.3.10.

LIST_TYPE_Type: list collection. See Clause 7.3.8.

NAMED_TYPE_Type: reference to an identified data type. See Clause 7.3.11.

NON_QUANTITATIVE_CODE_TYPE_Type: enumeration of string codes type: See Clause 7.3.3.

NON_QUANTITATIVE_INT_TYPE_Type: enumeration of integer codes type: See Clause7.3.7.

NON_TRANSLATABLE_STRING_TYPE_Type: non translatable string. See Clause 7.3.2.

NUMBER_TYPE_Type: number type. See Clause 7.3.4.

REAL_TYPE_Type: real without unit type. See Clause 7.3.4.

REAL_CURRENCY_TYPE_Type: real currency type. See Clause 7.3.5.

REAL_MEASURE_TYPE_Type: real with unit type. See Clause 7.3.6.

REMOTE_HTTP_ACCESS_TYPE_Type: string representing an URI. See Clause 7.3.2.

SET_TYPE_Type: set collection. See Clause 7.3.8.

SET_WITH_SUBSET_CONSTRAINT_TYPE_Type: explicitly defined set collection. See Clause 7.3.8.

STRING_TYPE_Type: string type. See Clause 7.3.2.

TRANSLATABLE_STRING_TYPE_Type: translatable string type. See Clause 7.3.2.

7.3.1 Boolean type

A Boolean type is defined using the **BOOLEAN_TYPE_Type** XML complex type. It is represented in Figure 55.

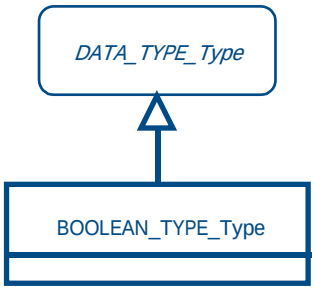

Figure 55 – Boolean type structure

7.3.2 String types

The value domain defined by strings is a sequence of any kind of characters. A basic OntoML string type is defined by the **STRING_TYPE_Type** XML complex type. It may be further qualified as a localized string (**TRANSLATABLE_STRING_TYPE_Type** XML complex type), a string that is not translatable (**NON_TRANSLATABLE_STRING_TYPE_Type** XML complex type) or a string that represents an HTTP address (**REMOTE_HTTP_ADDRESS_TYPE_Type** XML complex type). Figure 56 illustrates these resources.

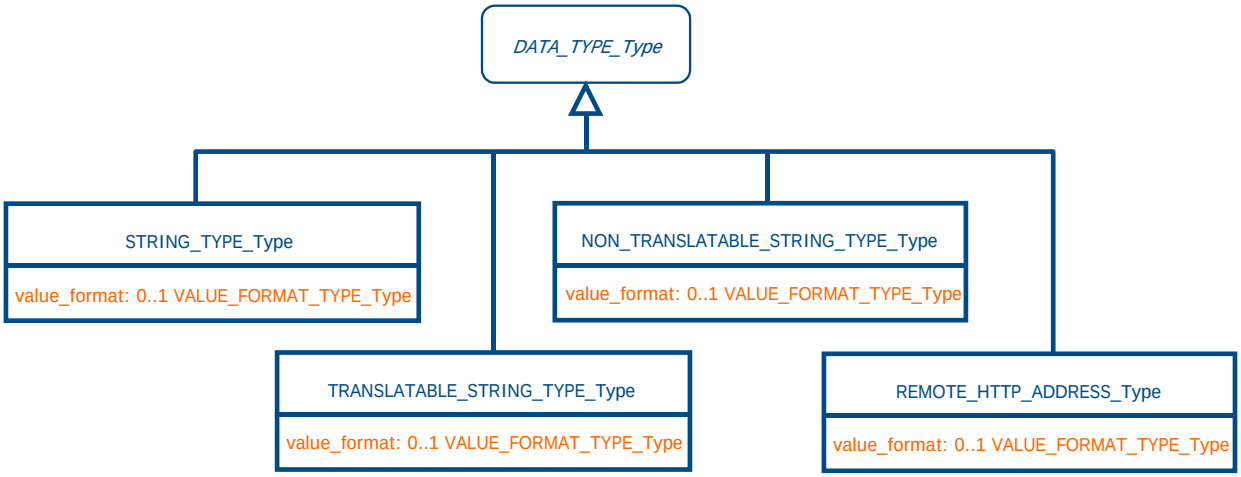

Figure 56 – String types structure

Internal item definition:
value_format: the specification of the type and length of the recommended presentation for displaying the value of a property. If present, this attribute provides guidance to the system about how the value should be displayed.

Internal type definition:
VALUE_FORMAT_TYPE_Type: identifies the values allowed for a value format. The length of a **VALUE_FORMAT_TYPE_Type** value shall not exceed 80 characters.

NOTE 1 **VALUE_FORMAT_TYPE_Type** values are defined according to ISO 6093 and ISO 9735.

Internal subtype definitions:
NON_TRANSLATABLE_STRING_TYPE_Type: provides for values of properties or user types that are of type STRING, but that are represented in the same way in any languages.

REMOTE_HTTP_ACCESS_TYPE_Type: provides for values of properties or user types that are of type STRING, but represents a URI.

STRING_TYPE_Type: provides for values of properties or user types that are of type STRING.

TRANSLATABLE_STRING_TYPE_Type: provides for values of properties or user types that are of type STRING, but that are supposed to be represented as different strings in different languages.

NOTE 2 The **source_language** XML element (defined in the **PROPERTY_DET** XML complex type, see Clause 5.7.4) of a property whose data type (**domain** XML element) is defined as a **TRANSLATABLE_STRING_TYPE_Type** plays the role of the root language in which string property values should be converted to identify same values when represented in different language.

7.3.3 Enumeration of string codes type

A string that must take its value among a set of enumerated codes is represented by the **NON_QUANTITATIVE_CODE_TYPE_Type** XML complex type. Each of those codes is associated to a meaning. It is illustrated in Figure 57.

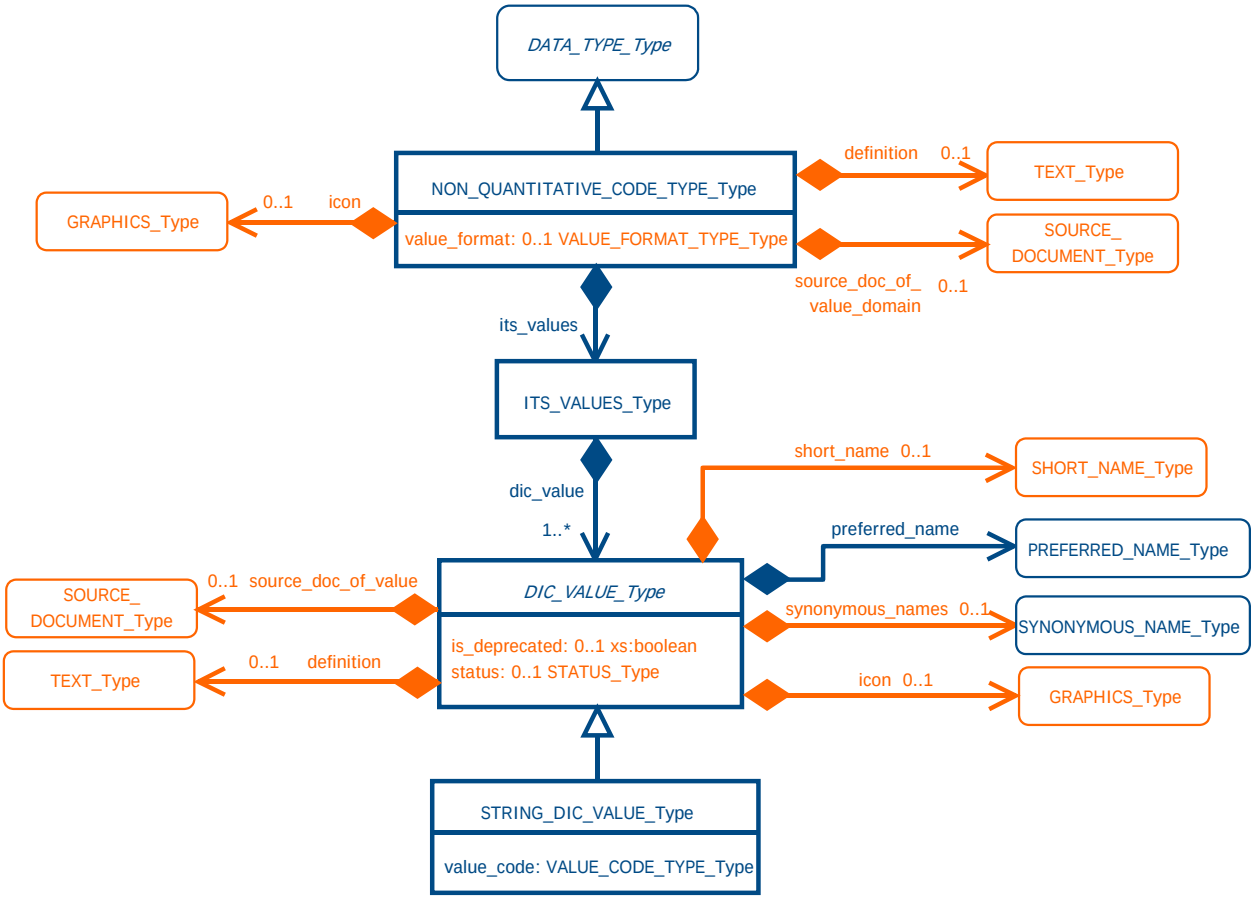

Figure 57 – Enumeration of string codes type structure

The enumeration items are defined in an XML element container called **its_values** whose content model is defined by is a **ITS_VALUES_Type** XML complex type. Each enumeration item is represented through a **dic_value** XML element. Its own content model is defined as a **DIC_VALUE_Type**, an more precisely, in case of the specification of an enumeration of string codes, by its specific **STRING_DIC_VALUE_Type** XML complex type subtype.

Internal item definition:
value_format: the specification of the type and length of the recommended presentation for displaying the value of a property. If present, this attribute provides guidance to the system about how the value should be displayed.

definition: the text describing this enumeration, possibly translated.

icon: a graphics representing the description associated with the enumeration.

source_doc_of_value_domain: the possible source document from which the enumeration definition comes.

its_values: enumeration items container.

its_values/dic_value: enumeration item.

NOTE 1 Each **dic_value** XML element content model shall be represented as a **STRING_DIC_VALUE_Type** XML complex type.

its_values/dic_value/preferred_name: the name of the enumeration item that is preferred for use, possibly translated.

its_values/dic_value/short_name: the abbreviation of the preferred name, possibly translated.

its_values/dic_value/synonymous_names: the set of synonymous names of the preferred name, possibly translated.

its_values/dic_value/icon: a graphics representing the description associated with the enumeration item names.

its_values/dic_value/definition: the text describing this enumeration item, possibly translated.

its_values/dic_value/is_deprecated: a Boolean that specifies, when true, that the enumeration item shall no longer be used

its_values/dic_value/status: defines the life cycle state of the enumeration item.

NOTE 2 Allowed **status** values are defined by private agreement between the dictionary supplier and dictionary users.

NOTE 3 If the **status** XML element is not provided, and if the enumeration item is not deprecated as denoted by a possible **is_deprecated** XML element, then the enumeration item has the same standardization status as the whole ontology into which it is used. In particular, if the ontology is standardized, this enumeration item is part of the current edition of the standard.

its_values/dic_value/source_doc_of_value: the possible source document from which the enumeration item definition comes.

its_values/dic_value/value_code: the enumeration item value.

Internal type definition:
STATUS_Type: a string (**xs:string** XML Schema datatype) that represents the values allowed for a status.

VALUE_FORMAT_TYPE_Type: identifies the values allowed for a value format. The length of a **VALUE_FORMAT_TYPE_Type** value shall not exceed 80 characters.

NOTE 4 **VALUE_FORMAT_TYPE_Type** values are defined according to ISO 6093 and ISO 9735.

External type definition:
SOURCE_DOCUMENT_Type: See Clause 7.2.3.1.

GRAPHICS_Type: See Clause 7.2.3.2.

PREFERRED_NAME_Type: See Clause 7.1.2.

SHORT_NAME_Type: See Clause 7.1.2.

SYNONYMOUS_NAME_Type: See Clause 7.1.2.

TEXT_Type: See Clause 7.1.2.

7.3.4 Numeric types

A numeric value may be represented as a general value (**NUMBER_TYPE_Type**), or as a real (**REAL_TYPE_Type**) or as an integer (**INT_TYPE_Type**) value.

Representation of numeric is given in Figure 58.

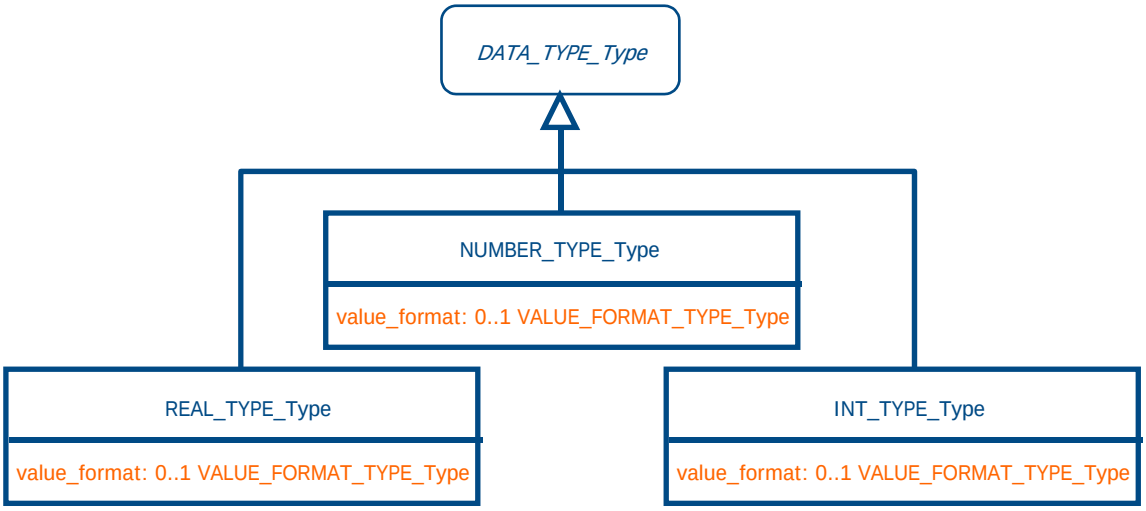

Figure 58 – Numeric types structure

Internal item definition:
value_format (**NUMBER_TYPE_Type**, **REAL_TYPE_Type**, **INT_TYPE_Type**): the specification of the type and length of the recommended presentation for displaying the value of a property. If present, this attribute provides guidance to the system about how the value should be displayed.

Internal type definition:
VALUE_FORMAT_TYPE_Type: identifies the values allowed for a value format. The length of a **VALUE_FORMAT_TYPE_Type** value shall not exceed 80 characters.

NOTE **VALUE_FORMAT_TYPE_Type** values are defined according to ISO 6093 and ISO 9735.

7.3.5 Numeric currency types

Both integer and real value domains may represent a currency. The former is represented by the OntoML **INT_CURRENCY_TYPE_Type** XML complex type, the latter by the OntoML **REAL_CURRENCY_TYPE_Type** XML complex type. Figure 59 Illustrates the numeric currency types representation.

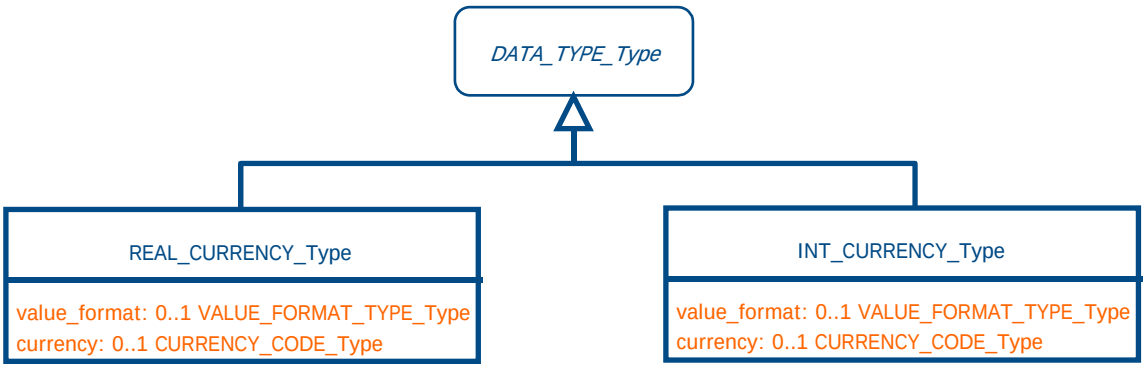

Figure 59 – Numeric currency types structure

Internal item definition:
value_format (**REAL_CURRENCY_Type**, **INT_CURRENCY_Type**): the specification of the type and length of the recommended presentation for displaying the value of a property. If present, this attribute provides for guidance to the system about how the value should be displayed.

currency: the associated code of the described currency.

NOTE 1 Currencies are expressed according ISO 4217.

NOTE 2 When not defined, the **currency** is intended to be explicitly represented at the library level.

Internal type definition:
VALUE_FORMAT_TYPE_Type: identifies the values allowed for a value format. The length of a **VALUE_FORMAT_TYPE_Type** value shall not exceed 80 characters.

NOTE 3 **VALUE_FORMAT_TYPE_Type** values are defined according to ISO 6093 and ISO 9735.

CURRENCY_CODE_Type: a string (**xs:string** XML Schema datatype) that represents the values allowed for a currency. This string length shall be equal to 3 characters.

EXAMPLE Currency values could be "CHF" for Swiss Francs, "CNY" for Yuan Renminbi (Chinese), "JPY" for Yen (Japanese), "SUR" for SU Rouble, "USD" for US Dollars, "EUR" for Euros etc ...

7.3.6 Numeric measure types

Integer and real value domains may represent a measure. The former is represented by the OntoML **INT_MEASURE_TYPE_Type**, the latter by the OntoML **REAL_MEASURE_TYPE_Type** XML complex type. illustrates the numeric measure types representation.

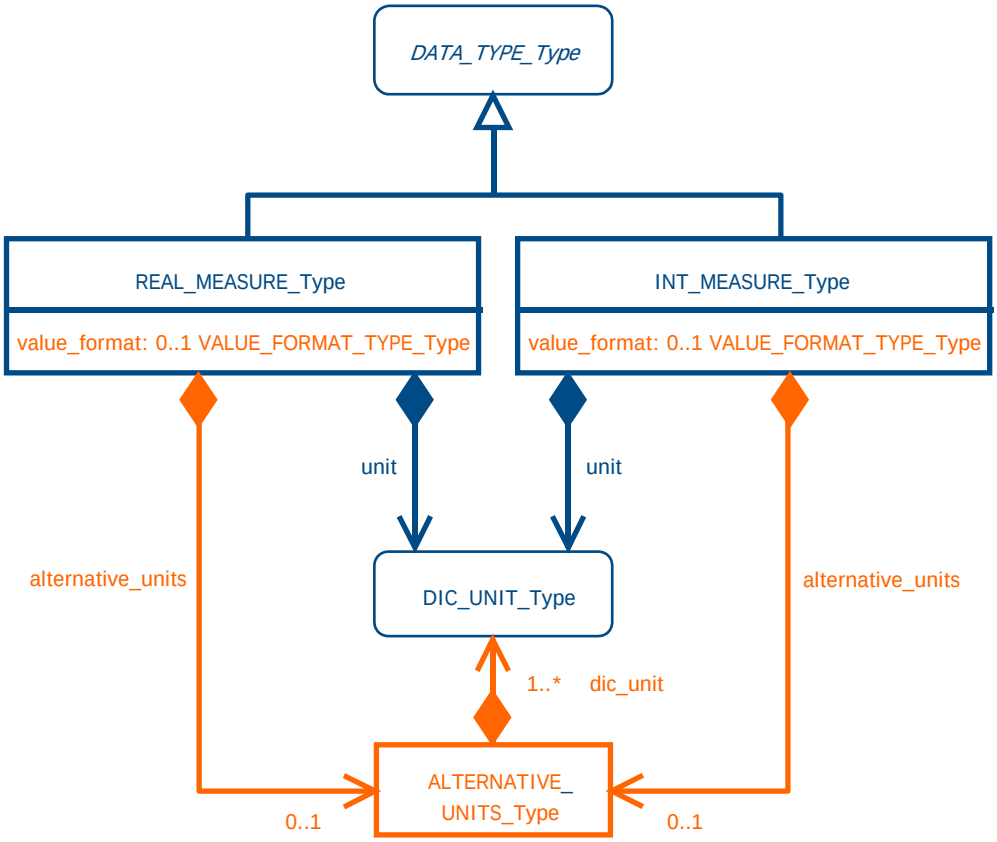

Figure 60 – Numeric measure types structure

Internal item definition:
value_format (**REAL_MEASURE_TYPE_Type**, **INT_MEASURE_TYPE_Type**): the specification of the type and length of the recommended presentation for displaying the value of a property. If present, this attribute provides guidance to the system about how the value should be displayed.

unit: the default unit of reference associated to the described measure.

NOTE 1 If the value of a property, whose value domain is a measure, is exchanged as a single number, this means that this value is expressed in this defined unit.

alternative_units: the set of other units that may be used to express the value of the property the whose value domain is a measure.

dic_unit (**ALTERNATIVE_UNITS_Type**): the specification of an alternative unit.

Internal type definition:
VALUE_FORMAT_TYPE_Type: identifies the values allowed for a value format. The length of a **VALUE_FORMAT_TYPE_Type** value shall not exceed 80 characters.

NOTE 2 **VALUE_FORMAT_TYPE_Type** values are defined according to ISO 6093 and ISO 9735.

ALTERNATIVE_UNITS_Type: the set of possible alternative units.

DIC_UNIT_Type: the specification of a dictionary unit. See clause 7.4.

7.3.7 Enumeration of integer codes type

An integer that must take its value among a set of enumerated codes is represented by the **NON_QUANTITATIVE_INT_TYPE_Type** XML complex type. Each of those codes is associated to a meaning. It is illustrated in Figure 61.

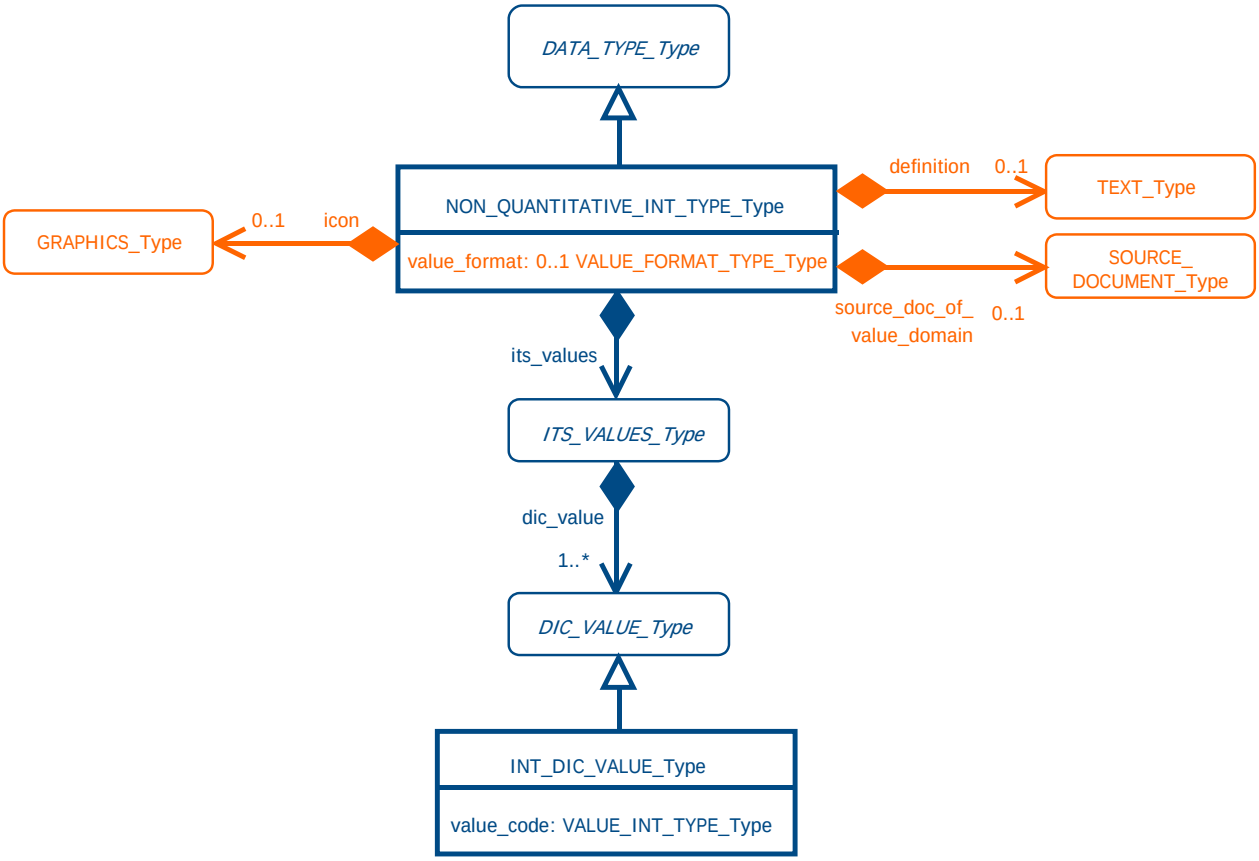

Figure 61 – Enumeration of integer codes type structure

The enumeration items are defined in an XML element container called **its_values** whose content model is defined by is a **ITS_VALUES_Type** XML complex type. Each enumeration item is represented through the **dic_value** XML element. Its own content model is defined as a **DIC_VALUE_Type**, an more precisely, in case of the specification of an enumeration of integer codes, by its specific **INT_DIC_VALUE_Type** XML complex type subtype.

Internal item definition:
value_format: the specification of the type and length of the recommended presentation for displaying the value of a property. If present, this attribute provides guidance to the system about how the value should be displayed.

definition: the text describing this enumeration, possibly translated.

source_doc_of_value_domain: the possible source document from which the enumeration definition comes.

icon: a graphics representing the description associated with the enumeration.

its_values: enumeration items container.

its_values/dic_value/value_code: the enumeration item integer value.

NOTE 1 Each **dic_value** XML element content model (defined in the **ITS_VALUES_Type** XML complex type) shall be represented as a **STRING_DIC_VALUE_Type** XML complex type.

Internal type definition:
GRAPHICS_Type: an XML abstract complex type representing an external resource that is a graphics.

VALUE_FORMAT_TYPE_Type: identifies the values allowed for a value format. The length of a **VALUE_FORMAT_TYPE_Type** value shall not exceed 80 characters.

NOTE 2 **VALUE_FORMAT_TYPE_Type** values are defined according to ISO 6093 and ISO 9735.

External type definition:
DIC_VALUE_Type: see Clause 7.3.3 .

SOURCE_DOCUMENT_Type: See Clause 7.2.3.1.

ITS_VALUES_Type: see Clause 7.3.3.

TEXT_Type: See Clause 7.1.2.

7.3.8 Collection types

Collections provide for the definition of data types that may be expressed as lists, sets, bags, arrays or constrained subsets of any kind of values. Five kind of collections are distinguished: bag, set, list, array and set with a subset constraint.

7.3.8.1 Bag type

A bag is an unordered collection of values that may contain duplicates A bag type is represented as a **BAG_TYPE_Type** XML complex type as illustrated in Figure 62.

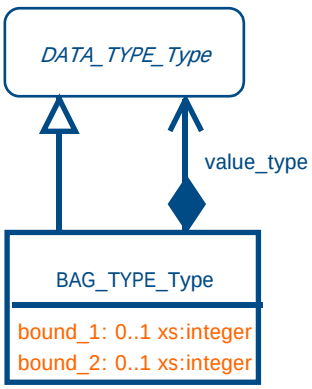

Figure 62 – Bag type structure

Internal item definition:
value_type: type of value (simple or complex) which is used for each element of the bag.

bound_1: the possible minimal cardinality of the bag.

NOTE 1 If **bound_1** is defined, it shall be greater or equal to 0. Otherwise, its default value is equal to 0.

bound_2: the possible maximal cardinality of the bag.

NOTE 2 If **bound_2** is defined, it shall be greater than 0. Otherwise, its default value is equal to unknown (unbounded).

NOTE 3 If **bound_2** is defined, **bound_1** must also be defined, and **bound_2** shall be greater than **bound_1**.

7.3.8.2 Set type

A set is an unordered collection of values that does not contain duplicates. A set type is represented as a **SET_TYPE_Type** XML complex type as illustrated in Figure 63.

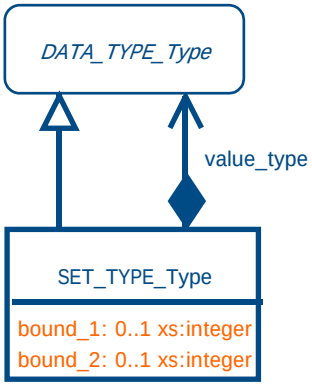

Figure 63 – Set type structure

Internal item definition:
value_type: type of value (simple or complex) which is used for each element of the set.

bound_1: the possible minimal cardinality of the set.

NOTE 1 If **bound_1** is defined, it shall be greater or equal to 0. Otherwise, its default value is equal to 0.

bound_2: the possible maximal cardinality of the set.

NOTE 2 If **bound_2** is defined, it shall be greater than 0. Otherwise, its default value is equal to unknown (unbounded).

NOTE 3 If **bound_2** is defined, **bound_1** must also be defined, and **bound_2** shall be greater than **bound_1**.

7.3.8.3 List type

A list is an ordered collection of values, possibly unique. A list type is represented as a **LIST_TYPE_Type** XML complex type as illustrated in Figure 64.

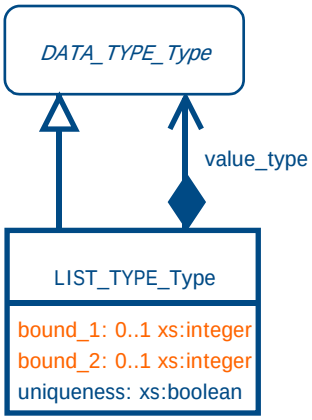

Figure 64 – List type structure

Internal item definition:
value_type: type of value (simple or complex) which is used for each element of the list.

bound_1: the possible minimal cardinality of the set.

NOTE 1 If **bound_1** is defined, it shall be greater or equal to 0. Otherwise, its default value is equal to 0.

bound_2: the possible maximal cardinality of the set.

NOTE 2 If **bound_2** is defined, it shall be greater than 0. Otherwise, its default value is equal to unknown (unbounded).

NOTE 3 If **bound_2** is defined, **bound_1** must also be defined, and **bound_2** shall be greater or equal than **bound_1**.

uniqueness: a Boolean flag to indicate whether all elements of the list must be unique (true) or whether duplicates are allowed (false).

7.3.8.4 Array type

An array is an ordered collection of possibly unique and optional values, of fixed length, whose members are indexed by a range of consecutive integers. An array type is represented as an **ARRAY_TYPE_Type** XML complex type as illustrated in Figure 65.

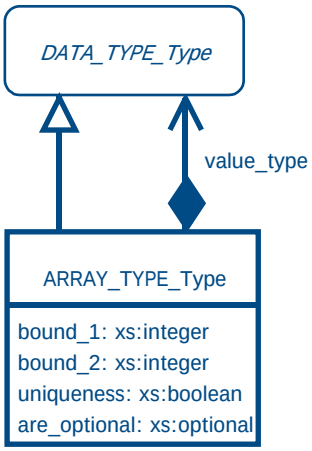

Figure 65 – Array type structure

Internal item definition:
value_type: type of value (simple or complex) which is used for each element of the array.

bound_1: the integer that defines the low index of the defined array type.

bound_2: the integer that defines the upper index of the defined array type.

uniqueness: a Boolean flag that indicates whether all elements of the array must be present (false) or whether some elements of the array may be missing (true).

are_optional: a Boolean flag that indicates whether all elements of the array must be present (false) or whether some elements of the array may be missing (true).

7.3.8.5 Set with subset constraint type

A set with a subset constraint is an unordered collection of values that does not contain duplicates and for which a subset of a minimal size to a maximal size may be extracted. A set with a subset constraint type is represented as a **SET_WITH_SUBSET_CONSTRAINT_TYPE_Type** XML complex type as illustrated in Figure 66.

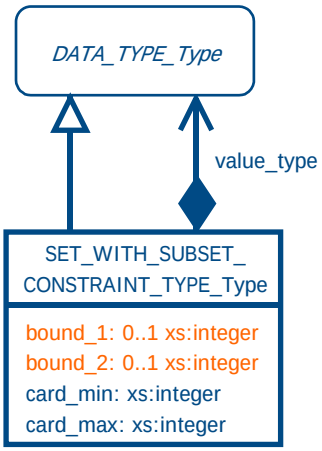

Figure 66 – Set with a subset constraint type structure

Internal item definition:
value_type: type of value (simple or complex) which is used for each element of the array.

bound_1: the integer that defines the low index of the defined set with a subset constraint type.

NOTE 1 If **bound_1** is defined, it shall be greater or equal to 0. Otherwise, its default value is equal to 0.

bound_2: the integer that defines the upper index of the defined set with a subset constraint type.

NOTE 2 If **bound_2** is defined, it shall be greater than 0. Otherwise, its default value is equal to unknown (unbounded).

NOTE 3 If **bound_2** is defined, **bound_1** must also be defined, and **bound_2** shall be greater than **bound_1**.

card_min: the minimal size of the subsets that may be extracted.

card_max: the maximal size of the subsets that may be extracted.

NOTE 4 **card_min** shall be less or equal than **card_max**.

NOTE 5 If **bound_1** is defined**, card_min** shall not be greater than **bound_1**.

NOTE 6 If **bound_2** is defined**, card_max** shall not be greater than **bound_2**.

7.3.9 Class instance type

A class instance type allows to define a property value domain that is a class instance.

NOTE A property whose type is a class instance type establishes a relationship between both classes. This relationship may be, for instance, a composition relationship.

EXAMPLE A *bolted assembly* may be constituted of a *screw* and a *nut*. Assuming that *bolted assembly*, *screw* and *nut* are parts families, the *bolted assembly* constituents may be represented firstly by defining two properties in the *bolted assembly* class (respectively *its_screw* and *its_nut*), and secondly by assigning to these properties a value domain that would be a **CLASS_INSTANCE_Type**, referencing respectively the *screw* class and the *nut* class.

A class instance type is represented as a **CLASS_INSTANCE_Type** XML complex type as it is illustrated in Figure 67.

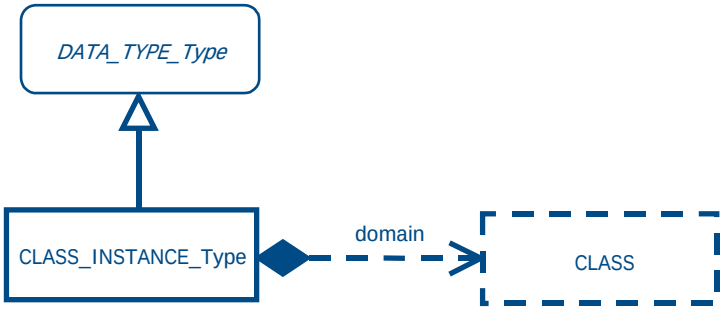

Figure 67 – Instance value domain structure

Internal item definition:
domain: a reference to the class ontology concept that defines the value domain of the data type.

7.3.10 Level type

A level type enables to represent an association of a property value that represents a physical quantity with one or more of the following qualifiers:

min: the minimal value of the physical quantity;

nom: the nominal value of the physical quantity;

type: the typical value of the physical quantity;

max: the maximal value of the physical quantity.

It is represented as a **LEVEL_TYPE_Type** XML complex type as it is illustrated in Figure 68.

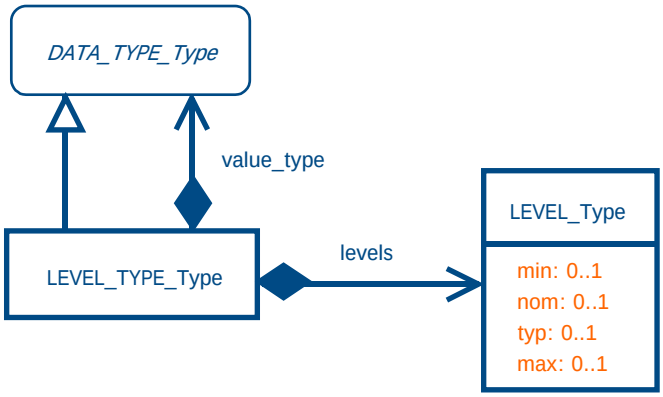

Figure 68 – Levels value domain structure

Internal item definition:
value_type: type of value which is associated to the level specification.

NOTE 1 The **value_type** shall be a numeric as defined in Clause 7.3.4.

levels: the list of qualifiers that shall be associated with the property.

min: the minimal qualifier assigned to the property value domain.

nom: the nominal qualifier assigned to the property value domain.

typ: the typical qualifier assigned to the property value domain.

max: the maximal qualifier assigned to the property value domain.

NOTE 2 There isn't any content model associated to the defined qualifiers. Therefore, no datatype is assigned to the XML elements corresponding to each qualifier.

NOTE 3 At least one qualifier shall be defined.

Internal item definition:
LEVEL_Type: the specification of the possible levels.

NOTE 4 Each level is represented as an optional and empty XML element.

7.3.11 Named data type

A named data type enables to represent a domain of values as a named type ontology concept. It is represented by a **NAMED_TYPE_Type** XML complex type as illustrated in Figure 69.

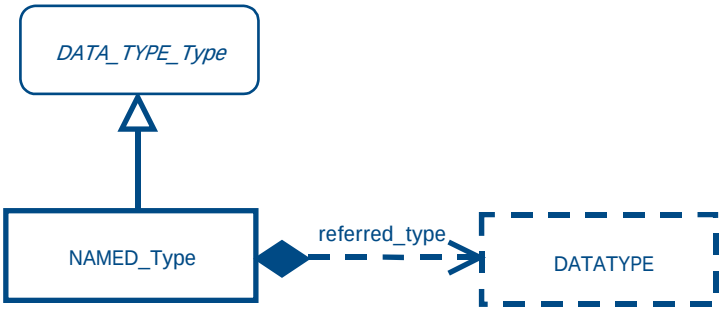

Figure 69 – Named value domain structure

Internal item definition:
referred_type: a reference to the datatype ontology concept that defines the value domain.

7.3.12 Advanced-level data types

The OntoML type system also allows to represent some other specific property value domains that may be used in advanced-level ontology classes (i.e., functional model classes and functional view classes). They are globally represented in Figure 70.

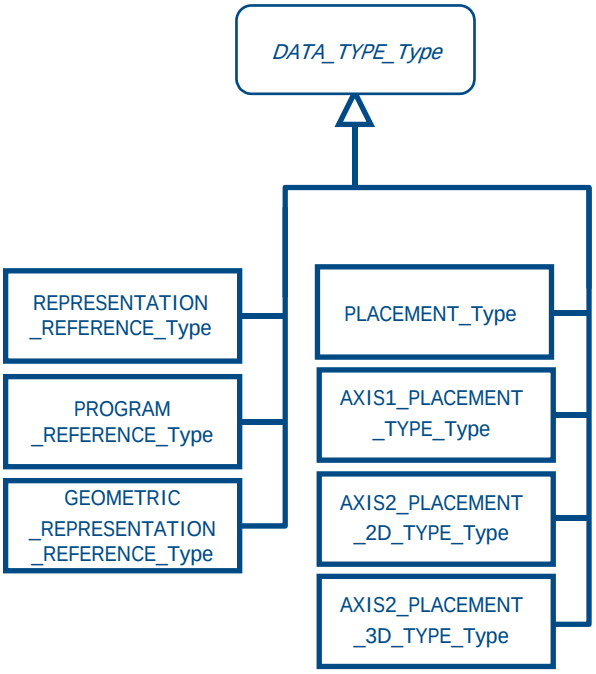

Figure 70 – Advanced value domain structures

Internal subtype definitions:
AXIS1_PLACEMENT_TYPE_Type: it allows to define a property value domain whose value is an instance of the *axis1_placement* EXPRESS entity and that enables the location and the orientation of geometric items with respect to a coordinate system.

NOTE 1 The *axis1_placement* EXPRESS entity is defined in ISO 10303-42, clause 4.4.13.

AXIS2_PLACEMENT_2D_TYPE_Type: it allows to define a property value domain whose value is an instance of the *axis2_placement_2D* EXPRESS entity and that enables the location and the orientation of geometric items with respect to a coordinate system.

NOTE 2 The *axis2_placement_2D* EXPRESS entity is defined in ISO 10303-42, clause 4.4.14.

AXIS2_PLACEMENT_3D_TYPE_Type: it allows to define a property value domain whose value is an instance of the *axis2_placement_3D* EXPRESS entity and that enables the location and the orientation of geometric items with respect to a coordinate system.

NOTE 3 The *axis2_placement_3D* EXPRESS entity is defined in ISO 10303-42, 4.4.15.

GEOMETRIC_REPRESENTATION_REFERENCE_Type: it allows to define a property value domain whose value is an instance of the *geometric_context* EXPRESS entity and that enables to define the context in which location and orientation of geometric items are defined.

NOTE 4 The *geometric_context* EXPRESS entity is defined in ISO 10303-42, clause 4.4.1.

PLACEMENT_Type: it allows to define a property value domain whose value is an instance of the *placement* EXPRESS entity and that enables the location and the orientation of geometric items with respect to a coordinate system.

NOTE 5 The *placement* EXPRESS entity is defined in ISO 10303-42, clause 4.4.12.

PROGRAM_REFERENCE_Type: it allows to define a property value domain whose value is an item implicit representation, i.e. a program reference external resource containing an algorithm that shall be triggered and provided with parameter values to generate each item representation.

NOTE 6 Program reference is defined in clause 7.3.12.

REPRESENTATION_REFERENCE_Type: it allows to define a property value domain whose value is an item explicit representation, i.e. a representation reference external resource.

NOTE 7 Representation reference is defined in clause 7.3.12.

7.4 Units

Properties for which the data type represents a measure are associated with units.

NOTE 1 OntoML units are defined according to the ISO 10303-41 information model for representing units.

In OntoML, the unit of a measure property is represented by a **DIC_UNIT_Type** XML complex type as illustrated in .

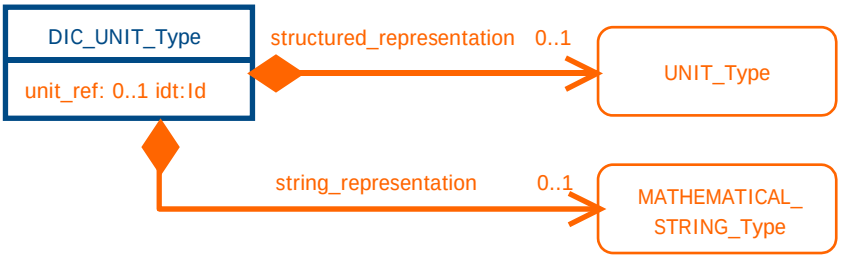

Figure 71 – General measure property unit structure

Internal item definition:
unit_ref: the possible reference to a unit identifier.

structured_representation: the explicit description of the measure property unit.

NOTE 2 Either a reference to a unit (**unit_ref**) or the explicit representation of the unit (**structured_representation**) or both shall be provided.

NOTE 3 If both a reference to a unit and the explicit representation of the unit are provided, in case of inconsistencies, the explicit representation takes precedence.

string_representation: string representation of the measure property unit.

External type definitions:
MATHEMATICAL_STRING_Type: the representation of a mathematical string. see Clause 7.8.2.

UNIT_Type: the unit specification. see Clause 7.4.1.

7.4.1 Unit structure

A unit is defined using the **UNIT_Type** abstract XML complex type. It is represented in Figure 72.

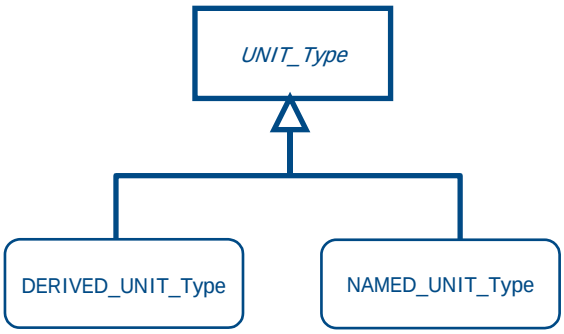

Figure 72 – Basic unit structures

External type definitions:
DERIVED_UNIT_Type: derived unit, see Clause 7.4.3.

EXAMPLE 1 Newton per square millimeter is a derived unit.

NAMED_UNIT_Type: named unit, see Clause 7.4.2.

EXAMPLE 2 Millimeter or Pascal are kinds of named unit.

7.4.2 Named unit

A named unit is a unit associated with the word, or group of words, by which the unit is identified. It is represented by the **NAMED_UNIT_Type** XML complex type as illustrated in Figure 73.

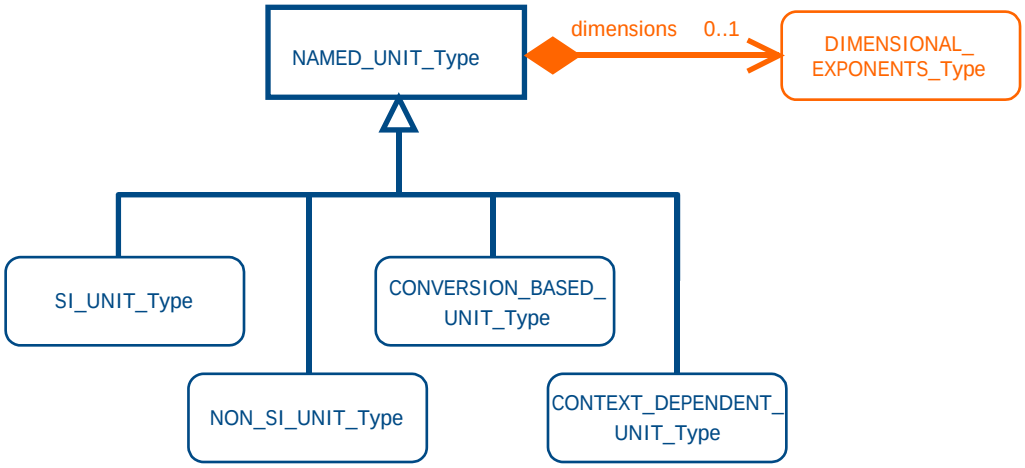

Figure 73 – Named unit general structure

Internal item definition:
dimensions: possible exponents of the base properties by which the named unit is defined.

Internal type definition:
DIMENSIONAL_EXPONENTS_Type: dimensional equation. See Clause 7.4.2.1.

External type definitions:
SI_UNIT_Type: an internationally standardized unit. See Clause 7.4.2.2.

NON_SI_UNIT_Type: a non internationally standardized unit. See Clause 7.4.2.3.

CONVERSION_BASED_UNIT_Type: a conversion based unit. See Clause 7.4.2.4.

CONTEXT_DEPENDENT_UNIT_Type: context dependent unit. See Clause 7.4.2.5.

7.4.2.1 Dimensional exponent

A named unit may be associated to its dimensional equation. It is represented by the **DIMENSIONAL_EXPONENTS_Type** as illustrated in Figure 74.

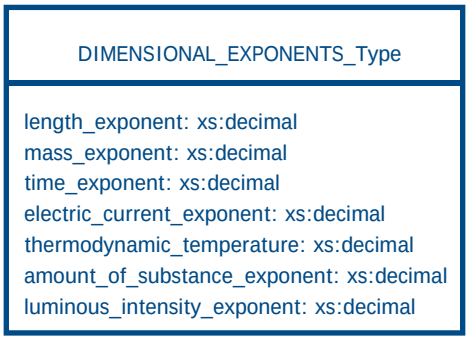

Figure 74 – Dimensional exponent structure

Internal item definitions:
length_exponent: the power of the length base quantity.

mass_exponent: the power of the mass base quantity.

time_exponent: the power of the time base quantity.

electric_current_exponent: the power of the electric current base quantity.

thermodynamic_temperature: the power of the thermodynamic temperature base quantity.

amount_of_substance_exponent: the power of the amount of substance base quantity.

luminous_intensity_exponent: the power of the luminous intensity base quantity.

7.4.2.2 Internationally standardized unit

An internationally standardized unit is the fixed quantity used as a standard in terms of which items are measured as defined by ISO 1000 (clause 2).

EXAMPLE 1 Millimeter is a internationally standardized unit.

NOTE 1 The inherited **dimensions** attribute shall not be set.

It is represented through a prefix and the associated SI unit. It is represented by the **SI_UNIT_Type** XML complex type as illustrated in Figure 75.

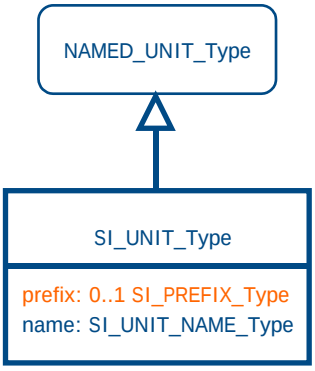

Figure 75 – International standardized unit structure

Internal item definitions:
prefix: the international standardized unit prefix.

name: the word or group of words by which the internationally standardized unit is referred to.

EXAMPLE 2 Millimeter is a SI unit: According to the **SI_UNIT_Type** XML complex type specification, the “MILLI” value would be assigned to the optional **prefix** XML element, and the “METRE” value would be assigned to the mandatory **name** XML element.

Internal type definitions:
SI_PREFIX_Type: the name of a prefix that may be associated with an internationally standardized unit. It is represented by an enumeration of the allowed internationally standardized unit prefix.

NOTE 2 The allowed international standardized unit prefixes are specified in ISO 1000 (clause 3).

SI_UNIT_NAME_Type: the name of an internationally standardized unit. It is represented by an enumeration of the allowed internationally standardized unit.

NOTE 3 The allowed internationally standardized unit names are specified in ISO 1000 (Clause 2).

7.4.2.3 Non internationally standardized unit

A non internationally standardized unit allows for the representation of units that are not SI units, nor conversion based units, nor length units. It is represented by the **NON_SI_UNIT_Type** XML complex type as illustrated in Figure 76.

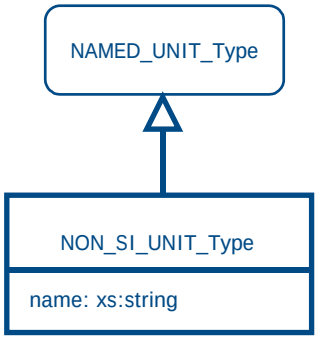

Figure 76 – Non international standardized unit structure

Internal item definitions:
name: the label used to name the described unit.

7.4.2.4 Conversion based unit

A conversion based unit is a unit defined on the base of another unit.

EXAMPLE 1 An inch is a conversion based unit.

It is represented by the **CONVERSION_BASED_UNIT_Type** XML complex type as illustrated in Figure 77.

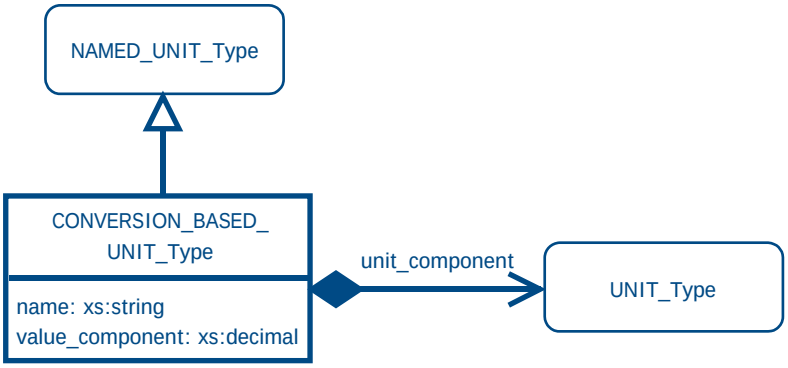

Figure 77 – Conversion based unit structure

Internal item definitions:
name: the word or group of words by which the conversion based unit is referred to.

value_component: value of the physical quantity that corresponds to one unit of the conversion based unit when expressed in the **unit_component** unit.

unit_component: the unit in which the physical quantity is expressed.

EXAMPLE 2 An inch is a conversion based unit. It is from the Imperial system, its **name** XML element value is equal to “inch", and it can be related to the SI unit, millimeter, through a measure value (**value_component** XML element) equal to 25.4 millimeter (**unit_component** XML element).

External type definition:
UNIT_Type: a general unit. See Clause 7.4.

7.4.2.5 Context dependent unit

A context dependent unit: it is a unit which is not related to the SI system.

It is represented by the **CONTEXT_DEPENDENT_UNIT_Type** XML complex type as illustrated in Figure 78.

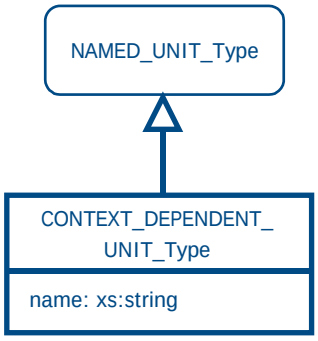

Figure 78 – Context dependent unit structure

Internal item definitions:
name: the word or group of words by which the context dependent unit is referred to.

EXAMPLE The number of parts in an assembly is a physical quantity measured in units that may be called “parts" but which cannot be related to an SI unit. The value “part” would be then assigned to the context dependent unit **name**.

7.4.3 Derived unit

A derived unit stands for an expression of units.

EXAMPLE 1 Newton per square millimeter is a derived unit.

It is represented by the **DERIVED_UNIT_Type** XML complex type as illustrated in Figure 79.

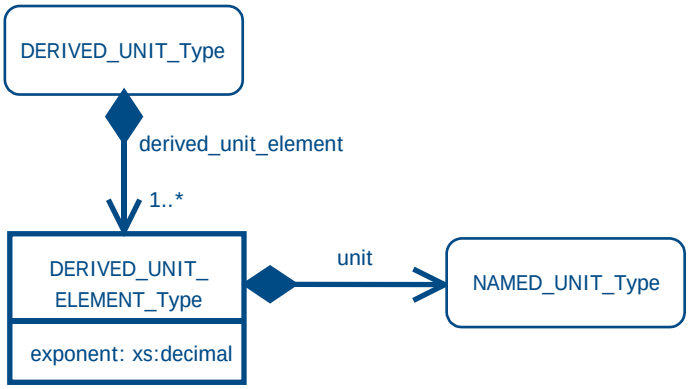

Figure 79 – Derived unit structure

Internal item definitions:
derived_unit_element: the unit quantity which makes up a derived unit.

derived_unit_element/exponent: the power that is applied to the **unit** XML element.

derived_unit_element/unit: the fixed quantity which is used as the mathematical factor.

EXAMPLE 2 *Newton per square millimeter* is a derived unit. It would then be represented by two **derived_unit_element**s: the former for representing the *Newton* SI unit, the latter for representing the *millimeter* SI unit to which a -2 **exponent** would also be assigned.

7.5 Constraints

OntoML constraints allow to further restrict the co-domain of any property defined in an ontology. Constraints may also be used to restrict co-domains when defined in a given class, and when the property domain is subclass of this class.

NOTE 1 OntoML allows to represent the whole set of constraints specified in the ISO CD 13584-42 information model.

The general constraints structure is illustrated in Figure 80.

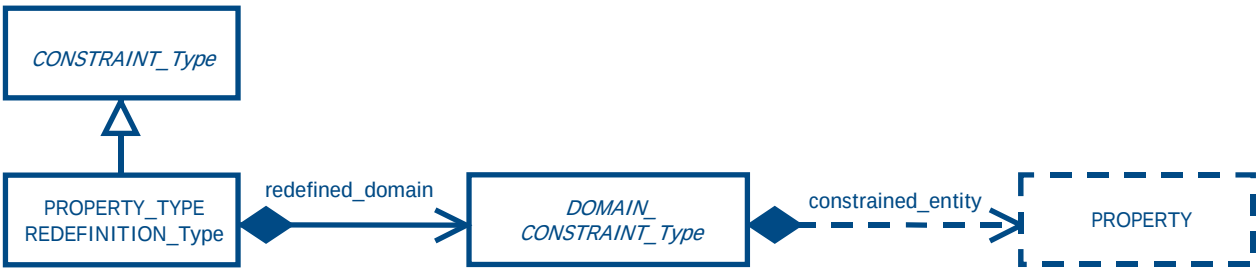

Figure 80 – General constraints structure

Internal item definitions:
redefined_domain: the constraint that applies on the data type of the constrained property.

redefined_domain/constrained_entity: the reference to the property ontology concept the type constraint applies to.

Internal type definitions:
PROPERTY_TYPE_REDEFINITION_Type: the particular constraint that allows to redefine, by restriction, the domain of values of a property.

NOTE 2 OntoML does only provide resources for representing such a constraint.

DOMAIN_CONSTRAINT_Type: defines a constraint that restricts a domain of an entity that may be either a property or a named type.

7.5.1 Subtypes of domain constraint type

OntoML provides resources for describing different kinds of domain constraints. Every specific constraint is defined as a subtype of the **DOMAIN_CONSTRAINT_Type** XML complex type. Figure 81 gives a global overview of the available property value constraints.

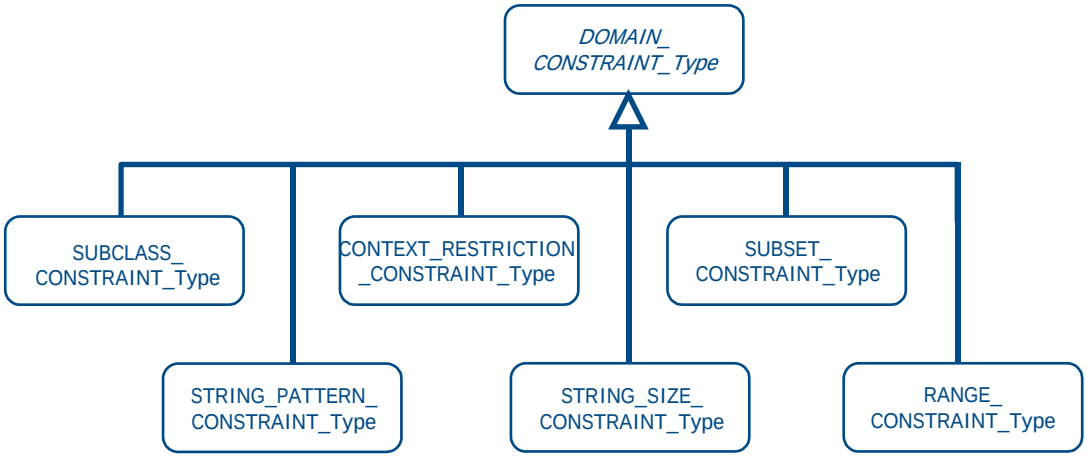

Figure 81 – Constraints representation

External type definitions:
CONTEXT_RESTRICTION_CONSTRAINT_Type: context restriction constraint specification. See Clause 7.5.4.

RANGE_CONSTRAINT_Type: range constraint specification. See Clause 7.5.6.

STRING_PATTERN_CONSTRAINT_Type: string pattern constraint specification. See Clause 7.5.3.

STRING_SIZE_CONSTRAINT_Type: string size constraint specification. See Clause 7.5.5.

SUBCLASS_CONSTRAINT_Type: subclass constraint specification. See Clause 7.5.2.

SUBSET_CONSTRAINT_Type: subset constraint specification. See Clause 7.5.7.

7.5.2 Subclass constraint

A subclass constraint applies to properties whose value domain is defined by a class instance type (see Clause 7.3.9). It specifies that the property value domain is restricted to a subclass of this class. It is represented by a **SUBCLASS_CONSTRAINT_Type** XML complex type as illustrated in Figure 82.

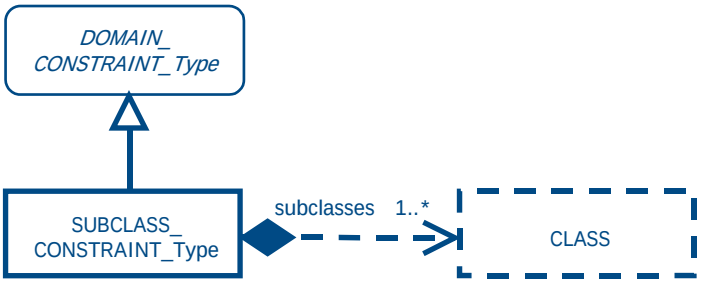

Figure 82 – Subclass constraint representation

Internal item definitions:
subclasses: the references to the class ontology concepts which redefine the new value domain of the constrained entity.

EXAMPLE Let's consider a property (**NON_DEPENDENT_P_DET_Type** XML complex type) called *screw_head* defined in the context of a *screw* class (**ITEM_CLASS_Type** XML complex type) whose value domain (**CLASS_INSTANCE_TYPE_Type** XML complex type) is a *head* class (**ITEM_CLASS_Type** XML complex type) defining the general characteristics of any kind of screw heads. Additionally, let's consider an *hexagonal_screw* class (**ITEM_CLASS_Type** XML complex type), subclass of the *screw* class, and an *hexagonal_head* class (**ITEM_CLASS_Type** XML complex type), subclass of the *head* class. In the *hexagonal_screw* class, the *screw_head* value domain could be restricted as being a subclass of the *head* class, i.e., the *hexagonal_head* class. This restriction would be expressed by defining a specific **SUBCLASS_CONSTRAINT_Type constraint** whose **subclass** XML element value would be a reference to the particular **ITEM_CLASS_Type** representing the *hexagonal_head* class.

7.5.3 String pattern constraint

A string pattern constraint applies to properties whose value domain is defined by a string.

NOTE 1 A string property value domain is either a string (see Clause 7.3.2), or a non translatable string (see Clause 7.3.2) or a translatable string (see Clause 7.3.2) , or a remote http address (see Clause 7.3.2), or enumeration of string codes (7.3.3).

A string pattern constraint specifies that the property value domain is restricted according to a given pattern. It is represented by a **STRING_PATTERN_CONSTRAINT_Type** XML complex type as illustrated in Figure 83.

NOTE 2 For properties whose data type is defined as a **TRANSLATED_STRING_TYPE_Type**, the constraint applies to any language-specific representation of the string.

NOTE 3 For properties whose data type is defined as a **NON_QUANTITATIVE_CODE_TYPE_type**, the constraint applies to the code.

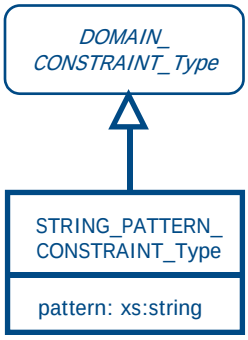

Figure 83 – String pattern constraint representation

Internal item definitions:
pattern: the pattern of string values that are allowed as values for the property identified by the constrained property.

NOTE 4 The **pattern** XML element value syntax is the LIKE operator syntax defined in the EXPRESS language reference manual defined in ISO 10303-11:2004;

EXAMPLE A property whose value domain is defined by a string (**STRING_TYPE_Type** XML complex type) and that is defined in a given class, may be restricted in a subclass of this class as being a string of digits by defining a **STRING_PATTERN_CONSTRAINT_Type** constraint and assigning the "#*" string value to its **pattern** XML element.

7.5.4 Context restriction constraint

A context restriction constraint applies to properties that play the role of a context dependent property.

NOTE 1 Context dependent properties are defined in Clause 5.7.4.

It specifies that the value domain(s) of a context parameter used for specifying the value of this context dependent property is(are) restricted using any kind of available domain constraint.

NOTE 2 Context parameters are defined in Clause 5.7.4.

It is represented by a **CONTEXT_RESTRICTION_CONSTRAINT_Type** XML complex type as illustrated in Figure 84.

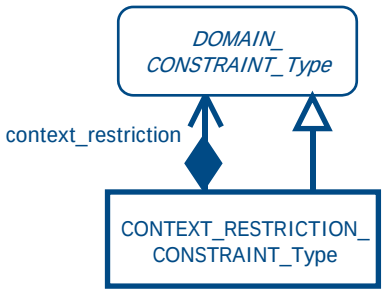

Figure 84 – Context restriction constraint representation

Internal item definitions:
context_restriction: property type redefinition that reduces the allowed domain of a context parameter of the constrained context dependent property.

EXAMPLE The *resistance* of a *thermistor* depends on (**DEPENDENT_P_DET_Type** XML complex type) the *ambient temperature* (**CONDITION_DET_Type** XML complex type) whose value domain is an **INT_TYPE_Type**. A **CONTEXT_RESTRICTION_CONSTRAINT_Type** constraint allows to require that this *ambient temperature* shall be 25°by applying a range constraint (**RANGE_CONSTRAINT_Type** XML complex type) on it.

7.5.5 String size constraint

A string size constraint applies to properties whose value domain is defined by a string.

NOTE 1 A string property value domain is either a string (see Clause 7.3.2), or a non translatable string (see Clause 7.3.2) or a translatable string (see Clause 7.3.2) , or a remote http address (see Clause 7.3.2), or enumeration of string codes (7.3.3).

A string size constraint specifies that the property value domain is restricted to possibly have a minimum and/or a maximum length. It is represented by a **STRING_SIZE_CONSTRAINT_Type** XML complex type as illustrated in Figure 85.

NOTE 2 For properties whose data type is defined as a **TRANSLATED_STRING_TYPE_Type**, the constraint applies to any language-specific representation of the string.

NOTE 3 For properties whose data type is defined as a **NON_QUANTITATIVE_CODE_TYPE_type**, the constraint applies to the code.

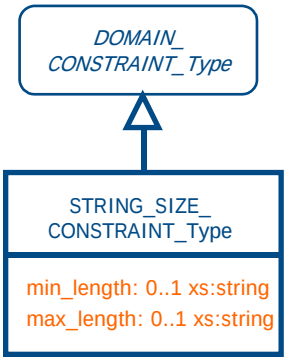

Figure 85 – String size constraint representation

Internal item definitions:
min_length: the minimal length for the strings that is allowed as value for the constrained property.

max_length: the maximal length for the strings that is allowed as value for the constrained property.

NOTE 4 **min_length** shall be greater than 0 and less or equal to **max_value**.

EXAMPLE A property whose value domain is a string (**STRING_TYPE_Type** XML complex type) and that is defined in a given class, may be restricted in a subclass of this class as having no more than 10 characters by defining a **STRING_SIZE_CONSTRAINT_Type** constraint and assigning 10 to the **max_length** XML element.

7.5.6 Range constraint

A range constraint applies to properties whose value domain is defined by a number.

NOTE 1 A string property value domain is either a number (see Clause 7.3.4), an integer (see Clause 7.3.4), a integer currency (see Clause 7.3.5), an integer measure (see Clause 7.3.6), a real (see Clause 7.3.4), a real currency (see Clause 7.3.5), a real measure (see Clause 7.3.6) or a enumeration of integer codes (see Clause 7.3.7).

A range constraint specifies that the property value domain is restricted to a subset of its values defined by a range. It is represented by a **RANGE_CONSTRAINT_Type** XML complex type as illustrated in Figure 86.

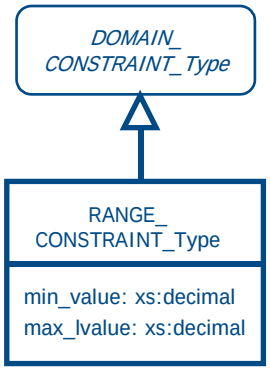

Figure 86 – Range constraint representation

Internal item definitions:
min_value: the number defining the low bound of the range of values.

max_value: the number defining the high bound of the range of values

NOTE 2 **min_value** shall be less or equal to **max_value**.

EXAMPLE A property whose value domain is an integer (**INT_TYPE_Type** XML complex type) and that is defined in a given class, may be restricted in a subclass of this class as a [10..50] range by defining a **RANGE_CONSTRAINT_Type** constraint and assigning 10 to the **min_value** and 50 to the **max_value** XML elements.

7.5.7 Subset constraint

A subset constraint applies to a property whose value domain is defined by a collection data type. It specifies that the property value domain is restricted to a discrete subset . This subset is specified by enumerating the allowed values, possibly assigning them a description. This constraint is represented by a **SUBSET_CONSTRAINT_Type** XML complex type as illustrated in Figure 87.

NOTE 1 For a currency (see Clause 7.3.5), or a measure (see Clause 7.3.6) associated with alternative unit, the constraint applies whatever be the currency or the unit.

NOTE 2 For translatable strings (see Clause 7.3.2), the constraint applies to any language-specific representation of the string.

NOTE 3 If another subset constraint is applied to a property already associated with a subset constraint in some superclass, both constraints apply. Thus the allowed set of values is the intersection of both subsets. Concerning the presentation order, and the possible meaning associated with each value, only those meanings defined in the lower subset constraint apply.

EXAMPLE A property whose value domain is an integer (**INT_TYPE_Type** XML complex type) and that is defined in a given class, may be restricted in a subclass of this class as a {10, 20, 30} value set by defining a **SUBSET_CONSTRAINT_Type** constraint and defining three possible values: 10, 20 and 30.

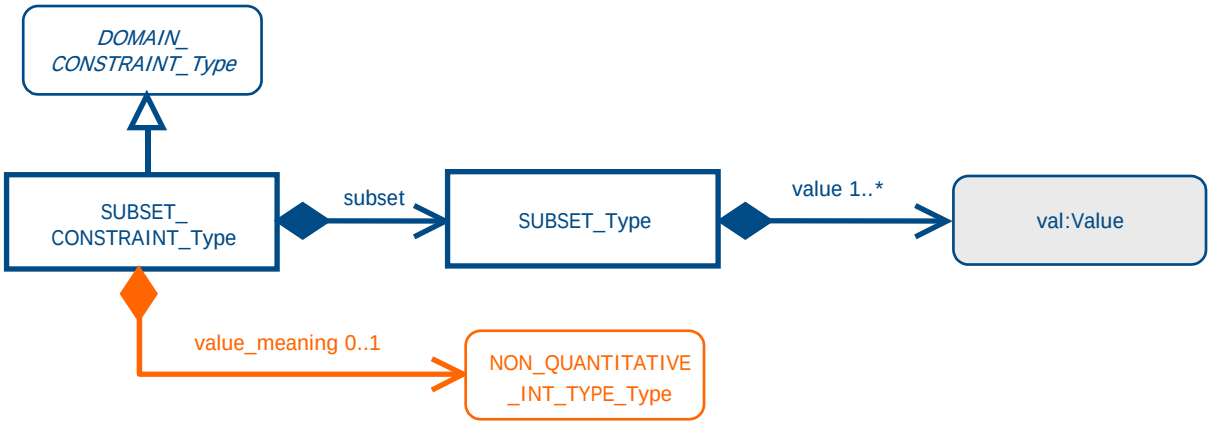

Figure 87 – Subset constraint representation

Internal item definitions:
subset: the list describing the subset of values that are allowed as possible values for the constrained property.

NOTE 4 The order defined by the list is the recommended order for presentation purposes.

subset/ value: value that is allowed as possible value for the constrained property.

NOTE 5 Representation of values shall comply with ISO 29002-10 developed as a joint effort of ISO 13584 and ISO 22745 development teams.

value_meaning: the optional description that may be associated with each value of the **subset** by means of an enumeration of integer codes, whose the i-th value describes the meaning of the i-th value of the **subset**.

Internal type definition:
SUBSET_Type: a set of values of a type specified by the **DATA_TYPE_Type** XML complex type.

7.6 A posteriori semantics relationship

An a posteriori semantic relationship is a relationship between two classes from which property equivalencies are modeled. A properties equivalence, or mapping, is performed by property pairs specification. More than one equivalence pair may be defined It is represented by an **A_POSTERIORI_SEMANTIC_RELATIONSHIP_Type** abstract XML complex type as illustrated in Figure 90.

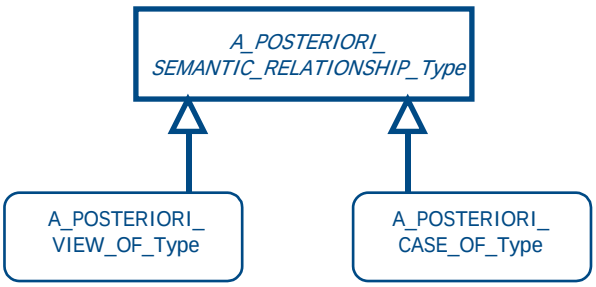

Figure 88 – A posteriori relationship general structure representation

A concrete a posteriori relationship is defined through one of the **A_POSTERIORI_SEMANTIC_RELATIONSHIP_Type** abstract XML complex type subtypes.

External type definitions:
A_POSTERIORI_CASE_OF_Type: a posteriori mapping in a *case-of* relationship. See Clause 7.6.1.

A_POSTERIORI_VIEW_OF_Type: a posteriori mapping in a *view-of* relationship. See Clause 7.6.2.

7.6.1 A posteriori mapping in a *case-of* relationship

An *a posteriori* case-of relationship allows to define *a posteriori* mappings between classes / properties belonging to different reference dictionaries. When *A* is case-of *B*, this means that all instances of *A* are also instances of *B*. In OntoML, it is represented by an **A_POSTERIORI_CASE_OF_Type** XML complex type as illustrated in Figure 89.

NOTE 1 The *case-of* relationship allows each organization to define its own reference dictionary while providing for data integration and for data exchange with other organizations.

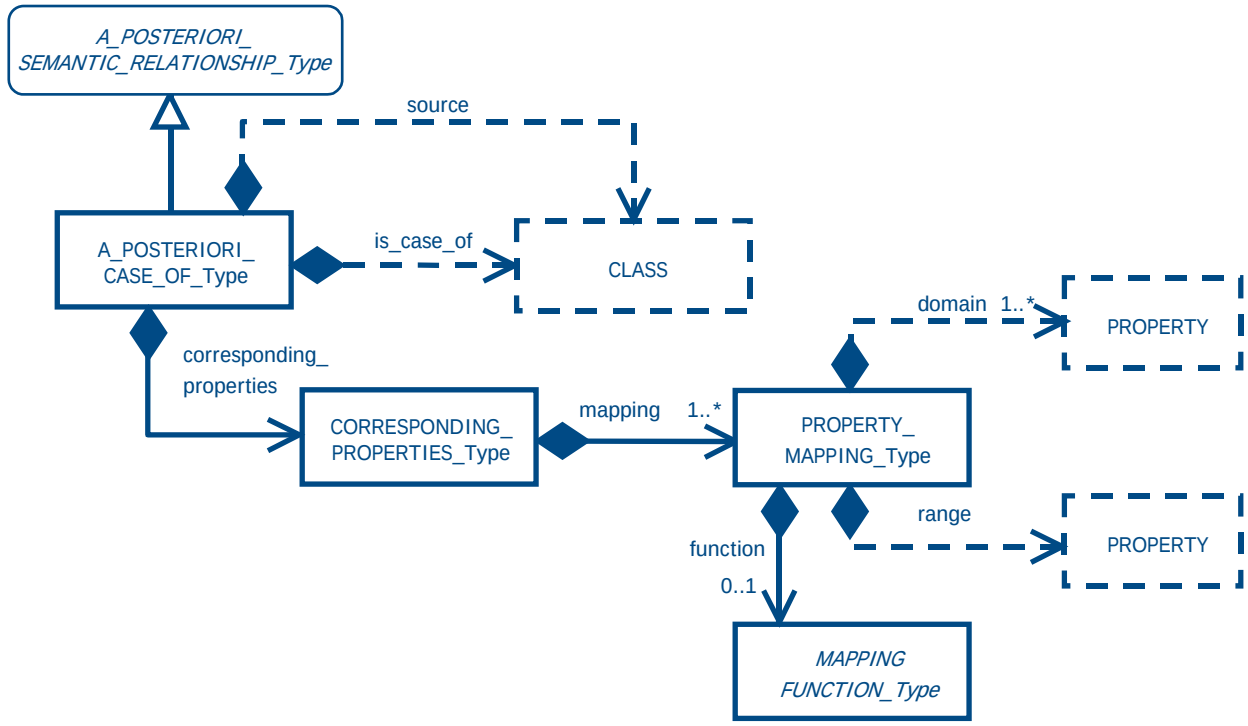

Figure 89 – A posteriori *case-of* relationship representation

Internal item definitions:
source: the reference to the class that is case-of the is-case-of class.

is_case_of: the reference to the class to which the **source** class is *case-of*.

NOTE 2 **source** and **is_case_of** referenced classes shall be both item classes or both functional model classes.

corresponding_properties: a set of property mappings.

corresponding_properties/mapping: a mapping that defines how the **range** property that belongs ton one of the class involved in a semantic relationship is computed from other properties that belong to the other class involved in the semantic relationship by means of a function.

NOTE 3 When the **function** is not provided, the **domain** shall contain a single property and the mapping means a property equivalence between the **domain** property and the **range** property.

NOTE 4 The property mapping is an oriented relationship equivalence. If a *P1* property is equivalent to a *P2* property and the *P2* property is equivalent to the *P1* property, the mapping of *P1* and *P2* properties shall be defined twice.

corresponding_properties/mapping/range: the reference to the property that must be computed.

corresponding_properties/mapping/domain: the reference to the property or properties from which the range property is computed.

corresponding_properties/mapping/function: an optional function that specifies how the **range** property is computes from the **domain** property(ies). When not provided, the **range** property is equivalent to the **domain** property that shall be unique.

Internal type definitions:
CORRESPONDING_PROPERTIES_Type: a container for the set of pairs of properties.

PROPERTY_MAPPING_Type: a property mapping specification.

MAPPING_FUNCTION_Type: an abstract XML complex type reserved for latter standardization.

7.6.2 A posteriori mapping in a *view-of* relationship

An *a posteriori* view-of relationship allows to associate *a posteriori* a functional model class to a product characterization class. In OntoML, it is represented by an **A_POSTERIORI_VIEW_OF_Type** XML complex type as illustrated in Figure 90.

EXAMPLE Assume that a class of functional models generate simplified model of a screw, independently of any particular kind of screw. The *a posteriori* view-of relationship would allow to connect these representations with a particular screw characterization class.

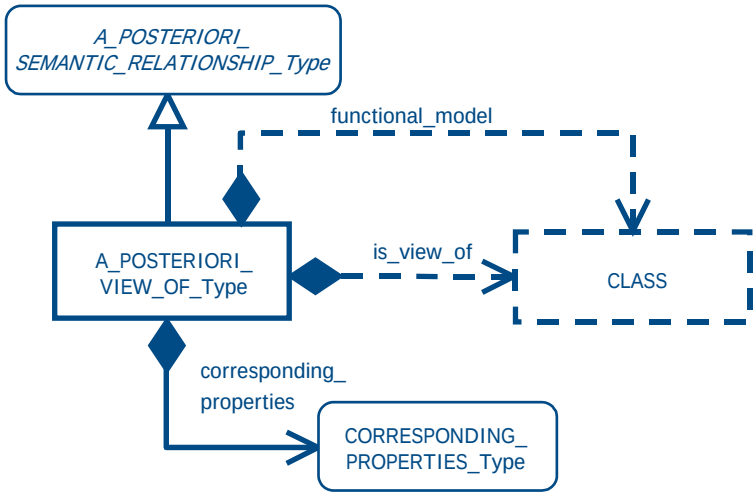

Figure 90 – A posteriori semantic relationships structure

Internal item definitions:
functional_model: the reference to the functional model class that will provide representations for the product of the is-view-of class.

is_view_of: the reference to the item class to which the **functional_model** class is *view-of*.

corresponding_properties: a set of property mappings.

External type definition:
CORRESPONDING_PROPERTIES_Type: property mappings. see Clause 7.6.1.

7.7 Data exchange specification identification

A data exchange specification allows to specify various characteristics of an OntoML document instance:

— the subset of the OntoML specification used,

— the possible view exchange protocols used when dealing with functional models associated to characterization classes,

— the implementation method namely in the case of OntoML, an OntoML document instance.

Two data exchange specification elements are defined. They allow to specify:

— characteristics of the exchanged ontology and/or library (see clause 7.7.1).

— characteristics of the possible view exchange protocols used (see Clause 7.7.2).

7.7.1 Simple-level ontology data exchange specification: library integrated information model identification

The library integrated information model identification identifies the model on which the exchange is based and the format used.

NOTE 1 Annex C specifies standard values that shall be used when exchanging OntoML data. These values are also defined below in notes.

It is represented by a **LIBRARY_IIM_IDENTIFICATION_Type** XML complex type as illustrated in Figure 91.

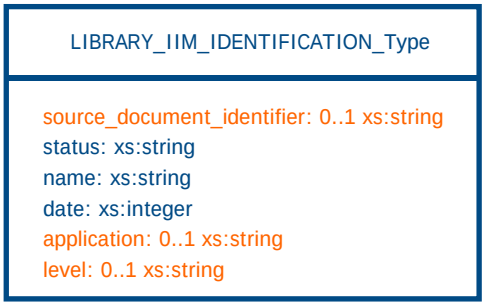

Figure 91 – Library integrated information model identification structure

Internal item definitions:
source_document_identifier: the identifier of the document that contains the data specification.

NOTE 2 For those documents issued by ISO TC184/SC4/WG2 this identifier shall be the integer part of the N number.

status: classification of the data specification with respect to its acceptance by the approving body of this International Standard, possibly followed by an integer version.

NOTE 3 A **status** may only take the following values: 'WD', 'CD', 'DIS', 'FDIS', 'IS', 'TS', 'PAS', 'ITA'.

name: the identifier of the data specification.

NOTE 4 For OntoML, this identifier shall be “ONTOML” in capital letters.

date: the year when the data specification reached its **status**.

application: an identifier to characterize an allowed functional subset of the complete data specification.

NOTE 5 Values shall be 1 if conformance class 1 is used (simple level ontology), 2 if conformance class 2 is used (advanced level ontology), 3 if conformance class 3 is used (simple level library), 4 if conformance class 4 is used (advanced level library).

level: an identifier that further characterizes an allowed subset of the **application** subset.

NOTE 6 This XML element shall not be used for OntoML exchanges.

7.7.2 Advanced-level ontology data exchange specification: view exchange protocol identification

A view exchange protocol is intended to specify which library external files are used within an OntoML document instance, which dictionary entries shall be recognized by a receiving system that claims conformance to this view exchange protocol, and which additional constraints are fulfilled by a library delivery file.

NOTE 1 An example of view exchange protocol is ISO 13584-101 for exchanging geometrical representations..

A view exchange protocol identification identifies a particular view exchange protocol. It is represented by a **VIEW_EXCHANGE_PROTOCOL_IDENTIFICATION_Type** XML complex type as illustrated in .

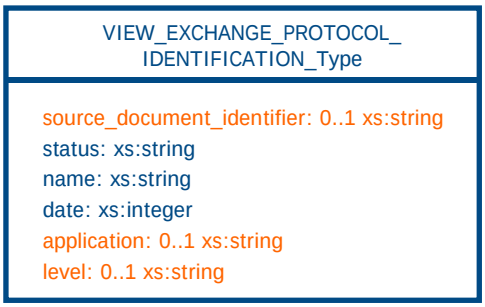

Figure 92 – View exchange protocol identification structure

Internal item definitions:
source_document_identifier: the identifier of the document that contains the data specification.

NOTE 2 For those documents issued by ISO TC184/SC4/WG2 this identifier shall be the integer part of the N number.

status: classification of the data specification with respect to its acceptance by the approving body of this International Standard, possibly followed by an integer version.

NOTE 3 A **status** may take the values: 'WD', 'CD', 'DIS', 'FDIS', 'IS', 'TS', 'PAS', 'ITA'.

name: the identifier of the data specification as defined in the view exchange protocol standards.

date: the year when the data specification reached its **status**.

application: an identifier to characterize an allowed functional subset of the complete data specification as possibly defined in the view exchange protocol standards.

level: an identifier that further characterizes an allowed subset of the **application** subset as possibly defined in the view exchange protocol standards.

7.8 Other structured information elements

This clause describes structured information elements that are used by the other information elements or ontology concept presented in the previous clauses.

7.8.1 Organization representation

An organization is an administrative structure. It is represented by the **ORGANIZATION_Type** XML complex type as illustrated in Figure 94.

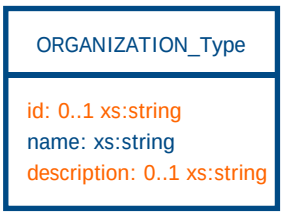

Figure 93 – Organization structure

Internal item definitions:
id: the identifier that distinguishes the organization.

name: the word or group of words by which the organization is referred to.

description: the text that relates the nature of the organization.

7.8.2 Mathematical string

A mathematical string construct provides resources for defining a representation for mathematical strings. It also allows a representation in the MathML format.

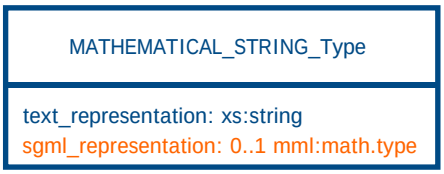

Figure 94 – Mathematical string structure

Internal item definitions:
text_representation: "linear" form of a mathematical string, using ISO 843-2: "Transliteration of Greek characters into Latin characters", if necessary.

sgml_representation: marked up according to the Mathematical Markup Language (MathML) Version 2.0 (Second Edition) specification (document Type Definition).

NOTE 1 MathML specification is available at the following URL: www.w3.org/TR/MathML2.

Internal item definitions:
mml:math.type: the Mathematical Markup Language construct that allows to specify mathematical expressions.

NOTE 2 **mml** specifies the namespace prefix associated to the following MathML namespace: http://www.w3.org/1998/Math/MathML.

8 OntoML exchange structure

OntoML defines a set of ontology-based XML Schemas for exchanging ontology and catalogue data. OntoML is modularized. It is constituted of a set of XML schemas, each of them carrying a part of the CIIM modeling constructs.

Beside the OntoML logical structure presented in the clauses 5, 6 and 7, this clause specifies physical level requirements that shall be fulfilled by any XML document instance that claims to be represented according to OntoML standard.

First, the identifier structure of the OntoML ontology concept is specified. Second, the specific namespace that allows to unambiguously reference resources coming from the OntoML vocabulary is defined. Third, the modular structure of OntoML is described. Finally, OntoML subsets and their corresponding conformance classes are specified.

8.1 Identifiers of ontology concepts

In this part of ISO 13584, global identifiers are used to identify and to reference ontology concepts. Their structure and content are specified in this clause.

OntoML identifiers are built according to the ISO/IEC 11179-5 International Registration Data Identifier (IRDI). In ISO/IEC 11179-5, a global identifier, called an IRDI, is assigned to administered items submitted to registration.

NOTE 2 All ontology concepts and ontology as a whole are identified by an IRDI.

Such an IRDI is threefold:

— a Registration Authority Identifier (**RAI**) intended to identify unambiguously the organization in charge of delivering data identifiers;

— a Data Identifier (**DI**) uniquely defined for a **RAI**, intended to identify an administered item;

— a Version Identifier (**VI**) intended to be used for change management of administered items.

Thus, an OntoML ontology entry identifier is defined according to the following structure:

```
1 ontoml_concept_identifier ::= RAI '#' DI '#' VI
```

NOTE 3 In the scope of this part of ISO 13584, the '#' is used as a separator between the different part of an OntoML ontology entry identifier.

In the next clauses, the structure of each part of these IRDI is presented.

8.1.1 Registration Authority Identifier (RAI) structure

An organization wishing to assign identifiers to OntoML ontology entry shall be itself unambiguously identified. This organization identification shall be defined according to the rules prescribed in ISO/IEC 6523.

ISO/IEC 6523 rules specify a structure for the identification of organizations. It consists of two mandatory parts possibly followed by two optional parts

The mandatory parts are as follows:

— the International Code Designator (**ICD**);

— the identification of an organization within an identification scheme: a data element containing an organization identifier (**OI**);

The optional parts are as follows:

— the identification of an organization part: a data element containing an organization part identifier (**OPI**);

— the **OPI** source indicator (**OPIS**): a data element containing a code value indicating the source of the **OPI**.

When an ontology is described in a standardized document, the ISO/IEC 6523 rules are not sufficient for representing accurately the registration authority identifier. Thus, the standard document number shall be supplied as part of the general registration authority identifier.

The RAI of ontology concepts and ontologies defined in a standard document shall consist of five parts:

— the International Code Designator (**ICD**) 112 where the standardization organization shall be identified;

— the organization identifier of the standardized organisation in the coding scheme 112

NOTE 1 In the coding scheme (**ICD**) 112, the ISO OI equals to 1, the IEC OI equals to 2, the ISO/IEC OI equals to 3.

— the number of the standard (**STDNB**);

— the part number (**STDPART**);

NOTE 2 If the standard part is not part of a multipart series, the part number is represented by the empty string.

— the edition number (**STDED**);

Thus, an OntoML registration authority identifier is defined according to the following syntactical structure:

```
2 RAI ::= ICD '-' OI ('[' OPI ('-' OPIS)? ']')? ('-STD:' STDNB '_' STDPART
'_' STDED)?
```

Where

— the “?” character stands for the optional operator;

— parenthesis are meta-notations characters for defining groups;

— characters between quotes (‘) are terminal characters that may appear in the RAI

— identifiers outside quotes must have the following structure:

  — **ICD**: a string of exactly 4 numeric characters.

NOTE 3 Leading 0 are concatenated to the ICD in case of an ICD length smaller than 4 characters.

- **OI**: a string up to 35 alphanumeric characters.
- **OPI**: a string up to than 35 alphanumeric characters.
- **OPIS**: a single numeric character.
- **STDNB**: a string up to 10 alphanumeric characters.

NOTE 4 For ISO and IEC standards, **STDNB** consists only of digits.

- **STDPART**: a string (that may be empty) up to 10of alphanumeric characters.

NOTE 5 For ISO and IEC standards, **STDPART** consists only of digits.

- **STDED**: a string up to 5 numeric characters.

NOTE 6 For ISO and IEC standards, **STDED** consists only of digits.

EXAMPLE 1 The French “FOO” company is identified (**ORGID**) by the “12345678901234” SIRENE number according to the French codification system for identifying companies (**ICD**= 0002”). The corresponding organization identifier would then be :

0002-12345678901234

EXAMPLE 2 The identifier (**ORGID**) allocated to a big device manufacturer is “123456789”. It is allocated as a DUNS number for identifying companies (**ICD**=”0060”). The particular plant of the manufacturer located in “Town” is also identified (**OPI**) by the following identifier: “12345”. Because this OPI identifier is allocated by the company itself, the **OPIS** is set to 1. The corresponding organization identifier would then be :

0060-123456789[12345-1]

EXAMPLE 3 IEC 61360-4 is a standard that specifies a data dictionary about electronic components. This standard is edited by IEC. IEC is identified (**ORGID**=“2”) identifier according to the ISO register for standards producing organization (**ICD**=”112”). This standard number (**STDNB**) is equal to “61360”, and the specific part (**STDPART**) in which the data dictionary is defined is numbered “4”. If we assume that we want to reference the first edition of this document (**STDED**=”1”), the corresponding organization identifier would then be :

0112-2-STD:61360_4_1

8.1.2 Version Identifier (VI) structure

Each ontology concept is versioned for life cycle management purposes. In order to unambiguously identify a version of each concept, its corresponding version shall also be represented.

The OntoML **VI** is defined as follows:

- **VI** : a string up to 9 alphanumeric characters.

8.1.3 Data Identifier (DI) structure

Each ontology entry defined in a given ontology shall be assigned an identifier called a data identifier. It is defined under the responsibility of an identified registration authority. Therefore, it is assumed that every data identifier assigned to an ontology concept instance shall be unique in the context of this registration authority.

In OntoML, the data identifier is defined as follows:

— the data identifier code (**CODE**);

— the identifier assigned to the context where the data identifier code (**CTXT**) was defined and within which the previous identifier is unique.

EXAMPLE In OntoML, each property is identified in the context of a given class. Both are associated with their own code. The property code is unique in the context of this class.

The OntoML data identifier is defined according to the following syntactical structure:

```
3 DI ::= (CTXT '*')? CODE
```

Where:

— **CTXT**: a string up to 14 alphanumeric characters.

— **CODE** : a string up to 14 alphanumeric characters.

8.1.3.1 DI for classes

In OntoML, a class has no definition context. Thus, its **DI** is equal to its **CODE**.

EXAMPLE ISO 13584-511 specifies an ontology dealing with fasteners. In this ontology, a product characterization class called “hexagon head bolt” is defined. Its code (**CODE**) is equal to “P511AAA156”, and its version (**VERS**) is set to “1”. The corresponding product characterization class identifier would then be :

0112-1-STD:13584_511_1#P511AAA156#1

8.1.3.2 DI for properties, named types and documents

In OntoML, properties, names types and documents are defined in the context of a class. Thus, their **DI** consists of three parts:

— the code of the class;

— a star (‘*’) character;

— the code of the property, named type or document.

EXAMPLE ISO 13584-501 specifies an ontology dealing with measuring instruments. In the “laboratory measuring instrument” product characterization class whose code is “P501_C000131”, a property called “accuracy rating property” is defined. This property is identified (**CODE**) by the “P501_P000178” identifier, and is assigned a version (**VERS**) equal to “000000001”. The “laboratory measuring instrument” product characterization class defines the definition context of the “accuracy rating property” property. The corresponding property identifier would then be :

112-1-STD:13584_501_1#P501_C000131*P501_P000178#000000001

8.1.3.3 DI for ontologies and libraries

OntoML also allows to associate an IRDI globally to an ontology or a library. The ontology or library code may be assigned by the RAI. The only constraint is that this code should be unique over all the classes, ontologies and libraries defined by this RAI. This code constitutes the data identifier (**DI**) of the ontology or library.

EXAMPLE 2 The first version of ontology identified by the “99999” code is defined by a big device manufacturer .This manufacturer is identified by “123456789”. It is allocated as a DUNS number for identifying companies (**ICD**=”0060”). The particular plant of the manufacturer located in “Town” is also identified (**OPI**) by the following identifier: “12345”. Because this OPI identifier is allocated by the company itself, the **OPIS** is set to 1. The corresponding ontology identifier would then be :

0060-123456789[12345-1]#99999#1

8.1.4 OntoML identifier representation

OntoML ontology concepts identifiers are represented using a mandatory XML attribute assigned to the corresponding ontology concept. The name of this attribute is always **id**. Its specific datatype depends on the kind of ontology concept that is intended to be identified. These types are as follows:

— **SupplierId** type for a supplier ontology concept,

— **ClassId** type for a class ontology concept,

— **PropertyId** type for a property ontology concept,

— **DatatypeId** type for a datatype ontology concept,

— **DocumentId** type for a document ontology concept.

Moreover, the identifier type of a dictionary and / or a library is defined as follows:

— OntologyId.

8.2 OntoML namespace

OntoML defines a namespace based on both a URN and a URI specifications.

OntoML processors must use the XML namespaces mechanism to recognize elements and attributes from this namespace. Vendors must not extend the OntoML namespace with additional elements or attributes.

This specification does not use any prefix for referring to elements in the OntoML namespace. However, OntoML documents are free to use any prefix, provided that there is a namespace declaration that binds the prefix to the URI of the OntoML namespace.

8.2.1 OntoML URN

The OntoML namespace has the following URN urn:iso:std:iso:cd:13584:-32.

NOTE The URN is defined according to the ISO URN scheme for resources defined in ISO TC184/SC4 standards.

8.2.2 OntoML URI

The OntoML namespace has the following URI http://www.plib.ensma.fr/2006/OntoML.

NOTE The “2006“ substring in the URI indicates the year in which the URI was allocated. It does not indicate the version of the OntoML module being used, which is specified by attributes.

8.3 Modular structure

OntoML is a set of XML schemas, called modules, that define general resources on which a top level schema is built. The list of modules is given hereafter:

— **ontoml.xsd**: the main XML schema. It defines the upper level of the OntoML ontology exchange format. It is based on the definition of a single **ontoml** XML element whose content is a mandatory header, followed by an optional dictionary specification, followed by an optional content specification.

NOTE An OntoML XML document must contain either a dictionary specification, or a library content specification, or both.

— **header.xsd**: this module contains management resources for characterizing the content (dates, authors …) of an OntoML XML file.

— **dictionary.xsd**: this module defines the container of any CIIM compliant ontology.

— **supplier.xsd**: this module contains the resources intended to represent an information source according to the structure defined in clause 5.7.1.

— **class.xsd**: this module contains the resources intended to represent an ontology class according to the structure defined in clause 5.7.2, whatever be its nature (a general model class, a functional model class or a functional view class).

— **property.xsd**: this module contains the resources intended to represent a property according to the structure defined in clause 5.7.4, whatever be its nature (a characteristic property, a context property, a context dependent property or a representation property).

— **datatype.xsd**: this module contains the constructs intended to represent data types according to the structure defined in clause 5.7.6.

— **document.xsd**: this module contains the constructs intended to represent a document according to the structure defined in clause 5.7.7.

— **datatypeSystem.xsd**: this module contains an XML representation of the complete OntoML type system presented in clause 7.3.

— **translations.xsd**: this module contains resources for defining multilingual representations of clear text as presented in clause 7.1.

— **externalFiles.xsd**: this module contains constructs for referencing external resources that may be either exchanged together with the OntoML document instance or referenced in the Internet; they are presented in clause 7.2.

— **units.xsd**: this module contains constructs for explicitly describing any kind of units according to the structure defined in clause 7.4.

— **aPosteriori.xsd**: this module contains constructs for representing a posteriori mappings between classes and properties defined in different ontologies according to the structure defined in clause 7.6.

— **constraints.xsd**: this module contains constructs intended to represent constrained properties according to the structure defined in clause 7.5.

— **baseTypes.xsd**: this module contains all the OntoML simple an complex type definitions shared by OntoML modules.

— **identifier.xsd**: this module provides the constructs needed for identifying and referencing CIIM ontology concepts as presented in clause 8.1;

— **content.xsd**: this module defines specifies the structure of families of products belonging to a library;

— **library.xsd**: this module defines structure for representing libraries of families of products, by their product characterizations possibly associated with the ontology where product characterizations classes and properties are defined.

Table 1 illustrates the reference relationships between these modules where gray squares specify the relationship from one module to another module.

Table 1 – OntoML modules cross-references

references →	baseTypes.xsd	identifiers.xsd	translations.xsd	units.xsd	externalFiles.xsd	header.xsd	supplier.xsd	dataTypeSystem.xsd	datatype.xsd	constraints.xsd	property.xsd	class.xsd	document.xsd	aPosteriori.xsd	dictionary.xsd	content.xsd	library.xsd	ontoml.xsd
baseTypes.xsd	■																	
identifiers.xsd		■																
translations.xsd	▒		■															
units.xsd	▒			■														
externalFiles.xsd	▒	▒		▒	■													
header.xsd		▒	▒		▒	■												
supplier.xsd	▒	▒					■											
dataTypeSystem.xsd	▒	▒	▒	▒	▒			■										
dataType.xsd	▒	▒	▒		▒			▒	■									
constraints.xsd	▒	▒						▒		■								
property.xsd	▒	▒	▒		▒			▒			■							
class.xsd	▒	▒	▒		▒					▒		■						
document.xsd	▒	▒	▒		▒								■					
aPosteriori.xsd		▒												■				
dictionary.xsd	▒						▒		▒		▒	▒	▒	▒	■			
content.xsd	▒	▒		▒	▒											■		
library.xsd																▒	■	
ontoml.xsd						▒									▒		▒	■

8.4 Levels of exchange and conformance classes

OntoML integrates a set of resource constructs into a single XML schema for representing ontologies and possibly supplier libraries for the purpose of exchange. In order to support various levels of exchange requirements, four functional subsets of OntoML have been identified as allowed levels of OntoML exchange. These various functional subsets are called conformance classes.

These conformance classes also specify allowed levels of OntoML implementation for those systems that claim conformance to the OntoML international standard.

The four conformance classes of OntoML are defined as follows:

— simple ontology: an ontology that define hierarchies of classes of items on the base of the common ISO/IEC dictionary schema together with the required external resources, documents, named types and collection data types, but without description of item representations and of

representation categories of items(i.e., functional view classes), corresponds to OntoML conformance class 1;

— advanced ontology: an ontology that corresponds to a simple ontology with the addition of resources for defining hierarchies of item representations (i.e., functional model classes) and of representation categories of items (i.e., functional view classes), corresponds to conformance class 2;

— simple library: a library content defining products on the basis of simple ontologies, possibly exchanged together with simple ontology definitions correspond to conformance class 3;

— advanced library: a library content defining products and product representations on the basis of advanced ontologies, possibly exchanged together with advanced ontology definitions corresponds to conformance class 4;

Table 2 summarizes the supported capabilities of the different conformance classes of OntoML.

Table 2 – Conformance options of OntoML

	Ontology		**Library**	
Conformance Class	Common ISO/IEC dictionary definitions Named type and documents Collections External resources	Representations Representation categories	Library of products	Library of representations
1	X			
2	X	X		
3	X		X	
4	X	X	X	X

Annex C defines the standard data that allow to precisely identify one conformance class among all the available conformance classes.

8.5 Conformance class requirements

8.5.1 Conformance class 1

Conformance class 1 addresses those implementations that support ontologies that define hierarchies of classes of items on the base of the common ISO/IEC dictionary schema together with the required external resources, document ontology concept, named type ontology concepts and collection data types. It shall support standardized data defined in Annex C. It shall support the following XML complex types:

— From *ontoml.xsd*:

  — **ONTOML_Type**

— From *header.xsd*:

  - **HEADER_Type**
  - **INFORMATION_Type**
  - **LIBRARY_IIM_IDENTIFICATION_Type**
- From *dictionary.xsd*:
  - **A_POSTERIORI_SEMANTIC_RELATIONSHIPS_Type**
  - **CONTAINED_CLASSES_Type**
  - **CONTAINED_DOCUMENTS_Type**
  - **CONTAINED_NAMED_TYPES_Type**
  - **CONTAINED_PROPERTIES_Type**
  - **CONTAINED_SUPPLIERS_Type**
  - **DICTIONARY_IN_STANDARD_FORMAT_Type**
  - **DICTIONARY_Type**
- From *supplier.xsd*:
  - **SUPPLIER_Type**
- From *class.xsd*:
  - **CLASS_CONSTANT_VALUES_Type**
  - **CLASS_Type**
  - **CLASS_VALUE_ASSIGNMENT_Type**
  - **CATEGORIZATION_CLASS_Type**
  - **ITEM_CLASS_CASE_OF_Type**
  - **ITEM_CLASS_Type**
- From *property.xsd*:
  - **PROPERTY_Type**
  - **NON_DEPENDENT_P_DET_Type**
  - **CONDITION_DET_Type**
  - **DEPENDENT_P_DET_Type**
  - **SYNONYMOUS_SYMBOLS_Type**
- From *datatype.xsd*:

- **DATATYPE_Type**

- From *datatypeSystem.xsd*:
  - **ALTERNATIVE_UNITS_Type**
  - **ARRAY_TYPE_Type**
  - **BAG_TYPE_Type**
  - **BOOLEAN_TYPE_Type**
  - **CLASS_INSTANCE_TYPE_Type**
  - **DATA_TYPE_Type**
  - **DIC_VALUE_Type**
  - **INT_CURRENCY_TYPE_Type**
  - **INT_MEASURE_TYPE_Type**
  - **INT_TYPE_Type**
  - **INT_DIC_VALUE_Type**
  - **ITS_VALUES_Type**
  - **LEVEL_Type**
  - **LEVEL_TYPE_Type**
  - **LIST_TYPE_Type**
  - **NAMED_TYPE_Type**
  - **NON_QUANTITATIVE_CODE_TYPE_Type**
  - **NON_QUANTITATIVE_INT_TYPE_Type**
  - **NON_TRANSLATABLE_STRING_TYPE_Type**
  - **NUMBER_TYPE_Type**
  - **REAL_CURRENCY_TYPE_Type**
  - **REAL_MEASURE_TYPE_Type**
  - **REAL_TYPE_Type**
  - **REMOTE_HTTP_ADDRESS_Type**
  - **SET_TYPE_Type**
  - **SET_WITH_SUBSET_CONSTRAINT_TYPE_Type**
  - **STRING_DIC_VALUE_Type**

- **STRING_TYPE_Type**
- **TRANSLATABLE_STRING_TYPE_Type**
- **TYPE_NAME_Type**
- **VALUE_TYPE_Type**

— From *translations.xsd*:

- **GENERAL_TEXT_Type**
- **PREFERRED_NAME_LABEL_Type**
- **PREFERRED_NAME_Type**
- **SHORT_NAME_LABEL_Type**
- **SHORT_NAME_Type**
- **SYNONYMOUS_NAME_LABEL_Type**
- **SYNONYMOUS_NAME_TYPE_Type**
- **TEXT_Type**
- **TRANSLATION_DATA_Type**
- **TRANSLATION_Type**

— From *externalFiles.xsd*:

- **EXTERNAL_GRAPHICS_Type**
- **EXTERNAL_RESOURCE_Type**
- **GRAPHIC_FILES_Type**
- **GRAPHICS_Type**
- **HTTP_FILE_Type**
- **IDENTIFIED_DOCUMENT_Type**
- **REFERENCED_DOCUMENT_Type**
- **REFERENCED_GRAPHICS_Type**
- **SOURCE_DOCUMENT_Type**

— From *units.xsd*:

- **CONTEXT_DEPENDENT_UNIT_Type**
- **CONVERSION_BASED_UNIT_Type**

- **DERIVED_UNIT_ELEMENT_Type**
- **DERIVED_UNIT_Type**
- **DIC_UNIT_Type**
- **DIMENSIONAL_EXPONENTS_Type**
- **NAMED_UNIT_Type**
- **NON_SI_UNIT_Type**
- **SI_UNIT_Type**
- **UNIT_Type**
- **UNITS_Type**

- From *constraints.xsd*:
  - **CONSTRAINT_Type**
  - **CONSTRAINTS_Type**
  - **CONTEXT_RESTRICTION_CONSTRAINT_Type**
  - **DOMAIN_CONSTRAINT_Type**
  - **PROPERTY_TYPE_REDEFINITION_Type**
  - **RANGE_CONSTRAINT_Type**
  - **STRING_PATTERN_CONSTRAINT_Type**
  - **STRING_SIZE_CONSTRAINT_Type**
  - **SUBCLASS_CONSTRAINT_Type**
  - **SUBSET_CONSTRAINT_Type**
  - **SUBSET_Type**
- From *baseTypes.xsd*:
  - **MATHEMATICAL_STRING_Type**
  - **ORGANIZATION_Type**
- From *document.xsd*:
  - **AUTHORS_Type**
  - **DOCUMENT_CONTENT_Type**
  - **DOCUMENT_Type**
  - **PERSON_Type**

- **REMOTE_LOCATIONS_Type**
- **STRINGS_Type**

— From *aPosteriori.xsd*:

- **A_POSTERIORI_CASE_OF_Type**
- **A_POSTERIORI_SEMANTIC_RELATIONSHIP_Type**
- **CORRESPONDING_PROPERTIES_type**
- **MAPPING_FUNCTION_Type**
- **PROPERTY_MAPPING_Type**

8.5.2 Conformance class 2

Conformance class 2 addresses those implementations that support ontologies that define hierarchies of classes of items on the base of the common ISO/IEC dictionary schema together with the required external resources, document ontology concept, named type ontology concepts collection data types, description of hierarchy of item representations and of representation categories of items. It shall support standardized data defined in Annex C. It shall also supports complex types and related constructs defined for conformance class 1, more the following complex types and related constructs:

— From *header.xsd*:

- **SUPPORTED_VEP_Type**
- **VIEW_EXCHANGE_PROTOCOL_IDENTIFICATION_Type**

— From *class.xsd*:

- **FM_CLASS_VIEW_OF_Type**
- **FUNCTIONAL_MODEL_CLASS_Type**
- **FUNCTIONAL_VIEW_CLASS_Type**
- **NON_INSTANTIABLE_FUNCTIONAL_VIEW_CLASS_Type**
- **REPRESENTATION_CONTEXT_Type**
- **V_C_V_RANGE_Type**
- **VIEW_CONTROL_VARIABLE_RANGE_Type**

— From *property.xsd*:

- **REPRESENTATION_P_DET_Type**

— From *datatypeSystem.xsd*:

- **AXIS1_PLACEMENT_TYPE_Type**
- **AXIS2_PLACEMENT_3D_TYPE_Type**

- **AXIS2_PLACEMENT_2D_TYPE_Type**
- **PLACEMENT_TYPE_Type**
- **REPRESENTATION_REFERENCE_TYPE_Type**
- **PROGRAM_REFERENCE_TYPE_Type**
- **GEOMETRIC_REPRESENTATION_CONTEXT_TYPE_Type**

- From *aPosteriori.xsd*:
  - **A_POSTERIORI_VIEW_OF_Type**

8.5.3 Conformance class 3

Conformance class 3 addresses those implementations that support exchange of products with dictionary definitions. It shall support standardized data defined in Annex C. It shall also supports complex types and related constructs defined for conformance class 1, more the following complex types and related constructs:

- From *externalFiles.xsd*:
  - **ILLUSTRATION_Type**
  - **MESSAGE_Type**
- From *externalFiles.xsd*:
  - **PROGRAM_REFERENCE_Type**
  - **REPRESENTATION_REFERENCE_Type**
- From *library.xsd*:
  - **CONTAINED_CLASS_EXTENSIONS_Type**
  - **LIBRARY_Type**
  - **LIBRARY_IN_STANDARD_FORMAT_Type**
- From *content.xsd*:
  - **CLASS_EXTENSION_Type**
  - **CLASS_PRESENTATION_ON_PAPER_Type**
  - **CLASS_PRESENTATION_ON_SCREEN_Type**
  - **CLASSIFICATION_Type**
  - **CREATE_ICON_Type**
  - **EXPLICIT_ITEM_CLASS_EXTENSION_Type**
  - **PROPERTY_VALUE_RECOMMENDED_PRESENTATION_Type**

- **PROPERTY_CLASSIFICATION_Type**
- **RECOMMENDED_PRESENTATION_Type**
- **cat:Catalogue**

NOTE **cat:Catalogue** is defined outside OntoML; The *cat* prefix stands for the namespace identified by the following URN: *urn:pcd:schema:product-characterization*.

8.5.4 Conformance class 4

Conformance class 4 addresses those implementations that support exchange of products and engineering models with dictionary definitions. It shall support standardized data defined in Annex C. It shall also supports complex types and related constructs defined for conformance class 3.

- From *content.xsd*:
  - **CONTEXT_PARAM_ICON_Type**
  - **EXPLICIT_FUNCTIONAL_MODEL_CLASS_EXTENSION_Type**
- From *externalFiles.xsd*:
  - **PROGRAM_REFERENCE_Type**
  - **REPRESENTATION_REFERENCE_Type**

Annex A
(normative)
Information object registration

A.1 Document identification

In order to provide for unambiguous identification of an information object in an open system, the object identifier

{ iso standard 13584 part (32) version (1) }

is assigned to this part of ISO 13584. The meaning of this value is defined in ISO 8824-1.

Annex B (normative)
The OntoML XML Schema

This annex contains the complete OntoML XML Schema. This schema is not documented in this annex. Nevertheless, a documented version may be downloaded at the following URL:

http://www.plib.ensma.fr/ontoml/ISO_CD_13584_32.zip.

Annex B lists the OntoML resource files as they are described in clause 8.3.

B.1 ontoml.xsd

It defines the upper level of the OntoML CIIM ontology exchange format. It is based on the definition of a single **ontoml** XML element whose content is an optional header, followed by an optional dictionary description, followed by an optional content specification.

```
<xs:schema xmlns="urn:iso:std:iso:cd:13584:-32" xmlns:cat="urn:pcd:schema:product-characterization"
xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="urn:iso:std:iso:cd:13584:-32"
elementFormDefault="unqualified" attributeFormDefault="unqualified">
	<xs:include schemaLocation="header.xsd"/>
	<xs:include schemaLocation="dictionary.xsd"/>
	<xs:include schemaLocation="library.xsd"/>
	<xs:element name="ontoml" type="ONTOML_Type"/>
	<xs:complexType name="ONTOML_Type">
		<xs:sequence>
			<xs:element name="header" type="HEADER_Type"/>
			<xs:element name="dictionary" type="DICTIONARY_Type" minOccurs="0"/>
			<xs:element name="library" type="LIBRARY_Type" minOccurs="0"/>
		</xs:sequence>
	</xs:complexType>
</xs:schema>
```

B.2 header.xsd

This module contains resources for characterizing the content (dates, authors …) of en OntoML XML file.

```
<xs:schema xmlns="urn:iso:std:iso:cd:13584:-32" xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:iso:std:iso:cd:13584:-32" elementFormDefault="unqualified"
attributeFormDefault="unqualified">
	<xs:include schemaLocation="translations.xsd"/>
	<xs:include schemaLocation="externalFiles.xsd"/>
	<xs:complexType name="HEADER_Type" abstract="false">
		<xs:sequence>
			<xs:element name="global_language" type="LANGUAGE_Type" minOccurs="0"/>
			<xs:element name="description" type="xs:string"/>
			<xs:element name="version" type="xs:string"/>
```

```
                        <xs:element name="name" type="xs:string"/>
                        <xs:element name="date_time_stamp" type="xs:dateTime"/>
                        <xs:element name="author" type="xs:string" maxOccurs="unbounded"/>
                        <xs:element name="organisation" type="xs:string" maxOccurs="unbounded"/>
                        <xs:element name="pre_processor_version" type="xs:string"/>
                        <xs:element name="originating_system" type="xs:string"/>
                        <xs:element name="authorisation" type="xs:string"/>
                        <xs:element name="ontoml_information" type="INFORMATION_Type"/>
                        <xs:element name="ontoml_structure" type="LIBRARY_IIM_IDENTIFICATION_Type"/>
                        <xs:element name="supported_vep" type="SUPPORTED_VEP_Type" minOccurs="0"/>
                </xs:sequence>
                <xs:attribute name="id" type="OntologyId" use="optional"/>
        </xs:complexType>
        <xs:complexType name="INFORMATION_Type">
                <xs:sequence>
                        <xs:element name="revision" type="REVISION_TYPE_Type"/>
                        <xs:element name="preferred_name" type="PREFERRED_NAME_Type"/>
                        <xs:element name="synonymous_names" type="SYNONYMOUS_NAME_Type"
minOccurs="0"/>
                        <xs:element name="short_name" type="SHORT_NAME_Type" minOccurs="0"/>
                        <xs:element name="icon" type="GRAPHICS_Type" minOccurs="0"/>
                        <xs:element name="note" type="TEXT_Type" minOccurs="0"/>
                        <xs:element name="remark" type="TEXT_Type" minOccurs="0"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="LIBRARY_IIM_IDENTIFICATION_Type" abstract="false">
                <xs:sequence>
                        <xs:element name="source_document_identifier" type="xs:string" minOccurs="0"/>
                        <xs:element name="status" type="xs:string"/>
                        <xs:element name="name" type="xs:string"/>
                        <xs:element name="date" type="xs:integer"/>
                        <xs:element name="application" type="xs:string" minOccurs="0"/>
                        <xs:element name="level" type="xs:string" minOccurs="0"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="SUPPORTED_VEP_Type">
                <xs:sequence>
                        <xs:element name="view_exchange_protocol_identification"
type="VIEW_EXCHANGE_PROTOCOL_IDENTIFICATION_Type" maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="VIEW_EXCHANGE_PROTOCOL_IDENTIFICATION_Type" abstract="false">
                <xs:sequence>
                        <xs:element name="source_document_identifier" type="xs:string" minOccurs="0"/>
                        <xs:element name="status" type="xs:string"/>
                        <xs:element name="name" type="xs:string"/>
                        <xs:element name="date" type="xs:integer"/>
                        <xs:element name="application" type="xs:string" minOccurs="0"/>
                        <xs:element name="level" type="xs:string" minOccurs="0"/>
                </xs:sequence>
        </xs:complexType>
</xs:schema>
```

B.3 dictionary.xsd

This module defines the container of any ISO 13584 compliant ontology.

```
<xs:schema xmlns="urn:iso:std:iso:cd:13584:-32" xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:iso:std:iso:cd:13584:-32" elementFormDefault="unqualified"
attributeFormDefault="unqualified">
        <xs:include schemaLocation="baseTypes.xsd"/>
        <xs:include schemaLocation="supplier.xsd"/>
        <xs:include schemaLocation="class.xsd"/>
        <xs:include schemaLocation="property.xsd"/>
```

```
	<xs:include schemaLocation="dataType.xsd"/>
	<xs:include schemaLocation="document.xsd"/>
	<xs:include schemaLocation="aPosteriori.xsd"/>
	<xs:complexType name="A_POSTERIORI_SEMANTIC_RELATIONSHIPS_Type">
		<xs:sequence>
			<xs:element name="a_posteriori_semantic_relationship"
type="A_POSTERIORI_SEMANTIC_RELATIONSHIP_Type" maxOccurs="unbounded"/>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="CONTAINED_CLASSES_Type">
		<xs:sequence>
			<xs:element name="class" type="CLASS_Type" maxOccurs="unbounded"/>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="CONTAINED_DOCUMENTS_Type">
		<xs:sequence>
			<xs:element name="document" type="DOCUMENT_Type" maxOccurs="unbounded"/>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="CONTAINED_NAMED_TYPES_Type">
		<xs:sequence>
			<xs:element name="data_type" type="DATATYPE_Type" maxOccurs="unbounded"/>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="CONTAINED_PROPERTIES_Type">
		<xs:sequence>
			<xs:element name="property" type="PROPERTY_Type" maxOccurs="unbounded"/>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="CONTAINED_SUPPLIERS_Type">
		<xs:sequence>
			<xs:element name="supplier" type="SUPPLIER_Type" maxOccurs="unbounded"/>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="DICTIONARY_IN_STANDARD_FORMAT_Type">
		<xs:complexContent>
			<xs:extension base="DICTIONARY_Type"/>
		</xs:complexContent>
	</xs:complexType>
	<xs:complexType name="DICTIONARY_Type" abstract="false">
		<xs:sequence>
			<xs:element name="is_complete" type="xs:boolean" minOccurs="0"/>
			<xs:element name="updates" type="DICTIONARY_REFERENCE_Type" minOccurs="0"/>
			<xs:element name="update_agreement" type="xs:string" minOccurs="0"/>
			<xs:element name="referenced_dictionaries" type="DICTIONARIES_REFERENCE_Type"
minOccurs="0"/>
			<xs:element name="responsible_supplier" type="SUPPLIER_REFERENCE_Type"
minOccurs="0"/>
			<xs:element name="contained_classes" type="CONTAINED_CLASSES_Type"/>
			<xs:element name="a_posteriori_semantic_relationships"
type="A_POSTERIORI_SEMANTIC_RELATIONSHIPS_Type" minOccurs="0"/>
			<xs:element name="contained_suppliers" type="CONTAINED_SUPPLIERS_Type"/>
			<xs:element name="contained_properties" type="CONTAINED_PROPERTIES_Type"
minOccurs="0"/>
			<xs:element name="contained_documents" type="CONTAINED_DOCUMENTS_Type"
minOccurs="0"/>
			<xs:element name="contained_named_types" type="CONTAINED_NAMED_TYPES_Type"
minOccurs="0"/>
		</xs:sequence>
	</xs:complexType>
```

```
</xs:schema>
```

B.4 supplier.xsd

This module contains the resources intended to represent a supplier according to ISO 13584.

```
<xs:schema xmlns="urn:iso:std:iso:cd:13584:-32" xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:iso:std:iso:cd:13584:-32" elementFormDefault="unqualified"
attributeFormDefault="unqualified">
	<xs:include schemaLocation="baseTypes.xsd"/>
	<xs:include schemaLocation="identifiers.xsd"/>
	<xs:include schemaLocation="translations.xsd"/>
	<xs:simpleType name="SUPPLIER_CODE_TYPE_Type">
		<xs:restriction base="xs:string">
			<xs:maxLength value="70"/>
			<xs:pattern value="[^\-]*"/>
			<xs:pattern value="[^\s]*"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:complexType name="SUPPLIER_Type" abstract="false">
		<xs:sequence>
			<xs:element name="date_of_original_definition" type="DATE_TYPE_Type"
minOccurs="0"/>
			<xs:element name="date_of_current_version" type="DATE_TYPE_Type" minOccurs="0"/>
			<xs:element name="date_of_current_revision" type="DATE_TYPE_Type" minOccurs="0"/>
			<xs:element name="revision" type="REVISION_TYPE_Type"/>
			<xs:element name="status" type="STATUS_Type" minOccurs="0"/>
			<xs:element name="is_deprecated" type="xs:boolean" minOccurs="0"/>
			<xs:element name="org" type="ORGANIZATION_Type"/>
			<xs:element name="internal_location" type="xs:string" minOccurs="0"/>
			<xs:element name="street_number" type="xs:string" minOccurs="0"/>
			<xs:element name="street" type="xs:string" minOccurs="0"/>
			<xs:element name="postal_box" type="xs:string" minOccurs="0"/>
			<xs:element name="town" type="xs:string" minOccurs="0"/>
			<xs:element name="region" type="xs:string" minOccurs="0"/>
			<xs:element name="postal_code" type="xs:string" minOccurs="0"/>
			<xs:element name="country" type="xs:string" minOccurs="0"/>
			<xs:element name="facsimile_number" type="xs:string" minOccurs="0"/>
			<xs:element name="telephone_number" type="xs:string" minOccurs="0"/>
			<xs:element name="electronic_mail_address" type="xs:string" minOccurs="0"/>
			<xs:element name="telex_number" type="xs:string" minOccurs="0"/>
		</xs:sequence>
		<xs:attribute name="id" type="SupplierId" use="required"/>
	</xs:complexType>
</xs:schema>
```

B.5 class.xsd

This module contains the resources intended to represent a class according to ISO 13584, whatever be its nature: a general model class, a functional model class or a functional view class.

```
<xs:schema xmlns="urn:iso:std:iso:cd:13584:-32" xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:iso:std:iso:cd:13584:-32" elementFormDefault="unqualified"
attributeFormDefault="unqualified">
	<xs:include schemaLocation="identifiers.xsd"/>
	<xs:include schemaLocation="baseTypes.xsd"/>
	<xs:include schemaLocation="translations.xsd"/>
	<xs:include schemaLocation="externalFiles.xsd"/>
	<xs:include schemaLocation="constraints.xsd"/>
```

```
	<xs:simpleType name="REPRESENTATION_Type">
		<xs:restriction base="xs:string">
			<xs:pattern value=".+_SCHEMA"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:complexType name="CLASS_CONSTANT_VALUES_Type">
		<xs:sequence>
			<xs:element name="class_value_assignment" type="CLASS_VALUE_ASSIGNMENT_Type"
maxOccurs="unbounded"/>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="CLASS_Type" abstract="true">
		<xs:sequence>
			<xs:element name="date_of_original_definition" type="DATE_TYPE_Type"
minOccurs="0"/>
			<xs:element name="date_of_current_version" type="DATE_TYPE_Type" minOccurs="0"/>
			<xs:element name="date_of_current_revision" type="DATE_TYPE_Type" minOccurs="0"/>
			<xs:element name="revision" type="REVISION_TYPE_Type"/>
			<xs:element name="status" type="STATUS_Type" minOccurs="0"/>
			<xs:element name="translation" type="TRANSLATION_Type" minOccurs="0"/>
			<xs:element name="source_language" type="LANGUAGE_Type" minOccurs="0"/>
			<xs:element name="is_deprecated" type="xs:boolean" minOccurs="0"/>
			<xs:element name="preferred_name" type="PREFERRED_NAME_Type"/>
			<xs:element name="synonymous_names" type="SYNONYMOUS_NAME_Type"
minOccurs="0"/>
			<xs:element name="short_name" type="SHORT_NAME_Type" minOccurs="0"/>
			<xs:element name="icon" type="GRAPHICS_Type" minOccurs="0"/>
			<xs:element name="definition" type="TEXT_Type"/>
			<xs:element name="source_doc_of_definition" type="SOURCE_DOCUMENT_Type"
minOccurs="0"/>
			<xs:element name="note" type="TEXT_Type" minOccurs="0"/>
			<xs:element name="remark" type="TEXT_Type" minOccurs="0"/>
			<xs:element name="its_superclass" type="CLASS_REFERENCE_Type" minOccurs="0"/>
			<xs:element name="described_by" type="PROPERTIES_REFERENCE_Type"
minOccurs="0">
				<xs:unique name="described_by_Unique">
					<xs:selector xpath="property"/>
					<xs:field xpath="@property_ref"/>
				</xs:unique>
			</xs:element>
			<xs:element name="defined_types" type="DATATYPES_REFERENCE_Type"
minOccurs="0">
				<xs:unique name="defined_types_Unique">
					<xs:selector xpath="data_type"/>
					<xs:field xpath="@datatype_ref"/>
				</xs:unique>
			</xs:element>
			<xs:element name="constraints" type="CONSTRAINTS_Type" minOccurs="0"/>
		</xs:sequence>
		<xs:attribute name="id" type="ClassId" use="required"/>
	</xs:complexType>
	<xs:complexType name="CLASS_VALUE_ASSIGNMENT_Type" abstract="false">
		<xs:sequence>
			<xs:element name="super_class_defined_property"
type="PROPERTY_REFERENCE_Type"/>
			<xs:element name="assigned_value" type="VALUE_CODE_TYPE_Type"/>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="CATEGORIZATION_CLASS_Type">
		<xs:complexContent>
			<xs:extension base="CLASS_Type">
				<xs:sequence>
					<xs:element name="categorization_class_superclasses"
type="CLASSES_REFERENCE_Type" minOccurs="0">
						<xs:unique name="categorization_class_superclasses_Unique">
							<xs:selector xpath="class"/>
							<xs:field xpath="@class_ref"/>
```

```
                                        </xs:unique>
                                    </xs:element>
                                </xs:sequence>
                            </xs:extension>
                        </xs:complexContent>
                    </xs:complexType>
                    <xs:complexType name="FM_CLASS_VIEW_OF_Type">
                        <xs:complexContent>
                            <xs:extension base="CLASS_Type">
                                <xs:sequence>
                                    <xs:element name="created_view" type="CLASS_REFERENCE_Type"/>
                                    <xs:element name="v_c_v_range" type="V_C_V_RANGE_Type"
minOccurs="0"/>
                                    <xs:element name="imported_properties_from_view"
type="PROPERTIES_REFERENCE_Type" minOccurs="0"/>
                                    <xs:element name="imported_types_from_view"
type="DATATYPES_REFERENCE_Type" minOccurs="0"/>
                                    <xs:element name="imported_documents_from_view"
type="DOCUMENTS_REFERENCE_Type" minOccurs="0"/>
                                    <xs:element name="case_of" type="CLASSES_REFERENCE_Type"
minOccurs="0"/>
                                    <xs:element name="imported_properties_from_models"
type="PROPERTIES_REFERENCE_Type" minOccurs="0"/>
                                    <xs:element name="imported_types_from_models"
type="DATATYPES_REFERENCE_Type" minOccurs="0"/>
                                    <xs:element name="imported_documents_from_models"
type="DOCUMENTS_REFERENCE_Type" minOccurs="0"/>
                                    <xs:element name="view_of" type="CLASS_REFERENCE_Type"/>
                                    <xs:element name="imported_properties_from_item"
type="PROPERTIES_REFERENCE_Type" minOccurs="0"/>
                                    <xs:element name="imported_types_from_item"
type="DATATYPES_REFERENCE_Type" minOccurs="0"/>
                                    <xs:element name="imported_documents_from_item"
type="DOCUMENTS_REFERENCE_Type" minOccurs="0"/>
                                </xs:sequence>
                            </xs:extension>
                        </xs:complexContent>
                    </xs:complexType>
                    <xs:complexType name="FUNCTIONAL_MODEL_CLASS_Type">
                        <xs:complexContent>
                            <xs:extension base="CLASS_Type">
                                <xs:sequence>
                                    <xs:element name="created_view" type="CLASS_REFERENCE_Type"/>
                                    <xs:element name="v_c_v_range" type="V_C_V_RANGE_Type"
minOccurs="0"/>
                                    <xs:element name="imported_properties_from_view"
type="PROPERTIES_REFERENCE_Type" minOccurs="0"/>
                                    <xs:element name="imported_types_from_view"
type="DATATYPES_REFERENCE_Type" minOccurs="0"/>
                                    <xs:element name="imported_documents_from_view"
type="DOCUMENTS_REFERENCE_Type" minOccurs="0"/>
                                    <xs:element name="case_of" type="CLASSES_REFERENCE_Type"
minOccurs="0"/>
                                    <xs:element name="imported_properties_from_models"
type="PROPERTIES_REFERENCE_Type" minOccurs="0"/>
                                    <xs:element name="imported_types_from_models"
type="DATATYPES_REFERENCE_Type" minOccurs="0"/>
                                    <xs:element name="imported_documents_from_models"
type="DOCUMENTS_REFERENCE_Type" minOccurs="0"/>
                                </xs:sequence>
                            </xs:extension>
                        </xs:complexContent>
                    </xs:complexType>
                    <xs:complexType name="FUNCTIONAL_VIEW_CLASS_Type">
                        <xs:complexContent>
                            <xs:extension base="CLASS_Type">
```

```
                        <xs:sequence>
                            <xs:element name="representation_type" type="REPRESENTATION_Type"
minOccurs="0"/>
                            <xs:element name="view_control_variables"
type="PROPERTIES_REFERENCE_Type" minOccurs="0">
                                <xs:unique name="view_control_variables_1_Unique">
                                    <xs:selector xpath="property"/>
                                    <xs:field xpath="@property_ref"/>
                                </xs:unique>
                            </xs:element>
                            <xs:element name="view_properties"
type="PROPERTIES_REFERENCE_Type" minOccurs="0">
                                <xs:unique name="view_properties_1_Unique">
                                    <xs:selector xpath="property"/>
                                    <xs:field xpath="@property_ref"/>
                                </xs:unique>
                            </xs:element>
                        </xs:sequence>
                    </xs:extension>
                </xs:complexContent>
            </xs:complexType>
            <xs:complexType name="ITEM_CLASS_CASE_OF_Type">
                <xs:complexContent>
                    <xs:extension base="CLASS_Type">
                        <xs:sequence>
                            <xs:element name="simplified_drawing" type="GRAPHICS_Type"
minOccurs="0"/>
                            <xs:element name="sub_class_properties"
type="PROPERTIES_REFERENCE_Type" minOccurs="0"/>
                            <xs:element name="class_constant_values"
type="CLASS_CONSTANT_VALUES_Type" minOccurs="0"/>
                            <xs:element name="coded_name" type="VALUE_CODE_TYPE_Type"
minOccurs="0"/>
                            <xs:element name="is_embedded" type="xs:boolean" minOccurs="0"/>
                            <xs:element name="is_case_of" type="CLASSES_REFERENCE_Type"/>
                            <xs:element name="imported_properties"
type="PROPERTIES_REFERENCE_Type" minOccurs="0"/>
                            <xs:element name="imported_types"
type="DATATYPES_REFERENCE_Type" minOccurs="0"/>
                            <xs:element name="imported_documents"
type="DOCUMENTS_REFERENCE_Type" minOccurs="0"/>
                        </xs:sequence>
                    </xs:extension>
                </xs:complexContent>
            </xs:complexType>
            <xs:complexType name="ITEM_CLASS_Type">
                <xs:complexContent>
                    <xs:extension base="CLASS_Type">
                        <xs:sequence>
                            <xs:element name="simplified_drawing" type="GRAPHICS_Type"
minOccurs="0"/>
                            <xs:element name="sub_class_properties"
type="PROPERTIES_REFERENCE_Type" minOccurs="0"/>
                            <xs:element name="class_constant_values"
type="CLASS_CONSTANT_VALUES_Type" minOccurs="0"/>
                            <xs:element name="coded_name" type="VALUE_CODE_TYPE_Type"
minOccurs="0"/>
                            <xs:element name="is_embedded" type="xs:boolean" minOccurs="0"/>
                        </xs:sequence>
                    </xs:extension>
                </xs:complexContent>
            </xs:complexType>
            <xs:complexType name="NON_INSTANTIABLE_FUNCTIONAL_VIEW_CLASS_Type">
                <xs:complexContent>
                    <xs:extension base="CLASS_Type">
                        <xs:sequence>
```

```
                                        <xs:element name="view_control_variables"
type="PROPERTIES_REFERENCE_Type" minOccurs="0">
                                                <xs:unique name="view_control_variables_Unique">
                                                        <xs:selector xpath="property"/>
                                                        <xs:field xpath="@property_ref"/>
                                                </xs:unique>
                                        </xs:element>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="REPRESENTATION_CONTEXT_Type" abstract="false">
                <xs:sequence>
                        <xs:element name="context_identifier" type="xs:string"/>
                        <xs:element name="context_type" type="xs:string"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="V_C_V_RANGE_Type">
                <xs:sequence>
                        <xs:element name="view_control_variable_range"
type="VIEW_CONTROL_VARIABLE_RANGE_Type" maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="VIEW_CONTROL_VARIABLE_RANGE_Type" abstract="false">
                <xs:sequence>
                        <xs:element name="parameter_type" type="PROPERTY_REFERENCE_Type"/>
                        <xs:element name="range_lobound" type="xs:integer"/>
                        <xs:element name="range_hibound" type="xs:integer"/>
                </xs:sequence>
        </xs:complexType>
</xs:schema>
```

B.6 property.xsd

This module contains the resources intended to represent a property according to ISO 13584, whatever be its nature: a characteristic property, a context property, a context dependent property or a representation property.

```
<xs:schema xmlns="urn:iso:std:iso:cd:13584:-32" xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:iso:std:iso:cd:13584:-32" elementFormDefault="unqualified"
attributeFormDefault="unqualified">
        <xs:include schemaLocation="identifiers.xsd"/>
        <xs:include schemaLocation="baseTypes.xsd"/>
        <xs:include schemaLocation="translations.xsd"/>
        <xs:include schemaLocation="externalFiles.xsd"/>
        <xs:include schemaLocation="datatypeSystem.xsd"/>
        <xs:simpleType name="DET_CLASSIFICATION_TYPE_Type">
                <xs:restriction base="xs:string">
                        <xs:length value="3"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:complexType name="CONDITION_DET_Type">
                <xs:complexContent>
                        <xs:extension base="PROPERTY_Type"/>
                </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="DEPENDENT_P_DET_Type">
                <xs:complexContent>
                        <xs:extension base="PROPERTY_Type">
                                <xs:sequence>
                                        <xs:element name="depends_on"
type="PROPERTIES_REFERENCE_Type" minOccurs="0"/>
```

```
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="NON_DEPENDENT_P_DET_Type">
            <xs:complexContent>
                <xs:extension base="PROPERTY_Type"/>
            </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="PROPERTY_Type" abstract="true">
            <xs:sequence>
                <xs:element name="date_of_original_definition" type="DATE_TYPE_Type"
minOccurs="0"/>
                <xs:element name="date_of_current_version" type="DATE_TYPE_Type"
minOccurs="0"/>
                <xs:element name="date_of_current_revision" type="DATE_TYPE_Type"
minOccurs="0"/>
                <xs:element name="revision" type="REVISION_TYPE_Type"/>
                <xs:element name="status" type="STATUS_Type" minOccurs="0"/>
                <xs:element name="translation" type="TRANSLATION_Type" minOccurs="0"/>
                <xs:element name="source_language" type="LANGUAGE_Type"
minOccurs="0"/>
                <xs:element name="is_deprecated" type="xs:boolean" minOccurs="0"/>
                <xs:element name="preferred_name" type="PREFERRED_NAME_Type"/>
                <xs:element name="synonymous_names" type="SYNONYMOUS_NAME_Type"
minOccurs="0"/>
                <xs:element name="short_name" type="SHORT_NAME_Type" minOccurs="0"/>
                <xs:element name="icon" type="GRAPHICS_Type" minOccurs="0"/>
                <xs:element name="definition" type="TEXT_Type"/>
                <xs:element name="source_doc_of_definition"
type="SOURCE_DOCUMENT_Type" minOccurs="0"/>
                <xs:element name="note" type="TEXT_Type" minOccurs="0"/>
                <xs:element name="remark" type="TEXT_Type" minOccurs="0"/>
                <xs:element name="preferred_symbol" type="MATHEMATICAL_STRING_Type"
minOccurs="0"/>
                <xs:element name="synonymous_symbols"
type="SYNONYMOUS_SYMBOLS_Type" minOccurs="0"/>
                <xs:element name="figure" type="GRAPHICS_Type" minOccurs="0"/>
                <xs:element name="det_classification"
type="DET_CLASSIFICATION_TYPE_Type" minOccurs="0"/>
                <xs:element name="domain" type="DATA_TYPE_Type"/>
                <xs:element name="formula" type="MATHEMATICAL_STRING_Type"
minOccurs="0"/>
            </xs:sequence>
            <xs:attribute name="id" type="PropertyId" use="required"/>
        </xs:complexType>
        <xs:complexType name="REPRESENTATION_P_DET_Type">
            <xs:complexContent>
                <xs:extension base="PROPERTY_Type"/>
            </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="SYNONYMOUS_SYMBOLS_Type">
            <xs:sequence>
                <xs:element name="mathematical_string"
type="MATHEMATICAL_STRING_Type" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
</xs:schema>
```

B.7 datatype.xsd

This module contains the resources intended to represent data types according to ISO 13584.

```
<xs:schema xmlns="urn:iso:std:iso:cd:13584:-32" xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:iso:std:iso:cd:13584:-32" elementFormDefault="unqualified"
attributeFormDefault="unqualified">
	<xs:include schemaLocation="identifiers.xsd"/>
	<xs:include schemaLocation="baseTypes.xsd"/>
	<xs:include schemaLocation="translations.xsd"/>
	<xs:include schemaLocation="externalFiles.xsd"/>
	<xs:include schemaLocation="datatypeSystem.xsd"/>
	<xs:complexType name="DATATYPE_Type" abstract="false">
		<xs:sequence>
			<xs:element name="date_of_original_definition" type="DATE_TYPE_Type"
minOccurs="0"/>
			<xs:element name="date_of_current_version" type="DATE_TYPE_Type" minOccurs="0"/>
			<xs:element name="date_of_current_revision" type="DATE_TYPE_Type" minOccurs="0"/>
			<xs:element name="revision" type="REVISION_TYPE_Type"/>
			<xs:element name="status" type="STATUS_Type" minOccurs="0"/>
			<xs:element name="translation" type="TRANSLATION_Type" minOccurs="0"/>
			<xs:element name="source_language" type="LANGUAGE_Type" minOccurs="0"/>
			<xs:element name="is_deprecated" type="xs:boolean" minOccurs="0"/>
			<xs:element name="preferred_name" type="PREFERRED_NAME_Type"/>
			<xs:element name="synonymous_names" type="SYNONYMOUS_NAME_Type"
minOccurs="0"/>
			<xs:element name="short_name" type="SHORT_NAME_Type" minOccurs="0"/>
			<xs:element name="icon" type="GRAPHICS_Type" minOccurs="0"/>
			<xs:element name="type_definition" type="DATA_TYPE_Type"/>
		</xs:sequence>
		<xs:attribute name="id" type="DatatypeId" use="required"/>
	</xs:complexType>
</xs:schema>
```

B.8 document.xsd

This module contains the resources intended to represent a document according to ISO 13584.

```
<xs:schema xmlns="urn:iso:std:iso:cd:13584:-32" xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:iso:std:iso:cd:13584:-32" elementFormDefault="unqualified"
attributeFormDefault="unqualified">
	<xs:include schemaLocation="identifiers.xsd"/>
	<xs:include schemaLocation="baseTypes.xsd"/>
	<xs:include schemaLocation="translations.xsd"/>
	<xs:include schemaLocation="externalFiles.xsd"/>
	<xs:simpleType name="ABSOLUTE_URL_TYPE_Type">
		<xs:restriction base="xs:string">
			<xs:pattern value="\c*://\c*"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:complexType name="AUTHORS_Type">
		<xs:sequence>
			<xs:element name="person" type="PERSON_Type" maxOccurs="unbounded"/>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="DOCUMENT_CONTENT_Type" abstract="false">
		<xs:complexContent>
			<xs:extension base="EXTERNAL_RESOURCE_Type">
				<xs:sequence>
					<xs:element name="revision" type="REVISION_TYPE_Type"/>
				</xs:sequence>
			</xs:extension>
		</xs:complexContent>
	</xs:complexType>
```

```
	<xs:complexType name="DOCUMENT_Type" abstract="false">
		<xs:sequence>
			<xs:element name="date_of_original_definition" type="DATE_TYPE_Type"
minOccurs="0"/>
			<xs:element name="date_of_current_version" type="DATE_TYPE_Type" minOccurs="0"/>
			<xs:element name="date_of_current_revision" type="DATE_TYPE_Type" minOccurs="0"/>
			<xs:element name="revision" type="REVISION_TYPE_Type"/>
			<xs:element name="status" type="STATUS_Type" minOccurs="0"/>
			<xs:element name="translation" type="TRANSLATION_Type" minOccurs="0"/>
			<xs:element name="source_language" type="LANGUAGE_Type" minOccurs="0"/>
			<xs:element name="is_deprecated" type="xs:boolean" minOccurs="0"/>
			<xs:element name="preferred_name" type="PREFERRED_NAME_Type"/>
			<xs:element name="synonymous_names" type="SYNONYMOUS_NAME_Type"
minOccurs="0"/>
			<xs:element name="short_name" type="SHORT_NAME_Type" minOccurs="0"/>
			<xs:element name="icon" type="GRAPHICS_Type" minOccurs="0"/>
			<xs:element name="definition" type="TEXT_Type"/>
			<xs:element name="note" type="TEXT_Type" minOccurs="0"/>
			<xs:element name="remark" type="TEXT_Type" minOccurs="0"/>
			<xs:element name="authors" type="AUTHORS_Type" minOccurs="0"/>
			<xs:element name="publishing_organisation" type="ORGANIZATION_Type"/>
			<xs:element name="content" type="DOCUMENT_CONTENT_Type" minOccurs="0"/>
		</xs:sequence>
		<xs:attribute name="id" type="DocumentId" use="required"/>
	</xs:complexType>
	<xs:complexType name="PERSON_Type" abstract="false">
		<xs:sequence>
			<xs:element name="id" type="xs:string"/>
			<xs:element name="last_name" type="xs:string" minOccurs="0"/>
			<xs:element name="first_name" type="xs:string" minOccurs="0"/>
			<xs:element name="middle_names" type="STRINGS_Type" minOccurs="0"/>
			<xs:element name="prefix_titles" type="STRINGS_Type" minOccurs="0"/>
			<xs:element name="suffix_titles" type="STRINGS_Type" minOccurs="0"/>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="REMOTE_LOCATIONS_Type">
		<xs:sequence>
			<xs:element name="remote_location" maxOccurs="unbounded">
				<xs:complexType>
					<xs:simpleContent>
						<xs:extension base="ABSOLUTE_URL_TYPE_Type">
							<xs:attribute name="language"
type="LANGUAGE_CODE_Type" use="required"/>
						</xs:extension>
					</xs:simpleContent>
				</xs:complexType>
			</xs:element>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="STRINGS_Type">
		<xs:sequence>
			<xs:element name="value" type="xs:string" maxOccurs="unbounded"/>
		</xs:sequence>
	</xs:complexType>
</xs:schema>
```

B.9 datatypeSystem.xsd

This module contains an XML representation of the complete ISO 13584 type system.

```
<xs:schema xmlns="urn:iso:std:iso:cd:13584:-32" xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:iso:std:iso:cd:13584:-32" elementFormDefault="unqualified"
attributeFormDefault="unqualified">
	<xs:include schemaLocation="identifiers.xsd"/>
	<xs:include schemaLocation="baseTypes.xsd"/>
```

```
<xs:include schemaLocation="translations.xsd"/>
<xs:include schemaLocation="externalFiles.xsd"/>
<xs:include schemaLocation="units.xsd"/>
<xs:simpleType name="CURRENCY_CODE_Type">
	<xs:restriction base="xs:string">
		<xs:maxLength value="3"/>
	</xs:restriction>
</xs:simpleType>
<xs:complexType name="ALTERNATIVE_UNITS_Type">
	<xs:sequence>
		<xs:element name="dic_unit" type="DIC_UNIT_Type" maxOccurs="unbounded"/>
	</xs:sequence>
</xs:complexType>
<xs:complexType name="ARRAY_TYPE_Type">
	<xs:complexContent>
		<xs:extension base="DATA_TYPE_Type">
			<xs:sequence>
				<xs:element name="bound_1" type="xs:integer"/>
				<xs:element name="bound_2" type="xs:integer"/>
				<xs:element name="value_type" type="DATA_TYPE_Type"/>
				<xs:element name="uniqueness" type="xs:boolean"/>
				<xs:element name="are_optional" type="xs:boolean"/>
			</xs:sequence>
		</xs:extension>
	</xs:complexContent>
</xs:complexType>
<xs:complexType name="AXIS1_PLACEMENT_TYPE_Type">
	<xs:complexContent>
		<xs:extension base="DATA_TYPE_Type"/>
	</xs:complexContent>
</xs:complexType>
<xs:complexType name="AXIS2_PLACEMENT_2D_TYPE_Type">
	<xs:complexContent>
		<xs:extension base="DATA_TYPE_Type"/>
	</xs:complexContent>
</xs:complexType>
<xs:complexType name="AXIS2_PLACEMENT_3D_TYPE_Type">
	<xs:complexContent>
		<xs:extension base="DATA_TYPE_Type"/>
	</xs:complexContent>
</xs:complexType>
<xs:complexType name="BAG_TYPE_Type">
	<xs:complexContent>
		<xs:extension base="DATA_TYPE_Type">
			<xs:sequence>
				<xs:element name="bound_1" type="xs:integer" minOccurs="0"/>
				<xs:element name="bound_2" type="xs:integer" minOccurs="0"/>
				<xs:element name="value_type" type="DATA_TYPE_Type"/>
			</xs:sequence>
		</xs:extension>
	</xs:complexContent>
</xs:complexType>
<xs:complexType name="BOOLEAN_TYPE_Type">
	<xs:complexContent>
		<xs:extension base="DATA_TYPE_Type">
			<xs:sequence>
				<xs:element name="value_format" type="VALUE_FORMAT_TYPE_Type"
minOccurs="0"/>
			</xs:sequence>
```

```
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="CLASS_INSTANCE_TYPE_Type">
            <xs:complexContent>
                <xs:extension base="DATA_TYPE_Type">
                    <xs:sequence>
                        <xs:element name="domain" type="CLASS_REFERENCE_Type"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="DATA_TYPE_Type" abstract="true"/>
        <xs:complexType name="DIC_VALUE_Type" abstract="true">
            <xs:sequence>
                <xs:element name="preferred_name" type="PREFERRED_NAME_Type"/>
                <xs:element name="synonymous_names" type="SYNONYMOUS_NAME_Type"
minOccurs="0"/>
                <xs:element name="short_name" type="SHORT_NAME_Type" minOccurs="0"/>
                <xs:element name="icon" type="GRAPHICS_Type" minOccurs="0"/>
                <xs:element name="source_doc_of_value" type="SOURCE_DOCUMENT_Type"
minOccurs="0"/>
                <xs:element name="definition" type="TEXT_Type" minOccurs="0"/>
                <xs:element name="status" type="STATUS_Type" minOccurs="0"/>
                <xs:element name="is_deprecated" type="xs:boolean" minOccurs="0"/>
            </xs:sequence>
        </xs:complexType>
        <xs:complexType name="GEOMETRIC_REPRESENTATION_CONTEXT_TYPE_Type">
            <xs:complexContent>
                <xs:extension base="DATA_TYPE_Type"/>
            </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="INT_CURRENCY_TYPE_Type">
            <xs:complexContent>
                <xs:extension base="DATA_TYPE_Type">
                    <xs:sequence>
                        <xs:element name="value_format" type="VALUE_FORMAT_TYPE_Type"
minOccurs="0"/>
                        <xs:element name="currency" type="CURRENCY_CODE_Type"
minOccurs="0"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="INT_MEASURE_TYPE_Type">
            <xs:complexContent>
                <xs:extension base="DATA_TYPE_Type">
                    <xs:sequence>
                        <xs:element name="value_format" type="VALUE_FORMAT_TYPE_Type"
minOccurs="0"/>
                        <xs:element name="unit" type="DIC_UNIT_Type"/>
                        <xs:element name="alternative_units" type="ALTERNATIVE_UNITS_Type"
minOccurs="0"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="INT_TYPE_Type">
            <xs:complexContent>
                <xs:extension base="DATA_TYPE_Type">
```

```
                        <xs:sequence>
                            <xs:element name="value_format" type="VALUE_FORMAT_TYPE_Type"
minOccurs="0"/>
                        </xs:sequence>
                    </xs:extension>
                </xs:complexContent>
            </xs:complexType>
            <xs:complexType name="INT_DIC_VALUE_Type" abstract="false">
                <xs:complexContent>
                    <xs:extension base="DIC_VALUE_Type">
                        <xs:sequence>
                            <xs:element name="value_code" type="xs:integer"/>
                        </xs:sequence>
                    </xs:extension>
                </xs:complexContent>
            </xs:complexType>
            <xs:complexType name="ITS_VALUES_Type">
                <xs:sequence>
                    <xs:element name="dic_value" type="DIC_VALUE_Type" maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:complexType>
            <xs:complexType name="LEVEL_Type">
                <xs:sequence>
                    <xs:element name="min" minOccurs="0"/>
                    <xs:element name="nom" minOccurs="0"/>
                    <xs:element name="typ" minOccurs="0"/>
                    <xs:element name="max" minOccurs="0"/>
                </xs:sequence>
            </xs:complexType>
            <xs:complexType name="LEVEL_TYPE_Type">
                <xs:complexContent>
                    <xs:extension base="DATA_TYPE_Type">
                        <xs:sequence>
                            <xs:element name="levels" type="LEVEL_Type"/>
                            <xs:element name="value_type" type="DATA_TYPE_Type"/>
                        </xs:sequence>
                    </xs:extension>
                </xs:complexContent>
            </xs:complexType>
            <xs:complexType name="LIST_TYPE_Type">
                <xs:complexContent>
                    <xs:extension base="DATA_TYPE_Type">
                        <xs:sequence>
                            <xs:element name="bound_1" type="xs:integer" minOccurs="0"/>
                            <xs:element name="bound_2" type="xs:integer" minOccurs="0"/>
                            <xs:element name="value_type" type="DATA_TYPE_Type"/>
                            <xs:element name="uniqueness" type="xs:boolean"/>
                        </xs:sequence>
                    </xs:extension>
                </xs:complexContent>
            </xs:complexType>
            <xs:complexType name="NAMED_TYPE_Type">
                <xs:complexContent>
                    <xs:extension base="DATA_TYPE_Type">
                        <xs:sequence>
                            <xs:element name="referred_type" type="DATATYPE_REFERENCE_Type"/>
                        </xs:sequence>
                    </xs:extension>
                </xs:complexContent>
```

```
        </xs:complexType>
        <xs:complexType name="NON_QUANTITATIVE_CODE_TYPE_Type">
                <xs:complexContent>
                        <xs:extension base="DATA_TYPE_Type">
                                <xs:sequence>
                                        <xs:element name="value_format" type="VALUE_FORMAT_TYPE_Type"
minOccurs="0"/>
                                        <xs:element name="its_values" type="ITS_VALUES_Type"/>
                                        <xs:element name="source_doc_of_value_domain"
type="SOURCE_DOCUMENT_Type" minOccurs="0"/>
                                        <xs:element name="definition" type="TEXT_Type" minOccurs="0"/>
                                        <xs:element name="icon" type="GRAPHICS_Type" minOccurs="0"/>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="NON_QUANTITATIVE_INT_TYPE_Type">
                <xs:complexContent>
                        <xs:extension base="DATA_TYPE_Type">
                                <xs:sequence>
                                        <xs:element name="value_format" type="VALUE_FORMAT_TYPE_Type"
minOccurs="0"/>
                                        <xs:element name="its_values" type="ITS_VALUES_Type"/>
                                        <xs:element name="source_doc_of_value_domain"
type="SOURCE_DOCUMENT_Type" minOccurs="0"/>
                                        <xs:element name="definition" type="TEXT_Type" minOccurs="0"/>
                                        <xs:element name="icon" type="GRAPHICS_Type" minOccurs="0"/>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="NON_TRANSLATABLE_STRING_TYPE_Type">
                <xs:complexContent>
                        <xs:extension base="DATA_TYPE_Type">
                                <xs:sequence>
                                        <xs:element name="value_format" type="VALUE_FORMAT_TYPE_Type"
minOccurs="0"/>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="NUMBER_TYPE_Type">
                <xs:complexContent>
                        <xs:extension base="DATA_TYPE_Type">
                                <xs:sequence>
                                        <xs:element name="value_format" type="VALUE_FORMAT_TYPE_Type"
minOccurs="0"/>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="PLACEMENT_TYPE_Type">
                <xs:complexContent>
                        <xs:extension base="DATA_TYPE_Type"/>
                </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="PROGRAM_REFERENCE_TYPE_Type">
                <xs:complexContent>
                        <xs:extension base="DATA_TYPE_Type"/>
                </xs:complexContent>
```

```
        </xs:complexType>
        <xs:complexType name="REAL_CURRENCY_TYPE_Type">
                <xs:complexContent>
                        <xs:extension base="DATA_TYPE_Type">
                                <xs:sequence>
                                        <xs:element name="value_format" type="VALUE_FORMAT_TYPE_Type"
minOccurs="0"/>
                                        <xs:element name="currency" type="CURRENCY_CODE_Type"
minOccurs="0"/>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="REAL_MEASURE_TYPE_Type">
                <xs:complexContent>
                        <xs:extension base="DATA_TYPE_Type">
                                <xs:sequence>
                                        <xs:element name="value_format" type="VALUE_FORMAT_TYPE_Type"
minOccurs="0"/>
                                        <xs:element name="unit" type="DIC_UNIT_Type"/>
                                        <xs:element name="alternative_units" type="ALTERNATIVE_UNITS_Type"
minOccurs="0"/>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="REAL_TYPE_Type">
                <xs:complexContent>
                        <xs:extension base="DATA_TYPE_Type">
                                <xs:sequence>
                                        <xs:element name="value_format" type="VALUE_FORMAT_TYPE_Type"
minOccurs="0"/>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="REMOTE_HTTP_ADDRESS_Type">
                <xs:complexContent>
                        <xs:extension base="DATA_TYPE_Type">
                                <xs:sequence>
                                        <xs:element name="value_format" type="VALUE_FORMAT_TYPE_Type"
minOccurs="0"/>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="REPRESENTATION_REFERENCE_TYPE_Type">
                <xs:complexContent>
                        <xs:extension base="DATA_TYPE_Type"/>
                </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="SET_TYPE_Type">
                <xs:complexContent>
                        <xs:extension base="DATA_TYPE_Type">
                                <xs:sequence>
                                        <xs:element name="bound_1" type="xs:integer" minOccurs="0"/>
                                        <xs:element name="bound_2" type="xs:integer" minOccurs="0"/>
                                        <xs:element name="value_type" type="DATA_TYPE_Type"/>
                                </xs:sequence>
```

```
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="SET_WITH_SUBSET_CONSTRAINT_TYPE_Type">
            <xs:complexContent>
                <xs:extension base="DATA_TYPE_Type">
                    <xs:sequence>
                        <xs:element name="bound_1" type="xs:integer" minOccurs="0"/>
                        <xs:element name="bound_2" type="xs:integer" minOccurs="0"/>
                        <xs:element name="value_type" type="DATA_TYPE_Type"/>
                        <xs:element name="cardinal_min" type="xs:integer" minOccurs="0"/>
                        <xs:element name="cardinal_max" type="xs:integer" minOccurs="0"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="STRING_DIC_VALUE_Type" abstract="false">
            <xs:complexContent>
                <xs:extension base="DIC_VALUE_Type">
                    <xs:sequence>
                        <xs:element name="value_code" type="VALUE_CODE_TYPE_Type"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="STRING_TYPE_Type">
            <xs:complexContent>
                <xs:extension base="DATA_TYPE_Type">
                    <xs:sequence>
                        <xs:element name="value_format" type="VALUE_FORMAT_TYPE_Type"
minOccurs="0"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="TRANSLATABLE_STRING_TYPE_Type">
            <xs:complexContent>
                <xs:extension base="DATA_TYPE_Type">
                    <xs:sequence>
                        <xs:element name="value_format" type="VALUE_FORMAT_TYPE_Type"
minOccurs="0"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="TYPE_NAME_Type">
            <xs:sequence>
                <xs:element name="element_value" type="xs:string" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
        <xs:complexType name="VALUE_TYPE_Type">
            <xs:choice>
                <xs:element name="value_code_type" type="VALUE_CODE_TYPE_Type"/>
                <xs:element name="integer_type" type="xs:integer"/>
            </xs:choice>
        </xs:complexType>
</xs:schema>
```

B.10 translations.xsd

This module contains resources for defining multilingual representations of clear text.

```
<xs:schema xmlns="urn:iso:std:iso:cd:13584:-32" xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:iso:std:iso:cd:13584:-32" elementFormDefault="unqualified"
attributeFormDefault="unqualified">
	<xs:include schemaLocation="baseTypes.xsd"/>
	<xs:include schemaLocation="identifiers.xsd"/>
	<xs:simpleType name="PREFERRED_NAME_TYPE_Type">
		<xs:restriction base="xs:string">
			<xs:maxLength value="70"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="SHORT_NAME_TYPE_Type">
		<xs:restriction base="xs:string">
			<xs:maxLength value="30"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="SYNONYMOUS_NAME_TYPE_Type">
		<xs:restriction base="xs:string">
			<xs:maxLength value="70"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:complexType name="GENERAL_TEXT_Type">
		<xs:simpleContent>
			<xs:extension base="xs:string">
				<xs:attribute name="language_code" type="LANGUAGE_CODE_Type"
use="optional"/>
				<xs:attribute name="country_code" type="COUNTRY_CODE_Type" use="optional"/>
			</xs:extension>
		</xs:simpleContent>
	</xs:complexType>
	<xs:complexType name="PREFERRED_NAME_LABEL_Type">
		<xs:simpleContent>
			<xs:extension base="PREFERRED_NAME_TYPE_Type">
				<xs:attribute name="language_code" type="LANGUAGE_CODE_Type"
use="optional"/>
				<xs:attribute name="country_code" type="COUNTRY_CODE_Type" use="optional"/>
			</xs:extension>
		</xs:simpleContent>
	</xs:complexType>
	<xs:complexType name="PREFERRED_NAME_Type">
		<xs:sequence>
			<xs:element name="label" type="PREFERRED_NAME_LABEL_Type"
maxOccurs="unbounded"/>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="SHORT_NAME_LABEL_Type">
		<xs:simpleContent>
			<xs:extension base="SHORT_NAME_TYPE_Type">
				<xs:attribute name="language_code" type="LANGUAGE_CODE_Type"
use="optional"/>
				<xs:attribute name="country_code" type="COUNTRY_CODE_Type" use="optional"/>
			</xs:extension>
		</xs:simpleContent>
	</xs:complexType>
	<xs:complexType name="SHORT_NAME_Type">
		<xs:sequence>
			<xs:element name="label" type="SHORT_NAME_LABEL_Type"
maxOccurs="unbounded"/>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="SYNONYMOUS_NAME_LABEL_Type">
		<xs:simpleContent>
```

```
                        <xs:extension base="SYNONYMOUS_NAME_TYPE_Type">
                                <xs:attribute name="language_code" type="LANGUAGE_CODE_Type"
use="optional"/>
                                <xs:attribute name="country_code" type="COUNTRY_CODE_Type" use="optional"/>
                        </xs:extension>
                </xs:simpleContent>
        </xs:complexType>
        <xs:complexType name="SYNONYMOUS_NAME_Type">
                <xs:choice>
                        <xs:element name="label" type="SYNONYMOUS_NAME_LABEL_Type"
maxOccurs="unbounded"/>
                </xs:choice>
        </xs:complexType>
        <xs:complexType name="TEXT_Type">
                <xs:sequence>
                        <xs:element name="text" type="GENERAL_TEXT_Type" maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="TRANSLATION_DATA_Type" abstract="false">
                <xs:sequence>
                        <xs:element name="language" type="LANGUAGE_Type"/>
                        <xs:element name="responsible_translator">
                                <xs:complexType>
                                        <xs:attribute name="supplier_ref" type="SupplierId" use="required"/>
                                </xs:complexType>
                        </xs:element>
                        <xs:element name="translation_revision" type="REVISION_TYPE_Type"/>
                        <xs:element name="date_of_current_translation_revision" type="DATE_TYPE_Type"
minOccurs="0"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="TRANSLATION_Type">
                <xs:sequence>
                        <xs:element name="translation_data" type="TRANSLATION_DATA_Type"
maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
</xs:schema>
```

B.11 externalFiles.xsd

This module contains constructs for referencing external resources.

```
<xs:schema xmlns="urn:iso:std:iso:cd:13584:-32" xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:iso:std:iso:cd:13584:-32" elementFormDefault="unqualified"
attributeFormDefault="unqualified">
        <xs:include schemaLocation="identifiers.xsd"/>
        <xs:include schemaLocation="baseTypes.xsd"/>
        <xs:include schemaLocation="units.xsd"/>
        <xs:simpleType name="EXTERNAL_ITEM_CODE_TYPE_Type">
                <xs:restriction base="xs:string">
                        <xs:maxLength value="18"/>
                        <xs:pattern value="[^\-]*"/>
                        <xs:pattern value="[^\s]*"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="HTTP_DIRECTORY_NAME_TYPE_Type">
                <xs:restriction base="xs:string">
                        <xs:maxLength value="128"/>
                        <xs:pattern value="[^\s]*"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="HTTP_FILE_NAME_TYPE_Type">
                <xs:restriction base="xs:anyURI">
                        <xs:maxLength value="128"/>
                        <xs:pattern value="[^\s]*"/>
```

```
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="ILLUSTRATION_TYPE_Type">
            <xs:restriction base="xs:string">
                <xs:enumeration value="SCHEMATIC_DRAWING"/>
                <xs:enumeration value="REALISTIC_PICTURE"/>
                <xs:enumeration value="NOT_STATIC_PICTURE"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="PROGRAM_REFERENCE_NAME_Type">
            <xs:restriction base="xs:string">
                <xs:pattern value="[^\-]*"/>
                <xs:pattern value="[^\s]*"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="SOURCE_DOC_TYPE_Type">
            <xs:restriction base="xs:string">
                <xs:maxLength value="80"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="STANDARD_SIZE_Type">
            <xs:restriction base="xs:string">
                <xs:enumeration value="a6_illustration"/>
                <xs:enumeration value="a9_illustration"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:complexType name="EXTERNAL_GRAPHICS_Type">
            <xs:complexContent>
                <xs:extension base="GRAPHICS_Type">
                    <xs:sequence>
                        <xs:element name="representation" type="GRAPHICS_FILES_Type"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="EXTERNAL_RESOURCE_Type" abstract="true">
            <xs:sequence>
                <xs:element name="file" type="HTTP_FILE_Type" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
        <xs:complexType name="GRAPHICS_FILES_Type">
            <xs:complexContent>
                <xs:extension base="EXTERNAL_RESOURCE_Type">
                    <xs:sequence/>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="GRAPHICS_Type" abstract="true"/>
        <xs:complexType name="HTTP_FILE_Type" abstract="false">
            <xs:sequence>
                <xs:element name="http_file_name" type="HTTP_FILE_NAME_TYPE_Type"/>
                <xs:element name="dir_name" type="HTTP_DIRECTORY_NAME_TYPE_Type"
minOccurs="0"/>
            </xs:sequence>
            <xs:attribute name="language_code" type="LANGUAGE_CODE_Type" use="optional"/>
            <xs:attribute name="country_code" type="COUNTRY_CODE_Type" use="optional"/>
        </xs:complexType>
        <xs:complexType name="IDENTIFIED_DOCUMENT_Type">
            <xs:complexContent>
                <xs:extension base="SOURCE_DOCUMENT_Type">
                    <xs:sequence>
                        <xs:element name="document_identifier"
type="SOURCE_DOC_TYPE_Type"/>
                    </xs:sequence>
                </xs:extension>
```

```
            </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="ILLUSTRATION_Type">
            <xs:complexContent>
                <xs:extension base="EXTERNAL_RESOURCE_Type">
                    <xs:sequence>
                        <xs:element name="code" type="EXTERNAL_ITEM_CODE_TYPE_Type"/>
                        <xs:element name="kind_of_content" type="ILLUSTRATION_TYPE_Type"/>
                        <xs:element name="width" type="xs:integer" minOccurs="0"/>
                        <xs:element name="height" type="xs:integer" minOccurs="0"/>
                    </xs:sequence>
                    <xs:attribute name="standard_size" type="STANDARD_SIZE_Type"
use="optional"/>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="MESSAGE_Type">
            <xs:complexContent>
                <xs:extension base="EXTERNAL_RESOURCE_Type">
                    <xs:sequence>
                        <xs:element name="code" type="EXTERNAL_ITEM_CODE_TYPE_Type"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="PROGRAM_REFERENCE_Type">
            <xs:complexContent>
                <xs:extension base="EXTERNAL_RESOURCE_Type">
                    <xs:sequence>
                        <xs:element name="code" type="EXTERNAL_ITEM_CODE_TYPE_Type"/>
                        <xs:element name="syntactical_name"
type="PROGRAM_REFERENCE_NAME_Type"/>
                        <xs:element name="in_parameters"
type="PROPERTIES_REFERENCE_Type" minOccurs="0"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="REFERENCED_DOCUMENT_Type">
            <xs:complexContent>
                <xs:extension base="SOURCE_DOCUMENT_Type">
                    <xs:sequence>
                        <xs:element name="document_reference"
type="DOCUMENT_REFERENCE_Type"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="REFERENCED_GRAPHICS_Type">
            <xs:complexContent>
                <xs:extension base="GRAPHICS_Type">
                    <xs:sequence>
                        <xs:element name="graphics_reference"
type="DOCUMENT_REFERENCE_Type"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="REPRESENTATION_REFERENCE_Type">
            <xs:complexContent>
                <xs:extension base="EXTERNAL_RESOURCE_Type">
                    <xs:sequence>
                        <xs:element name="code" type="EXTERNAL_ITEM_CODE_TYPE_Type"/>
                        <xs:element name="representation_id" type="xs:string" minOccurs="0"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
```

```
	</xs:complexType>
	<xs:complexType name="SOURCE_DOCUMENT_Type" abstract="true"/>
</xs:schema>
```

B.12 units.xsd

This module contains resources for explicitly describing any kind of units according to ISO 10303-41.

```
<xs:schema xmlns="urn:iso:std:iso:cd:13584:-32" xmlns:idt="urn:pcd:schema:identifier"
xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="urn:iso:std:iso:cd:13584:-32"
elementFormDefault="unqualified" attributeFormDefault="unqualified">
	<xs:import namespace="urn:pcd:schema:identifier" schemaLocation="..\ISO29002\identifier.xsd"/>
	<xs:include schemaLocation="baseTypes.xsd"/>
	<xs:simpleType name="POSITIVE_LENGTH_MEASURE_Type">
		<xs:restriction base="xs:decimal">
			<xs:minExclusive value="0.0"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="POSITIVE_PLANE_ANGLE_MEASURE_Type">
		<xs:restriction base="xs:decimal"/>
	</xs:simpleType>
	<xs:simpleType name="POSITIVE_RATIO_MEASURE_Type">
		<xs:restriction base="xs:decimal"/>
	</xs:simpleType>
	<xs:simpleType name="SI_UNIT_NAME_Type">
		<xs:restriction base="xs:string">
			<xs:enumeration value="METRE"/>
			<xs:enumeration value="GRAM"/>
			<xs:enumeration value="SECOND"/>
			<xs:enumeration value="AMPERE"/>
			<xs:enumeration value="KELVIN"/>
			<xs:enumeration value="MOLE"/>
			<xs:enumeration value="CANDELA"/>
			<xs:enumeration value="RADIAN"/>
			<xs:enumeration value="STERADIAN"/>
			<xs:enumeration value="HERTZ"/>
			<xs:enumeration value="NEWTON"/>
			<xs:enumeration value="PASCAL"/>
			<xs:enumeration value="JOULE"/>
			<xs:enumeration value="WATT"/>
			<xs:enumeration value="COULOMB"/>
			<xs:enumeration value="VOLT"/>
			<xs:enumeration value="FARAD"/>
			<xs:enumeration value="OHM"/>
			<xs:enumeration value="SIEMENS"/>
			<xs:enumeration value="WEBER"/>
			<xs:enumeration value="TESLA"/>
			<xs:enumeration value="HENRY"/>
			<xs:enumeration value="DEGREE_CELSIUS"/>
			<xs:enumeration value="LUMEN"/>
			<xs:enumeration value="LUX"/>
			<xs:enumeration value="BECQUEREL"/>
			<xs:enumeration value="GRAY"/>
			<xs:enumeration value="SIEVERT"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="SI_PREFIX_Type">
		<xs:restriction base="xs:string">
			<xs:enumeration value="EXA"/>
			<xs:enumeration value="PETA"/>
			<xs:enumeration value="TERA"/>
			<xs:enumeration value="GIGA"/>
			<xs:enumeration value="MEGA"/>
			<xs:enumeration value="KILO"/>
```

```
                        <xs:enumeration value="HECTO"/>
                        <xs:enumeration value="DECA"/>
                        <xs:enumeration value="DECI"/>
                        <xs:enumeration value="CENTI"/>
                        <xs:enumeration value="MILLI"/>
                        <xs:enumeration value="MICRO"/>
                        <xs:enumeration value="NANO"/>
                        <xs:enumeration value="PICO"/>
                        <xs:enumeration value="FEMTO"/>
                        <xs:enumeration value="ATTO"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:complexType name="CONTEXT_DEPENDENT_UNIT_Type">
                <xs:complexContent>
                        <xs:extension base="NAMED_UNIT_Type">
                                <xs:sequence>
                                        <xs:element name="name" type="xs:string"/>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="CONVERSION_BASED_UNIT_Type">
                <xs:complexContent>
                        <xs:extension base="NAMED_UNIT_Type">
                                <xs:sequence>
                                        <xs:element name="name" type="xs:string"/>
                                        <xs:element name="value_component" type="xs:decimal"/>
                                        <xs:element name="unit_component" type="UNIT_Type"/>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="DERIVED_UNIT_ELEMENT_Type" abstract="false">
                <xs:sequence>
                        <xs:element name="unit" type="NAMED_UNIT_Type"/>
                        <xs:element name="exponent" type="xs:decimal"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="DERIVED_UNIT_Type" abstract="false">
                <xs:complexContent>
                        <xs:extension base="UNIT_Type">
                                <xs:sequence>
                                        <xs:element name="derived_unit_element"
type="DERIVED_UNIT_ELEMENT_Type" maxOccurs="unbounded"/>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="DIC_UNIT_Type" abstract="false">
                <xs:sequence>
                        <xs:element name="unit_ref" type="idt:Id" minOccurs="0"/>
                        <xs:element name="structured_representation" type="UNIT_Type" minOccurs="0"/>
                        <xs:element name="string_representation" type="MATHEMATICAL_STRING_Type"
minOccurs="0"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="DIMENSIONAL_EXPONENTS_Type" abstract="false">
                <xs:sequence>
                        <xs:element name="length_exponent" type="xs:decimal"/>
                        <xs:element name="mass_exponent" type="xs:decimal"/>
                        <xs:element name="time_exponent" type="xs:decimal"/>
                        <xs:element name="electric_current_exponent" type="xs:decimal"/>
                        <xs:element name="thermodynamic_temperature_exponent" type="xs:decimal"/>
                        <xs:element name="amount_of_substance_exponent" type="xs:decimal"/>
                        <xs:element name="luminous_intensity_exponent" type="xs:decimal"/>
                </xs:sequence>
        </xs:complexType>
```

```
	<xs:complexType name="NAMED_UNIT_Type" abstract="false">
		<xs:complexContent>
			<xs:extension base="UNIT_Type">
				<xs:sequence>
					<xs:element name="dimensions" type="DIMENSIONAL_EXPONENTS_Type"
minOccurs="0"/>
				</xs:sequence>
			</xs:extension>
		</xs:complexContent>
	</xs:complexType>
	<xs:complexType name="NON_SI_UNIT_Type">
		<xs:complexContent>
			<xs:extension base="NAMED_UNIT_Type">
				<xs:sequence>
					<xs:element name="name" type="xs:string"/>
				</xs:sequence>
			</xs:extension>
		</xs:complexContent>
	</xs:complexType>
	<xs:complexType name="SI_UNIT_Type">
		<xs:complexContent>
			<xs:extension base="NAMED_UNIT_Type">
				<xs:sequence>
					<xs:element name="prefix" type="SI_PREFIX_Type" minOccurs="0"/>
					<xs:element name="name" type="SI_UNIT_NAME_Type"/>
				</xs:sequence>
			</xs:extension>
		</xs:complexContent>
	</xs:complexType>
	<xs:complexType name="UNIT_Type" abstract="true"/>
	<xs:complexType name="UNITS_Type">
		<xs:sequence>
			<xs:element name="unit" type="UNIT_Type" maxOccurs="unbounded"/>
		</xs:sequence>
	</xs:complexType>
</xs:schema>
```

B.13 aPosteriori.xsd

This module contains resources for representing a posteriori mappings between properties defined in different ontologies.

```
<xs:schema xmlns="urn:iso:std:iso:cd:13584:-32" xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:iso:std:iso:cd:13584:-32" elementFormDefault="unqualified"
attributeFormDefault="unqualified">
	<xs:include schemaLocation="identifiers.xsd"/>
	<xs:complexType name="A_POSTERIORI_CASE_OF_Type">
		<xs:complexContent>
			<xs:extension base="A_POSTERIORI_SEMANTIC_RELATIONSHIP_Type">
				<xs:sequence>
					<xs:element name="source" type="CLASS_REFERENCE_Type"/>
					<xs:element name="is_case_of" type="CLASS_REFERENCE_Type"/>
					<xs:element name="corresponding_properties"
type="CORRESPONDING_PROPERTIES_type" minOccurs="0"/>
				</xs:sequence>
			</xs:extension>
		</xs:complexContent>
	</xs:complexType>
	<xs:complexType name="A_POSTERIORI_SEMANTIC_RELATIONSHIP_Type" abstract="true"/>
	<xs:complexType name="A_POSTERIORI_VIEW_OF_Type">
		<xs:complexContent>
			<xs:extension base="A_POSTERIORI_SEMANTIC_RELATIONSHIP_Type">
```

```
                        <xs:sequence>
                            <xs:element name="functional_model" type="CLASS_REFERENCE_Type"/>
                            <xs:element name="is_view_of" type="CLASS_REFERENCE_Type"/>
                            <xs:element name="corresponding_properties"
type="CORRESPONDING_PROPERTIES_type" minOccurs="0"/>
                        </xs:sequence>
                    </xs:extension>
                </xs:complexContent>
            </xs:complexType>
            <xs:complexType name="CORRESPONDING_PROPERTIES_type">
                <xs:sequence>
                    <xs:element name="mapping" type="PROPERTY_MAPPING_Type"
maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:complexType>
            <xs:complexType name="PROPERTY_MAPPING_Type">
                <xs:sequence>
                    <xs:element name="domain" type="PROPERTIES_REFERENCE_Type"
maxOccurs="unbounded"/>
                    <xs:element name="range" type="PROPERTY_REFERENCE_Type"/>
                    <xs:element name="function" type="MAPPING_FUNCTION_Type" minOccurs="0"/>
                </xs:sequence>
            </xs:complexType>
            <xs:complexType name="MAPPING_FUNCTION_Type" abstract="true"/>
</xs:schema>
```

B.14 constraints.xsd

This module contains resources intended to represent constrain properties.

```
<xs:schema xmlns="urn:iso:std:iso:cd:13584:-32" xmlns:val="urn:pcd:schema:value"
xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="urn:iso:std:iso:cd:13584:-32"
elementFormDefault="unqualified" attributeFormDefault="unqualified">
            <xs:include schemaLocation="baseTypes.xsd"/>
            <xs:include schemaLocation="datatypeSystem.xsd"/>
            <xs:include schemaLocation="identifiers.xsd"/>
            <xs:import namespace="urn:pcd:schema:value" schemaLocation="..\ISO29002\value.xsd"/>
            <xs:complexType name="CONSTRAINT_Type" abstract="true"/>
            <xs:complexType name="CONSTRAINTS_Type">
                <xs:sequence>
                    <xs:element name="constraint" type="CONSTRAINT_Type" maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:complexType>
            <xs:complexType name="CONTEXT_RESTRICTION_CONSTRAINT_Type">
                <xs:complexContent>
                    <xs:extension base="DOMAIN_CONSTRAINT_Type">
                        <xs:sequence>
                            <xs:element name="context_restriction"
type="DOMAIN_CONSTRAINT_Type"/>
                        </xs:sequence>
                    </xs:extension>
                </xs:complexContent>
            </xs:complexType>
            <xs:complexType name="DOMAIN_CONSTRAINT_Type" abstract="true">
                <xs:sequence>
                    <xs:element name="constrained_entity" type="PROPERTY_REFERENCE_Type"/>
                </xs:sequence>
            </xs:complexType>
            <xs:complexType name="PROPERTY_TYPE_REDEFINITION_Type">
                <xs:complexContent>
                    <xs:extension base="CONSTRAINT_Type">
                        <xs:sequence>
```

```
                                        <xs:element name="redefined_domain"
type="DOMAIN_CONSTRAINT_Type"/>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="RANGE_CONSTRAINT_Type">
                <xs:complexContent>
                        <xs:extension base="DOMAIN_CONSTRAINT_Type">
                                <xs:sequence>
                                        <xs:element name="min_value" type="xs:decimal"/>
                                        <xs:element name="max_value" type="xs:decimal"/>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="STRING_PATTERN_CONSTRAINT_Type">
                <xs:complexContent>
                        <xs:extension base="DOMAIN_CONSTRAINT_Type">
                                <xs:sequence>
                                        <xs:element name="pattern" type="xs:string"/>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="STRING_SIZE_CONSTRAINT_Type">
                <xs:complexContent>
                        <xs:extension base="DOMAIN_CONSTRAINT_Type">
                                <xs:sequence>
                                        <xs:element name="min_length" type="xs:integer" minOccurs="0"/>
                                        <xs:element name="max_length" type="xs:integer" minOccurs="0"/>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="SUBCLASS_CONSTRAINT_Type">
                <xs:complexContent>
                        <xs:extension base="DOMAIN_CONSTRAINT_Type">
                                <xs:sequence>
                                        <xs:element name="subclass" type="CLASSES_REFERENCE_Type"/>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="SUBSET_CONSTRAINT_Type">
                <xs:complexContent>
                        <xs:extension base="DOMAIN_CONSTRAINT_Type">
                                <xs:sequence>
                                        <xs:element name="subset" type="SUBSET_Type" minOccurs="0"/>
                                        <xs:element name="value_meaning"
type="NON_QUANTITATIVE_INT_TYPE_Type" minOccurs="0"/>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="SUBSET_Type">
                <xs:sequence maxOccurs="unbounded">
                        <xs:group ref="val:value"/>
                </xs:sequence>
        </xs:complexType>
</xs:schema>
```

B.15 baseTypes.xsd

This module contains all the basic types definitions shared by all the OntoML modules.

```
<xs:schema xmlns="urn:iso:std:iso:cd:13584:-32" xmlns:mml="http://www.w3.org/1998/Math/MathML"
xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="urn:iso:std:iso:cd:13584:-32"
elementFormDefault="unqualified" attributeFormDefault="unqualified">
	<xs:import namespace="http://www.w3.org/1998/Math/MathML"
schemaLocation="..\mathml2\mathml2.xsd"/>
	<xs:simpleType name="COUNTRY_CODE_Type">
		<xs:restriction base="xs:string">
			<xs:maxLength value="2"/>
			<xs:pattern value="[^\-]*"/>
			<xs:pattern value="[^\s]*"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="DATE_TYPE_Type">
		<xs:restriction base="xs:date"/>
	</xs:simpleType>
	<xs:simpleType name="LANGUAGE_CODE_Type">
		<xs:restriction base="xs:string">
			<xs:maxLength value="3"/>
			<xs:pattern value="[^\-]*"/>
			<xs:pattern value="[^\s]*"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="REVISION_TYPE_Type">
		<xs:restriction base="xs:string">
			<xs:maxLength value="3"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="STATUS_Type">
		<xs:restriction base="xs:string">
			<xs:pattern value="[^\-]*"/>
			<xs:pattern value="[^\s]*"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="VALUE_CODE_TYPE_Type">
		<xs:restriction base="xs:string">
			<xs:maxLength value="18"/>
			<xs:pattern value="[^\-]*"/>
			<xs:pattern value="[^\s]*"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="VALUE_FORMAT_TYPE_Type">
		<xs:restriction base="xs:string">
			<xs:maxLength value="80"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:complexType name="LANGUAGE_Type">
		<xs:attribute name="language_code" type="LANGUAGE_CODE_Type" use="required"/>
		<xs:attribute name="country_code" type="COUNTRY_CODE_Type" use="optional"/>
	</xs:complexType>
	<xs:complexType name="MATHEMATICAL_STRING_Type" abstract="false">
		<xs:sequence>
			<xs:element name="text_representation" type="xs:string"/>
			<xs:element name="sgml_representation" type="mml:math.type" minOccurs="0"/>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="ORGANIZATION_Type" abstract="false">
		<xs:sequence>
			<xs:element name="id" type="xs:string" minOccurs="0"/>
			<xs:element name="name" type="xs:string"/>
			<xs:element name="description" type="xs:string" minOccurs="0"/>
		</xs:sequence>
	</xs:complexType>
</xs:schema>
```

B.16 identifiers.xsd

This module provides basic resources intended to be used for identifying or referencing CIIM ontology concepts.

```
<xs:schema xmlns="urn:iso:std:iso:cd:13584:-32" xmlns:idt="urn:pcd:schema:identifier"
xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="urn:iso:std:iso:cd:13584:-32"
elementFormDefault="unqualified" attributeFormDefault="unqualified">
	<xs:import namespace="urn:pcd:schema:identifier" schemaLocation="..\ISO29002\identifier.xsd"/>
	<xs:simpleType name="SupplierId">
		<xs:restriction base="idt:Id">
			<xs:pattern value="([0-9]{4})-([a-zA-Z0-9]{1,35})(\[([a-zA-Z0-9]{1,35})(-[019])?\])?(-STD:([a-
zA-Z0-9]{1,10})_([a-zA-Z0-9]{0,10})_([0-9]{1,5}))?"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="OntologyId">
		<xs:restriction base="idt:Id">
			<xs:pattern value="([0-9]{4})-([a-zA-Z0-9]{1,35})(\[([a-zA-Z0-9]{1,35})(-[019])?\])?(-STD:([a-
zA-Z0-9]{1,10})_([a-zA-Z0-9]{0,10})_([0-9]{1,5}))?#([a-zA-Z0-9_]{1,14})#[a-zA-Z0-9]{1,9}"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="ClassId">
		<xs:restriction base="idt:Id">
			<xs:pattern value="([0-9]{4})-([a-zA-Z0-9]{1,35})(\[([a-zA-Z0-9]{1,35})(-[019])?\])?(-STD:([a-
zA-Z0-9]{1,10})_([a-zA-Z0-9]{0,10})_([0-9]{1,5}))?#([a-zA-Z0-9_]{1,14})#[a-zA-Z0-9]{1,9}"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="PropertyId">
		<xs:restriction base="idt:Id">
			<xs:pattern value="([0-9]{4})-([a-zA-Z0-9]{1,35})(\[([a-zA-Z0-9]{1,35})(-[019])?\])?(-STD:([a-
zA-Z0-9]{1,10})_([a-zA-Z0-9]{0,10})_([0-9]{1,5}))?#([a-zA-Z0-9_]{1,14}\*)?([a-zA-Z0-9_]{1,14})#[a-zA-Z0-9]{1,9}"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="DocumentId">
		<xs:restriction base="idt:Id">
			<xs:pattern value="([0-9]{4})-([a-zA-Z0-9]{1,35})(\[([a-zA-Z0-9]{1,35})(-[019])?\])?(-STD:([a-
zA-Z0-9]{1,10})_([a-zA-Z0-9]{0,10})_([0-9]{1,5}))?#([a-zA-Z0-9_]{1,14}\*)?([a-zA-Z0-9_]{1,14})#[a-zA-Z0-9]{1,9}"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="DatatypeId">
		<xs:restriction base="idt:Id">
			<xs:pattern value="([0-9]{4})-([a-zA-Z0-9]{1,35})(\[([a-zA-Z0-9]{1,35})(-[019])?\])?(-STD:([a-
zA-Z0-9]{1,10})_([a-zA-Z0-9]{0,10})_([0-9]{1,5}))?#([a-zA-Z0-9_]{1,14}\*)?([a-zA-Z0-9_]{1,14})#[a-zA-Z0-9]{1,9}"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:complexType name="CLASS_REFERENCE_Type">
		<xs:attribute name="class_ref" type="ClassId" use="required"/>
	</xs:complexType>
	<xs:complexType name="CLASSES_REFERENCE_Type">
		<xs:sequence>
			<xs:element name="class" type="CLASS_REFERENCE_Type" maxOccurs="unbounded"/>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="DATATYPE_REFERENCE_Type">
		<xs:attribute name="datatype_ref" type="DatatypeId" use="required"/>
	</xs:complexType>
	<xs:complexType name="DATATYPES_REFERENCE_Type">
		<xs:sequence>
			<xs:element name="datatype" type="DATATYPE_REFERENCE_Type"
maxOccurs="unbounded"/>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="DICTIONARY_REFERENCE_Type">
		<xs:attribute name="dictionary_ref" type="OntologyId" use="required"/>
	</xs:complexType>
```

```
	<xs:complexType name="DICTIONARIES_REFERENCE_Type">
		<xs:sequence>
			<xs:element name="dictionary" type="DICTIONARY_REFERENCE_Type"
maxOccurs="unbounded"/>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="DOCUMENT_REFERENCE_Type">
		<xs:attribute name="document_ref" type="DocumentId" use="required"/>
	</xs:complexType>
	<xs:complexType name="DOCUMENTS_REFERENCE_Type">
		<xs:sequence>
			<xs:element name="document" type="DOCUMENT_REFERENCE_Type"
maxOccurs="unbounded"/>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="PROPERTY_REFERENCE_Type">
		<xs:attribute name="property_ref" type="PropertyId" use="required"/>
	</xs:complexType>
	<xs:complexType name="PROPERTIES_REFERENCE_Type">
		<xs:sequence>
			<xs:element name="property" type="PROPERTY_REFERENCE_Type"
maxOccurs="unbounded"/>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="SUPPLIER_REFERENCE_Type">
		<xs:attribute name="supplier_ref" type="SupplierId" use="required"/>
	</xs:complexType>
	<xs:complexType name="SUPPLIERS_REFERENCE_Type">
		<xs:sequence>
			<xs:element name="supplier" type="SUPPLIER_REFERENCE_Type"
maxOccurs="unbounded"/>
		</xs:sequence>
	</xs:complexType>
</xs:schema>
```

B.17 content.xsd

This module defines structure for representing class extensions as a set of property – value couples.

```
<xs:schema xmlns="urn:iso:std:iso:cd:13584:-32" xmlns:cat="urn:pcd:schema:item"
xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="urn:iso:std:iso:cd:13584:-32"
elementFormDefault="unqualified" attributeFormDefault="unqualified">
	<xs:include schemaLocation="identifiers.xsd"/>
	<xs:include schemaLocation="baseTypes.xsd"/>
	<xs:include schemaLocation="externalFiles.xsd"/>
	<xs:include schemaLocation="units.xsd"/>
	<xs:import namespace="urn:pcd:schema:item" schemaLocation="..\ISO29002\catalogue.xsd"/>
	<xs:simpleType name="VERSION_TYPE_Type">
		<xs:restriction base="xs:string">
			<xs:maxLength value="9"/>
			<xs:pattern value="[0-9]+"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:complexType name="CLASS_EXTENSION_Type" abstract="true">
		<xs:sequence>
			<xs:element name="dictionary_definition" type="CLASS_REFERENCE_Type"/>
			<xs:element name="content_version" type="VERSION_TYPE_Type" minOccurs="0"/>
			<xs:element name="content_revision" type="REVISION_TYPE_Type" minOccurs="0"/>
			<xs:element name="recommended_presentation"
type="RECOMMENDED_PRESENTATION_Type" minOccurs="0"/>
			<xs:element name="classification" type="CLASSIFICATION_Type" minOccurs="0"/>
			<xs:element name="instance_identification" type="PROPERTIES_REFERENCE_Type"/>
			<xs:element name="population" type="cat:Detail"/>
			<xs:element name="table_like" type="xs:boolean"/>
		</xs:sequence>
	</xs:complexType>
```

```
        <xs:complexType name="CLASS_PRESENTATION_ON_PAPER_Type">
                <xs:sequence>
                        <xs:element name="illustration" type="ILLUSTRATION_Type" maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="CLASS_PRESENTATION_ON_SCREEN_Type">
                <xs:sequence>
                        <xs:element name="illustration" type="ILLUSTRATION_Type" maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="CLASSIFICATION_Type">
                <xs:sequence>
                        <xs:element name="property_classification" type="PROPERTY_CLASSIFICATION_Type"
maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="CONTEXT_PARAM_ICON_Type">
                <xs:sequence>
                        <xs:element name="a6_illustration" type="ILLUSTRATION_Type"
maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="CREATE_ICON_Type">
                <xs:sequence>
                        <xs:element name="a6_illustration" type="ILLUSTRATION_Type"
maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="EXPLICIT_FUNCTIONAL_MODEL_CLASS_EXTENSION_Type">
                <xs:complexContent>
                        <xs:extension base="CLASS_EXTENSION_Type">
                                <xs:sequence>
                                        <xs:element name="referenced_representation"
type="PROPERTY_REFERENCE_Type" minOccurs="0"/>
                                        <xs:element name="available_views_icon" type="ILLUSTRATION_Type"
minOccurs="0"/>
                                        <xs:element name="available_views_msg" type="MESSAGE_Type"
minOccurs="0"/>
                                        <xs:element name="context_param_icon"
type="CONTEXT_PARAM_ICON_Type" minOccurs="0"/>
                                        <xs:element name="context_param_msg" type="MESSAGE_Type"
minOccurs="0"/>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="EXPLICIT_ITEM_CLASS_EXTENSION_Type">
                <xs:complexContent>
                        <xs:extension base="CLASS_EXTENSION_Type">
                                <xs:sequence>
                                        <xs:element name="access_icon" type="ILLUSTRATION_Type"
minOccurs="0"/>
                                        <xs:element name="content_msg" type="MESSAGE_Type" minOccurs="0"/>
                                        <xs:element name="create_icon" type="CREATE_ICON_Type"
minOccurs="0"/>
                                        <xs:element name="create_msg" type="MESSAGE_Type" minOccurs="0"/>
                                        <xs:element name="class_presentation_on_paper"
type="CLASS_PRESENTATION_ON_PAPER_Type" minOccurs="0"/>
                                        <xs:element name="class_presentation_on_screen"
type="CLASS_PRESENTATION_ON_SCREEN_Type" minOccurs="0"/>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="PROPERTY_VALUE_RECOMMENDED_PRESENTATION_Type"
abstract="false">
```

```
                <xs:sequence>
                        <xs:element name="prop_def" type="PROPERTY_REFERENCE_Type"/>
                        <xs:element name="recommended_presentation_unit" type="UNIT_Type"/>
                        <xs:element name="recommended_presentation_format"
type="VALUE_FORMAT_TYPE_Type"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="PROPERTY_CLASSIFICATION_Type" abstract="false">
                <xs:sequence>
                        <xs:element name="its_value" type="xs:integer"/>
                        <xs:element name="prop_def" type="PROPERTY_REFERENCE_Type"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="RECOMMENDED_PRESENTATION_Type">
                <xs:sequence>
                        <xs:element name="property_value_recommended_presentation"
type="PROPERTY_VALUE_RECOMMENDED_PRESENTATION_Type" maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
</xs:schema>
```

B.18 library.xsd

```
<xs:schema xmlns="urn:iso:std:iso:cd:13584:-32" xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:iso:std:iso:cd:13584:-32" elementFormDefault="unqualified"
attributeFormDefault="unqualified">
        <xs:include schemaLocation="content.xsd"/>
        <xs:complexType name="CONTAINED_CLASS_EXTENSIONS_Type">
                <xs:sequence>
                        <xs:element name="class_extension" type="CLASS_EXTENSION_Type"
maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="LIBRARY_IN_STANDARD_FORMAT_Type">
                <xs:complexContent>
                        <xs:extension base="LIBRARY_Type"/>
                </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="LIBRARY_Type" abstract="false">
                <xs:sequence>
                        <xs:element name="contained_class_extensions"
type="CONTAINED_CLASS_EXTENSIONS_Type" maxOccurs="unbounded"/>
                        <xs:element name="responsible_supplier" type="SUPPLIER_REFERENCE_Type"/>
                </xs:sequence>
        </xs:complexType>
</xs:schema>
```

Annex C (normative)
Standard data requirements for OntoML

Standard data are the XML elements that shall be recognized by any implementation that claims conformance to some conformance class of OntoML.

Standard data shall be specified for this library integrated information model for each conformance class.

C.1 Conformance class specification table

Table C.1 specifies the values of the **name** and **application** XML elements defined in the **LIBRARY_IIM_IDENTIFICATION_Type** XML complex type that are allowed for use in a **LIBRARY_IIM_IDENTIFICATION_Type** to reference OntoML in either of its conformance classes.

Table C.1 – ISO 13584 LIIM 32 conformance class specification

Conformance Class	LIBRARY_IIM_IDENTIFICATION_Type name XML element mandatory value	LIBRARY_IIM_IDENTIFICATION_Type application XML element mandatory value
1	'ONTOML'	'1'
2	'ONTOML'	'2'
3	'ONTOML'	'3'
4	'ONTOML'	'4'

C.1 Standard data for conformance classes 1 to 4

The **LIBRARY_IIM_IDENTIFICATION_Type** XML complex type values allowed for use in a library delivery file conform to OntoML defined in this part of ISO 13584 shall obey the constraints defined in this clause.

An informal constraint is defined on the **LIBRARY_IIM_IDENTIFICATION_Type** XML complex type to be allowed for use to reference conformance class 1 to 4 of OntoML defined in this part of ISO 13584.

A **LIBRARY_IIM_IDENTIFICATION_Type** XML complex type is allowed for use to reference conformance class 1 to 4 of OntoML if the following conditions hold:

— the **name** XML element of the **LIBRARY_IIM_IDENTIFICATION_Type** XML complex type that references library integrated model LIIM 32 shall be equal to 'ONTOML', and

— the **status** XML element of the **LIBRARY_IIM_IDENTIFICATION_Type** XML complex type shall be equal to 'WD', 'CD' or 'DIS',-'FDIS', 'IS', 'TS', 'PAS' or 'ITA'-and

— the **application** XML element of the **LIBRARY_IIM_IDENTIFICATION_Type** XML complex type shall be equal to ‘1', '2', '3', or '4'.

Annex D (normative)
Structural transformation of the CIIM model from OntoML XML Schema to EXPRESS

This annex specifies how the OntoML constructs shall be converted into EXPRESS data. Such a conversion allows to build XML tools that generate EXPRESS representations of OntoML document instances in order to check the semantic consistency of its content with respect to the integrity constraints defined in the CIIM.

D.1 Difference between OntoML and CIIM information elements

OntoML is an XML Schema allowing to represent, within an XML document instance, all the information elements represented in a CIIM EXPRESS physical file.

As a rule, all the information elements of a CIIM EXPRESS physical file are explicitly represented in an OntoML document instance, and all the constraints that these information elements are supposed to fulfill must also hold for OntoML document instances content.

Nevertheless, four differences were decided during the OntoML design:

1. sharing *Vs* duplication of some information elements

Both in OntoML and in CIIM EXPRESS physical files, ontology concepts are defined only once and referenced several times. Concerning the other pieces of information of the CIIM:

— in EXPRESS several ontology concepts may share some EXPRESS entity.

EXAMPLE The **item_names** entity data type is shared some ontology cocnepts.

NOTE 1 **item_names** is defined in ISO 13584-42:2003, Clause D.3.8.2.5.

— in XML, it was decided that each ontology concept will only reference other ontology concepts. All the others pieces of information that are referenced by an ontology concept in a CIIM EXPRESS physical file are embedded in XML. Thus, their content is possibly duplicated if the same piece of information is referenced by several CIIM ontology concept.

NOTE 2 The duplication of pieces of information does not change the semantics of the underlying CIIM EXPRESS data model.

2. Use of pre-existing XML capabilities to characterize internet resources

In the CIIM, some powerful but complex EXPRESS mechanisms were defined in order to be able to represent both the external files information and the way to process them. In OntoML, this representation has been replaced by the use of the MIME protocol mechanism that is sufficient to interpret external files content.

3. Removing some constraints that cannot be checked in XML

In the CIIM model, the **prefix_ordered_class_list** constraint stipulates that classes contained in an OntoML document instance are sorted in such a way that no forward reference from one class to some other classes appear. Due to the fact that such a constraint cannot be checked in XML, it is removed from the OntoML specification and OntoML document instances are not supposed to fulfil this constraint. If needed, this order may be compiled and ensured, when the OntoML content is translated into EXPRESS for constraint checking.

4. Simplification of the CIIM model

The following simplification were assumed.

— we assume that when a document is visible, it is also applicable;

— translations representations have been simplified;

— resources for the representation of documentation that can be associated to any ontology concept has been simplified;

— removing of CIIM constructs that are not pertinent in the OntoML context;

EXAMPLE The entity instance type CIIM data type has been omitted in the OntoML type system.

— simplification of the representation of CIIM global identifiers (known as basic semantic units) of ontology concepts.

D.2 Introductory example

This annex of OntoML specifies a formal mapping from the XML pieces of information, included in an OntoML document instance, to those EXPRESS pieces of information that would be included in a CIIM EXPRESS physical file used to exchange the same information.

This clause identifies, on a very simple example, the various components of such a mapping.

Let's consider the following UML model used to illustrate a simple information model (Figure D.1):

NOTE 1 Such an information model could be represented in EXPRESS.

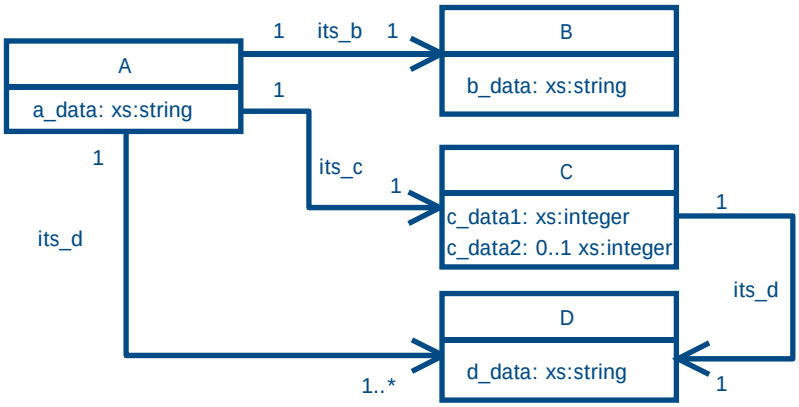

Figure D.1 – A UML information model example

A composite class *A* is defined through a *a_data* string attribute, and an *its_b*, an *its_c* and an *its_d* attributes whose the data types are respectively classes *B*, *C* and *D*. Class *B* is described through a *b_data* string attribute. Class *C* is described through a *c_data1 and* an optional *c_data2* integer attributes, and an *its_d* attributes whose the data type is class *D*. Class *D* is described through a *d_data* string attribute.

The OntoML-like representation of this information model, using the same transformation principles than those used to design OntoML from the CIIM EXPRESS model, could be as illustrated in the following UML-like model (Figure D.2).

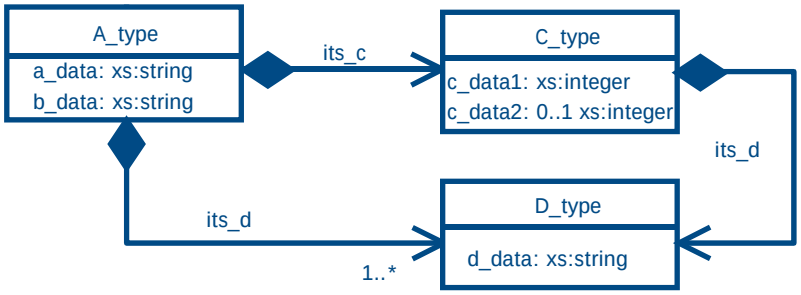

Figure D.2 – An UML-like representation of the information model

NOTE 2 Notations used in this UML-like model are those defined in Clause 5.3.1.

A corresponding XML Schema representation, could be the following (Figure D.3):

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:element name="a" type="A_type"/>
    <xs:complexType name="A_type">
        <xs:sequence>
            <xs:element name="a_data" type="xs:string"/>
            <xs:element name="b_data" type="xs:string"/>
            <xs:element name="its_c" type="C_type"/>
            <xs:element name="its_d" type="its_D_type">
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="C_type">
        <xs:sequence>
            <xs:element name="c_data1" type="xs:int"/>
            <xs:element name="c_data2" type="xs:int" minOccurs="0"/>
            <xs:element name="its_d" type="D_type">
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="its_D_type">
        <xs:sequence>
            <xs:element name="d" type="D_type" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="D_type">
        <xs:sequence>
            <xs:element name="d_data" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

Figure D.3 – An XML Schema example

The *a* XML element has been defined for playing both the role of a data container and the role of the root of the XML document: its type is the *A_type* XML complex type.

Thus it is possible to represent the same information using two representation formats: the ISO 10303-21 EXPRESS instance syntax on the base of the UML model defined in Figure D.1, and the XML document syntax on the base of the XML Schema defined in Figure D.3. Table D.1 represents an example of these two representation formats.

Table D.1 – XML and corresponding ISO 10303 instances

XML document	ISO 10303-21 instances
<a> <a_data>string 1</a_data> <b_data>string 2</b_data> <its_c> <c_data1>10</c_data1> <c_data2>20</c_data2> <its_d> <d_data>string 3</d_data> </its_d> </its_c> <its_d> <d> <d_data>string 4</d_data> </d> <d> <d_data>string 5</d_data> </d> </its_d> 	#1=A('string 1', #2, #6, (#4, #5)); #2=D('string 2'); #3=D('string 3'); #4=D('string 4'); #5=D('string 5'); #6=C(10, 20, #3);

We note that defining mapping rules from each XML piece of information into EXPRESS pieces of information needs the following:

— capability to instanciate the EXPRESS representation of the container XML element, i.e., the *a* XML element, and to reference it;

— capability, in the XML document, to identify any piece of information embedded, directly or indirectly, within the *a* container and, in the EXPRESS file, to instanciate and to identify any piece of information referenced directly or indirectly from the *a* entity instance;

— capability to express how each particular value represented in XML must be converted to be represented in EXPRESS.

In the mapping defined in this annex:

— all the mappings start from an embedding XML element that is either an ontology concept or the dictionary root element. The EXPRESS image of this embedding element is denoted **SELF**.

— identification of XML embedded element uses the location where the mapping rules are represented in OntoML (Clause D.3) and, when need XPath / XSLT notations.

— identification of EXPRESS items referenced by the EXPRESS image of the embedding XML element use the EXPRESS path syntax, starting from **SELF** (see Clause D.4).

— instance creation and value representation use a set of specific functions defined in Clause D.4.

Moreover, clause D.3 defines the overall structure of OntoML embedding elements.

D.3 Mapping rules location in OntoML

Every OntoML element definition is associated to a (some) mapping rule(s). This mapping rule is expressed using the *annotation* element proposed by the XML Schema specification. Figure D.4

illustrates the mapping location that would be defined in the *A_Type* XML complex type of the XML Schema example proposed in Figure D.3.

```
<xs:complexType name="A_type">
        <xs:sequence>
                <xs:element name="a_data" type="xs:string">
                        <xs:annotation>
                                <xs:appinfo> mapping rule(s)</xs:appinfo>
                        </xs:annotation>
                </xs:element>
                <xs:element name="b_data" type="xs:string">
                        <xs:annotation>
                                <xs:appinfo> mapping rule(s)</xs:appinfo>
                        </xs:annotation>
                </xs:element>
                <xs:element name="its_c" type="C_type">
                        <xs:annotation>
                                <xs:appinfo> mapping rule(s)</xs:appinfo>
                        </xs:annotation>
                </xs:element>
                <xs:element name="its_d" type="its_D_type">
                        <xs:annotation>
                                <xs:appinfo> mapping rule(s)</xs:appinfo>
                        </xs:annotation>
                </xs:element>
        </xs:sequence>
</xs:complexType>
```

Figure D.4 – Mapping representation in OntoML

The *appinfo* sub-element of the *annotation* element is intended to contain all the mapping rules.

The mapping rules involves two aspects:

— the link between an XML element and the corresponding EXPRESS attribute in the ISO 13584 data model: it is specified by EXPRESS target paths (using the group reference and attribute reference mechanism defined in ISO 10303-11); it is the *localization part* of the mapping;

— the assignment of the XML element value(s) to the corresponding EXPRESS item(s) in the CIIM; it is the *value-conversion part* of the mapping.

D.4 Link between an OntoML element and a CIIM attribute

The OntoML to CIIM mapping is based on the definition of EXPRESS-based CIIM target paths and XPath-based OntoML source paths. These paths are used for expressing the correspondence between an XML element and an EXPRESS item.

D.4.1 OntoML source path

The XML source path is defined by the location of the annotation. Figure D.5 outlines the XML source path concept:

```
<xs:element name="b_data" type="xs:string">
    <xs:annotation>
        <xs:appinfo>mapping rule(s)</xs:appinfo>
    </xs:annotation>
</xs:element>
```

Figure D.5 – XML source Path

This annotation being assigned to the *b_data* element, the source path is the *B_data* element.

D.4.2 EXRESS target path

The role of the target path is to localize the target EXPRESS construct that corresponds to the source XML construct defined by the location of the annotation. The target EXPRESS construct is defined by a path that is built using EXPRESS attribute reference mechanism that use the classical dot notation (“.”) used to defined the entity datatype of the entity instance to be created.

NOTE 1 Attribute reference is defined in ISO 10303-11 Clause 12.7.3.

OntoML defines some XML elements as global elements and the other as local elements.

Only global elements are referenced by means of identifiers. All the other elements are embedded within global elements.

Thus, in the EXPRESS representation of an OntoML document instance:

— each OntoML global element will be represented by instantiating a corresponding EXPRESS entity; this instance will be denoted by **SELF** and will define the context in which the embedded XML element of the OntoML global element are mapped.

— each piece of information that is not a global element is embedded within an XML global element and will be mapped onto an EXPRESS image of its embedding XML global element.

This global structure is shown in Figure D.6.

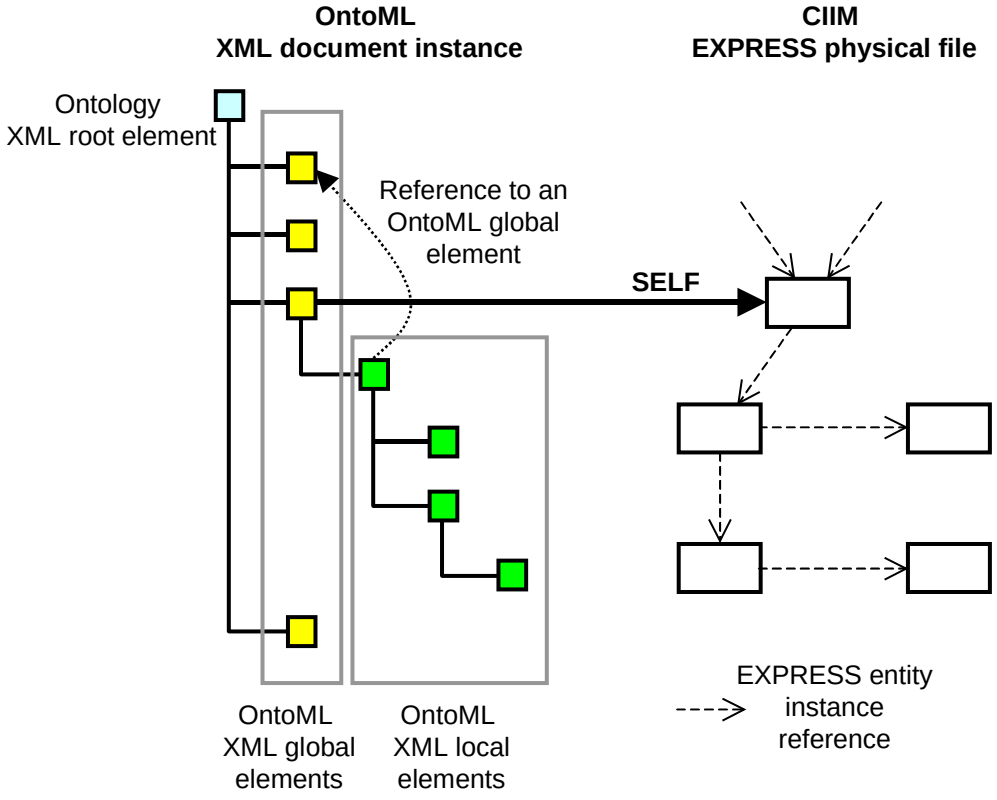

Figure D.6 – Global Vs local XML elements

OntoML specifies seven global elements:

— **supplier**, representing the supplier ontology concept;

— **class**, representing the class ontology concept;

— **property**, representing the property ontology concept;

— **data_type**, representing the named value domain ontology concept;

— **document**, representing the document ontology concept;

— **ontoml**, representing the root element of an ontology and / or library.

The EXPRESS path is defined in a modular way: each embedded element of a local element contains a local path that starts by "." and that contains the EXPRESS path from the EXPRESS image of the XML local element to the EXPRESS attribute corresponding to the embedded element.

The complete path from the EXPRESS image of an XML global element instance to the EXPRESS image of an XML element which is indirectly embedded in the XML global element is built by concatenation of the global path, and of all the local paths encountered when moving in the XML tree structure from the initial XML global element to the final embedded XML element.

The complete path structure is built from an ontology concept instance as defined in Clause D.4.2.1, and from local paths as defined in Clause D.4.2.2. The particular part structure for embedded element belonging to a collection of element is defined in Clause D.4.2.3. The compete path structure built by aggregation of these subpaths is defined in Clause D.4.2.4.

D.4.2.1 Ontology concept instance

In OntoML, XML global elements represent ontology concepts. The mapping of an XML global element to its corresponding EXPRESS image is done through the **SELF** keyword. It represents the instance of an EXPRESS entity, image of the XML global element.

The table below gives the meaning of the CIIM SELF instance depending on the OntoML context where it is defined. The context is defined by a pair: an XML complex type and an XML local element. It is defined in Table D.2.

Table D.2– SELF meaning in its use context

OntoML context	"SELF" CIIM EXPRESS entity type
XML complex type : *CONTAINED_SUPPLIERS_Type* XML element: *supplier*	**supplier_element**
XML complex type : *CONTAINED_CLASSES_Type* XML element: *class*	**class** or one of its subtype
XML complex type : *CONTAINED_PROPERTIES_Type* XML element: *property*	**property_det** or one of its subtype

XML complex type : *CONTAINED_DATATYPES_Type* XML element: *datatype*	**data_type_element**
XML complex type : *CONTAINED_DOUMENTS_Type* XML element: *document*	**document_element**
XML root element: *ontoml*	**dictionary** or one of its subtype

NOTE 1 If the OntoML document instance contains only a dictionary specification, the **SELF** instance is an instance of the **dictionary** or of the **dictionary_in_standard_format** CIIM EXPRESS entity data type.

NOTE 2 If the OntoML document instance contains only a library specification, the **SELF** instance is an instance of the **library** or of the **library_in_standard_format** CIIM EXPRESS entity data type.

NOTE 3 If the OntoML document instance contains both a dictionary and a library specification, the **SELF** instance is an instance of the **library** or of the **library_in_standard_format** CIIM EXPRESS entity data type.

EXAMPLE 1 In OntoML, class ontology concept is represented by the **class** XML global element. In the CIIM EXPRESS information model, this class ontology concept is represented by one instance of the **class** entity data type, or one of its subtypes. This EXPRESS instance is said the ***SELF*** instance. It represents the **class** XML element in an EXPRESS based universe.

XML global elements support polymorphism through the possibly associated **xsi:type** XML attribute/

NOTE 4 *xsi* stands for the prefix associated to the XML Schema:Structure specification that defines several attributes for direct use in XML document. Its namespace is: <http://www.w3.org/2001/XMLSchema-instance>.

It means that the EXPRESS mapping of an XML global element must take into account this datatype information. Consequently, the following applies:

— when no **xsi:type** XML attribute is used to specify an XML global element, the **SELF** instance corresponds to an instance of the EXPRESS entity image of the XML global element complex type specification;

— when an **xsi:type** XML attribute is used to specify an XML global element, the **SELF** instance corresponds to an instance of the EXPRESS entity image of the referenced XML complex type;

EXAMPLE 2 The property ontology concept is partially defined as follows:

```
<xs:element name="property" type="PROPERTY_Type" maxOccurs="unbounded">
        <xs:annotation>
                <xs:documentation>A reference to a property</xs:documentation>
                <xs:appinfo>SELF</xs:appinfo>
        </xs:annotation>
</xs:element>
<xs:complexType name="PROPERTY_Type" abstract="true">
…
</xs:complexType>

<xs:complexType name="CONDITION_DET_Type">
        <xs:complexContent>
                <xs:extension base="PROPERTY_Type"/>
```

```
        </xs:complexContent>
</xs:complexType>

<xs:complexType name="NON_DEPENDENT_P_DET_Type">
        <xs:complexContent>
                <xs:extension base="PROPERTY_Type"/>
        </xs:complexContent>
</xs:complexType>
```

Let's now consider the following OntoML fragment:

```
                <property xsi:type="NON_DEPENDENT_P_DET_Type" …>
```

In such a case, the **SELF** instance is an instance of the EXPRESS image associated to the OntoML **NON_DEPENDENT_P_DET_Type** XML complex type, i.e., an instance of the CIIM **non_dependent_p_det** EXPRESS entity data type.

D.4.2.2 Information elements of an ontology: local EXPRESS target path

A local CIIM target path is assigned to every local XML element defined in the XML complex type assigned either to a global XML element or to another local XML element.

It specifies a mapping link between the EXPRESS image of a local XML element and an EXPRESS attribute.

NOTE 1 This EXPRESS target path is said local, because it defines only locally the partial mapping.

The local CIIM target path has the structure defined in Figure D.7:

.local_path.attribute

Figure D.7 – Local EXPRESS target path structure

Where:

— *"."*: the instance of the EXPRESS entity image of the XML complex type used to represent the local XML element;

NOTE 2 If the complex type represents the specification of a global XML element, "." Represents the SELF instance.

— *sub_path*: the path defined from the EXPRESS entity instance to the EXPRESS entity instance where the target attribute is defined;

— *attribute*: the name of the EXPRESS attribute to which the global CIIM path defines a mapping.

EXAMPLE 1 Real measures may be used to represent either a property value domain. In OntoML, real measures are represented on the base of the **REAL_MEASURE_TYPE_Type** XML complex type. This complex type defines a **unit** local XML element representing the specific unit of the real measure. The mapping of the unit XML element to the corresponding EXPRESS attribute is specified as follows:

.UNIT

The **"."** instance is an instance of the EXPRESS image associated to the OntoML **REAL_MEASURE_TYPE_Type** XML complex type, i.e., an instance of the CIIM **real_measure_type** EXPRESS entity data type.

D.4.2.3 Local EXPRESS target path for indexing a collection of XML elements

Some XML elements are defined as collections of other embedded XML elements. They correspond to EXPRESS attributes whose types are collections. Each of their embedded element corresponds to one element of the corresponding EXPRESS collection. Therefore, it is needed to specify a mapping between each embedded XML element and the corresponding EXPRESS collection element.

For that purpose, EXPRESS paths are completed using the EXPRESS aggregate indexing operator.

NOTE 1 Aggregate indexing operator is defined in ISO 10303-11, Clause 12.6.1.

It consists of the collection value being indexed (the target EXPRESS attribute) and the index specification. The index specification is either set to a “i” integer variable or to an integer constant.

In case of an integer variable, its range is implicitly defined as follows:

— its minimum value is equal to the minimum bound of the target collection structure if it is an array, else is it is equal to 1;

— its maximum value is equal to the minimum value, minus one, plus the number of embedded XML elements that appear in the XML element for which the mapping is defined.

EXAMPLE This example illustrates an EXPRESS path for indexing collection of XML elements.

```
<xs:element name="translation" type="TRANSLATION_Type" minOccurs="0">
        <xs:annotation>
                <xs:appinfo>.ADMINISTRATION.TRANSLATION[i]</xs:appinfo>
        </xs:annotation>
</xs:element>
<xs:complexType name="TRANSLATION_Type">
        <xs:sequence>
                <xs:element name="translation_data" type="TRANSLATION_DATA_Type"
                        maxOccurs="unbounded"/>
        </xs:sequence>
</xs:complexType>
<xs:complexType name="TRANSLATION_DATA_Type" abstract="false">
        <xs:sequence>
                …
                <xs:element name="translation_revision" type="REVISION_TYPE_Type">
                        <xs:annotation>
                                <xs:appinfo>.TRANSLATION_REVISION</xs:appinfo>
                        </xs:annotation>
                </xs:element>
                …
        </xs:sequence>
</xs:complexType>
```

The OntoML **translation** element corresponds (**.ADMINISTRATION.TRANSLATION[i]**) to the EXPRESS **translation** attribute defined in the **administrative_data** EXPRESS entity data type and referenced by the **administration** EXPRESS attribute. The EXPRESS **translation** attribute datatype is a collection of **translation_data** EXPRESS entity instances. Each EXPRESS entity instance is described by a set of attributes. Among them, there is the **translation_revision** EXPRESS attribute. In OntoML, each item of the **translation** XML element collection is represented by an embedded **translation_data** XML element. The **translation_data** XML element is defined according to the **TRANSLATION_DATA_Type** XML complex type specification that describes a **translation_revision** XML element whose EXPRESS image is the previously described **translation_revision** EXPRESS attribute (**.TRANSLATION_REVISION**).

NOTE 2 Indexing a 2D aggregate structure would be expressed as follows: *attribute_name*[i][j].

D.4.2.4 Complete EXPRESS target path structure

A complete EXPRESS target path specifies the EXPRESS path from the **SELF** instance to the EXPRESS attribute that is the image of a final XML local element.

The complete path from the EXPRESS image of an XML global element instance to the EXPRESS image of an XML element (which is indirectly embedded in the XML global element) is built by concatenation of the **SELF** instance, and of all the local paths encountered when moving in the XML tree structure from the initial XML global element to the final embedded XML element.

The complete CIIM target path has the structure defined in Figure D.8:

SELF.sub_path.attribute

Figure D.8 – Complete EXPRESS target path structure

Where:

— *SELF*: the EXPRESS instance corresponding to the global OntoML element;

— *sub_path*: the concatenation of all the local paths encountered when moving in the XML tree structure from the initial XML global element to the final embedded XML element;

— *attribute*: the name of the target EXPRESS attribute corresponding to the final embedded XML element.

EXAMPLE 1 A class is an ontology concept that is associated with a name. In the CIIM EXPRESS information model, this name is represented in the **ITEM_NAMES** entity datatype by a **preferred_name** attribute. In OntoML, this ontology concept is represented by a **class** global XML element whose content model is defined by a **CLASS_Type** complex type. This complex type specifies an XML element called **preferred_name** that represents this class name. The complete EXPRESS target path would be implicitly defined as follows:

SELF.NAMES.PREFERRED_NAME

This complete CIIM target path specifies the link between the **preferred_name** element of an OntoML class ontology concept and the **preferred_name** attribute of the EXPRESS **item_names** entity. **SELF** represents a class entity instance (or an instance of one of its subtypes).

EXAMPLE 2 Assuming that the translation XML local element presented in the example defined in Clause D.4.2.3, is directly embedded in an XML global element representing for instance a property ontology concept, the complete path that leads to the translation_revision XML local element would be:

SELF.ADMINISTRATION.TRANSLATION[i].TRANSLATION_REVISION

D.4.3 Assignment of an OntoML element value to an EXPRESS attribute

Assigning an OntoML element value to an EXPRESS attribute requires to specify:

OntoML source and an EXPRESS target attribute paths defining the information elements to be mapped;

— an assignment operator;

— a syntax for accessing information units in the OntoML compliant XML instance document;

— specific EXPRESS constructors for those information elements that are not represented in a straightforward manner in OntoML according to the CIIM.

This clause specified all these assignment aspects.

D.4.3.1 Assignment operator

The assuagement of a value represented in an OntoML document to an EXPRESS target path is performed using the ":=" assignment operator.

D.4.3.2 Assignment operation

The assignment of an EXPRESS value to an EXPRESS target path is only defined for those XML element that define the final target of the corresponding complete EXPRESS target path. The assignment is performed using the following syntax:

EXPRESS target path := EXPRESS value

Where:

— *EXPRESS target path*: a local target path assigned to an XML element;

NOTE 1 No other mapping is defined for the embedded XML element of this XML element

— *EXPRESS value*: a value being either referenced (simple value) or computed (complex value) by a specific mapping function.

EXAMPLE 1 The revision number of class ontology concept is represented by an EXPRESS attribute called **revision** whose data type is a simple string. It is represented in OntoML by an XML element (**revision**) whose content model is also a simple string. The mapping between the OntoML representation of a revision consists of assigning the OntoML **revision** string value to the EXPRESS **revision** attribute.

EXAMPLE 2 The preferred name of a class ontology concept is represented by an EXPRESS attribute called preferred_name whose data type is the **translatable_label** entity. It is represented in OntoML by an XML element (**preferred_name**) whose content model is not represented using the same constructs for simplification purposes. Consequently, the mapping can not be represented simply using an EXPRESS target path. A specific mapping function must be used.

D.4.3.3 Retrieving OntoML information

Defining a mapping between an OntoML document instance and EXPRESS instances requires to retrieve the XML document data, and then to assign them to EXPRESS entity attributes.

For that purpose, OntoML mapping rules use the XPath syntax. Every XPath is defined locally to the XML element for which the mapping rule is defined. The XPath syntax is restricted to the following constructs:

— **.**: returns the current node;

— ***@attribute_name****:* returns the <*attribute_name*> attribute value of the current node*;*

— ***:** returns all the children elements of the current element, whatever be their names.

— ***/***: separator used to specify the XPath localization steps;

— ***element_name***: returns all the <*element_name*> children nodes of the contextual node.

D.4.3.3 Assigning OntoML information to EXPRESS target paths

This clause defines the different means used to assigned values referenced from an XML document to EXPRESS target paths.

D.4.3.3.1 Assigning a simple OntoML value to a simple EXPRESS attribute

An XML simple value to an EXPRESS attribute assignment is implicitly done for those XML element whose content model is defined as simple. For simplification purposes, such a simple assignment does not require to use the assignment operator.

EXAMPLE The date of original definition of an ontology concept is mapped as follows:

```
<xs:element name="class" type="CLASS_Type" maxOccurs="unbounded">
        <xs:annotation>
                <xs:appinfo>SELF</xs:appinfo>
        </xs:annotation>
</xs:element>
<xs:complexType name="CLASS_Type" abstract="true">
        <xs:sequence>
        …
           <xs:element name="date_of_original_definition" type="DATE_TYPE_Type" minOccurs="0">
                <xs:annotation>
           <xs:appinfo>.TIME_STAMPS.DATE_OF_ORIGINAL_DEFINITION</xs:appinfo>
                </xs:annotation>
           </xs:element>
        …
</xs:complexType>
```

Let's now consider this OntoML fragment:

```
<class xsi:type="…" id="…">
…
        <revision>1</revision>
…
</class>
```

The mapping of the OntoML revision value to the corresponding complete EXPRESS target path is implicitly defined as follows:

SELF.TIME_STAMPS.DATE_OF_ORIGINAL_DEFINITION := 1

D.4.3.3.2 Assigning an OntoML value to a complex EXPRESS attribute

Some information elements are not represented in a straightforward manner in OntoML according to the CIIM. The mapping can not be expressed defining only simple EXPRESS target paths and assigning simple XML element values. It requires to specify mapping functions. Their role is to retrieve information from the OntoML document, to process it, and to assign an instance value to the attribute referenced by the complete EXPRESS target path. Their corresponding algorithm may be more or less complex.

NOTE The mapping does not define the algorithm of each of these mapping functions, but only their signature and their behavior.

A complex EXPRESS attribute is an attribute whose data type is an entity datatype. The assignment of an OntoML value to such an attribute requires to build a type-compatible instance. The general structure of such an assignment is defined as follows:

EXPRESS target path := <function_name>({parameters})

Where

— *EXPRESS target path*: a local target path assigned to an XML element, the targeted attribute datatype being an EXPRESS entity;

— <*function_name*>: a mapping function whose role is to create a set of EXPRESS instances and to assign one of them to the attribute referenced in the EXPRESS target path;

— {*parameters*}: the set of effective parameters corresponding to XML items (element or attribute) value retrieved (using XPath operator) from the OntoML document instance.

EXAMPLE The preferred name of a class ontology concept is represented by an EXPRESS attribute called preferred_name whose data type is the **translatable_label** entity. It is represented in OntoML by an XML element (**preferred_name**) whose content model is not represented using the same constructs for simplification purposes. Consequently, the mapping can not be represented simply using an EXPRESS target path. A specific mapping function is used:

```
<xs:element name="class" type="CLASS_Type" maxOccurs="unbounded">
    <xs:annotation>
        <xs:appinfo>SELF</xs:appinfo>
    </xs:annotation>
</xs:element>
<xs:complexType name="CLASS_Type" abstract="true">
    <xs:sequence>
    …
        <xs:element name="preferred_name" type="PREFERRED_NAME_Type">
            <xs:annotation>
                <xs:appinfo>.NAMES.PREFERRED_NAME := createLabel(*)</xs:appinfo>
            </xs:annotation>
        </xs:element>
    …
</xs:complexType>
```

Let's now consider this OntoML fragment:

```
<class xsi:type="…" id="…">
…
    <preferred_name>
        <label language="en">bearing</label>
    </preferred_name>
…
</class>
```

The mapping of the OntoML preferred name complex value to the corresponding complete EXPRESS target path is implicitly defined as follows:

SELF.NAMES.PREFERRED_NAME := createLabel(*)

The EXPRESS **preferred_name** attribute value is intended to be built using the **createLabel** mapping function. This function takes as an effective parameter a set of nodes referenced by the "*" XPath (the set of children nodes of the contextual XML element, i.e., the nodes children of the **preferred_name** XML element), and process (create required EXPRESS instances) it according to the CIIM. It returns a type compatible EXPRESS **preferred_name** attribute value (a **translated_label** entity instance) that is assigned to the specified complete EXPRESS target path.

The following subclauses specify these mapping functions.

D.4.3.3.2.1 BSU from an ontology concept identifier mapping

Every ontology concept is associated to an unambiguous identifier whose structure is defined in this part of ISO13584.

In OntoML, these identifiers are represented by a string, whereas they are structurally and descriptively specified in the CIIM. Thus, we define a function that is intended to build the EXPRESS

entity data type instance resources for representing an ontology concept identifier from an identifier represented by a string. Its signature is the following:

<ontologyConcept>BSUFromId(ontoMLId: string): basic_semantic_unit

where:

— <*ontologyConcept*>: the specific ontology concept for which the CIIM identifier is built. It may take the following values:

  — supplier ontology concept: “supplier”;

  — class ontology concept: “class”;

  — property ontology concept: “property”;

  — datatype ontology concept: “datatype”;

  — document ontology concept: “document”;

— *ontoMLId*: a string representing an OntoML concept identifier;

NOTE 1 The **ontoMLId** effective value is intended to be retrieved from an OntoML document instance using XPath localization path.

— *basic_semantic_unit*: the instance type returned by this function call.

NOTE 2 **basic_semantic_unit** is defined in ISO13584-42, clause D.3.3.2.1.

Depending on the <ontologyconcept>, the **<ontologyConcept>BSUFromId** function returns an instance of:

— the **supplier_BSU** entity datatype for the supplier ontology concept;

— the **class_BSU** entity data type for the class ontology concept;

— the **property_BSU** entity data type for the property ontology concept;

— the **datatype_BSU** entity data type for the datatype ontology concept;

— the **document_BSU** entity data type for the document ontology concept.

If the OntoML identifier has already been mapped in an instance of one of these CIIM entity datatype, it shall not be created twice, but its corresponding CIIM entity instance shall be retrieved and returned by the function.

Additionally, depending on the ontology concept identified, the following apply:

— class ontology concept: if the identified supplier in the OntoML class identifier has not already been mapped into an EXPRESS entity instance, it shall be created and then referenced, otherwise, the existing EXPRESS entity instance shall be only referenced;

— property, datatype or document ontology concept: if the identified supplier and the identified class in the OntoML property, datatype or document identifier have not already been mapped into EXPRESS entity instances, they shall be created and then referenced, otherwise, the existing EXPRESS entity instances shall be only referenced.

Table D.3 lists OntoML ontology concept identifiers (see Clause 8.1) and their corresponding CIIM EXPRESS instances.

Table D.3 – OntoML identifiers mapping

OntoML identifiers	EXPRESS instances
`SupplierId ::= icd oi [opi [opis]] [std]`	`#supp=SUPPLIER_BSU(CIIMrai, *);` *CIIMrai* is built from *supplierId* according to ISO 13584-26 rules.
`classId ::= rai # di #vi`	`#cl=CLASS_BSU(di, vi, #supp);` *#supp* is a reference to an instance of a **supplier_BSU** identified in the *rai* part of the *dictionaryId* identifier
`propertyId ::= rai # di # vi`	`#prop=PROPERTY_BSU(diProp, vi, #cl);` *diProp* is the CIIM property code identified in the *di* part of the OntoML *propertyId*. *#cl* is a reference to an instance of a **class_BSU** identified in the *rai* and *di* part of the OntoML *propertyId*.
`documentId ::= rai # di # vi`	`#doc=DOCUMENT_BSU(diDoc, vi, #cl);` *diDoc* is the CIIM document code identified in the *di* part of the OntoML *documentId*. *#cl* is a reference to an instance of a **class_BSU** identified in the *rai* and *di* part of the OntoML *documentId*.
`datatypeId ::= rai # di # vi`	`#type=DATA_TYPE_BSU(diType, vi, #cl);` *diType* is the CIIM data type code identified in the *di* part of the OntoML *datatypeId*. *#cl* is a reference to an instance of a **class_BSU** identified in the *rai* and *di* part of the OntoML *datatypeId*.

EXAMPLE The class ontology concept identifier is represented as follows:

```
<xs:element name="class" type="CLASS_Type" maxOccurs="unbounded">
        <xs:annotation>
                <xs:appinfo>SELF</xs:appinfo>
        </xs:annotation>
</xs:element>
<xs:complexType name="CLASS_Type" abstract="true">
        …
        <xs:attribute name="id" type="ClassId" use="required">
                <xs:annotation>
                        <xs:appinfo>.IDENTIFIED_BY := classBSUFromId(string(@id))</xs:appinfo>
                </xs:annotation>
```

```
        </xs:attribute>
    </xs:complexType>
```

The complete EXPRESS target path defined for the **id** (the class ontology concept identifier) XML attribute and its corresponding assignment is therefore:

SELF.IDENTIFIED_BY := classBSUFromId(string(@id))

It means the **identified_by** EXPRESS attribute of the **SELF** instance (an instance representing the class ontology concept) is set to the value returned by the **classBSUFromId** function. This function takes as an argument the result of the defined XPath (“string(@id)”), i.e., the OntoML **id** attribute value.

Let’s now consider this OntoML fragment:

```
<class ... id="0002-38491502100024#BEARING#001">
...
</class>
```

If we assume that a **supplier_BSU** entity instance has already been created (*#supp*), the mapping would look like:

```
#cl = CLASS_BSU('BEARING', '001', #supp);
SELF.IDENTIFIED_BY := #cl
```

Where

— *#cl*: an EXPRESS **class_BSU** instance identifier;

— *#supp*: an EXPRESS **supplier_BSU** instance identifier (assumed to exist).

D.4.3.3.2.2 Class BSUs from class identifiers mapping

The CIIM EXPRESS model requires to reference every defined or referenced classes from the **dictionary** EXPRESS entity, through its **contained_classes** attribute. The **classBSUsFromIds** is used for that purpose. It retrieves all the OntoML class identifiers and returns their corresponding CIIM representation (**class_BSU** instances). Its signature is the following:

classBSUsFromIds(ontoMLIds: XPath): LIST OF UNIQUE class_BSU

where:

— *ontoMLIds*: an XPath allowing to retrieve all the OntoML class identifiers;

NOTE 1 In OntoML, such an identifier is represented by an XML attribute.

— *class_BSU*: the instance type returned by this function call.

NOTE 2 **Class_BSU** is defined in ISO 13584-42, Clause D.3.5.1.1.

NOTE 3 The **classBSUsFromIds** is used only once, in the mapping specification of the **contained_classes** XML element defined in the **DICTIONARY_Type** XML complex type.

Table D.4 presents an application example of the **classBSUsFromIds** function, assuming that this mapping function is defined in the context of an OntoML **contained_classes** XML element, as follows: `.CONTAINED_CLASSES := classBSUsFromIds(*/@id).`

Table D.4 – OntoML list of class identifiers mapping

OntoML	EXPRESS instances
<contained_classes> <class id=”rai#di1#vi” …> … </class> <class id=” rai#di2#vi” …> … </class> … <class id=” rai#din#vi” …> … </class> </contained_classes>	#cl1=CLASS_BSU(di1, vi, #supp); #cl2=CLASS_BSU(di2, vi, #supp); … #cln=CLASS_BSU(din, vi, #supp); *#supp* is a reference to an instance of a **supplier_BSU** identified in the *rai* part of the *dictionaryId* identifier *classBSUsFromIds* function result: [#cl1, #cl2, …, #cln]

D.4.3.3.2.3 Dictionary and library identification mapping

The **dictionaryCodeFromId** function allows to build an EXPRESS **dictionary_identification** entity instance form an OntoML dictionary identifier. Its signature is the following:

DictionaryCodeFromId(ontoMLDicLibId: string): dictionary_identification

Where:

— *ontoMLDicLibId*: an XPath allowing to access the OntoML dictionary and / or library identifier;

NOTE In OntoML, such an identifier is represented by an XML attribute.

— *dictionary_identifier*: the instance type returned by this function call.

NOTE **dictionary_identification** is defined in ISO13584-24, Clause 11.5.

Table D.5 presents the dictionary and library identifier (see Clause 8.1) and its corresponding CIIM EXPRESS representation.

Table D.5 – OntoML ontology identifier mapping

OntoML	EXPRESS instances
ontologyId ::= rai # di #vi	#dic=DICTIONARY_IDENTIFICATION(di, vi, revision, #supp); *#supp* is a reference to an instance of a **supplier_BSU** identified in the *rai* part of the o*ntologyId* identifier. The *revision* attribute is not mapped by this function.

D.4.3.3.2.4 Label and translated label mapping

The **createLabel** function allows to build the CIIM EXPRESS resources corresponding to some clear label information, possibly translated. Its signature is the following:

createLabel(ontoMLLabel: XPath): translatable_label

Where:

— *ontoMLLabel*: an XPath that references a set of OntoML **label** XML element (possibly associated to a **language** XML attribute) intended to be processed.

— *translatable_label*: the general data type returned by this function call. In case of not translated text, it returns a CIIM **label_type** value. In case of a translated text, it returns a CIIM **translated_label** entity instance value.

NOTE **translatable_label** is defined in ISO13584-42, Clause D.4.1.4.

Table D.6 presents labels and translated labels and their corresponding CIIM EXPRESS representation.

Table D.6 – OntoML label and translated label mapping

OntoML	EXPRESS instances
<…> <label> a label </label> </…>	LABEL('a label') The CIIM representation of a non translated label is a string whose specific data type is **LABEL**.
<…> <label language="en"> a label </label> <label language="fr"> un label </label> </…>	#tlabel=TRANSLATED_LABEL(('a label', 'un label'), #pt); #pt=PRESENT_TRANSLATIONS(('en', 'fr')); The CIIM representation of a translated label consists of two CIIM EXPRESS entity data type instances: **translated_label** and **present_translations**.

EXAMPLE The class ontology concept preferred name is represented as follows:

```
<xs:element name="class" type="CLASS_Type" maxOccurs="unbounded">
        <xs:annotation>
                <xs:appinfo>SELF</xs:appinfo>
        </xs:annotation>
</xs:element>
<xs:complexType name="CLASS_Type" abstract="true">
        …
        <xs:element name="preferred_name" type="PREFERRED_NAME_Type">
                <xs:annotation>
                        <xs:appinfo>.NAMES.PREFERRED_NAME := createLabel(*)</xs:appinfo>
                </xs:annotation>
        </xs:element>
        …
</xs:complexType>
```

```
<xs:complexType name="PREFERRED_NAME_Type">
        <xs:sequence>
                <xs:element name="label" type="PREFERRED_NAME_LABEL_Type"
                        maxOccurs="unbounded"/>
        </xs:sequence>
</xs:complexType>
<xs:complexType name="PREFERRED_NAME_LABEL_Type">
        <xs:simpleContent>
                <xs:extension base="PREFERRED_NAME_TYPE_Type">
                        <xs:attribute name="language" type="LANGUAGE_CODE_Type"
                                use="optional"/>
                </xs:extension>
        </xs:simpleContent>
</xs:complexType>
```

The complete EXPRESS target path defined for the **preferred_name** (the class preferred name) XML element and its corresponding assignment is therefore:

SELF.NAMES.PREFERRED_NAME := createLabel(*)

Let's now consider this OntoML fragment:

```
<class …>
…
        <preferred_name>
                <label language="en">roller bearing</label>
        </preferred_name>
…
</class>
```

The result of the expressed mapping could be as follows:

```
#tlabel = TRANSLATED_LABEL(('roller bearing'), #pt);
#pt = PRESENT_TRANSLATIONS(('en'));
SELF.NAMES.PREFERRED_NAME := #tlabel
```

D.4.3.3.2.5 Text and translated text mapping

The **createText** function allows to build the CIIM EXPRESS resources corresponding to some clear text information, possibly translated. Its signature is the following:

createText(ontoMLText: XPath): translatable_text

Where:

— *ontoMLText*: an XPath that references a set of OntoML **text** XML element (possibly associated to a **language** XML attribute) intended to be processed.

— *translatable_Text*: the general data type returned by this function call. In case of not translated text, it returns a CIIM **text_type** value. In case of a translated text, it returns a CIIM **translated_text** entity instance value.

NOTE **translatable_text** is defined in ISO13584-42, Clause D.4.1.6.

Table D.6 presents texts and translated texts and their corresponding CIIM EXPRESS representation.

Table D.7 – OntoML text and translated text mapping

OntoML	EXPRESS instances
<…> <text> a text </text> </…>	TEXT('a label') The CIIM representation of a non translated label is a string whose specific data type is **LABEL**.
<…> <text language=”en”> a text </label> <text language=”fr”> un texte </text> </…>	#tlabel=TRANSLATED_TEXT(('a text, 'un texte'), #pt); #pt=PRESENT_TRANSLATIONS(('en', 'fr')); The CIIM representation of a translated text consists of two CIIM EXPRESS entity data type instances: **translated_text** and **present_translations**.

D.4.3.3.2.6 Synonymous names mapping

The **createSynonymous** function allows to build the CIIM EXPRESS resources corresponding to some synonymous labels information, possibly translated. Its signature is the following:

createSynonymous(ontoMLLabel: XPath): SET OF syn_name_type

Where:

— *ontoMLLabel*: an XPath that references a set of OntoML **label** XML element (possibly associated to a **language** XML attribute) intended to be processed.

— *syn_name_type*: the general data type returned by this function call. In case of not translated text, the function returns a set of CIIM **label** type value. In case of a translated text, it returns a set of CIIM **label_with_language** entity instance value.

NOTE **syn_name_type** is defined in ISO13584-42, Clause D.3.8.1.15.

Table D.8 presents synonymous names and translated synonymous names and their corresponding CIIM EXPRESS representation.

Table D.8 – OntoML synonymous and translated synonymous mapping

OntoML	EXPRESS instances
<…> <label> a synonymous </label> <label> another synonymous </label> </…>	[LABEL('a synonymous'), LABEL('another synonymous')] The CIIM representation of a non translated synonymous names is a set of strings whose specific data type is **LABEL**.
<…> <label language="en"> a synonymous </label> <label language="fr"> un synonyme </label> <label language="fr"> un autre synonyme </label> </…>	#syn1=LABEL_WITH_LANGUAGE('a synonymous', 'en'); #syn2=LABEL_WITH_LANGUAGE('un synonyme', 'fr'); #syn3=LABEL_WITH_LANGUAGE('un autre synonyme', 'fr'); #pt=PRESENT_TRANSLATIONS(('en', 'fr')); The CIIM representation of a translated text consists of two CIIM EXPRESS entity data type instances: **label_with_language** and **present_translations**.

D.4.3.3.2.7 Documentation mapping

The association of documentation to ontology concepts may be done in three different ways.

— either by referencing a document represented as an instance of an ontology concept: for that purpose, the **referenced_graphics** and **referenced_document** CIIM EXPRESS entities are used;

— or by referencing an well-identified document: for that purpose, the **identified_document** CIIM EXPRESS entity is used;

— or by referencing an http resource: for that purpose, the **external_graphics**, **document_content**, **message**, **illustration**, **a6_illustration** and the **a9_illustration** CIIM EXPRESS entities are used;

For this latter functions group, protocol and translation information must be provided according to the CIIM.

These CIIM EXPRESS resources may be classified, according to the CIIM, as follows:

— graphics: the EXPRESS resources that represent a CIMM **graphics**: **referenced_graphics** and **external_graphics**;

— documents: the EXPRESS resources that represent a CIMM **document**: **referenced_document** and **identified_document**;

— document ontology concept content specification: the EXPRESS resource that represents a CIIM **document_content**;

— class extension external resources: the EXPRESS resources that represent a CIMM **class_extension_external_item**: **message**, **illustration**, **a6_uillustration** and **a9_illustration**.

Mapping functions are defined according to this classification:

— a mapping function intended to create a graphics:

createGraphics(ontoMLGraphicsType: XPath): graphics

where:

— *ontoMLGraphicsType*: an XPath that references the *xsi:type* attribute of the element to which this function is applied; it allows to specify what kind of graphics is intended to be built;

— *graphics*: the entity data type of the instance returned by this function call; if the function applies to an XML element whose content model is defined as a referenced graphics (**xsi:type = REFERENCED_GRAPHICS_Type**), it returns a CIIM **referenced_graphics** entity instance value; if it applies to an XML element whose content model is defined as a an external graphics (**xsi:type = EXTERNAL_GRAPHICS_Type**), it returns a CIIM **external_graphics** entity instance value;

— a mapping function intended to create a document:

createDocument(ontoMLDocumentType: XPath): document

where:

— *ontoMLDocumentType*: an XPath that references the *xsi:type* attribute of the element to which this function is applied; it allows to specify what kind of document is intended to be built;

— *document*: the entity data type of the instance returned by this function call; if the function applies to an XML element whose content model is defined as a referenced document (**xsi:type = REFERENCED_DOCUMENT_Type**), it returns a CIIM **referenced_document** entity instance value; if it applies to an XML element whose content model is defined as a identified document (**xsi:type = IDENTIFIED_DOCUMENT_Type**), it returns a CIIM **identified_document** entity instance value;

— a mapping function intended to create a document ontology concept content specification:

createDocumentContent(doc: string): document_content

where:

— *doc*: an XPath retrieving the OntoML document identifier for which the document content is specified;

— *document_content*: the entity data type of the instance returned by this function call;

— a mapping function intended to create class extension external resources:

createExtResource(ontoMLExtResourceType: XPath): class_extension_external_item

where:

— *ontoMLExtResourceType*: an XPath that references the *xsi:type* attribute of the element to which this function is applied; it allows to specify what kind of document is intended to be built;

— *class_extension_external_item*: the entity data type of the instance returned by this function call; if the function applies to an XML element whose content model is defined as a message

(**xsi:type = MESSAGE_Type**), it returns a CIIM **message** entity instance value; if it applies to an XML element whose content model is defined as an illustration (**xsi:type = ILLUSTRATION_Type**) where no **standard_size** XML attribute is defined, it returns a CIIM **illustration** entity instance value; if it applies to an XML element whose content model is defined as an illustration (**xsi:type = ILLUSTRATION_Type**) where the **standard_size** XML attribute is defined, it returns a CIIM **a6_illustration** entity instance value if this attribute value is equal to "**a6_illustration**", or a CIIM **a9_illustration** entity instance value if this attribute value is equal to "**a9_illustration**".

These four functions are intended to process the OntoML sub document part that consists of the sub XML elements tree whose root is the OntoML element where one of those mapping functions is called.

Mapping functions building **external_graphics**, **document_content**, **message**, **illustration**, **a6_illustration** and the **a9_illustration** CIIM entity data type instances deal with reference to http resources. According to the CIIM, it is required for each of these functions to build the following entity instances corresponding to the representation of the HTTP protocol. It is presented in Table D.9.

Table D.9 – OntoML HTTP proptocol mapping

OntoML	EXPRESS instances

No explicit information in OntoML.	`#http_prot_name=ITEM_NAMES(LABEL('Hypertext Transfer Protocol'), (), LABEL('HTTP/1.1'), $, $);`
	`#org=ORGANIZATION('', 'World Wide Web Consortium', 'W3C');`
	`#http_protocol=HTTP_PROTOCOL(#org, 'USA', 'NONE', '001', $, #http_prot_name, $, 2621);`
	
	The **http_protocol** instance is intended to be referenced.

Mapping functions building **external_graphics**, **document_content**, **message**, **illustration**, **a6_illustration** and the **a9_illustration** CIIM entity data type instances deal also with translations. According to the CIIM, it is required for each of these functions to build the following entity instances corresponding to the representation of external resource translations. It is represented in Table D.10.

Table D.10 – OntoML translated and not translated files mapping

OntoML	EXPRESS instances
<file> <file_name>filename.ext </file_name> <dir_name>directory name</dir_name> </file> **dir_name** is an optional OntoML element.	#ext_cont=NOT_TRANSLATED_EXTERNAL_CONTENT((#lang_spec_cont)); #lang_spec_cont=LANGUAGE_SPECIFIC_CONTENT((#http_file), #http_file, 'utf-8'); #http_file=HTTP_FILE('filename.ext', $, mime_type, mime_subtype, $, 'filename.ext', # http_class_dir, $); *mime_type* and *mime_subtype* are intended to be computed from the file name. #http_class_dir=HTTP_CLASS_DIRECTORY('directory_name', #class); If the directory name is not provided, it shall be created and managed by the mapping program.
<file language="en"> <file_name>filename_en.ext </file_name> <dir_name>directory_name_1 </dir_name> </file> <file language="fr"> <file_name>filename_fr.ext </file_name> <dir_name>directory_name_2 </dir_name> </file> **dir_name** is an optional OntoML element.	#pt=PRESENT_TRANSLATIONS(('en', 'fr')); #lang_spec_cont_1=LANGUAGE_SPECIFIC_CONTENT((#http_file_1), #http_file_1, "utf-8"); #lang_spec_cont_2=LANGUAGE_SPECIFIC_CONTENT((#http_file_2), #http_file_2, 'utf-8'); #http_file_1=HTTP_FILE('***filename_en.ext***', $, mime_type, mime_subtype, $, 'filename_en.ext', # http_class_dir_1, $); #http_file_2=HTTP_FILE('***filename_fr.ext***', $, mime_type, mime_subtype, $, 'filename_fr.ext', # http_class_dir_2, $); *mime_type* and *mime_subtype* are intended to be computed from the file name. #http_class_dir_1=HTTP_CLASS_DIRECTORY("directory_name_1", #class); #http_class_dir_2=HTTP_CLASS_DIRECTORY("directory_name_2", #class); If the directory name is not provided, it shall be created and managed by the mapping program.

Table D.11 presents for each OntoML documentation constructs the corresponding CIIM EXPRESS representation. When required, they reference CIIM EXPRESS entity instances previously introduced.

Table D.11 – OntoML external resource mapping

OntoML	EXPRESS instances
< … xsi:type= **"REFERENCED_GRAPHICS_Type"**> <graphics_reference document_ref="documentId"/> <…/>	#ref_graph=REFERENCED_GRAPHICS(#doc); #doc is built from the OntoML *documentId*, according to D.4.3.3.2.1. The **createGraphics** function would return the *#ref_graph* entity instance.
< … xsi:type= **"REFERENCED_DOCUMENT_Type"**> <document_reference document_ref="documentId"/> <…/>	#ref_doc=REFERENCED_DOCUMENT(#doc); #doc is built from the OntoML *documentId*, according to D.4.3.3.2.1. The **createDocument** function would return the *#ref_doc* entity instance.
< … xsi:type= **"IDENTIFIED_DOCUMENT_Type"**> <document_identifier> anIdentifier </document_identifier> <…/>	#doc_id=IDENTIFIED_DOCUMENT(<anIdentifier>**)** The **createDocument** function would return the *#doc_id* entity instance.
< … xsi:type= **"EXTERNAL_GRAPHICS_Type"**> <file> ... </file> ... <file> ... </file> <…/>	#ext_graph=EXTERNAL_GRAPHICS(#graph_files); #graph_files=GRAPHIC_FILES(#http_protocol, #ext_cont); *#http_protocol* specifies the HTTP protocol *#ext_content* specifies the possible translations. The **createGraphics** function would return the *#ext_graph* entity instance.
< … xsi:type= **"DOCUMENT_CONTENT_Type"**> <file> ... </file> ... <file> ... </file> <revision>*rev*</revision> <…/>	#doc_cont=DOCUMENT_CONTENT(#doc, #http_protocol, #ext_cont, rev); *#doc represents an instance of the document ontology concept identifier (see Clause* D.4.3.3.2.1*)* *#http_protocol* specifies the HTTP protocol *#ext_content* specifies the possible translations. The **createDocumentContent** function would return the *#doc_cont* entity instance.
< … xsi:type= **"MESSAGE_Type"**> <file> ... </file> ... <file> ... </file> <code>a code</code> <…/>	#mess=MESSAGE(#http_protocol, #ext_cont, <aCode>); *#http_protocol* specifies the HTTP protocol *#ext_content* specifies the possible

	translations. The **createExtResource** function would return the *#mess* entity instance.
< … xsi:type= **"ILLUSTRATION_Type"**> <file> ... </file> ... <code>aCode</code> <kind_of_content> *aKind*</kind_of_content> <width>*aWidth*</width> <height>aHeight</height> <…/>	#ill=ILLUSTRATION(#http_protocol, #ext_cont, <aCode>, <aKind>, #width, #height); #width=LENGTH_MEASURE_WITH_UNIT(LENGTH_MEASURE(<aWidth>), #lu_width); #lu_width=(LENGTH_UNIT()SI_UNIT(.MILLI., .METRE.)); #height=LENGTH_MEASURE_WITH_UNIT(LENGTH_MEASURE(<aHeighth>), #lu_height); #lu_height=(LENGTH_UNIT()SI_UNIT(.MILLI., .METRE.)); *#http_protocol* specifies the HTTP protocol *#ext_content* specifies the possible translations. The **createExtResource** function would return the *#ill* entity instance.
< … xsi:type= **"ILLUSTRATION_Type"** standard_size="**a6_illustratio**n"> <file> ... </file> ... <code>aCode</code> <kind_of_content> *aKind*</kind_of_content> <width>*aWidth*</width> <height>aHeight</height> <…/>	#a6ill=A6_ILLUSTRATION(#http_protocol, #ext_cont, <aCode>, <aKind>, #width, #height); *#http_protocol* specifies the HTTP protocol *#ext_content* specifies the possible translations. *#width* and *#height* are defined above. The **createExtResource** function would return the *#a6ill* entity instance.
< … xsi:type= **"ILLUSTRATION_Type"** standard_size="**a9_illustration**"> <file> ... </file> ... <code>aCode</code> <kind_of_content> *aKind*</kind_of_content> <width>*aWidth*</width> <height>aHeight</height> <…/>	#a9ill=A9_ILLUSTRATION(#http_protocol, #ext_cont, <aCode>, <aKind>, #width, #height); *#http_protocol* specifies the HTTP protocol *#ext_content* specifies the possible translations. *#width* and *#height* are defined above. The **createExtResource** function would return the *#a9ill* entity instance.

D.4.3.3.2.8 A posteriori case of relationship mapping

The OntoML representation of a posteriori semantic relationship is not the same than in the CIIM EXPRESS model. Consequently, the mapping may only be expressed using a mapping function. The signature of this mapping function is given below:

createAPosteriori(ontoMLAposteriori: XPath): a_posteriori_semantic_relationship

where:

— *ontoMLAposteriori*: an XPath that references the **xsi:type** attribute of the element to which this function is applied; it allows to specify what kind of a posteriori semantic relationships is intended to be built;

— a_*posteriori_semantic_relationship*: the entity data type of the instance returned by this function call; if the function applies to an XML element whose content model is defined as an *a posteriori* case of semantic relationship (**xsi:type = A_POSTERIORI_CASE_OF_Type**), it returns a CIIM **a_posteriori_case_of** entity instance value; if it applies to an XML element whose content model is defined as an *a posteriori* case of semantic relationship (**xsi:type = A_POSTERIORI_VIEW_OF_Type**) it returns a CIIM **a_posteriori_view_of** entity instance value.

Table D.12 presents *a posteriori* case of OntoML representations and its corresponding CIIM EXPRESS representation.

Table D.12 – OntoML a posteriori case-of relationship mapping

OntoML	EXPRESS instances
< … xsi:type= **"A_POSTERIORI_CASE_OF_Type"**> <source class_ref="classId1"/> <is_case_of class_ref="classId2"/> <corresponding_properties> <mapping> <domain> <property property_ref="propId1"> </domain> <range property_ref="propId2"/> </mapping> </corresponding_properties> <…/>	#apcof=A_POSTERIORI_CASE_OF(#cl1, #cl2, ((#prop1, #prop2))); *#cl1* and *#cl2* are respectively built from the OntoML *classId1* and *classId2* identifiers, according to D.4.3.3.2.1. *#prop1* and *#prop2* are respectively built from the OntoML *propId1* and *propId2* identifiers, according to D.4.3.3.2.1. The **createAPosteriori** function would return the *#apcof* entity instance.

Table D.13 presents *a posteriori* view of OntoML representations and its corresponding CIIM EXPRESS representation.

Table D.13 – OntoML a posteriori view-of relationship mapping

OntoML	EXPRESS instances
< … xsi:type= **"A_POSTERIORI_VIEW_OF_Type"**> <functional_model class_ref="classId1"/> <is_view_of class_ref="classId2"/> <corresponding_properties> <mapping> <domain>	#apcof=A_POSTERIORI_VIEW_OF(#cl1, #cl2, ((#prop1, #prop2))); *#cl1* and *#cl2* are respectively built from the OntoML *classId1* and *classId2* identifiers, according to D.4.3.3.2.1. *#prop1* and *#prop2* are respectively built from the OntoML *propId1* and *propId2* identifiers, according

<property property_ref=”propId1”> </domain> <range property_ref=”propId2”/> </mapping> </corresponding_properties> <…/>	OntoML *propId1* and *propId2* identifiers, according to D.4.3.3.2.1. The **createAPosteriori** function would return the *#apcof* entity instance.

D.4.3.3.2.9 Instances mapping

OntoML does not define any structure for representing values and instances, but it uses the resources defined in the common product exchange format intended to be jointly defined by ISO 13584 and ISO 22745.

The property values and class instances mapping is outside the scope of OntoML Nevertheless, two abstract mapping functions are provided for consistency purposes.

The **createValue** function allows to map values defined in the common product exchange format to the corresponding CIIM entity instance(s). Its signature is the following:

createValue(OntoMLValues: XPath): LIST OF primitive_value

Where:

— *ontoMLValues*: an XPath that references a set of **value** XML elements intended to be processed.

NOTE 1 The **value** XML element is defined in the common product exchange format intended to be jointly defined by ISO 13584 and ISO 22745.

— *property_value*: the general data type returned by this function call that represents a CIIM compliant representation of the value.

NOTE 2 **primitive_value** is defined in ISO13584-24, Clause 6.3.2.

The **createPopulation** function allows to map instances defined in this common format to the corresponding CIIM entity instance(s). Its signature is the following:

createPopulation(OntoMLInstances: XPath): LIST OF UNIQUE dic_class_intance

— *ontoMLInstances*: an XPath that references a set of **item** XML elements intended to be processed.

NOTE 3 The **item** XML element is defined in the common product exchange format intended to be jointly defined by ISO 13584 and ISO 22745.

— *dic_class_instance*: the general data type returned by this function call that represents a CIIM compliant representation of the instance.

NOTE 4 **dic_class_instance** is defined in ISO13584-24, Clause 6.4.7.1.

D.4.4 OntoML resource mapping for un-referenced CIIM EXPRESS items

In OntoML, some un-referenced CIIM EXPRESS resources are represented. The mapping can therefore not be expressed in a common way, on the base of some ontology concepts or on the base of the general dictionary structure.

For that purpose, in place of defining functions whose result is intended to be assigned to an EXPRESS target path, procedures are specified.

D.4.4.1 Global Ontology language

The **create_global_language_assignment** procedure is used to create a CIIM **GLOBAL_LANGUAGE_ASSIGNMENT** EXPRESS entity data type instance from a specified language. Its signature is the following:

create_global_language_assignment(language : XPath)

where:

— *doc*: an XPath retrieving the OntoML general language attribute value.

Table D.14 gives an example of applying this procedure (it is assumed that this mapping procedure is assigned to the XML language attribute of the ontoml XML element: **create_global_language_assignment(.)**).

Table D.14 – OntoML global language mapping

OntoML	EXPRESS instances
<ontoml global_language="en"> … </ontoml>	#gla=GLOBAL_LANGUAGE_ASSIGNMENT('en');

D.5 OntoML complex types and CIIM entity datatype correspondence table

Table D.15 lists the correspondence between the whole set of concrete OntoML complex types and the CIIM EXPRESS constructs (entities or data types).

Table D.15 – OntoML complex types / CIIM entity datatypes correspondence

OntoML complex type	EXPRESS entity data type
SUPPLIER_Type	supplier_element
CLASS_CONSTANT_VALUES_Type	class_constant_values
CLASS_VALUE_ASSIGNMENT_Type	class_value_assignment
CATEGORIZATION_CLASS_Type	categorization_class
FM_CLASS_VIEW_OF_Type	fm_class_view_of
FUNCTIONAL_MODEL_CLASS_Type	functional_model_class
FUNCTIONAL_VIEW_CLASS_Type	functional_view_class
ITEM_CLASS_CASE_OF_Type	item_class_case_of_type
ITEM_CLASS_Type	item_class
NON_INSTANTIABLE_FUNCTIONAL_VIEW_CLASS_Type	non_instanciable_functional_view_class
REPRESENTATION_CONTEXT_Type	representation_context
VIEW_CONTROL_VARIABLE_RANGE_Type	view_control_variable_range
CONDITION_DET_Type	condition_DET
DEPENDENT_P_DET_Type	dependent_P_DET
NON_DEPENDENT_P_DET_Type	non_dependent_P_DET
REPRESENTATION_P_DET_Type	representation_P_DET

DATATYPE_Type	data_type_element
DOCUMENT_CONTENT_Type	document_content
DOCUMENT_Type	document_element
PERSON_Type	person
GLOBAL_LANGUAGE_ASSIGNMENT_Type	global_language_assignment
PREFERRED_NAME_LABEL_Type	translatable_label
SHORT_NAME_LABEL_Type	translatable_label
SYNONYMOUS_NAME_LABEL_Type	syn_name_type
GENERAL_TEXT_Type	translatable_text
TRANSLATION_DATA_Type	translation_data
EXTERNAL_GRAPHICS_Type	external_graphics
GRAPHICS_FILES_Type	graphic_files
HTTP_FILE_Type	http_file
IDENTIFIED_DOCUMENT_Type	identified_document
ILLUSTRATION_Type	Illustration
MESSAGE_Type	message
PROGRAM_REFERENCE_Type	program_reference
REFERENCED_DOCUMENT_Type	referenced_document
REFERENCED_GRAPHICS_Type	referenced_graphics
REPRESENTATION_REFERENCE_Type	representation_reference
A_POSTERIORI_CASE_OF_Type	a_posteriori_case_of
A_POSTERIORI_VIEW_OF_Type	a_posteriori_view_of
MATHEMATICAL_STRING_Type	mathematical_string
ORGANIZATION_Type	organization
CONTEXT_RESTRICTION_CONSTRAINT_Type	context_restriction_constraint
DOMAIN_CONSTRAINT_Type	domain_constraint
ENTITY_SUBTYPE_CONSTRAINT_Type	entity_subtype_contraint
PROPERTY_TYPE_REDEFINITION_Type	property_type_redefinition
RANGE_CONSTRAINT_Type	range_constraint
STRING_PATTERN_CONSTRAINT_Type	string_pattern_constraint
STRING_SIZE_CONSTRAINT_Type	string_size_constraint
SUBCLASS_CONSTRAINT_Type	subclass_constraint
SUBSET_CONSTRAINT_Type	subset_constraint
DICTIONARY_Type	dictionary
DICTIONARY_IN_STANDARD_FORMAT_Type	dictionary_in_standard_format
LIBRARY_IIM_IDENTIFICATION_Type	library_iim_identification
LIBRARY_IN_STANDARD_FORMAT_Type	library_in_standard_format
LIBRARY_Type	library

VIEW_EXCHANGE_PROTOCOL_IDENTIFICATION_Type	view_exchange_protocol_identification
ARRAY_TYPE_Type	array_type
AXIS1_PLACEMENT_TYPE_Type	axis1_placement_type
AXIS2_PLACEMENT_2D_TYPE_Type	axis2_placement_2D_type
AXIS2_PLACEMENT_3D_TYPE_Type	axis2_placement_3D_type
BAG_TYPE_Type	bag_type
BOOLEAN_TYPE_Type	boolean_type
CLASS_INSTANCE_TYPE_Type	class_instance_type
ENTITY_INSTANCE_TYPE_Type	entity_instance_type
GEOMETRIC_REPRESENTATION_CONTEXT_TYPE_Type	geometric_representation_context_type
INT_CURRENCY_TYPE_Type	int_currency_type
INT_MEASURE_TYPE_Type	int_measure_type
INT_TYPE_Type	int_ type
INT_DIC_VALUE_Type	dic_value
LEVEL_Type	level
LEVEL_TYPE_Type	level_type
LIST_TYPE_Type	list_type
NAMED_TYPE_Type	named_type
NON_QUANTITATIVE_CODE_TYPE_Type	non_quantitative_code_type
NON_QUANTITATIVE_INT_TYPE_Type	non_quantitative_int_type
NON_TRANSLATABLE_STRING_TYPE_Type	non_translatable_string_type
NUMBER_TYPE_Type	number_type
PLACEMENT_TYPE_Type	placement_type
PROGRAM_REFERENCE_TYPE_Type	program_reference_type
REAL_CURRENCY_TYPE_Type	real_currency_type
REAL_MEASURE_TYPE_Type	real_measure_type
REAL_TYPE_Type	real_type
REMOTE_HTTP_ADDRESS_Type	remote_http_address
REPRESENTATION_REFERENCE_TYPE_Type	representation_reference_type
REPRESENTATION_TYPE_Type	representation_type
SET_TYPE_Type	set_type
SET_WITH_SUBSET_CONSTRAINT_TYPE_Type	set_with_subset_constraint_type
STRING_DIC_VALUE_Type	dic_value
STRING_TYPE_Type	string_type
TRANSLATABLE_STRING_TYPE_Type	translatable_string
CONTEXT_DEPENDENT_UNIT_Type	context_dependent_unit

CONVERSION_BASED_UNIT_Type	conversion_based_unit
DERIVED_UNIT_Type	derived_unit
DERIVED_UNIT_ELEMENT_Type	derived_unit_element
DIC_UNIT_Type	dic_unit
DIMENSIONAL_EXPONENTS_Type	dimensional_exponents
NAMED_UNIT_Type	named_unit
NON_SI_UNIT_Type	non_si_unit
SI_UNIT_Type	si_unit

Annex E (informative)
XML file example

E.1 Example of an XML file compliant with OntoML: general model

This annex presents, on an example, an overview of the different resources and steps involved in the description of a parts library according to the ISO 13584 series.

The content of this clause is as follows:

— example introduction;

— OntoML representation of these product characterization classes.

E.1.1 Example introduction

Figure E.1 intended to be described. It is a characterization class of washers, denoted *paw*, that is sold by a bearing supplier and that is used as a bearing in some mechanical contexts. It is intended to be represented in a single language, English.

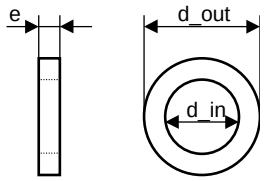

Figure E.1 – General model example: ontology definition

This product characterization class is described by three characteristics:

— the inner diameter (*d_in* symbol), a real measure whose unit is millimeter;

— the outer diameter (*d_out* symbol), a real measure whose unit is millimeter;

— the thickness (*e* symbol), a real measure whose unit is millimeter.

This product characterization class may be also associated to a drawing (the one presented in Figure E.1), called *paw.jpg*.

For the purpose of this example, we propose to define a very simple ontology as follow:

— a product characterization class that plays the role of the ontology root, and whose name will be *bearing*; it defines and it is described by two properties:

  — the inner diameter (*d_in*);

  — the outer diameter (*d_out*).

— a product characterization class corresponding to the *paw* parts family, subclass of the *bearing* class; it defines and it is described by a single property:

  — the thickness (*e*).

The ontology concepts that need to be used in this example are: dictionary, supplier, class and property. For each of them, we define (according to the identifiers structure defined in Clause 8.1) the following identifiers:

— parts supplier: “0002-38491502100024”;

— dictionary: “0002-38491502100024#DICO#001”;

— *bearing* characterization class: “0002-38491502100024#BEARING#001”;

— *paw* characterization class: “0002-38491502100024#PAW#001”;

— *inner diameter* property: “0002-38491502100024#BEARING*DIN#001”;

— *outer diameter* property: “0002-38491502100024#BEARING*DOUT#001”;

— *thickness* property: “0002-38491502100024#PAW*THICK#001”.

According to this ontology structure, we propose to describe the following five products (Figure E.2):

d_in	e	d_out
10	1	15
11	1	16.5
13	2	19.5
17	3	25.5
19	4	28.5

Figure E.2 – General model example: product specification

NOTE OntoML does only provide resources for the representation of information about the container structure of products (product characterization class extension). Product description as property reference and typed value pairs is defined outside the scope of this part of ISO 13584.

E.1.2 OntoML representation

In this clause, the complete OntoML representation of the ontology example introduced in clause E.1.1 is given. This XML file is conformant with conformance class 3 defined in this part of ISO 13584.

```
<ontoml:ontoml xmlns:ontoml="urn:iso:std:iso:cd:13584:-32" xmlns:bt="urn:pcd:schema:basic-type"
xmlns:idt="urn:pcd:schema:identifier" xmlns:cat="urn:pcd:schema:item" xmlns:val="urn:pcd:schema:value"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="urn:iso:std:iso:cd:13584:-32
D:\plib\plibDoc\P32\schema\CD\ontoml.xsd">
    <header id="0002-38491502100024#DICO#001">
        <global_language language_code="en"/>
        <description>This file illustrates the use of OntoML</description>
        <version>1</version>
        <name>generalModel.xml</name>
        <date_time_stamp>2006-11-13T12:20:46+01:00</date_time_stamp>
        <author>Eric SARDET</author>
        <organisation>CRITT Informatique CRCFAO</organisation>
        <pre_processor_version/>
        <originating_system>XMLSpy v2006 sp2 U</originating_system>
```

```
            <authorisation>Eric Sardet - sardet@ensma.fr</authorisation>
            <!-- The XML file is compliant with ISO13584-32, conformance class 3. The library_structure XML element specifies this particular conformance class. -->
            <ontoml_information>
                  <!--We assume that the dictionary/library revision is equal to 1-->
                  <revision>1</revision>
                  <preferred_name>
                        <label>Basic dictionary example</label>
                  </preferred_name>
            </ontoml_information>
            <ontoml_structure>
                  <status>CD</status>
                  <name>ONTOML</name>
                  <date>2007</date>
                  <application>3</application>
            </ontoml_structure>
      </header>
      <!-- The ontology is defined using the dictionary XML element. Because it represents both the ontology itself and instances compliant to this ontology, the specific content model used is based on the LIBRARY_IN_STANDARD_FORMAT_Type complex type definition. An identifier is assigned to the dictionary using the id XML attribute. -->
      <dictionary xsi:type="ontoml:DICTIONARY_IN_STANDARD_FORMAT_Type">
            <!--We assume that the information described is a complete library-->
            <is_complete>true</is_complete>
            <!-- The supplier responsible for the description of the ontology data is identified using the supplier_ref XML attribute -->
            <responsible_supplier supplier_ref="0002-38491502100024"/>
            <!-- All the characterization classes described in an ontology are gathered in the "contained_classes" XML element.-->
            <contained_classes>
                  <!-- This class XML element defines the "bearing" class. It is defined as an item class ("ITEM_CLASS_Type" complex type). It is identified using the "id" XML attribute.-->
                  <class xsi:type="ontoml:ITEM_CLASS_Type" id="0002-38491502100024#BEARING#001">
                        <!-- We assume that the provided characterizationclass revision is equal to 1 -->
                        <revision>1</revision>
                        <!-- This characterization class preferred name is not translated and set to "bearing" -->
                        <preferred_name>
                              <label>bearing</label>
                        </preferred_name>
                        <!-- The "bearing" characterization class definition is not translated. -->
                        <definition>
                              <text>general bearing parts family</text>
                        </definition>
                        <!-- The "bearing" characterization class is described by two properties, each of them being identified unambiguously ("property_ref" XMLattribute). In this example, the properties definitions are provided, but it is not mandatory. The listed properties are: inner diameter and outer diameter.-->
                        <described_by>
                              <property property_ref="0002-38491502100024#BEARING*DIN#001"/>
                              <property property_ref="0002-38491502100024#BEARING*DOUT#001"/>
                        </described_by>
                  </class>
                  <!-- This class XML element defines the "paw" class. It is defined as an item class ("ITEM_CLASS_Type" complex type). It is identified using the id XML attribute.-->
                  <class xsi:type="ontoml:ITEM_CLASS_Type" id="0002-38491502100024#PAW#001">
                        <!-- We assume that the provided characterization class revision is equal to "1" -->
                        <revision>1</revision>
                        <!-- This characterization class preferred name is not translated and set to "paw" -->
                        <preferred_name>
                              <label>paw</label>
                        </preferred_name>
                        <!-- The "paw" characterization class definition is not translated. -->
                        <definition>
                              <text>paw parts family</text>
                        </definition>
```

```
                        <!-- The "paw" characterization class is a subclass of the "bearing" characterization class: it is specified using the "its_superclass" element whose "class_ref" attibute references the "bearing" characterization class-->
                        <its_superclass class_ref="0002-38491502100024#BEARING#001"/>
                        <!-- The "paw" characterization class is described by a single property being identified unambiguously ("property_ref" XMLattribute). In this example, the properties definitions are provided, but it is not mandatory. The listed properties are: inner diameter and outer diameter.-->
                        <described_by>
                                <property property_ref="0002-38491502100024#PAW*THICK#001"/>
                        </described_by>
                        <!-- A drawing, stored in a file called "paw.jpeg" located in the same directory that the current XML file, is assigned to the "paw" characterization class.-->
                        <simplified_drawing xsi:type="ontoml:EXTERNAL_GRAPHICS_Type">
                                <!-- The drawing consists of a single external resource called "paw.jpg"-->
                                <representation>
                                        <file>
                                                <http_file_name>paw.jpeg</http_file_name>
                                        </file>
                                </representation>
                        </simplified_drawing>
                </class>
        </contained_classes>
        <!-- A name, that is only defined in english, is provided to the dictionary -->
        <!--Every supplier referenced in this ontology shall be described in the "contained_suppliers" XML element. -->
        <contained_suppliers>
                <supplier id="0002-38491502100024">
                        <revision>1</revision>
                        <!-- The supplier is only described through its name. -->
                        <org>
                                <name>CRITT Informatique CRCFAO</name>
                        </org>
                </supplier>
        </contained_suppliers>
        <!-- All the properties described in an ontology are gathered in the "contained_properties" XML element.-->
        <contained_properties>
                <property xsi:type="ontoml:NON_DEPENDENT_P_DET_Type" id="0002-38491502100024#BEARING*DIN#001">
                        <!-- We assume that the provided property revision is equal to "1" -->
                        <revision>1</revision>
                        <!-- This property preferred name is not translated and set to "inner diameter" -->
                        <preferred_name>
                                <label>inner diameter</label>
                        </preferred_name>
                        <!-- The "inner diameter" property definition is not translated. -->
                        <definition>
                                <text>the bearing inner diameter</text>
                        </definition>
                        <!-- The "d_in" preferred symbol is assigned to this property -->
                        <preferred_symbol>
                                <text_representation>d_in</text_representation>
                        </preferred_symbol>
                        <!-- A real measure value domain (REAL_MEASURE_TYPE_Type XML complex type), expressed in millimetre, is assigned to this property. -->
                        <domain xsi:type="ontoml:REAL_MEASURE_TYPE_Type">
                                <unit>
                                        <structured_representation xsi:type="ontoml:SI_UNIT_Type">
                                                <!-- The particular named unit used is an SI Unit, as defined by the SI_UNIT_Type XML complex type-->
                                                <prefix>MILLI</prefix>
                                                <name>METRE</name>
                                        </structured_representation>
                                </unit>
                        </domain>
                </property>
```

```
                    <property xsi:type="ontoml:NON_DEPENDENT_P_DET_Type" id="0002-
38491502100024#BEARING*DOUT#001">
                        <!-- We assume that the provided property revision is equal to "1" -->
                        <revision>1</revision>
                        <!-- This property preferred name is not translated and set to "outer diameter" -->
                        <preferred_name>
                            <label>outer diameter</label>
                        </preferred_name>
                        <!-- The "outer diameter" property definition is not translated. -->
                        <definition>
                            <text>the bearing outer diameter</text>
                        </definition>
                        <!-- The "d_in" preferred symbol is assigned to this property -->
                        <preferred_symbol>
                            <text_representation>d_out</text_representation>
                        </preferred_symbol>
                        <!-- A real measure value domain (REAL_MEASURE_TYPE_Type XML complex
type), expressed in millimetre, is assigned to this property. -->
                        <domain xsi:type="ontoml:REAL_MEASURE_TYPE_Type">
                            <unit>
                                <structured_representation xsi:type="ontoml:SI_UNIT_Type">
                                    <!-- The particular named unit used is an SI Unit, as defined by
the SI_UNIT_Type XML complex type-->
                                    <prefix>MILLI</prefix>
                                    <name>METRE</name>
                                </structured_representation>
                            </unit>
                        </domain>
                    </property>
                    <property xsi:type="ontoml:NON_DEPENDENT_P_DET_Type" id="0002-
38491502100024#PAW*THICK#001">
                        <!-- We assume that the provided property revision is equal to "1" -->
                        <revision>1</revision>
                        <!-- This property preferred name is not translated and set to "thickness" -->
                        <preferred_name>
                            <label>thickness</label>
                        </preferred_name>
                        <!-- The "thickness" property definition is not translated. -->
                        <definition>
                            <text>the paw thickness</text>
                        </definition>
                        <!-- The "e" preferred symbol is assigned to this property -->
                        <preferred_symbol>
                            <text_representation>e</text_representation>
                        </preferred_symbol>
                        <!-- A real measure value domain, expressed in millimetre
(REAL_MEASURE_TYPE_Type XML complex type), is assigned to this property. -->
                        <domain xsi:type="ontoml:REAL_MEASURE_TYPE_Type">
                            <unit>
                                <structured_representation xsi:type="ontoml:SI_UNIT_Type">
                                    <!-- The particular named unit used is an SI Unit, as defined by
the SI_UNIT_Type XML complex type-->
                                    <prefix>MILLI</prefix>
                                    <name>METRE</name>
                                </structured_representation>
                            </unit>
                        </domain>
                    </property>
                </contained_properties>
                <!-- The ontology is provided in a single language: english.-->
            </dictionary>
            <!-- The "content" XML element allows to represent products according to some product charaterization
classes described in a given ontology-->
            <library xsi:type="ontoml:LIBRARY_IN_STANDARD_FORMAT_Type">
                <contained_class_extensions>
```

```
<!-- We describe the products that relate to a product characterization class. Every product is described as a set of (property reference, type value) pairs. -->
<class_extension xsi:type="ontoml:EXPLICIT_ITEM_CLASS_EXTENSION_Type">
<!-- The particular product characterization class for which the products are specified is referenced using the "class_ref" XML attribute.-->
<dictionary_definition class_ref="0002-38491502100024#PAW#001"/>
<!-- We assume that the provided content revision is equal to "1" -->
<content_revision>1</content_revision>
<!-- the "instance_identification" defines which properties of the describes products play the role of a primary key. In our example, the "inner diameter" property plays such a role. This property is therefore referenced unambiguously using the "property_ref" XML attribute.-->
<instance_identification>
<property property_ref="0002-38491502100024#BEARING*DIN#001"/>
</instance_identification>
<!-- A product charaterization class consists of a set of product. they are all gathered in the "population" XML element. -->
<population>
<!-- Every product is defines using external resources for describing a product as a set of (property reference, type value) pairs.Moreover, every product references the class to which it belongs using the "class-ref" attribute.-->
<cat:item class-ref="0002-38491502100024#PAW#001">
<!-- The first value is about the "inner diameter" property that is referenced using the "property-ref' attribute -->
<cat:property-value property-ref="0002-38491502100024#BEARING*DIN#001">
<!-- The valiue is a real measure, whose unit is "mm". This value is equal to "10". -->
<val:measure-number-value uom-code="mm">
<val:real-value>10.0</val:real-value>
</val:measure-number-value>
</cat:property-value>
<!-- The second value is about the "thickness" property that is referenced using the "property-ref' attribute -->
<cat:property-value property-ref="0002-38491502100024#PAW*THICK#001">
<!-- The valiue is a real measure, whose unit is "mm". This value is equal to "10". -->
<val:measure-number-value uom-code="mm">
<val:real-value>1.0</val:real-value>
</val:measure-number-value>
</cat:property-value>
<!-- The third value is about the "outer diameter" property that is referenced using the "property-ref' attribute -->
<cat:property-value property-ref="0002-38491502100024#BEARING*DOUT#001">
<!-- The valiue is a real measure, whose unit is "mm". This value is equal to "10". -->
<val:measure-number-value uom-code="mm">
<val:real-value>15.0</val:real-value>
</val:measure-number-value>
</cat:property-value>
</cat:item>
<!-- Second product description -->
<cat:item class-ref="0002-38491502100024#PAW#001">
<cat:property-value property-ref="0002-38491502100024#BEARING*DIN#001">
<val:measure-number-value uom-code="mm">
<val:real-value>11.0</val:real-value>
</val:measure-number-value>
</cat:property-value>
<cat:property-value property-ref="0002-38491502100024#PAW*THICK#001">
<val:measure-number-value uom-code="mm">
<val:real-value>1.0</val:real-value>
</val:measure-number-value>
</cat:property-value>
```

```
                                        <cat:property-value property-ref="0002-
38491502100024#BEARING*DOUT#001">
                                                <val:measure-number-value uom-code="mm">
                                                        <val:real-value>16.5</val:real-value>
                                                </val:measure-number-value>
                                        </cat:property-value>
                                </cat:item>
                                <!-- Third product description -->
                                <cat:item class-ref="0002-38491502100024#PAW#001">
                                        <cat:property-value property-ref="0002-
38491502100024#BEARING*DIN#001">
                                                <val:measure-number-value uom-code="mm">
                                                        <val:real-value>13.0</val:real-value>
                                                </val:measure-number-value>
                                        </cat:property-value>
                                        <cat:property-value property-ref="0002-
38491502100024#PAW*THICK#001">
                                                <val:measure-number-value uom-code="mm">
                                                        <val:real-value>2.0</val:real-value>
                                                </val:measure-number-value>
                                        </cat:property-value>
                                        <cat:property-value property-ref="0002-
38491502100024#BEARING*DOUT#001">
                                                <val:measure-number-value uom-code="mm">
                                                        <val:real-value>19.5</val:real-value>
                                                </val:measure-number-value>
                                        </cat:property-value>
                                </cat:item>
                                <!-- Fourth product description -->
                                <cat:item class-ref="0002-38491502100024#PAW#001">
                                        <cat:property-value property-ref="0002-
38491502100024#BEARING*DIN#001">
                                                <val:measure-number-value uom-code="mm">
                                                        <val:real-value>17.0</val:real-value>
                                                </val:measure-number-value>
                                        </cat:property-value>
                                        <cat:property-value property-ref="0002-
38491502100024#PAW*THICK#001">
                                                <val:measure-number-value uom-code="mm">
                                                        <val:real-value>3.0</val:real-value>
                                                </val:measure-number-value>
                                        </cat:property-value>
                                        <cat:property-value property-ref="0002-
38491502100024#BEARING*DOUT#001">
                                                <val:measure-number-value uom-code="mm">
                                                        <val:real-value>25.5</val:real-value>
                                                </val:measure-number-value>
                                        </cat:property-value>
                                </cat:item>
                                <!-- Fifth product description -->
                                <cat:item class-ref="0002-38491502100024#PAW#001">
                                        <cat:property-value property-ref="0002-
38491502100024#BEARING*DIN#001">
                                                <val:measure-number-value uom-code="mm">
                                                        <val:real-value>19.0</val:real-value>
                                                </val:measure-number-value>
                                        </cat:property-value>
                                        <cat:property-value property-ref="0002-
38491502100024#PAW*THICK#001">
                                                <val:measure-number-value uom-code="mm">
                                                        <val:real-value>4.0</val:real-value>
                                                </val:measure-number-value>
                                        </cat:property-value>
                                        <cat:property-value property-ref="0002-
38491502100024#BEARING*DOUT#001">
                                                <val:measure-number-value uom-code="mm">
```

```
                                                    <val:real-value>28.5</val:real-value>
                                                </val:measure-number-value>
                                            </cat:property-value>
                                        </cat:item>
                                    </population>
                                    <table_like>true</table_like>
                                </class_extension>
                            </contained_class_extensions>
                            <responsible_supplier supplier_ref="0002-38491502100024"/>
                        </library>
</ontoml:ontoml>
```

Bibliography

[1] ISO 29002-10, *Concept dictionaries – Part 10: Product characteristic data exchange model*.

[2] ISO/IEC CD Guide 77-2*, Guide for the specification of product properties and classes — Technical Guide.*

Index

CIIM ontology concept....3
class....3
common ISO/IEC ontology model....3
EXPRESS attribute....3
EXPRESS entity....3
GId....3
global identifier....3
library....4
ontology....4
OntoML document instance....4
product characterization....4
product ontology....4
property....4
reference dictionary....5
XML attribute....5
XML complex type....5
XML element....5
XML simple type....5

********* Do Not Delete This Line! ******* Ne Pas Supprimer Cette Ligne ******* Do Not Delete This Line!****