



BSI Standards Publication

Industrial automation systems and integration — Parts library

Part 32: Implementation resources: OntoML: Product ontology markup language

National foreword

This British Standard is the UK implementation of ISO 13584-32:2010.

The UK participation in its preparation was entrusted to Technical Committee AMT/4, Industrial data and manufacturing interfaces.

A list of organizations represented on this committee can be obtained on request to its committee manager.

Contractual and legal considerations

This publication has been prepared in good faith, however no representation, warranty, assurance or undertaking (express or implied) is or will be made, and no responsibility or liability is or will be accepted by BSI in relation to the adequacy, accuracy, completeness or reasonableness of this publication. All and any such responsibility and liability is expressly disclaimed to the full extent permitted by the law.

This publication is provided as is, and is to be used at the recipient's own risk.

The recipient is advised to consider seeking professional guidance with respect to its use of this publication.

This publication is not intended to constitute a contract. Users are responsible for its correct application.

© The British Standards Institution 2023
Published by BSI Standards Limited 2023

ISBN 978 0 580 65938 6

ICS 25.040.40

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 30 April 2023.

Amendments/corrigenda issued since publication

Date	Text affected
------	---------------

**Industrial automation systems and
integration — Parts library —**

**Part 32:
Implementation resources: OntoML:
Product ontology markup language**

*Systèmes d'automatisation industrielle et intégration — Bibliothèque
de composants —*

*Partie 32: Ressources d'implémentation: OntoML: Langage de marquage
ontologique*



Reference number
ISO 13584-32:2010(E)

Documents
Document Sharing Hub
© ISO 2010

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.



COPYRIGHT PROTECTED DOCUMENT

© ISO 2010

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword.....	vii
Introduction	ix
1 Scope	1
2 Normative references	2
3 Terms and definitions	2
4 Abbreviated terms	7
5 OntoML implementation levels	7
6 Overview of OntoML ontology representation	8
7 Overview of OntoML libraries representation	57
8 Other structured information elements	66
9 OntoML exchange structure	135
10 Dictionary Change Management Rules.....	152
Annex A (normative) Information object registration	163
Annex B (normative) Computer interpretable listings	164
Annex C (normative) Standard data requirements for OntoML	166
Annex D (normative) Value representation of ISO 13584 / IEC 61360 entities and data types on ISO/TS 29002-10 shared XML schemas	167
Annex E (normative) Ontology specification of extended values used in OntoML	192
Annex F (normative) Structural transformation of the CIIM model from OntoML XML Schema to EXPRESS.....	199
Annex G (normative) OntoML exchange levels	233
Annex H (normative) Value format specification	235
Annex I (informative) XML file example	249
Annex J (informative) Information to support implementations	256

Figures

Figure 1 — CIIM ontology concepts description	9
Figure 2 — UML-like representation of an XML complex type	10
Figure 3 — UML-like representation of a reference to an XML complex type	10
Figure 4 — UML-like representation of an external reference to an XML complex type	10
Figure 5 — UML-like representation of XML attributes and simple type XML elements	11
Figure 6 — XML representation of XML attributes and simple type XML elements	11
Figure 7 — UML-like representation of an XML complex type XML element	11
Figure 8 — XML representation of an XML complex type element	12
Figure 9 — UML-like representation of XML elements cardinality	12
Figure 10 — XML representation of XML elements cardinality	12
Figure 11 — UML-like representation of XML complex type extensions	13
Figure 12 — XML representation of XML complex type extensions	13
Figure 13 — Identification of a CIIM ontology concept	14
Figure 14 — CIIM ontology concept reference	14
Figure 15 — Reference between CIIM ontology concepts	15
Figure 16 — UML-like representation of a simple reference between CIIM ontology concepts	16
Figure 17 — XML representation of a simple reference between CIIM ontology concepts	16
Figure 18 — UML-like representation of a multi-valued reference between CIIM ontology concepts	17
Figure 19 — XML representation of a multi-valued reference between CIIM ontology concepts	17
Figure 20 — Ontology structure UML diagram	19
Figure 21 — Ontology header structure	20
Figure 22 — Root element of an ontology	22
Figure 23 — Supplier ontology concept UML diagram	25
Figure 24 — Simple class ontology concept UML diagram	27
Figure 25 — Example of a supplier ontology using categorization classes	32
Figure 26 — Categorization class	33
Figure 27 — Item class case-of UML diagram	34
Figure 28 — Class value assignment structure	37
Figure 29 — Advanced-level ontology class concept UML diagram: functional view class	40
Figure 30 — Advanced class ontology concept UML diagram: functional model class	41
Figure 31 — Advanced class ontology concept UML diagram: functional model class <i>view-of</i>	44
Figure 32 — View control variable structure	46
Figure 33 — Simple property ontology concept UML diagram	48
Figure 34 — Advanced property ontology concept UML diagram	51
Figure 35 — Data type UML diagram	52
Figure 36 — Simple-level document UML diagram	54
Figure 37 — Root element of library	57
Figure 38 — General class extension structure	58
Figure 39 — Properties classification	60
Figure 40 — Properties presentation	61
Figure 41 — Products representation structure	62
Figure 42 — Functional models structure UML diagram	64
Figure 43 — Language specification	66
Figure 44 — Translation resources	67
Figure 45 — Translation data structure	69
Figure 46 — Simple-level ontology external resources	70
Figure 47 — Simple-level ontology external resources: HTTP file structure	71
Figure 48 — Simple-level ontology external resources: illustration	72
Figure 49 — Simple-level ontology external resources: message	73
Figure 50 — Simple-level ontology external resources: external files	73
Figure 51 — External resources: source document	74
Figure 52 — External resources: identified document	74
Figure 53 — External resources: referenced document	75
Figure 54 — External resources: graphics	76
Figure 55 — External resources: external graphics	76
Figure 56 — External resources: referenced graphics	77

Figure 57 — OntoML datatype system	78
Figure 58 — Boolean type structure	80
Figure 59 — String types structure	81
Figure 60 — Date and time types structure	82
Figure 61 — Enumeration of string codes type structure	84
Figure 62 — Numeric types structure	86
Figure 63 — Numeric currency types structure	88
Figure 64 — Numeric measure types structure	90
Figure 65 — Enumeration of integer codes type structure	92
Figure 66 — Bag type structure	94
Figure 67 — Set type structure	95
Figure 68 — List type structure	96
Figure 69 — Array type structure	97
Figure 70 — Set with a subset constraint type structure	98
Figure 71 — Instance value domain structure	99
Figure 72 — Levels value domain structure	100
Figure 73 — Named type structure	101
Figure 74 — Advanced-level data types structure	102
Figure 75 — General measure property unit structure	105
Figure 76 — Basic unit structures	105
Figure 77 — Named unit general structure	106
Figure 78 — Dimensional exponent structure	107
Figure 79 — International standardized unit structure	107
Figure 80 — Non international standardized unit structure	108
Figure 81 — Conversion based unit structure	109
Figure 82 — Context dependent unit structure	110
Figure 83 — Derived unit structure	110
Figure 84 — General constraints structure	111
Figure 85 — Constraint reference structure	112
Figure 86 — Class constraint structure	113
Figure 87 — Configuration control constraint structure	113
Figure 88 — Property constraint structure	115
Figure 89 — Context restriction constraint structure	115
Figure 90 — Integrity constraint structure	116
Figure 91 — Domain constraints	117
Figure 92 — Subclass constraint representation	118
Figure 93 — String pattern constraint representation	119
Figure 94 — Cardinality constraint representation	120
Figure 95 — String size constraint representation	121
Figure 96 — Range constraint representation	122
Figure 97 — Enumeration constraint representation	123
Figure 98 — <i>A posteriori</i> relationship general structure representation	126
Figure 99 — <i>A posteriori case-of</i> relationship representation	128
Figure 100 — <i>A posteriori</i> semantic relationships structure	130
Figure 101 — Library integrated information model identification structure	131
Figure 102 — View exchange protocol identification structure	132
Figure 103 — Organization structure	133
Figure 104 — Mathematical string structure	133
Figure 105 — Geometric context structure	134
Figure 106 — Geometric unit context structure	134
Figure 107 — Classifying a dictionary change	158
 Figure E.1 — Planning model of the ontology of extended values	 193
 Figure F.1 — A UML information model example	 200
Figure F.2 — An UML-like representation of the information model	201
Figure F.3 — An XML Schema example	201
Figure F.4 — Mapping representation in OntoML	203
Figure F.5 — XML source Path	203

Figure F.6 — Global Vs local XML elements	204
Figure F.7 — Local EXPRESS target path structure	207
Figure F.8 — Complete EXPRESS target path structure	209
Figure I.1 — General model example: ontology definition	249
Figure I.2 — General model example: product specification	250

Tables

Table 1 — OntoML modules cross-references	143
Table 2 — Conformance options of OntoML	144
Table 3 — Revision and version	155
Table B.1 — XML schema defined in this part of ISO 13584	164
Table B.2 — XML schemas defined outside of this part of ISO 13584	165
Table C.1 — ISO 13584 LIIM 32 conformance class specification	166
Table E.1 — OntoML extendedvalues: class identifiers	198
Table E.2 — OntoML extendedvalues: property identifiers	198
Table F.1 — XML and corresponding ISO 10303-21 instances	202
Table F.2 — SELF meaning in its use context	205
Table F.3 — OntoML identifiers mapping	213
Table F.4 — OntoML list of class identifiers mapping	215
Table F.5 — OntoML ontology identifier mapping	216
Table F.6 — OntoML label and translated label mapping	216
Table F.7 — OntoML text and translated text mapping	218
Table F.8 — OntoML synonymous and translated synonymous mapping	219
Table F.9 — OntoML keywords and translated keywords mapping	220
Table F.10 — OntoML HTTP protocol mapping	222
Table F.11 — OntoML translated and not translated files mapping	222
Table F.12 — OntoML external resource mapping	223
Table F.13 — OntoML <i>a posteriori</i> case-of relationship mapping	226
Table F.14 — OntoML <i>a posteriori</i> view-of relationship mapping	226
Table F.15 — OntoML global language mapping	228
Table F.16 — OntoML complex types / CIIM entity datatypes correspondence	228
Table H.1 — ISO/IEC 14977 EBNF syntactic metalanguage	236
Table H.2 — Transposing European style digits into Arabic digits	243
Table H.3 — Number value examples	244
Table H.4 — Characters from other rows of the Basic Multilingual Plane of ISO/IEC 10646-1	245

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 document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO 13584-32 was prepared by Technical Committee ISO/TC 184, *Automation systems and integration*, Subcommittee SC 4, *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: Information supplier identification*
- *Part 31: Implementation resources: Geometric programming interface*
- *Part 32: Implementation resources: OntoML: Product ontology markup language*
- *Part 35: Implementation resources: Spreadsheet interface for parts library [Technical Specification]*
- *Part 42: Description methodology: Methodology for structuring parts families*
- *Part 101: Geometrical view exchange protocol by parametric program*
- *Part 102: View exchange protocol by ISO 10303 conforming specification*
- *Part 501: Reference dictionary for measuring instruments — Registration procedure*
- *Part 511: Mechanical systems and components for general use — Reference dictionary for fasteners*

The structure of ISO 13584 is described in ISO 13584-1. The numbering of the parts of ISO 13584 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 reference dictionaries.

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

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

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.

ISO 13584 is organized as a series of parts, each published separately. The parts of ISO 13584 fall into one of the following series: conceptual descriptions, logical resources, implementation resources, description methodology, view exchange protocol, and reference dictionaries. 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. It is advisable to be familiar with ISO/IEC Guide 77-2 when making use of this part of ISO 13584.

Industrial automation systems and integration — Parts library —

Part 32:

Implementation resources: OntoML: Product ontology markup language

1 Scope

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

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

- representation of the common ISO 13584/IEC 61360 model using UML notations;
- definition of two levels of implementation of the common ISO 13584/IEC 61360 model, respectively called the simple level and the advanced level;
- specification of XML markup declarations that enable both simple ontologies and advanced ontologies compliant with the common ISO 13584/IEC 61360 model to be exchanged using XML;
- specification of XML markup declarations that enable the exchange of both ontologies compliant with the common ISO 13584/IEC 61360 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 ISO 13584/IEC 61360 model compliant ontologies is defined in ISO 13584-25.

- specification of XML global elements allowing to use OntoML as an exchange format for representing responses to queries performed using the ISO/TS 29002-20 concept dictionary resolution mechanism;
- specification of a formal mapping allowing to associate each OntoML elements and attributes of the corresponding entities and attributes of the common ISO 13584/IEC 61360 model EXPRESS data model.

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

- rules used to build OntoML from the common ISO 13584/IEC 61360 model;
- the specification of the program intended to interpret all the mapping operators defined in OntoML for building the corresponding ISO 10303-21 instances of the EXPRESS representation of the common ISO 13584/IEC 61360 model;
- the exchange of individual products whose characterizations are defined by means of ontologies compliant with the common ISO 13584/IEC 61360 model.

NOTE 3 Individual products can be exchanged using the ISO/TS 29002-10 product exchange format.

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 10303-11:1994, *Industrial automation systems and integration — Product data representation and exchange — Part 11: Description methods: The EXPRESS language reference manual*

ISO/IEC 14977, *Information technology — Syntactic metalanguage — Extended BNF*

ISO/TS 29002-5, *Industrial automation systems and integration — Exchange of characteristic data — Part 5: Identification scheme*

ISO/TS 29002-10:2009, *Industrial automation systems and integration — Exchange of characteristic data — Part 10: Characteristic data exchange format*

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 Edited 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 and definitions

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

3.1

class

abstraction of a set of similar products

NOTE Adapted from ISO 13584-42:2010, definition 3.6.

3.2

class member

product that complies with the abstraction defined by a class

[ISO 13584-42:2010, definition 3.8]

3.3

common ISO 13584/IEC 61360 model

data model for product ontology, using the information modelling language EXPRESS, resulting from a joint effort between ISO/TC 184/SC4/WG2 and IEC SC3D

NOTE 1 Adapted from ISO 13584-42:2010, definition 3.10.

NOTE 2 The previous version of the common ISO 13584/IEC 61360 ontology model is published in both IEC 61360-5 and ISO 13584-25:2004. A new version, in line with this version of OntoML and with ISO 13584-42:2010 is under work.

3.4

CIIM ontology concept

basic unit of knowledge represented in an ontology based on the common ISO 13584/IEC 61360 ontology model, CIIM ontology concepts are information source (supplier), class, property, 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 can be referenced several times in the same exchange file. Therefore, a referencing mechanism is defined in OntoML.

3.5

EXPRESS attribute

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

NOTE An attribute describes only a 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.

3.6

EXPRESS entity

class of information defined by common properties

[ISO 10303-11:1994, definition 3.2.5]

3.7

global identifier

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

NOTE All CIIM ontology concepts are associated with a global identifier.

3.8

is-a relationship

class inclusion relationship associated with inheritance: if A1 is-a A, then each product belonging to A1 belongs to A, and all that is described in the context of A is automatically duplicated in the context of A1

NOTE 1 This mechanism is usually called "inheritance".

NOTE 2 In the common ISO 13584/IEC 61360 dictionary model, the is-a relationship can only be defined between characterization classes. It is advisable that it define a single hierarchy and it ensures that both visible and applicable properties are inherited.

[ISO 13584-42:2010, definition 3.23]

3.9

is-case-of relationship

property importation mechanism: if A1 is case-of A, then the definition of A products also covers A1 products, thus A1 can import any property from A

NOTE 1 The goal of the case-of relationship is to allow connecting together several class inclusion hierarchies while insuring that referenced hierarchies can be updated independently.

NOTE 2 There is no constraint stating that the case-of relationship is intended to define single hierarchies.

NOTE 3 In the common ISO 13584/IEC 61360 dictionary model, the case-of relationship can be used in particular in four cases: (1) to link a characterization class to a categorization class, (2) to import, in the context of some standardized reference dictionaries, some properties already defined in other standardized reference dictionaries, (3) to connect a user reference dictionary to one or several standardized reference dictionaries, (4) to describe a product using the properties of different classes: when products of class A1 fulfil two different functions, and are thus logically described by properties associated with two different classes, A and B, A1 can be connected by is-a to, for example, A, and by case-of to B.

NOTE 4 Adapted from ISO 13584-42:2010, definition 3.24.

3.10 is-view-of

relationship providing a formal expression of the fact that an object is a representation of another object according to a particular perspective

EXAMPLE A set of geometric entities might provide a drafting representation of a particular screw. If both the set of geometric entities and the particular screw are represented as objects, the is-view-of relationship holds between the former object and the latter object (in a drafting perspective).

[ISO 13584-24:2003, definition 3.64]

3.11 library

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

NOTE 1 A library is also called a catalogue.

NOTE 2 In the OntoML schemas, to highlight the difference between the ontology part and the content part, the ontology part is embedded in the "dictionary" XML element, and the content part is embedded in the "library" XML element.

3.12 OntoML document instance

XML document that complies with the OntoML XML Schema

3.13 product categorization part categorization categorization

recursive partition of a set of products into subsets for a specific purpose

NOTE 1 Subsets which appear in a product categorization are called product categorization classes, or product categories.

NOTE 2 A product categorization is not a product ontology. It cannot be used for characterizing products.

NOTE 3 No property is associated with categorizations.

NOTE 4 Several categorizations of the same set of products are possible according to their target usage.

EXAMPLE The UNSPSC classification, defined by the United Nations, is an example of product categorization that was developed for spend analysis.

NOTE 5 Using the is-case-of relationship, several product characterization class hierarchies can be connected to a categorization hierarchy to generate a single structure.

[ISO 13584-42:2010, definition 3.32]

3.14**product categorization class****part categorization class****categorization class**

class of products that constitutes an element of a categorization

EXAMPLE *Manufacturing Components and Supplies and Industrial Optics*, are examples of a product categorization class defined in the UNSPSC.

NOTE 1 No rule is given in this part of ISO 13584 about how to select categorization classes. This concept is introduced (1) to clarify its difference with characterization class, and (2) to explain that the same characterization class can be connected to any number of categorization classes.

NOTE 2 There is no property associated with a categorization class.

[ISO 13584-42:2010, definition 3.33]

3.15**product characterization****part 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 parentheses stand for the identifier of the bolt properties defined in ISO 4014.

[ISO 13584-42:2010, definition 3.34]

3.16**product characterization class****part characterization class****characterization class**

class of products that fulfil the same function and that share common properties

NOTE Product characterization classes can be defined at various levels of details, thus defining a class inclusion hierarchy.

EXAMPLE *Metric threaded bolt/screw* and *hexagon head bolt* are examples of product characterization classes defined in ISO 13584-511. The first characterization class is included in the second one. *Transistor* and *bipolar power transistor* are examples of product characterization classes defined in IEC 61360-4-DB. The second one is included in the first one.

[ISO 13584-42:2010, definition 3.35]

3.17**product ontology****part ontology****ontology**

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

NOTE 1 Product ontologies are based on a class-instance model that allows one to recognize and to designate the sets of products, called characterization classes, that have a similar function (e.g. *ball bearing*, *capacitor*), but also to discriminate within a class the various subsets of products, called instances, that are considered identical. It is advisable that the rules defined in ISO 1087-1 be used for formulating designations and definitions of characterization classes. Instances have no definitions. They are designated by the class to which they belong and a set of property-value pairs.

NOTE 2 Ontologies are not concerned with words but with concepts, independent of any particular language.

NOTE 3 “Consensual” means that the conceptualization is agreed upon in some community.

NOTE 4 “Formal” means that the ontology is intended to be machine interpretable. Some level of machine reasoning is logically possible over ontology, e.g. consistency checking or inferencing.

EXAMPLE 1 Consistency checking is a kind of machine reasoning.

NOTE 5 “Identified” means that each ontology characterization class and properties is associated with a globally unique identifier allowing one to reference this concept from any context.

NOTE 6 In OntoML, advanced ontologies are those ontologies that use all the modelling mechanisms defined in the common ISO 13584/IEC 61360 ontology model. OntoML also defines a simple functional subset of this model allowing to define simple ontologies.

NOTE 7 In this part of ISO 13584, each product ontology addressing a particular product domain compliant with the common ISO 13584/IEC 61360 dictionary model is called a reference dictionary for that domain.

EXAMPLE 2 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 8 Adapted from ISO 13584-42:2010, definition 3.36.

3.18

property

defined parameter suitable for the description and differentiation of products

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

NOTE 2 A property is defined by the totality of its associated attributes. The types and number of attributes that describe a property with high accuracy are documented in this part of ISO 13584.

NOTE 3 The term “property” used in this part of ISO 13584 and the term “data element type” used in IEC 61360 are synonyms.

NOTE 4 Adapted from ISO 13584-42:2010, definition 3.37.

3.19

reference dictionary

product ontology compliant with the common ISO 13584/IEC 61360 dictionary model

NOTE In the ISO 13584 standard series, a product ontology that addresses a particular product domain, based on the common ISO 13584/IEC 61360 dictionary model, is called a reference dictionary for that domain.

[ISO 13584-42:2010, definition 3.41]

3.20

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.

3.21

XML complex type

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

3.22**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 can contain other XML elements defined by either an XML simple type or an XML complex type.

3.23**XML simple type**

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.

4 Abbreviated terms

CIIM	Common ISO 13584/IEC 61360 Model
IRDI	International Registration Data Identifier
SI	Système International d'Unités (International System of Units)
STEP	Standard for The Exchange of Product model data
UNSPSC	classification of products and services defined by the United Nations
URI	Uniform Resource Identifier
URN	Uniform Resource Name
XML	Extensible Markup Language

5 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 discipline 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 this part of ISO 13584.

All the modelling mechanisms that do not belong to the simple level are referenced as “advanced” when they are presented. Clause 9.5 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 makes it possible to model two kinds of information:

- ontologies,
- libraries, which 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/TS 29002-10.

NOTE 2 ISO/TS 29002 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 13584-24:2003 and ISO 13584-25:2004.

6 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 CIIM ontology concept structure are introduced.

6.1 CIIM ontology concepts

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

- supplier,
- class,
- property,
- identified data types,
- 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 9.1.

NOTE The OntoML global identifier of a CIIM ontology concept contains the same information as 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.

6.2 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 representations are used to reference any other piece of information.

Figure 1 illustrates these two kinds of representation.

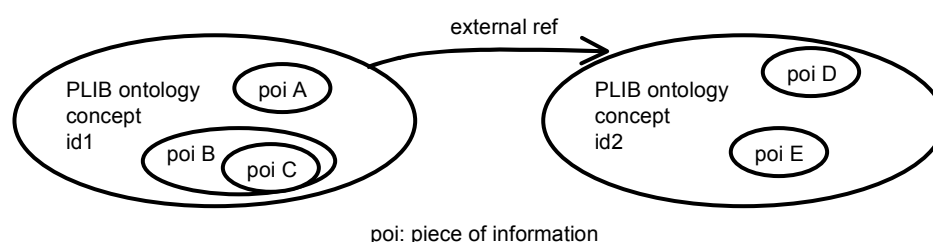


Figure 1 — CIIM ontology concepts description

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 can 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.

6.3 UML-like graphical representation of OntoML constructs

In this part of ISO 13584, OntoML is described using UML notations. The basic UML notations are enriched in order to:

- highlight the difference between an XML element and an XML attribute;
- explicitly represent references between XML complex type represented in different diagrams;
- represent reference(s) between CIIM ontology concepts using the specified identification mechanism;
- simplify the diagram when representing references between CIIM ontology concepts.

These graphical notations are called UML-like notations.

This clause presents the UML-like graphical representation of OntoML constructs. It also describes the mechanism used for representing in OntoML references between CIIM 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.

6.3.1 Graphical notations

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

6.3.1.1 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 can 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 defined 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

6.3.1.2 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 colour. It is illustrated in Figure 4 below.

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

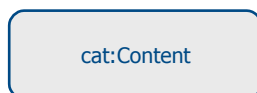


Figure 4 — UML-like representation of an external reference to an XML complex type

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

6.3.1.3 Representation of XML attributes and XML elements whose content models are XML simple types

Both XML attributes and XML elements whose contents 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 (nested) elements, the name of the former is prefixed by the “@” character.

NOTE The ‘@’ character is not part of the attribute name. Therefore, it is not represented in the OntoML XML Schema.

In Figure 5 below, a **PROPERTY_Type** is an abstract 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 **PropertyId** XML simple type.

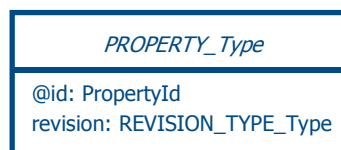


Figure 5 — UML-like representation of XML attributes and simple type XML elements

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 XML elements

6.3.1.4 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 **ANY_TYPE_Type** XML complex type.

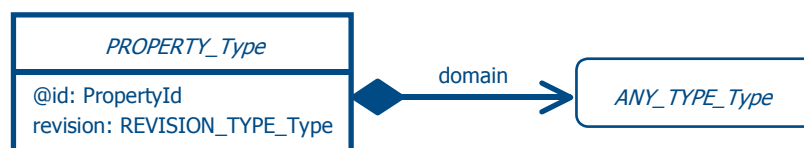


Figure 7 — UML-like representation of an XML complex type XML element

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="ANY_TYPE_Type"/>
  </xs:sequence>
  <xs:attribute name="id" type="PropertyId" use="required"/>
</xs:complexType>
```

Figure 8 — XML representation of an XML complex type element

6.3.1.5 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.

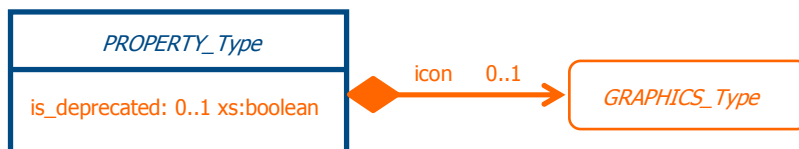


Figure 9 — UML-like representation of XML elements cardinality

NOTE 1 The cardinality relationships expressed in Figure 9 stand for optionality.

NOTE 2 Colour conventions are defined in Clause 6.3.3.

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

```
<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

6.3.1.6 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.

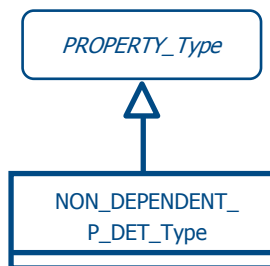


Figure 11 — UML-like representation of XML complex type extensions

EXAMPLE 2 Figure 12 below shows the XML document instance corresponding to Figure 11.

```

<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

6.3.2 Reference mechanism between CIIM ontology concepts

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

6.3.2.1 Identification of a CIIM ontology concept

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

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

- the names of these XML complex types reflect the names of their target concept types,
- each of these XML complex types contains an attribute whose value is the global identifier of the particular CIIM 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: it is of **SUPPLIER_Type** XML complex type; the type of the identifier is **SupplierId**;
- class: it is of **CLASS_Type** XML complex type; the type of the identifier is **ClassId**;
- property: it is of **PROPERTY_Type** XML complex type; the type of the identifier is **PropertyId**;
- data type: it is of **DATATYPE_Type** XML complex type; the type of the identifier is **DatatypeId**;
- document: it is of **DOCUMENT_Type** XML complex type; the type of the identifier is **DocumentId**.

NOTE The structure of these identifier types are given in Clause 9.1.

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 a **@id** attribute whose data type is **ClassId**.



Figure 13 — Identification of a CIIM ontology concept

6.3.2.2 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 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 CIIM 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.

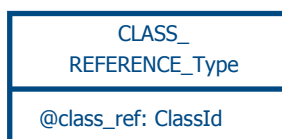


Figure 14 — CIIM ontology concept reference

6.3.2.3 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.

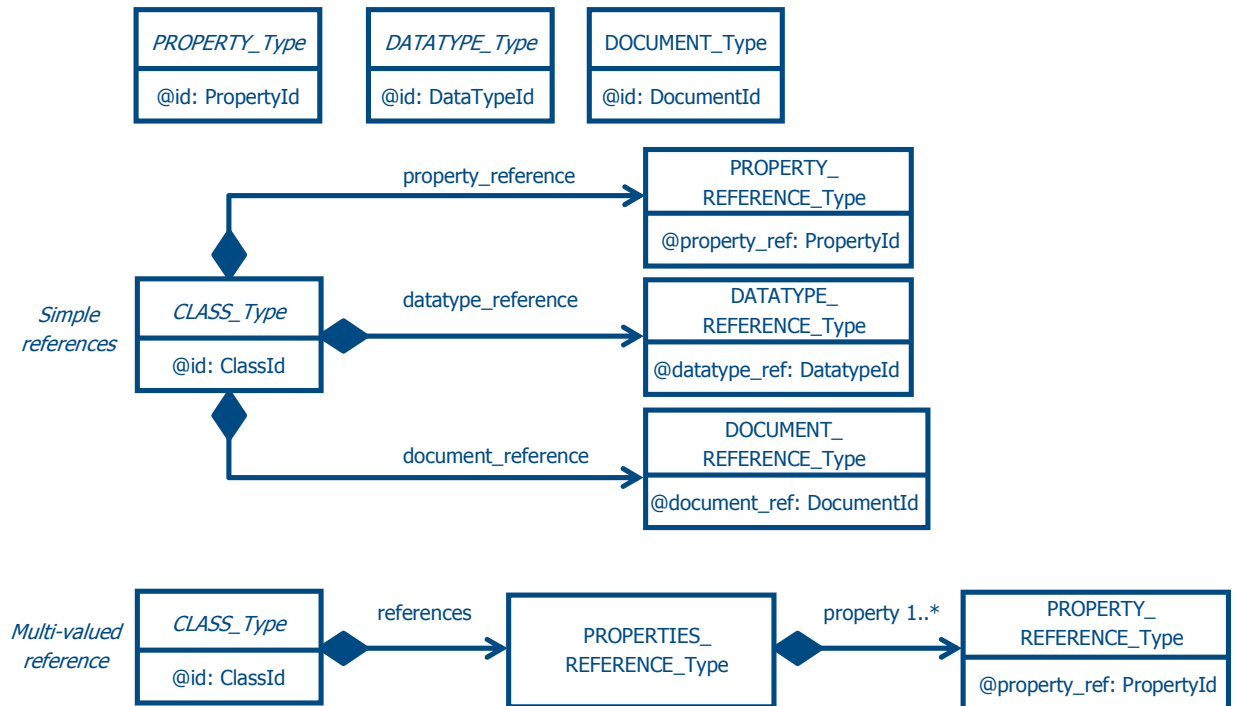


Figure 15 — Reference between CIIM ontology concepts

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, a data type, a document and a class ontology concepts, respectively represented by a **PROPERTY_Type**, **DATATYPE_Type**, **DOCUMENT_Type** and a **CLASS_Type** XML complex types are defined. All are identified by an **id** XML attribute whose type depends on the identified CIIM ontology concept. Two cases of relationships are presented:

- *simple reference*: it is a one to one relationship from a class to a property, a data type or a document. The relationship is represented by an XML element (respectively *property_reference*, *datatype_reference* and *document_reference*) whose content definition is respectively a **PROPERTY_REFERENCE_Type**, **DATATYPE_REFERENCE_Type** and a **DOCUMENT_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 its content definition is a **PROPERTIES_REFERENCE_Type** XML complex type.

6.3.2.4 Simplified graphical representation of references between CIIM ontology concepts

As explained in Clause 6.3.2.3, 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**;
- 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.

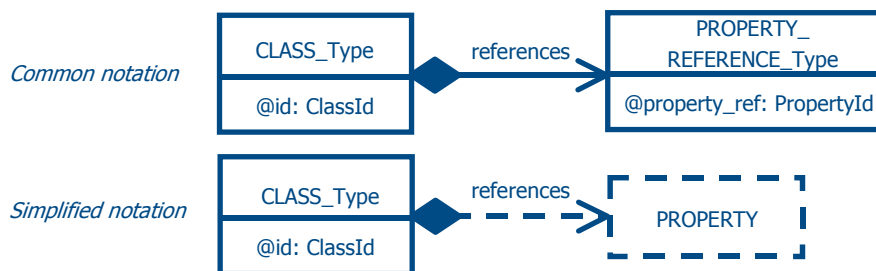


Figure 16 — UML-like representation of a simple reference between CIIM ontology concepts

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.

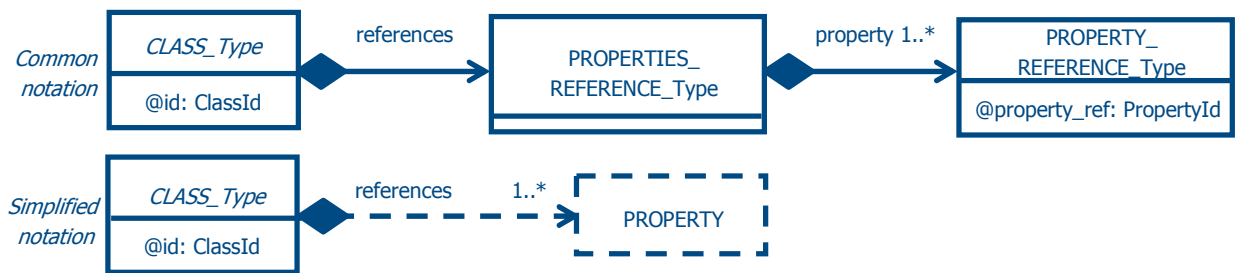


Figure 18 — UML-like representation of a multi-valued reference between CIIM ontology concepts

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

6.3.3 UML diagrams colour conventions

In UML diagrams, a colour convention is used to highlight those XML attributes and XML elements 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.

6.3.4 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 6.6 is titled "Root element 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.

6.3.4.1 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 can 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 structures are defined in the same figure as the one that defines their embedding complex-type.

EXAMPLE 1 In Figure 21, the **HEADER_Type** embeds an **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 21: it embeds different XML elements: **synonymous_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 6.5 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 6.5, 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 6.5, it is specified that **GRAPHICS_TYPE** is defined in Clause 8.2.2.2.

6.3.4.2 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 slash ('/')

EXAMPLE 1 Under the "Internal item definition" header of Clause 6.6, 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 parentheses.

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)**.

6.3.4.3 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.

6.3.4.4 External type definition

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

6.3.4.5 Constraint specification

Under the "Constraint specification" header of the clause, additional constraints that can not be represented using the UML conventions are listed.

6.4 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.

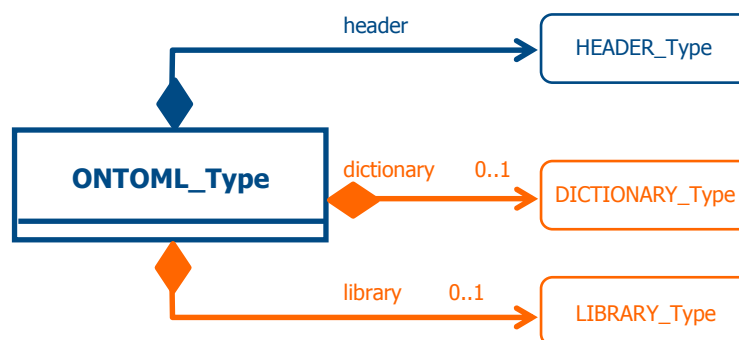


Figure 20 — Ontology structure UML diagram

Internal item definitions:

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

header: general information about the file that is exchanged.

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

External type definitions:

DICTIONARY_Type: the specification of the OntoML ontology, see 6.6.

HEADER_Type: the specification of the OntoML XML document header, see 6.5.

LIBRARY_Type: the specification of the OntoML library, see 7.

Constraint specification:

Either a dictionary XML element exists, or a library XML element exists or both exist.

6.5 OntoML header

The OntoML header provides the version of this part of ISO 13584 used to create the OntoML document instance and the human readable information about this 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.

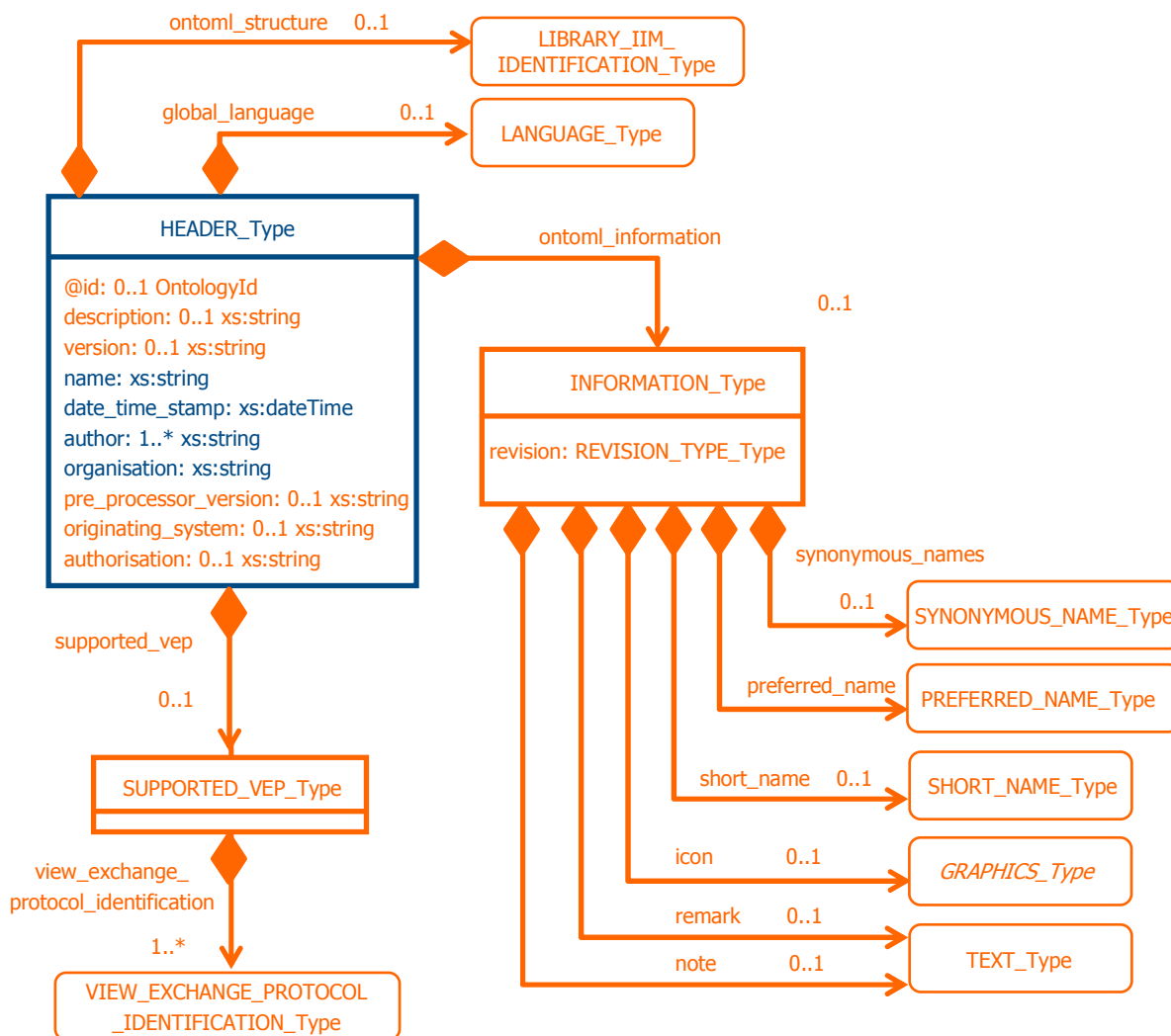


Figure 21 — Ontology header structure

Internal item definitions:

@id: the possible identifier of the dictionary to which the classes defined belong to.

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

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

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

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

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

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

ontoml_information/icon: an optional graphics which represents the description associated with the different names provided for describing the OntoML dictionary and / or library.

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

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

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

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

ontoml_information/short_name: the abbreviation of the preferred name.

ontoml_information/synonymous_names: the set of synonymous names.

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

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

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

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

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

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.

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

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 data type) 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 9.1.

GRAPHICS_Type: see 8.2.2.2.

LANGUAGE_Type: see 8.1.1.

LIBRARY_IIM_IDENTIFICATION_Type: see 8.7.1.

PREFERRED_NAME_Type: see 8.1.2.

SHORT_NAME_Type: see 8.1.2.

SYNONYMOUS_NAME_Type: see 8.1.2.

TEXT_Type: see 8.1.2.

VIEW_EXCHANGE_PROTOCOL_IDENTIFICATION_Type: see 8.7.2.

6.6 Root element 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.

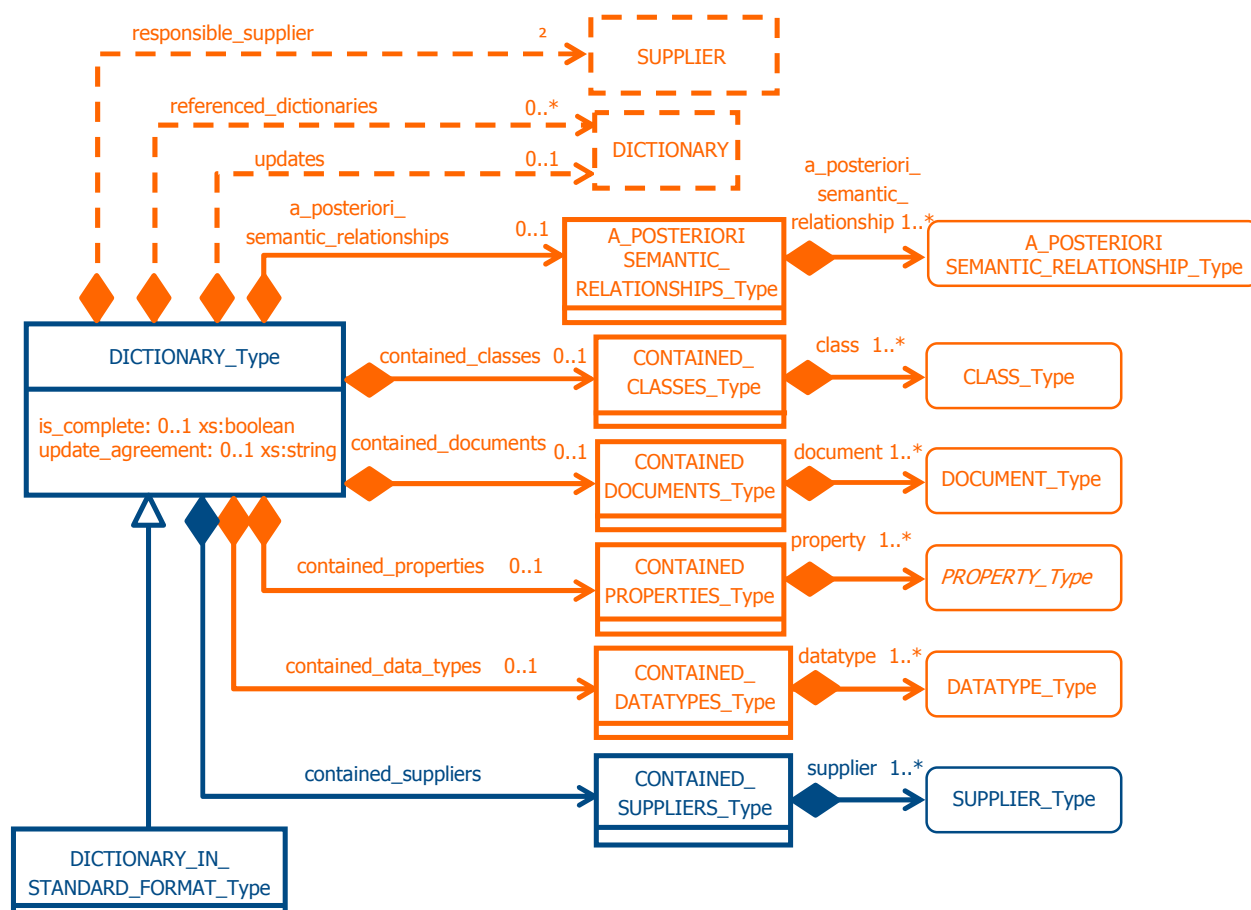


Figure 22 — Root element of an ontology

Internal item definitions:

a_posteriori_semantic_relationships: the list of *a posteriori* relationships contained in the dictionary.

a_posteriori_semantic_relationships/a_posteriori_semantic_relationship: an *a posteriori* relationship contained in the dictionary.

contained_classes: the list of class descriptions contained in the dictionary.

contained_classes/class: a class description contained in the dictionary.

contained_datatypes: the list of data type descriptions contained in the dictionary.

contained_datatypes/datatype: a data type description contained in the dictionary.

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

contained_documents/document: 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_suppliers: the list of supplier descriptions contained in the dictionary.

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

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.

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

responsible_supplier: the possible data supplier responsible for the ontology concepts.

NOTE 2 The supplier of all or part of the dictionary content is referenced as the **responsible_supplier** only when he/she is the responsible of the OntoML document instance. Else, he/she is referenced in the **contained_supplier** XML element.

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.

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

NOTE 3 The **updates** XML element can only exist when the **identified_by** XML element exists, and when the **is_complete** XML element is valued to false.

Internal type definitions:

CONTAINED_CLASSES_Type: sequence of class descriptions.

CONTAINED_DATATYPES_Type: sequence of data type descriptions.

CONTAINED_DOCUMENTS_Type: sequence of document descriptions.

CONTAINED_PROPERTIES_Type: sequence of property descriptions.

CONTAINED_SUPPLIERS_Type: sequence of supplier descriptions.

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 8.6.

CLASS_Type: dictionary class description, see 6.7.2.

DATATYPE_Type: dictionary datatype description, see 6.7.6.

DOCUMENT_Type: dictionary document description, see 6.7.7.

PROPERTY_Type: dictionary property description, see 6.7.4.

SUPPLIER_Type: supplier description, see 6.7.1.

Constraint specifications:

When the header part of the OntoML exchange file provides a product ontology identifier (**id** XML attribute of the **HEADER_Type** XML complex type), the **is_complete** XML element information must be provided.

When the header part of the OntoML exchange file provides a product ontology identifier (**id** XML attribute of the **HEADER_Type** XML complex type), the referenced **responsible_supplier** shall be the same than the supplier identified in the ontology identifier.

Updates shall not exist when the product ontology is not identified (**id** XML attribute of the **HEADER_Type** XML complex type), or when the **is_complete** XML element is set to true.

If both the product ontology is identified (**id** XML attribute of the **HEADER_Type** XML complex type) and **updates** has been defined, the identified product ontology shall have the same code and the same supplier as the one referenced by the **updates** XML element, and it shall have a version greater than the one that appears in the **updates** referenced dictionary.

6.7 OntoML representation of CIIM ontology concepts

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

6.7.1 Supplier

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

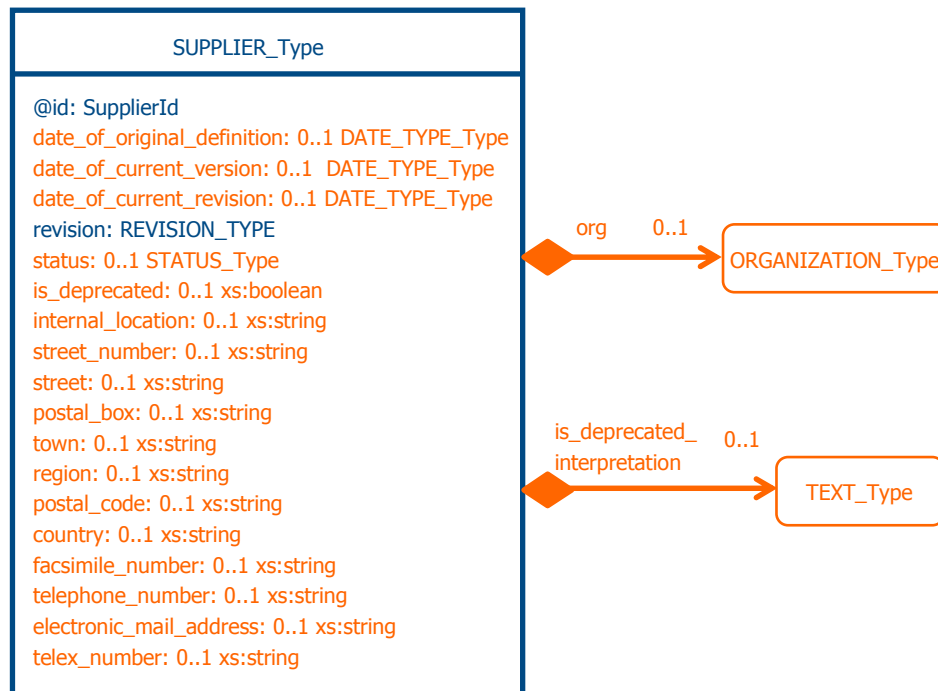


Figure 23 — Supplier ontology concept UML diagram

Internal item definitions:

@id: the supplier identifier.

country: the name of a country.

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.

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

facsimile_number: the number at which facsimiles can be received.

internal_location: organization-defined address for internal mail delivery.

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

is_deprecated_interpretation: specifies the deprecation rationale and how instance values of the deprecated supplier, and of its corresponding identifier, should be interpreted.

org: organizational data of this supplier.

postal_box: the number of a postal box.

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

region: the name of a region.

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.

street: the name of a street.

street_number: the number of a building in a street.

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

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

town: the name of a town.

Internal type definitions:

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

REVISION_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:

SupplierId: see 9.1.

ORGANIZATION_Type: see 8.8.1.

Constraint specification:

At least **internal_location**, **street_number**, **street**, **postal_box**, **town**, **region**, **postal_code**, **country**, **facsimile_number**, **telephone_number**, **electronic_mail_address** or **telex_number** shall have a value.

When **is_deprecated** exists, **is_deprecated_interpretation** shall exist.

Instance values of **is_deprecated_interpretation** element shall be defined at the time where deprecation decision was taken.

6.7.2 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 systems. Such a classification does not imply any additional properties.
- *item class case-of*: a special kind of item class that, besides inheriting properties from its possible *is-a* parent, borrows some properties from some other existing classes that encompass the item class case-of within their own scope.

NOTE Product characterization class and categorization class are defined in Clause 5 of ISO/IEC Guide 77-2:2008.

6.7.2.1 Item class

Figure 24 describes both the structure of a class and of an item class.

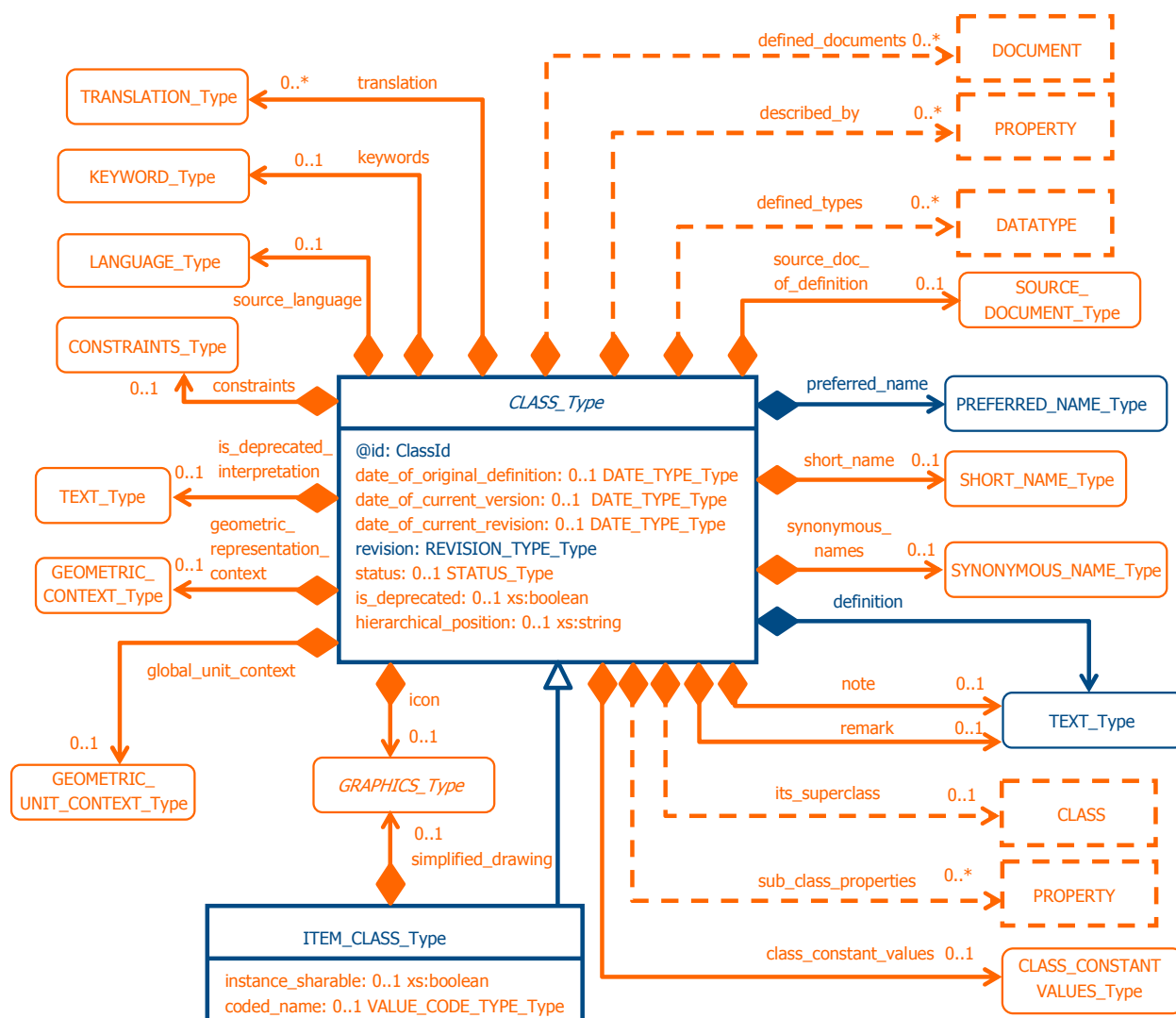


Figure 24 — Simple class ontology concept UML diagram

An item class (represented by the **ITEM_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.

class_constant_values: assignments in the current class for class-valued properties declared in superclasses.

NOTE 2 **class_constant_values** defines class selectors, as specified in Clause 5.5 of ISO/IEC Guide 77-2:2008.

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

NOTE 3 Each constraint in the **constraints** collection must be fulfilled by class instances. Thus the **constraints** collection is a conjunction of constraints.

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

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

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

definition: the text describing this class, possibly translated.

defined_documents: the set of references to the additional documents that can be used throughout the inheritance tree descending from this class.

NOTE 4 Every document referenced in the **defined_documents** collection is said applicable to the class.

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

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

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

NOTE 7 A property may also be applicable to a class when this property is imported from another class through an **ITEM_CLASS_CASE_OF_Type** class (see 6.7.2.3), a **FUNCTIONAL_MODEL_CLASS_Type** (see 6.7.3.2) or a **FM_CLASS_VIEW_OF_Type** (see 6.7.3.3). Therefore the properties referenced by the **described_by** attribute do not define all the applicable properties for a class.

NOTE 8 The list order is the presentation order of the properties suggested by the supplier.

geometric_representation_context: the specification of the reference coordinate system for every property of the class whose datatype is a STEP positioning entity, i.e., either a **PLACEMENT_Type_Type**, an **AXIS1_PLACEMENT_Type_Type**, an **AXIS2_PLACEMENT_2D_Type_Type** or an **AXIS2_PLACEMENT_3D_Type_Type**.

NOTE 9 STEP positioning entities are defined in Annex E, Clause E1.

NOTE 10 The positioning of the reference coordinate system with respect to the object defined by the class is described informally in the **description** element of the **geometric_representation_context**.

EXAMPLE 1 Consider an item class that describes cupboards whose top faces are rectangular. The supplier wants to define by **placements** the eight vertex of the shipping box for each cupboard. The geometric representation context, allowing then to define the eight vertex, might be defined by the following description: "the origin of the reference coordinate system is the intersection of the two diagonal lines of the cupboard top face, z axis moves upward, x axis is horizontal in the front direction of the cupboard".

NOTE 11 The **geometric_representation_context** is the OntoML representation of the **geometric_representation_context** defined in ISO 10303-42 for geometric representations. In ISO 10303-42, this context applies to all geometric representation items referenced by the representation. In OntoML, this context applies to all STEP positioning entities that are values of properties of the class where the **geometric_representation_context** is defined.

global_unit_context: the specification of the length unit, and possibly angle unit, that are assigned to the geometric representation context of all the STEP positioning entities of the class.

NOTE 12 The **global_unit_context** is the OntoML representation of the **global_unit_assigned_context** defined in ISO 10303-42 for geometric representations. In ISO 10303-42, this context applies to all geometric representation items referenced by the representation. In OntoML, this context applies to all STEP positioning entities that are values of properties of the class where the **global_unit_context** is defined.

NOTE 13 If **global_unit_context** is not provided, the default value for length measure is millimetre and for planar angle measure it is degree.

hierarchical_position: the coded representation of the class position in a class inclusion hierarchy to which it belongs

NOTE 14 This kind of coded name is used in particular in product categorization hierarchies for representing the class inclusion structure through some coding convention.

EXAMPLE 2 In UNSPSC, Manufacturing Components and Supplies has the hierarchical position 31000000, Hardware has the hierarchical position 31160000 and Bolt the hierarchical position 31161600. By convention, this representation of the hierarchical position allows to represent that Manufacturing Components and Supplies is at the first level of the hierarchy, that Hardware is at the second level of the hierarchy and is included in Manufacturing Components and Supplies, and that Bolt is at the third level of the hierarchy and is included in Hardware.

NOTE 15 A **hierarchical_position** of a class changes when the class structure of an ontology is changed. Thus it cannot be used as a stable identifier for classes.

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

instance_sharable (ITEM_CLASS_Type): when false, it specifies that instances of the item class are features; when not provided or true it specifies that instances of the item class are stand-alone items.

NOTE 16 In the common ISO13584/IEC61360 dictionary model, it is implementation dependent to decide whether several real world items modelled by the same set of property-values pairs should be represented in the data exchange file by several XML item description constructs or by the same XML item description construct. Thus, a single XML item description construct whose **instance_sharable** equals false and that is referenced by several XML item description constructs at the data model level is interpreted as representing several real world items.

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

is_deprecated_interpretation: specifies the deprecation rationale and how instance values of the deprecated class, and of its corresponding identifier, should be interpreted.

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

keywords: a set of keywords, possibly in several languages, allowing to retrieve the class.

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

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

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

revision: the revision number of the present class definition.

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

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

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

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

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

NOTE 17 Allowed **status** values are defined by private agreement between the dictionary supplier and dictionary users.

NOTE 18 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.

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

NOTE 19 **sub_class_properties** defines class selectors, as specified in Clause 5.5 of ISO/IEC Guide 77-2:2008.

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

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

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.

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 35 characters.

External type definitions:

ClassId: see 9.1.

ConstraintId: see 9.1.

CLASS_CONSTANT_VALUES_Type: see 6.7.2.4.

CONSTRAINT_Type: see 8.5.

GEOMETRIC_CONTEXT: see 8.8.3.

GEOMETRIC_UNIT_CONTEXT: see 8.8.4.

GRAPHICS_Type: see 8.2.2.2.

KEYWORD_Type: see 8.1.2.

LANGUAGE_Type: see 8.1.1.

PREFERRED_NAME_Type: see 8.1.2.

SHORT_NAME_Type: see 8.1.2.

SOURCE_DOCUMENT_Type: see 8.2.2.1.

SYNONYMOUS_NAME_Type: see 8.1.2.

TEXT_Type: see 8.1.2.

TRANSLATION_Type: see 8.1.3.

Constraint specifications:

The inheritance structure defined by the class hierarchy (defined through the **its_superclass** XML element inherited from the **CLASS_Type** XML complex type) shall not contain cycles.

Only those properties that are visible for a class may become applicable to this class by virtue of being referenced in the list specified by the **described_by** collection.

Only those data types that are visible for a class may become applicable to this class by virtue of being referenced in the list specified by the **defined_types** collection.

Only those properties that are not applicable for a class by inheritance may become applicable to this class by virtue of being referenced in the list specified by the **described_by** collection.

Only those data types that are not applicable for a class by inheritance may become applicable to this class by virtue of being referenced in the list by specified the **defined_types** collection.

Only context dependent properties (**DEPENDENT_P_DET_Type**, see 6.7.4) whose all context parameters (**CONDITION_DET_Type**, see 6.7.4) are applicable in the class may become applicable for this class by virtue of being referenced by its **described_by** XML element.

The **constraints** collection shall define restrictions that are compatible with the domain of values of the properties to which they apply.

All the properties referenced in the **precondition** collection of a constraint for which base type is a **CONFIGURATION_CONTROL_CONSTRAINT_Type** shall be applicable to the class.

All the properties referenced in the **constraints** collection shall be either visible or applicable to the class.

The properties referenced in the **sub_class_properties** collection shall also be referenced in the **described_by** collection.

The properties referenced in the **class_constant_values** collection were declared as class-valued in some superclass of the current class, or in the current class itself.

If a property referenced in the **class_constant_values** collection was already assigned a value in a superclass, the value assigned in the current class should be the same.

When **is_deprecated** exists, **is_deprecated_interpretation** shall exist.

Instance values of **is_deprecated_interpretation** element shall be defined at the time where deprecation decision was taken.

If **PLACEMENT_Type** or **AXIS1_PLACEMENT_Type** or **AXIS2_PLACEMENT_2D_Type** or **AXIS2_PLACEMENT_3D_Type** is the datatype assigned to a property referenced in the **described_by** collection, the corresponding **geometric_representation_context** XML element shall be provided.

6.7.2.2 Categorization class

A categorization class enables the modelling of a grouping of a set of objects that constitutes an element of a categorization.

EXAMPLE 1 Manufacturing Components and Supplies, Industrial optics, are example of product categorization class defined in UNSPSC.

Neither properties nor data types, nor constraints are associated with such a class. Moreover, categorization classes may not be related to each other by the is-a inheritance relationship, but they may only be related to each other through the is-case-of class relationship. A specific XML element, called **categorization_class_superclasses** allows to record the categorization classes that are superclasses of a categorization class in a case-of hierarchy.

NOTE Using the case-of resource constructs, item classes may also be connected to categorization classes.

EXAMPLE 2 The following example shows how characterization classes and categorization classes may be connected to achieve some particular goals. A ball bearing supplier wants to design its own ontology and to make it easy to retrieve and easy to use. To achieve these goals, he/she wants to use standard properties and to be connected to standard classifications. The supplier provides only ball bearings, but some bearings are sealed, some others are not. Particular properties may be associated with sealed bearings and with not sealed bearings, but these categories do not exist as classes in standard bearing ontologies. Thus, the bearing supplier processes as follows. (1) He/she designs a proprietary ontology consisting of three characterization classes: *my_bearing*, *my_sealed_bearings*, *my_non_sealed_bearing*. The two latter are connected to the former by the is-a inheritance relationship, and all the properties assigned to the former are inherited by the latter. (2) To use some of the properties defined in the future ISO/TS 23768-1, the bearing supplier specifies that his/her class *my_bearings* is case-of the standard bearing class *ball bearing* defined in ISO/TS 23768. Through this case-of relationship, he/she may import in his/her class *my_bearings* the standard-defined properties: *bore diameter*, *outside diameter*, *ISO tolerance class*. Moreover, he/she creates those needed properties that are not defined in the standard. (3) To facilitate the retrieval of the server that display the supplier's catalogue, he/she represents a small fragment of the UNSPSC classification, and a case-of relationship between the UNSPSC class *ball bearings* and its own class *my_bearing*. The result is presented in Figure 25 below.

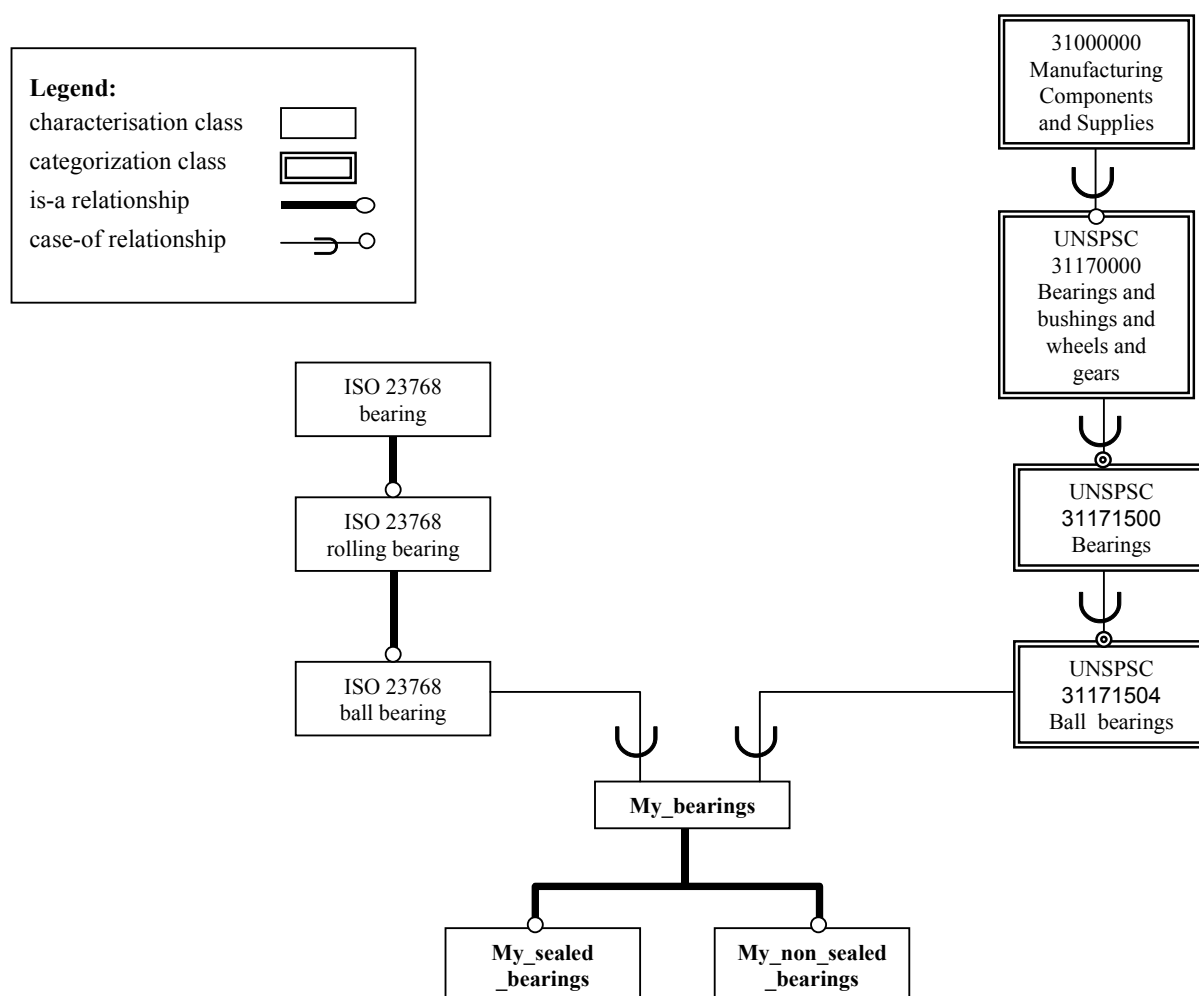


Figure 25 — Example of a supplier ontology using categorization classes

Figure 26 describes the structure of a categorization class.

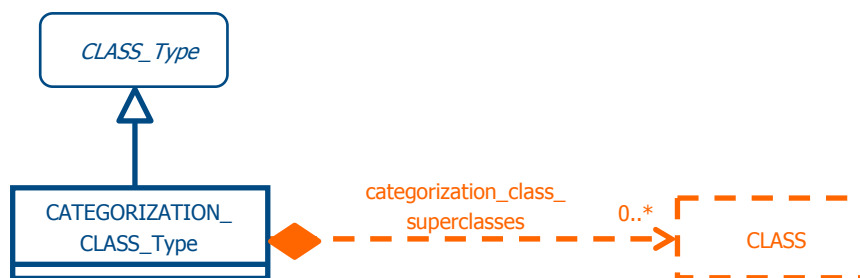


Figure 26 — Categorization class

A 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.

Internal item definitions:

categorization_class_superclasses: the categorization classes that are one step above the categorization class in a case-of class hierarchy.

Constraint specifications:

Only identifiers corresponding to classes for which the base type is **CATEGORIZATION_CLASS_Type** may be referenced in the **categorization_class_superclasses** collection.

The inherited **its_superclass** XML element shall not be specified when defining a categorization class.

The inherited **described_by** XML element shall not be specified when defining a categorization class.

The inherited **defined_types** XML element shall not be specified when defining a categorization class.

The inherited **sub_class_properties** XML element shall not be specified when defining a categorization class.

The inherited **class_constant_values** XML element shall not be specified when defining a categorization class.

The inherited **constraints** XML element shall not be specified when defining a categorization class.

A categorization class shall not be the property definition class of any property.

6.7.2.3 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 27.

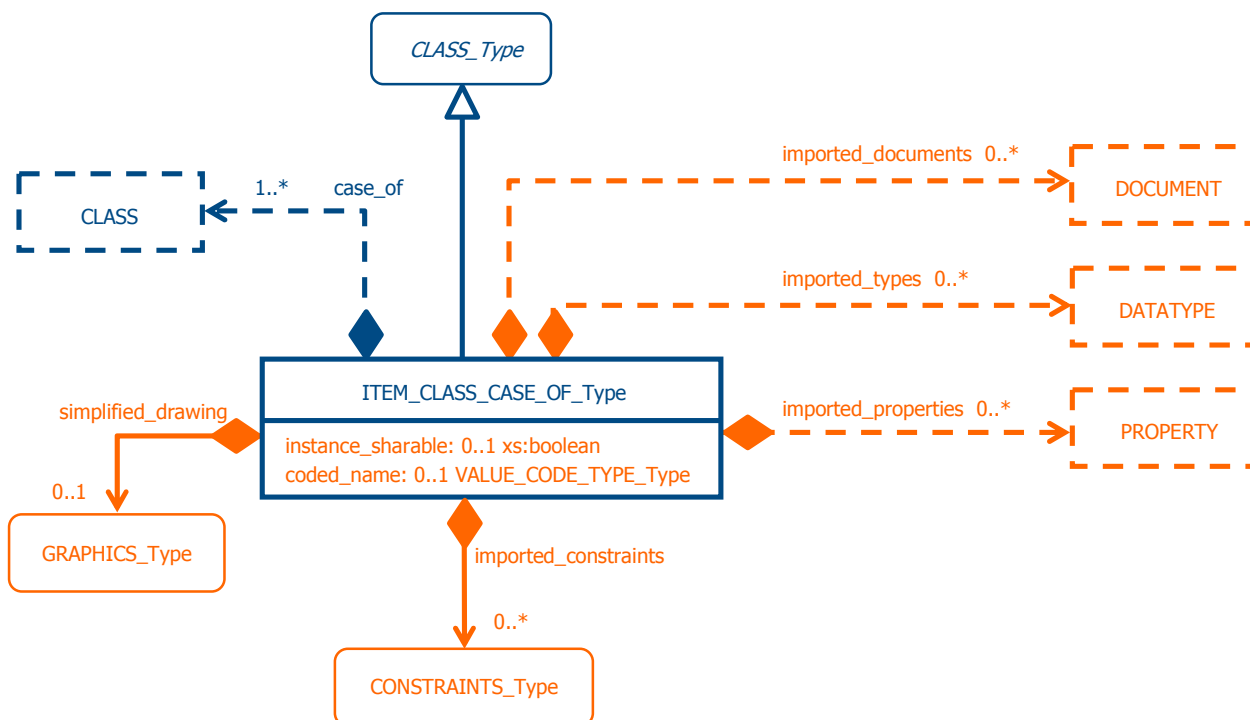


Figure 27 — Item class case-of UML diagram

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 3.5.2 of ISO/IEC Guide 77-2:2008.

imported_constraints: the set of constraints that are imported from the item class(es) the defined item class is case-of.

NOTE 2 Unlike other imported entities, the **imported_constraints** constraints cannot be selected when an item class case of is designed. These constraints are all the constraints that restrict the domains of any of the properties defined in the **imported_properties** collection in the classes of the **case_of** XML element from which they are imported.

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_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).

instance_sharable (ITEM_CLASS_Type): when false, it specifies that instances of the item class case-of are features; when not provided or true it specifies that instances of the item class case-of are stand-alone items.

NOTE 3 In the common ISO13584/IEC61360 dictionary model, it is implementation dependent to decide whether several real world instances of features modeled by the same set of property-values pairs are represented by several EXPRESS pieces of data or by the same piece of data in the data exchange file. Thus, an instance of an item class case-of whose **instance_sharable** equals *false* and that is referenced by several instances of item classes case-of at the data model level is interpreted as several real world instances of the same feature.

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

Internal type definitions:

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 35 characters.

External type definitions:

CLASS_Type: see 6.7.2.1.

CLASS_VALUE_ASSIGNMENT_Type: see 6.7.2.4.

CONSTRAINTS_Type: see 8.5.1.

GRAPHICS_Type: see 8.2.2.2.

Constraint specifications:

The inheritance structure defined by the class hierarchy (defined through the **its_superclass** XML element inherited from the **CLASS_Type** XML complex type) shall not contain cycles.

Only those properties that are visible for a class may become applicable to this class by virtue of being referenced in the list specified by the **described_by** collection.

Only those data types that are visible for a class may become applicable to this class by virtue of being referenced in the list specified by the **defined_types** collection.

Only those properties that are not applicable for a class by inheritance may become applicable to this class by virtue of being referenced in the list specified by the **described_by** collection.

Only those data types that are not applicable for a class by inheritance may become applicable to this class by virtue of being referenced in the list by specified the **defined_types** collection.

The **constraints** collection shall define restrictions that are compatible with the domain of values of the properties to which they apply.

All the properties referenced in the **precondition** collection of a constraint for which base type is a **CONFIGURATION_CONTROL_CONSTRAINT_Type** shall be applicable to the class.

All the properties referenced in the **constraints** collection shall be either visible or applicable to the class.

The properties referenced in the **class_constant_values** collection were declared as class-valued in some superclass of the current class, or in the current class itself.

If a property referenced in the **class_constant_values** collection was already assigned a value in a superclass, the value assigned in the current class should be the same.

Only an identifier corresponding to a class for which the base type is **ITEM_CLASS_Type** or an **ITEM_CLASS_CASE_OF_Type** may be referenced in the **its_superclass** XML element.

Only an identifier corresponding to a class for which the base type is **ITEM_CLASS_Type**, an **ITEM_CLASS_CASE_OF_Type** or a **CATEGORIZATION_CLASS_Type** may be referenced in the **case_of** collection.

The properties referenced in the **sub_class_properties** collection shall be referenced either in the **described_by** collection, or in the **imported_properties** collection.

All the class valued properties declared by being referenced in the **sub_class_properties** collection that are also referenced in the **imported_properties** collection shall be class valued properties in all the **case_of** classes where they are applicable.

The values assigned to an imported property in the **class_constant_value** collection shall not be different than the possible value assigned to the same property in the referenced classes.

All the properties referenced in the **imported_properties** collection that are class valued properties in a class referenced from the **case_of** collection, must be class valued properties in the current class.

All the properties referenced in the **imported_properties** that are assigned a class constant value in a class from the **case_of** collection, must be assigned the same class constant value class value in the current class.

Each constraint specified in the **imported_constraints** collection through the **constraint** XML element shall be defined either as a referenced constraint using the **constraint_ref** XML attribute or as a defined constraint using the **constraint_definition** XML element, not both.

6.7.2.4 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; the property is described as any other property, but with one restriction: the property is referenced in the **sub_class_properties** XML element of the associated class.
- a typed value may be assigned in every subclasses of this given class: 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_VALUES_Type** XML complex type as illustrated in Figure 28.

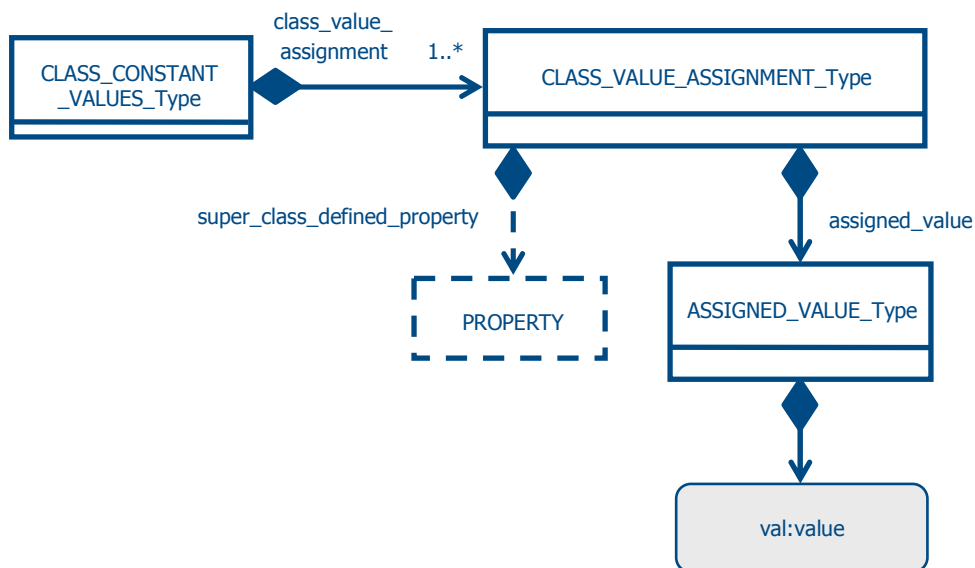


Figure 28 — Class value assignment structure

Internal item definitions:

class_value_assignment: the specification of all the class value assignments.

class_value_assignment/assigned_value: the value assigned to the property, valid for the whole class referring this class value assignment in its **class_constant_values** collection.

NOTE 1 The **assigned_value** belongs 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 value (**assigned_value** XML element) is assigned.

NOTE 2 A property is defined as being a subclass property when it appears 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:

ASSIGNED_VALUE_Type: the value to be assigned.

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 does not exceed 35 characters.

External type definitions:

val:value: specification of a typed value.

NOTE 3 **val:value** is defined in the ISO/TS 29002-10 product exchange format.

Constraint specification:

The value assigned by means of the **assigned_value** XML element shall be type compatible with the value domain of the associated property (referenced by the **super_class_defined_property** XML element).

6.7.3 Advanced-level ontology class

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

Characterization classes allow to capture the various kinds of items of an application domain, and to identify those characteristic properties that provide for discriminating items of the class by their values. Identical values for all characteristics properties means that items are identical. Different values for some properties means that items are different. Therefore, characteristic properties are supposed to be rigid (invariant for a given item).

EXAMPLE 1 When looking at the mechanical fasteners domain, we may identify the *metric threaded screw* as a characterization class of fasteners, and properties like *threaded length*, *total length*, *threaded diameter*, *material*, *part number* as characteristic properties allowing to characterize different ~~subsets of identical~~ items.

But properties of an item are not only characteristic properties. According to the discipline-oriented point of view we have on an item, a number of other properties prove useful. If an ontology user is in charge of *procurement* of screws, the properties *price* and *delivery delay* are necessary. If an ontology user is responsible for *inventory management*, the *inventory size* (the number of particular screws that are currently available in the inventory) and the *quantity of order* of this screw are relevant properties. If an ontology user designs products using a given CAD system, and wants to insert one screw in the current product, the *geometric shape* of the screw is necessary to make the design process efficient.

Compared with characteristic properties, the properties considered above have two differences that suggest criteria for identifying them:

- each of these properties makes sense only for some particular points of view, usages or disciplines of the item user;
- most of these properties are not characteristics of an item: they may change without changing the target item.

To allow an ontology designer to separate this kind of properties from the characteristic properties and to structure all the discipline-oriented properties that may be associated with an item, OntoML provides two additional categories of classes:

- functional view classes that provide for representing discipline-oriented points of view on items;

EXAMPLE 2 Procurement, inventory, marketing; geometry (3D, simplified) or geometry (2D, front view, precise), are examples of points of view.

NOTE 1 As shown in the geometry example, specifying a point of view may require not only a *name* ("geometry"), but also values of some variables such that *geometry_level* (2D, 3D), *side* (top, front, ...) or *level_of_detail*. Such variables are called view control variables.

- functional model classes that provide for representing items of some characterization class(es) according to the discipline-oriented point of view defined by a functional view class.

NOTE 2 Each functional model class references one functional view class for defining its discipline-oriented point of view.

NOTE 3 If a functional model class does not reference a characterization class in an exchanged ontology, it is intended to be associated with a characterization class on the target user system.

EXAMPLE 3 A *procurement* functional model class might contain values for the following properties: *part number* (imported from a *metric threaded screw* characterization class), *price*, *delivery delay*, for each screw in the *metric threaded screw* class.

EXAMPLE 4 The set of 2D precise front geometrical representations of all the screws in a *metric threaded screw* class could constitute a functional model class that provides a geometrical representation of items of this class.