TECHNICAL SPECIFICATION

ISO/TS 19139

First edition
2007-04-15

Geographic information — Metadata — XML schema implementation

Information géographique — Métadonnées — Implémentation de schémas XML

Reference number
ISO/TS 19139:2007(E)

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.

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

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.

In other circumstances, particularly when there is an urgent market requirement for such documents, a technical committee may decide to publish other types of normative document:

— an ISO Publicly Available Specification (ISO/PAS) represents an agreement between technical experts in an ISO working group and is accepted for publication if it is approved by more than 50 % of the members of the parent committee casting a vote;

— an ISO Technical Specification (ISO/TS) represents an agreement between the members of a technical committee and is accepted for publication if it is approved by 2/3 of the members of the committee casting a vote.

An ISO/PAS or ISO/TS is reviewed after three years in order to decide whether it will be confirmed for a further three years, revised to become an International Standard, or withdrawn. If the ISO/PAS or ISO/TS is confirmed, it is reviewed again after a further three years, at which time it must either be transformed into an International Standard or be withdrawn.

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/TS 19139 was prepared by Technical Committee ISO/TC 211, *Geographic information/Geomatics*.

Introduction

The importance of metadata describing digital geographic data is explained in detail in the text of ISO 19115. ISO 19115 provides a structure for describing digital geographic data by defining metadata elements and establishing a common set of metadata terminology, definitions and extension procedures. ISO 19115 is abstract in that it provides a worldwide view of metadata relative to geographic information, but no encoding.

Since ISO 19115 does not provide any encoding, the actual implementation of geographic information metadata could vary based on the interpretation of metadata producers. In an attempt to facilitate the standardization of implementations, this comprehensive metadata implementation specification provides a definitive, rule-based encoding for applying ISO 19115. This Technical Specification provides Extensible Markup Language (XML) schemas that are meant to enhance interoperability by providing a common specification for describing, validating and exchanging metadata about geographic datasets, dataset series, individual geographic features, feature attributes, feature types, feature properties, etc.

ISO 19115 defines general-purpose metadata in the field of geographic information. More detailed metadata for geographic data types and geographic services are defined in other ISO 19100 series standards and user extensions (ISO 19115). This Technical Specification is also intended to define implementation guidelines for general-purpose metadata. Where necessary, interpretations of some other ISO 19100 series standards are incorporated.

ISO 19118 describes the requirements for creating encoding rules based on UML schemas and the XML-based encoding rules as well as providing an introduction to XML. This Technical Specification utilizes the encoding rules defined in ISO 19118 and provides the specific details of their application with regard to deriving XML schema for the UML models in ISO 19115.

Geographic information — Metadata — XML schema implementation

1 Scope

This Technical Specification defines Geographic MetaData XML (gmd) encoding, an XML schema implementation derived from ISO 19115.

2 Conformance

Conformance with this Technical Specification shall be checked using all the relevant tests specified in Annex A. The framework, concepts, and methodology for testing, and the criteria to be achieved to claim conformance are specified in ISO 19105.

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

ISO 3166 (all parts), *Codes for the representation of names of countries and their subdivisions*

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

ISO/IEC 10646, *Information technology — Universal Multiple-Octet Coded Character Set (UCS)*

ISO/TS 19103, *Geographic information — Conceptual schema language*

ISO 19105, *Geographic information — Conformance and testing*

ISO 19107, *Geographic information — Spatial schema*

ISO 19108, *Geographic information — Temporal schema*

ISO 19109, *Geographic information — Rules for application schema*

ISO 19110, *Geographic information — Methodology for feature cataloguing*

ISO 19111:—[1)], *Geographic information — Spatial referencing by coordinates*

ISO 19115:2003, *Geographic information — Metadata*

1) To be published. (Revision of ISO 19111:2003)

ISO 19115:2003/Cor. 1:2006, *Geographic information — Metadata — Technical Corrigendum 1*

ISO 19117, *Geographic information — Portrayal*

ISO 19118:2005, *Geographic information — Encoding*

ISO 19136:—[2)], *Geographic information — Geography Markup Language (GML)*

W3C XMLName, *Namespaces in XML. W3C Recommendation* (14 January 1999)

W3C XMLSchema-1, *XML Schema Part 1: Structures. W3C Recommendation* (2 May 2001)

W3C XMLSchema-2, *XML Schema Part 2: Datatypes. W3C Recommendation* (2 May 2001)

W3C XML, *Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation* (6 October 2000)

W3C XLink, *XML Linking Language (XLink) Version 1.0. W3C Recommendation* (27 June 2001)

4 Terms and definitions

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

4.1
namespace
collection of names, identified by a URI reference, that are used in XML documents as element names and attribute names

[W3C XML]

4.2
package
general purpose mechanism for organizing elements into groups

[ISO/TS 19103, definition 4.2.22]

EXAMPLE Identification information; Metadata entity set information; Constraint information.

4.3
realization
semantic relationship between classifiers, wherein one classifier specifies a contract that another classifier guarantees to carry out

[Booch 1999]

4.4
polymorphism
characteristic of being able to assign a different meaning or usage to something in different contexts – specifically, to allow an entity such as a variable, a function, or an object to have more than one form

NOTE There are several different kinds of polymorphism.

[http://searchsmallbizit.techtarget.com]

2) To be published.

5 Symbols and abbreviated terms

5.1 Acronyms

UML	Unified Modelling Language
XCT	XML Class Type
XCPT	XML Class Property Type
XCGE	XML Class Global Element
XML	Extensible Markup Language
XPath	XML Path Language
XSD	XML Schema Definition
XSL	Extensible Style Language
XSLT	XSL Transformation

5.2 Namespace abbreviations

In the lists below, the item on the left describes the common namespace prefix used to describe the elements in the namespace. The second item is an English description of the namespace prefix, and the item in parenthesis is the URI of the actual namespace. These URIs do not correspond necessarily to an effective location of the schemas.

This first list corresponds to the namespaces defined by this Technical Specification.

gco	Geographic Common extensible markup language	(http://www.isotc211.org/2005/gco)
gmd	Geographic MetaData extensible markup language	(http://www.isotc211.org/2005/gmd)
gmx	Geographic Metadata XML Schema	(http://www.isotc211.org/2005/gmx)
gss	Geographic Spatial Schema extensible markup language	(http://www.isotc211.org/2005/gss)
gsr	Geographic Spatial Referencing extensible markup language	(http://www.isotc211.org/2005/gsr)
gts	Geographic Temporal Schema extensible markup language	(http://www.isotc211.org/2005/gts)

This second list corresponds to external namespaces used by this Technical Specification.

gml	Geography Markup Language	(use the GML namespace URI stated in ISO 19136)
xlink	XML Linking Language	(use the XLINK namespace URI stated in the W3C XLink recommendation)
xs	W3C XML base schemas	(use the XML schema namespace URI stated in the W3C XMLSchema-1 and W3C XMLSchema-2 recommendations)

5.3 UML model relationships

The diagrams that appear in this Technical Specification are presented using the Unified Modelling Language (UML) as the conceptual schema language as defined in ISO/TS 19103. ISO 19115:2003, Figure 2, also displays the UML notation that is used to describe the metadata. In addition to the UML described in ISO/TS 19103 and shown in ISO 19115, this Technical Specification uses the notation shown in Figure 1.

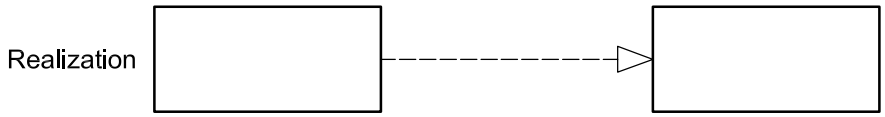

Figure 1 — UML Notation for Realization

The class that is the source of the connection (shown on the left in Figure 1) is guaranteed to carry out (or implement) the specification of the class at the destination of the connection (shown on the right in Figure 1).

5.4 UML model stereotypes

A UML stereotype is an extension mechanism for existing UML concepts (see ISO 19115). In addition to the stereotypes already defined in ISO 19115 for describing metadata, this Technical Specification defines stereotypes necessary for a rules-based encoding into XML schema.

The elements of the UML diagrams depicted in Clause 9 can carry stereotypes specifying an XML implementation. Those stereotypes are carried by classes representing XML elements or XML types, UML attributes, UML links (realizations or dependencies) and UML packages.

In this Technical Specification the following stereotypes of classes are used.

a) <<xs:choice>>: the class represents an implementation type encoded as an XML choice block. Each property of the class is implemented as an element of the choice.

b) <<xs:complexType>>: the class represents an implementation type encoded as an XML complex type.

c) <<xs:element>>: the class represents an XML global element.

d) <<xs:simpleType>>: the class represents an implementation type encoded as an XML simple type.

e) <<xs:simpleContent>>: the class represents an implementation type encoded as an XML complex type with simple content.

In this Technical Specification the following stereotypes of attributes are used.

f) <<xs:attribute>>: the property is encoded as an XML attribute.

g) <<xs:element>>: the property is encoded as an XML element with a name and a type (<xs:element name=”propertyName” type=”propertyType”/>)

h) <<pattern>>: this stereotype only applies to XML simple types derived from built-in XML types. A property with the stereotype <<pattern>> restricts the range of a built-in simple type.

i) <<XCGE>>: the property is encoded as a reference to an XML global element (<xs:element ref=”XCGE”/>).

In this Technical Specification the following stereotypes of links are used.

j) <<XCT>>: (carried by realization relationships). The XCT of the abstract concept to implement is substituted by the specified external implementation.

k) <<XCGE>>: (carried by realization relationships). The XCGE of the abstract concept to implement is substituted by the specified external implementation.

l) <<XCPT>>: (carried by realization relationships). The XCPT of the abstract concept to implement is substituted by the specified external implementation.

m) <<implement>>: (carried by dependency relationships). The source represents an XML schema implementing the abstract concepts defined in the target.

n) <<include>>: (carried by dependency relationships). The source and the target represent XML schemas. The source includes (<xs:include … />) the target.

o) <<import>>: (carried by dependency relationships). The source and the target represent sets of XML objects grouped within the same namespace. The source imports (<xs:import … />) the target.

In this Technical Specification the following stereotypes of packages are used.

p) <<xmlSchema>>: The package represents an XML schema.

q) <<xmlNamespace>>: The package represents a set of XML objects grouped within the same namespace.

6 Requirements

6.1 Introduction to gmd

Geographic metadata is represented in ISO 19115 as a set of UML packages containing one or more UML classes. ISO 19115 provides a universal, encoding-independent view of geographic information metadata. This Technical Specification provides a universal implementation of ISO 19115 through an XML schema encoding that conforms to the rules described in ISO 19118.

While the details of XML namespaces are not included in this Technical Specification, the contents of several namespaces are defined here. A namespace is really a collection of names which can be used in XML documents as element or attribute names. The namespace is used to identify the names with a particular schema. A namespace is a URI, and the ones utilized in this Technical Specification are listed in 5.2. A URI is often cumbersome for reading, writing and including in human discussion, so this Technical Specification will more often refer to common namespace prefixes when identifying particulars about the contents of a namespace. The primary namespace defined in this Technical Specification is http://www.isotc211.org/2005/gmd and the namespace prefix is **gmd** which stands for ***G**eographic **M**eta**D**ata extensible markup language*.

XML schema offers many alternatives for structuring information for exchange. ISO 19118 defines a set of encoding rules for transforming a UML conceptual schema from the ISO 19100 series of documents into an XML schema. Even within the pared down limitations of ISO 19118, there are still choices for the creation of specific XML schemas. Clauses 7, 8 and 9 describe the details of encoding the ISO 19115 UML conceptual schema and the UML models depicted in the respective ISO 19100 series of International Standards listed in Clause 3, into a set of XML schemas. A description of geographic ISO/TS 19139 XML resources and examples of metadata instance documents are included in Annexes C and D.

Before delving into the details of the encoding it is important to understand why certain encoding rules are utilized in the development of gmd. Gaining an understanding of the rules will make the capabilities, limitations and best-practice use of gmd clear. Some of the major goals for gmd were interoperability with other ISO 19100 series specifications, predictability, extensibility and usability. Further details of these goals are described in 6.2 to 6.8.

6.2 Rule-based

This XML schema implementation is a rule-based encoding built from the UML models in the ISO 19100 series of International Standards as required by ISO 19118. Using this methodology achieves a couple of the goals mentioned in 6.1. First, the resulting gmd schemas are based directly on other ISO 19100 series International Standards and therefore increase the chance for interoperability. Second, the resulting schema is predictable since any class, attribute, association, etc. is encoded just as other UML elements of the same type are encoded.

Although not discussed in detail in this Technical Specification, having a rule-based encoding also allows the XML schemas to be generated in an automated or semi-automated fashion.

6.3 Quality

Quality in terms of the XML schema implementation implies simple XML schemas as well as the human readability of the XML files. The structural complexity of the ISO 19100 series, particularly ISO 19115, implies that it is not possible to provide simple XML schemas for geographic metadata. But the encoding rules are defined so that a user can directly create and/or understand the content of an XML instance document using the UML models in ISO 19115 as a basis. Additionally, an implementer can determine the XML schema implementation of the UML models in ISO 19115 by knowing the encoding rules.

Another aspect to the quality of gmd is completeness. This Technical Specification encodes the entire UML model from ISO 19115 without regard to a particular usage for gmd or a particular application schema that will utilize gmd.

6.4 Web implementations

One of the goals stated in 6.1 is usability. Usability, as it pertains to the design of gmd, focuses on the exchange of geographic metadata with the understanding that this will often happen in a web-like environment. While there is no restraint against creating geographic metadata instance documents based on gmd which never transfer across a network, there are many aspects to the design that are intended to aid internet and web-like transfer of data.

6.5 Use of external XML implementations

Another design principle that aids interoperability and usability is the use of existing XML schemas. If an XML schema standard already exists that encodes a part of the ISO 19100 series pertaining to geographic metadata then it is advantageous to incorporate that XML schema standard into gmd. If gmd uses the external XML schema directly, then interoperability is enhanced. It is also likely that software already exists that can process instance documents that conform to the external XML. Furthermore, if the external schema is well designed it might be more efficient than XML schema generated from a series of encoding rules and this might help achieve the goal of usability.

While using an implementation that already exists has some important advantages, the external XML schemas should not violate the primary design principles of gmd. For example, if an external XML schema implements part of the ISO 19100 series but does so in a cumbersome, unusable manner then it is not incorporated into gmd. Additionally, if an external XML schema does not readily meet the requirements stated in 6.6 for multilingual support then it is not incorporated into gmd.

6.6 Multilingual support

Cultural and linguistic adaptability is a basic requirement for any textual metadata elements. In Annex J of ISO 19115:2003, there is an informative discussion of multilingual textual metadata elements. In order to enhance the chances for interoperability of implementations it is important that a normative mechanism for multilingual support be included in this Technical Specification. The details of multilingual support are described in 7.3 and 9.8.6 but for the sake of understanding the design of gmd it is important to understand that special consideration is given to this requirement. The specific mechanisms used to achieve this goal are polymorphism and codelist registers. Polymorphism is introduced in 6.7 and codelist registers are described in 7.4.4.4.

It is also important to understand that the multilingual support that exists in W3C XML is not sufficient for the expression of geographic metadata. In W3C XML, “A special attribute named xml:lang may be inserted in documents to specify the language used in the contents and attribute values of any element in an XML document [W3C XML]”. If a particular element can only occur once based on the encoding rules discussed in Clause 8 then the technique of using the special xml:lang attribute to indicate the language does not allow for the specification of the same element in two or more languages.

6.7 Polymorphism

The term polymorphism is formally defined in Clause 4. In general terms, polymorphism means the ability to assume different forms. In terms of this Technical Specification, the first obvious use for polymorphism is for providing cultural and linguistic adaptability. This allows implementers to provide geographic metadata in one or more languages without violating any cardinal rules defined in ISO 19115. Polymorphism provides more than just support for multiple languages. It also allows user communities to better refine geographic metadata to meet their organizational needs. For example, ISO 19115 contains an *individualName* attribute of type *CharacterString* in the *CI_ResponsibleParty* class, but within an organization individuals may be described in a more compartmentalized form (e.g. by first, middle and last names). Polymorphism allows implementers to extend the more general format of *individualName* within their namespaces while still fully utilizing gmd and still providing usable and understandable instance documents for users outside of their organization. The characteristics of gmd that allow for polymorphism primarily derive from the property type encodings described in 8.4.

6.8 Rules for application schema

ISO 19109 defines the rules for application schema and is comprised of two categories of models that are related to metadata:

— **a General Feature Model** that determines the particular way metadata and quality elements relate to geographic features;

— **two interchange models**: the traditional data transfer model and the interoperability model, each of them implying an interrelation of the metadata with its resources.

With regards to the ISO 19109 General Feature Model, the following specific types of attribute can be defined for a feature type:

— metadata attributes (as instances of GF_MetadataAttributeType) whose data type is MD_Metadata or one of its subclasses;

— quality attributes (as instances of GF_QualitiyAttributeType) whose data type is DQ_Element or one of its subclasses.

The use of geographic metadata XML schema in the context of these rules for application schema basically consists of the use of the XML schema definitions corresponding to the metadata and quality attribute data types. The use of XML schema definitions when encoding feature types using ISO 19136 is described in 9.9.2.

With regards to the ISO 19109 interchange models, the *interoperability* model is based on data interchange by transactions and is designed for a large number of transactions involving simple interchange. In contrast, the *transfer model* is designed for a lesser number of transactions with large amounts of well-organized data. The ISO 19115 metadata conceptual schema and this Technical Specification are clearly designed to be a starting point for providing more coherent transfer of geospatial data among and within user communities (i.e. *transfer model*).

Additionally, it is important to understand that the *interoperability* model applies to the interaction between the user application and a service provider, and that the interaction is fully determined by the service interface. While the service interface may not *require* the use of a specific schema, the adoption of the geographic metadata XML schema within an information-sharing community is highly recommended when applicable. Interoperable interchanges by transfer needs to go further in terms of standardization. This is the purpose of the “Extensions for metadata-based transfers of geospatial information” presented in 7.4.

7 Extensions to the UML models in the ISO 19100 series of International Standards

7.1 Introduction to extensions

It has already been stated that ISO 19115 provides an internationally-accepted, encoding-independent view of geographic information metadata and that this Technical Specification provides a worldwide implementation of ISO 19115 through an XML schema encoding. Before delving into the specifics of the encoding it is important to recognise that once a specific implementation technology is identified there may be some specific extensions required for the ISO 19100 series UML models. The purpose of these extensions may vary but they are mainly intended to support the requirements stated in Clause 6. An extension might be created to facilitate interoperability, ease of use, web-like environments, etc. The UML diagrams shown throughout Clause 7 are the extensions identified to support an XML schema encoding of ISO 19115 and its related ISO 19100 series of International Standards.

7.2 Extensions specific to the web environment

There are several extensions of the CharacterString class from ISO/TS 19103 that are necessary to add convenience when working with XML documents. These extensions are specific to the World Wide Web environment where XML documents are typically processed. Figure 2 defines the metadata required to describe elements specific to working in a web environment. The data dictionary for this diagram is located in B.2.1.

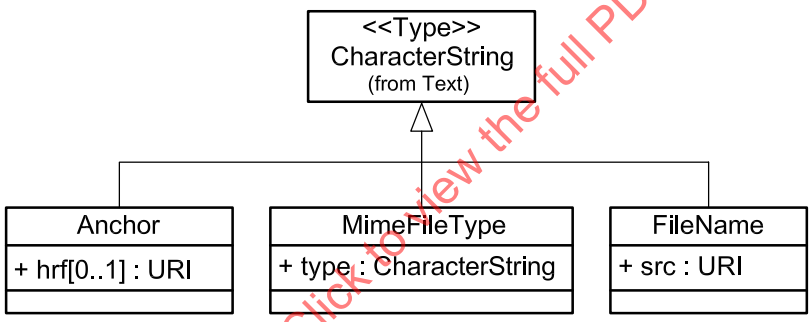

Figure 2 — Extensions to support web environments

The Anchor class is needed to support hyper-linking capabilities and to ensure a web-like implementation of ISO 19115's RS_Identifier and MD_Identifier classes. Because Anchor is a subclass of CharacterString, the *code* attribute of RS_Identifier and MD_Identifier can be implemented by instantiating Anchor and thus provide a reference to the place where the information related to the code is available.

The FileName class is needed to support explicitly referencing of an external file corresponding to a property containing the name of the file. This is valuable in the case of ISO 19115's MD_BrowseGraphic class with regards to the fileName attribute. A human readable file name might be a useful value for population of this attribute, but the additional src attribute provided by the FileName class can provide a machine-readable absolute path to the location of the file.

The MimeFileType class is needed to support identification of the file type using the mime media type name and subtype name. This is useful in the case of ISO 19115's MD_BrowseGraphic class with regards to the fileType attribute. The value of the fileType attribute might be “JPEG” and the type attribute of the MimeFileType class allows for the machine-readable Mime-type content-type expression such as “image/jpeg”.

7.3 Cultural and linguistic adaptability extensions

7.3.1 Free text

The free text element in the domain of a CharacterString property type in ISO 19115 is intended to support a textual metadata element having multiple instances of the same information in different locales. A locale (identified as PT_Locale) is a combination of language, potentially a country, and a character encoding (i.e. character set) in which localized character strings are expressed. Annex J of ISO 19115:2003, describes this Free Text concept (identified as PT_FreeText) but does not include the conceptual schema to accompany the description. This Technical Specification makes the use of PT_FreeText normatively by providing a conceptual schema in Figure 3 as well as a corresponding data dictionary (see B.2.2).

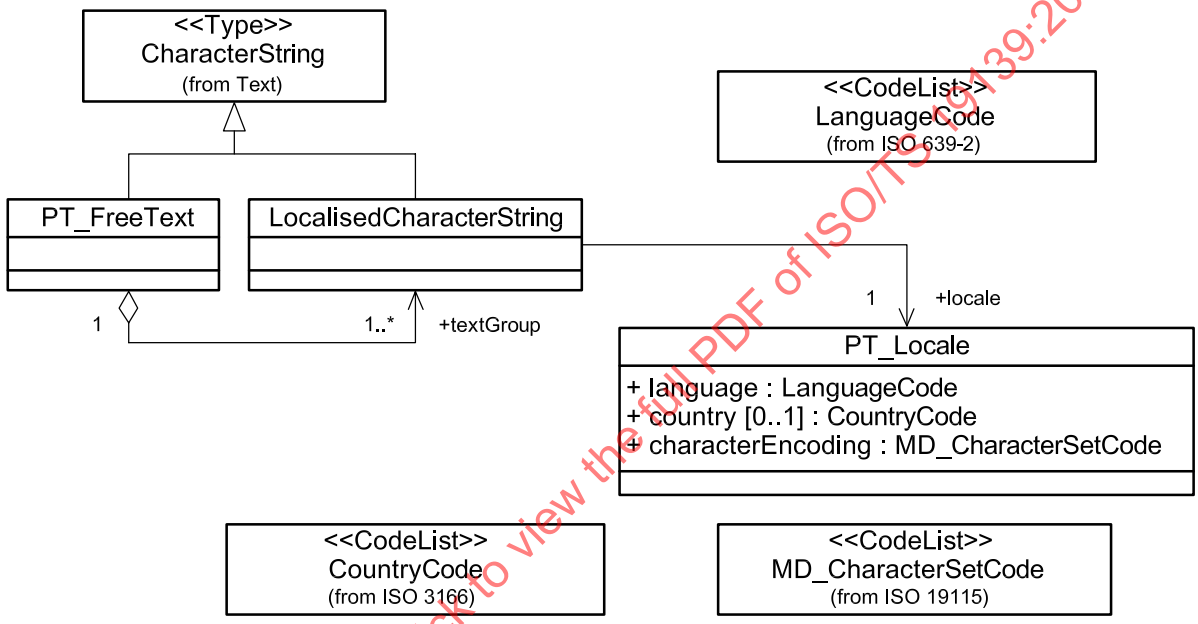

Figure 3 — Free text and localized character string

As a consequence of introducing the Locale concept (PT_Locale), the Group concept (identified as PT_Group) from ISO 19115:2003, Annex J is replaced by the concept of Localised String (identified as LocalisedCharacterString). LocalisedCharacterString is a subtype of CharacterString whose value is expressed in a single locale. An instance of a Free Text consequently is a CharacterString (with its value expressed in a default language and character set that could be defined in an instance of MD_Metadata), which also aggregates a series of Localised Character String translations via the *textGroup* role.

7.3.2 Multilingual metadata sets

An optional but repeatable attribute, locale, presented in Figure 4, was added to the class MD_Metadata of ISO 19115 by corrigendum. This attribute is instantiated if – and only if – the metadata set is multilingual (at least one of the metadata elements is an instance of PT_FreeText or one of its inherited classes).

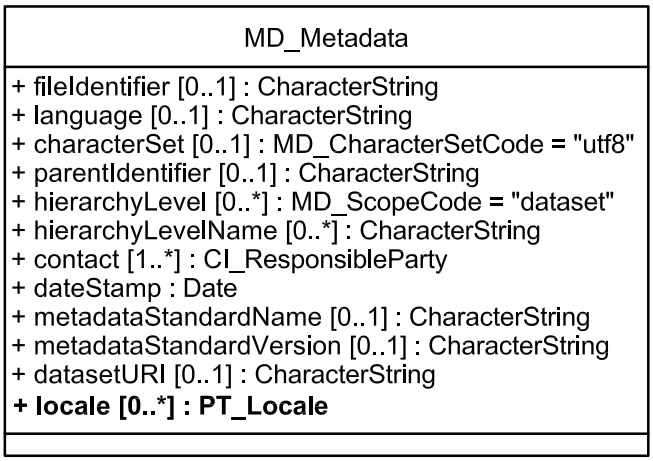

Figure 4 — Attributes of MD_Metadata in the geographic metadata XML schema

7.3.3 Management of localized strings

An instance of free text is composed of default character strings and their translations in different locales through the use of localized strings. This construct implies a distribution of localized strings throughout any given multilingual metadata set. However, a more common way of managing multilingual sets of information consists of grouping the localized strings per their locales. In order to ease the management of localized strings, this Technical Specification describes the concept of locale container (identified as PT_LocaleContainer). A locale container aggregates a set of localized strings related to a given locale (*locale* attribute of PT_LocaleContainer). There is no direct relationship between a locale container and a metadata set except that a locale container may aggregate localized strings of a metadata set.

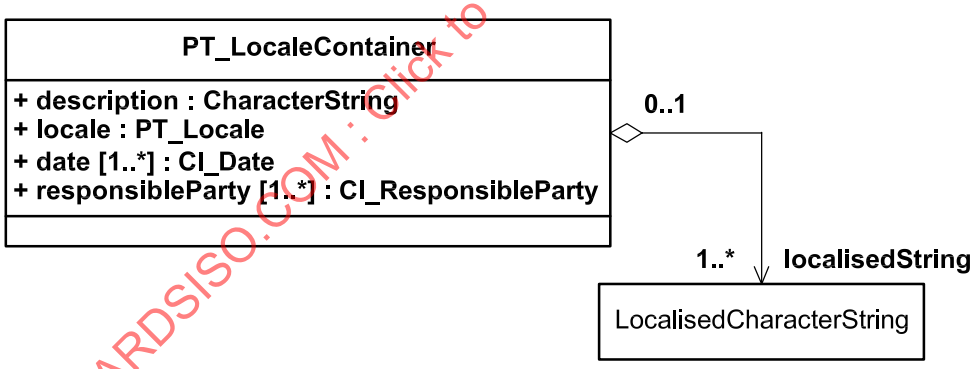

Figure 5 — Translation container

This translation container concept is particularly useful in terms of the XML implementation of ISO 19115, but it is applicable to any other implementation. Indeed, an XML file can only support data expressed in a single character set, which is generally declared in the XML file header. Having all the localized strings stored in a single XML file would limit the use of a single character set such as UTF-8. In order to avoid this,

— the LocalisedCharacterString class is implemented specifically to allow a by-reference containment of the PT_FreeText.textGroup property, and

— the PT_LocaleContainer is the recommended root element to be instantiated in a dedicated XML file.

The localized string related to a given locale can be stored in a corresponding locale container (i.e. XML file) and referenced from the PT_FreeText.textGroup property instances.

7.4 Extensions for metadata-based transfers of geospatial information

7.4.1 Transfer of datasets and aggregate datasets

Figure 6 reproduces Figure 3 of ISO 19115:2003, focusing on the classes and associations needed for metadata-driven transfers of geospatial information. Datasets (DS_Dataset) may be part of aggregates (DS_Aggregate) which may be subset or superset aggregates. Both aggregates and datasets are linked to one or more sets of metadata elements (MD_Metadata).

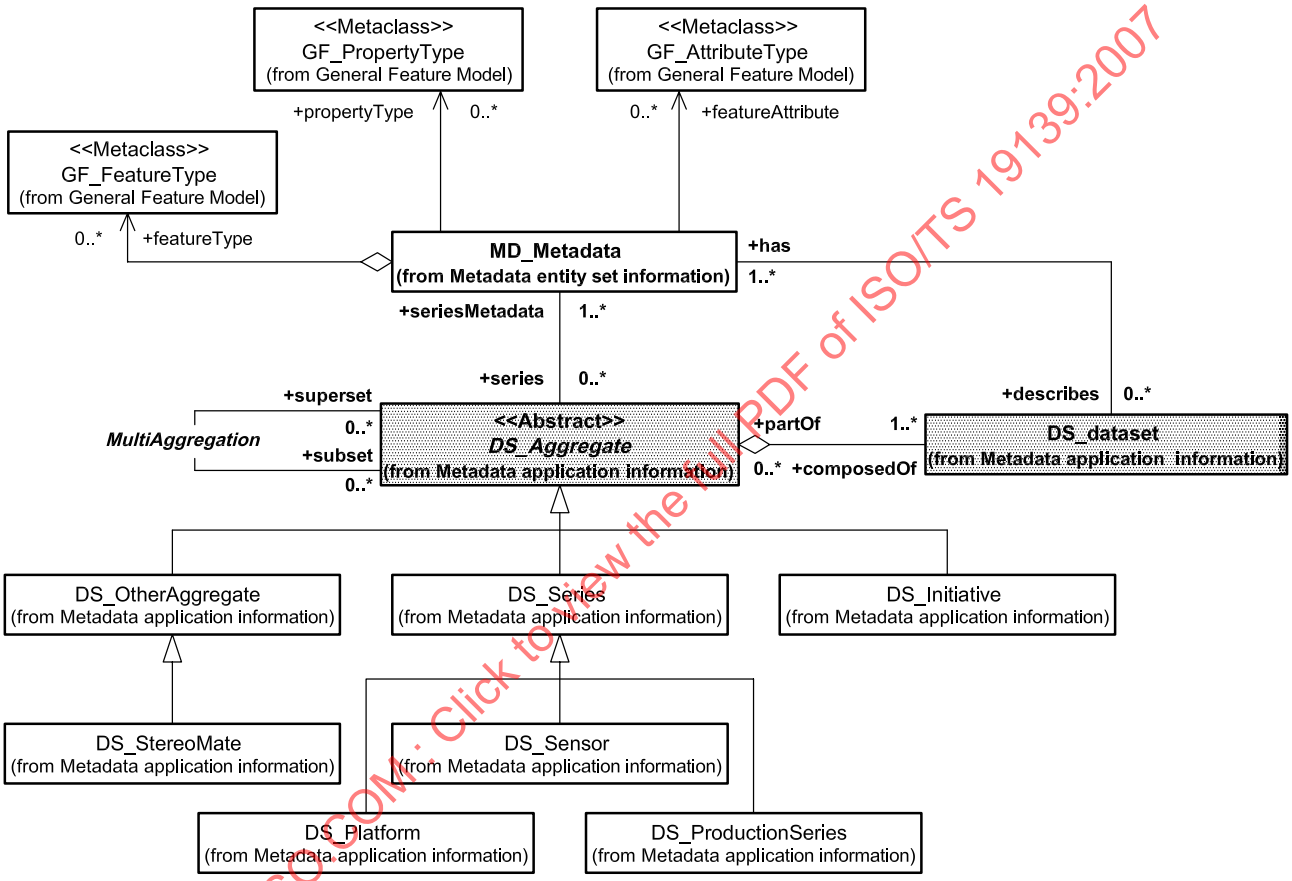

Figure 6 — Datasets, aggregates and their metadata

Metadata-based transfers of geospatial information require the extension of ISO 19115 that is shown in Figure 7.

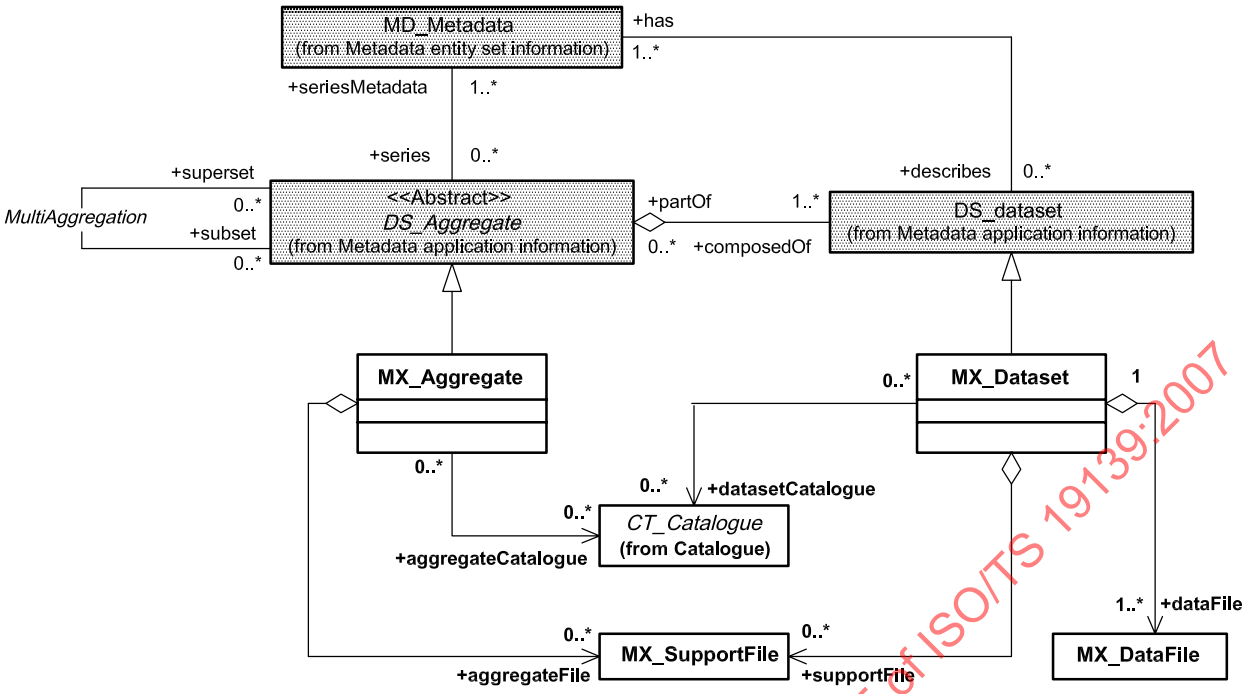

Figure 7 — The generic interchange organization

In order to supply a generic interchange organization for transfer, two new concepts are introduced: **transfer dataset** (MX_Dataset) and **transfer aggregate** (MX_Aggregate).

In the context of an interchange by transfer, the dataset data is organized in data files (MX_DataFile). Both transfer datasets and aggregates may be accompanied by support files (MX_SupportFile) which may contain resources needed to exploit them, or complementary information. The data files and the support files are described in 7.4.3.

In practice, the information needed to exploit a dataset or an aggregate is not limited to their metadata.

Particularly:

— the metadata cites the feature and portrayal catalogues but does not embed them;

— the metadata instances reference information such as codelists, units of measure and coordinate reference systems that all need to be accessed.

All of those resources may be managed externally in on-line registries, but it is usually necessary, in the context of interchange by transfer, to be able to provide that information within the transfer datasets and transfer aggregates. The abstract concept of catalogue (CT_Catalogue detailed in 7.4.4) corresponds exactly to those resources needed to exploit the datasets, aggregates and their metadata. Herein, catalogues are used in the context of interchange by transfer and are associated to transfer datasets (MX_Dataset) and transfer aggregates (MX_Aggregate).

7.4.2 Management of aggregates

The subclassification of DS_Aggregate shown in Figure 6 is fully valid conceptually. Yet, in the context of a data transfer, aggregation of datasets follows constraints (e.g. transfer media capacity) and requirements which outweigh the pure design of geospatial aggregations. The transfer interchange mechanism is based on transfer aggregates, but the initial nature of the geospatial aggregation can be expressed through the hierarchyLevel attribute of MD_Metadata using the extension of the MD_ScopeCode codelist specified in Figure 8.

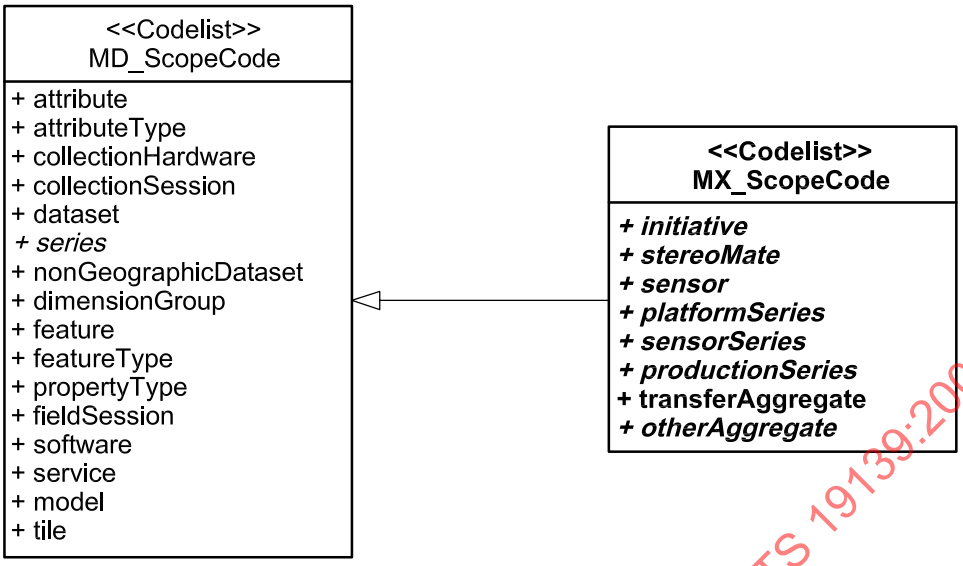

Figure 8 — Extended scope code

Note that the transferAggregate specific value indicates that the aggregate makes sense only in the context of the actual transfer interchange.

7.4.3 Transfer files

Figure 9 describes the support and data files concept. The data files are potentially related to one or many feature types. Each data file has its own format which is generally described within the metadata of the dataset.[3)]

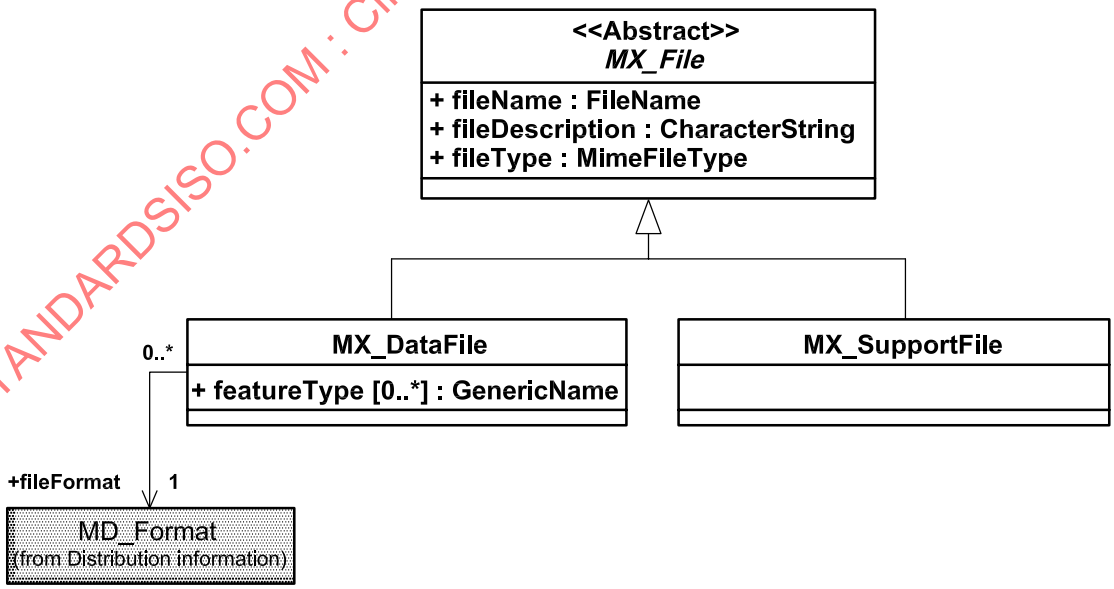

Figure 9 — Transfer files

3) It is also valid to describe the format of data files in the metadata of an aggregate in the case where the format is homogenous over the full aggregate.

7.4.4 Catalogues

7.4.4.1 General concepts

The abstract concept of catalogue (CT_Catalogue) is shown in Figure 10. In the context of this Technical Specification it is a key component of the standard interchange of geospatial information by transfer. Its design is derived from the feature catalogue concept defined in ISO 19110:

— several optional attributes (language, characterSet and locale) are added to support the cultural and linguistic adaptability requirement;

— several attributes of the class FC_FeatureCatalog are considered specific to a feature catalogue.

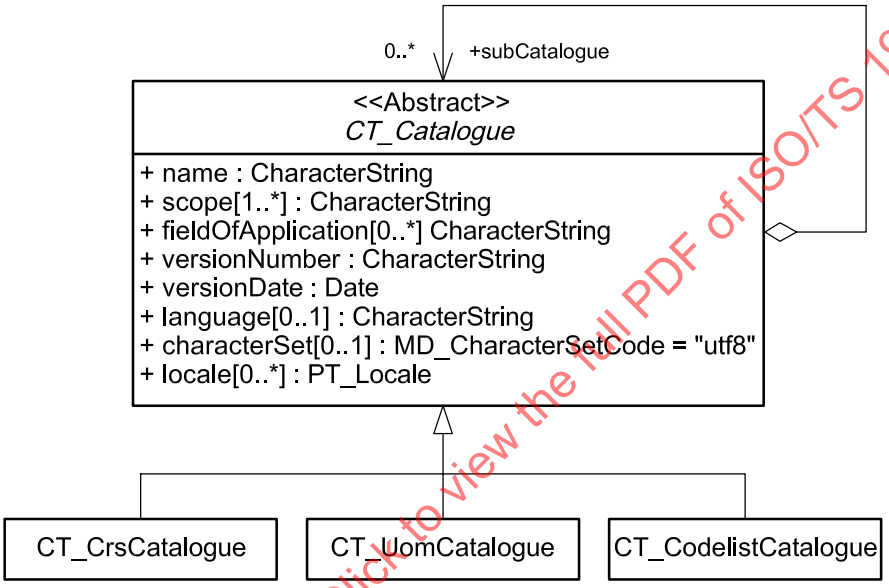

Figure 10 — Catalogues

This Technical Specification provides the conceptual schema of three concrete catalogues as follows.

— A CRS catalogue (CT_CrsCatalogue) contains the description of coordinate reference systems as well as the description of their components (datum, coordinate system, etc.). In the context of interchange by transfer of this specification (detailed in 7.4.1) a CRS catalogue contains the description of items used in the transfer dataset or the transfer aggregate.

— An UoM catalogue (CT_UomCatalogue) contains the description of units of measure. In the context of interchange by transfer of this specification (detailed in 7.4.1) a UoM catalogue contains the description of items used in the transfer dataset or the transfer aggregate.

— A codelist catalogue (CT_CodelistCatalogue) contains the description of codelists, including their name, and the definition as well as the description of their values. In the context of interchange by transfer of this specification (detailed in 7.4.1) a codelist catalogue contains the description of items used in the transfer dataset or the transfer aggregate.

Detailed descriptions of these concrete catalogues are provided respectively in 7.4.4.2 to 7.4.4.4.

The abstract concept of catalogue has also been defined as the basis for harmonization of the different ISO 19100 series catalogue concepts, such as PF_PortrayalCatalogue (ISO 19117) and FC_FeatureCatalogue (ISO 19110). The XML schema implementations of ISO 19110 and ISO 19117 are out

of the scope of this Technical Specification, but it is expected that the XML feature and portrayal catalogues will be transferable as their concepts (PF_PortrayalCatalogue and FC_FeatureCatalogue) are inherited from CT_Catalogue.

The abstract concept of a catalogue can also serve as a basis for establishing a data quality measures catalogue (e.g. as ISO 19138). This would allow a quality measure catalogue to be transferred as part of the transfer datasets and aggregates defined herein. More generally, the extension of the catalogue concept and the use of catalogues is recommended for any set of information which is managed or is manageable in on-line registries, when the catalogues enable the exploitation of the transfer datasets and aggregates.

7.4.4.2 UoM catalogue

A unit of measure catalogue (CT_UomCatalogue) as shown in Figure 11 aggregates a list of unit of measure items. Those items are defined in ISO 19136.

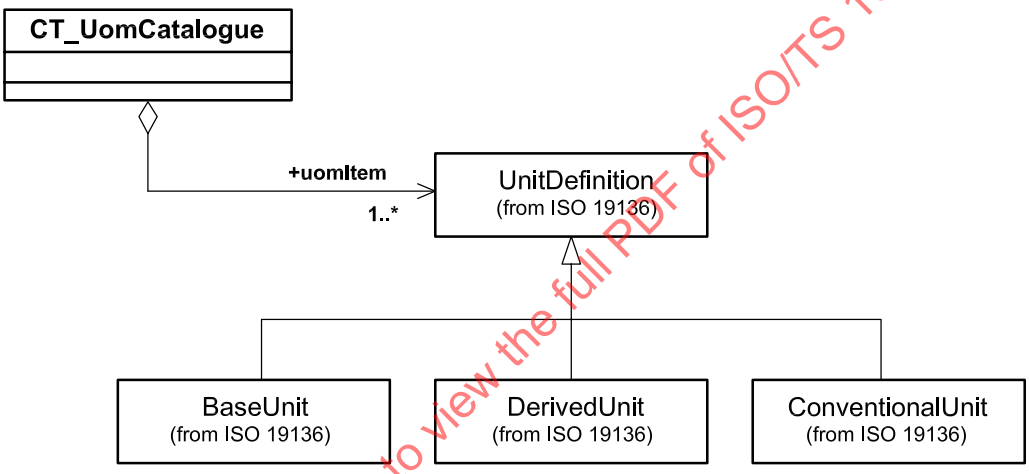

Figure 11 — UoM catalogue

7.4.4.3 CRS catalogue

The concept of CRS catalogue (CT_CrsCatalogue) is shown in Figure 12. A CRS Catalogue has no specific attributes, but it aggregates different types of geodetic items, all inherited from the fundamental classes defined in ISO 19111.

7.4.4.4 Codelist catalogue

The concept of codelist catalogue (CT_CodelistCatalogue) is shown in Figure 13. A codelist catalogue has no specific attributes, but it aggregates a single type of codelist item (CT_Codelist) based on the general concept of catalogue item. A codelist item also aggregates a single type of item (CT_CodelistValue) which corresponds to the values of the codelist.

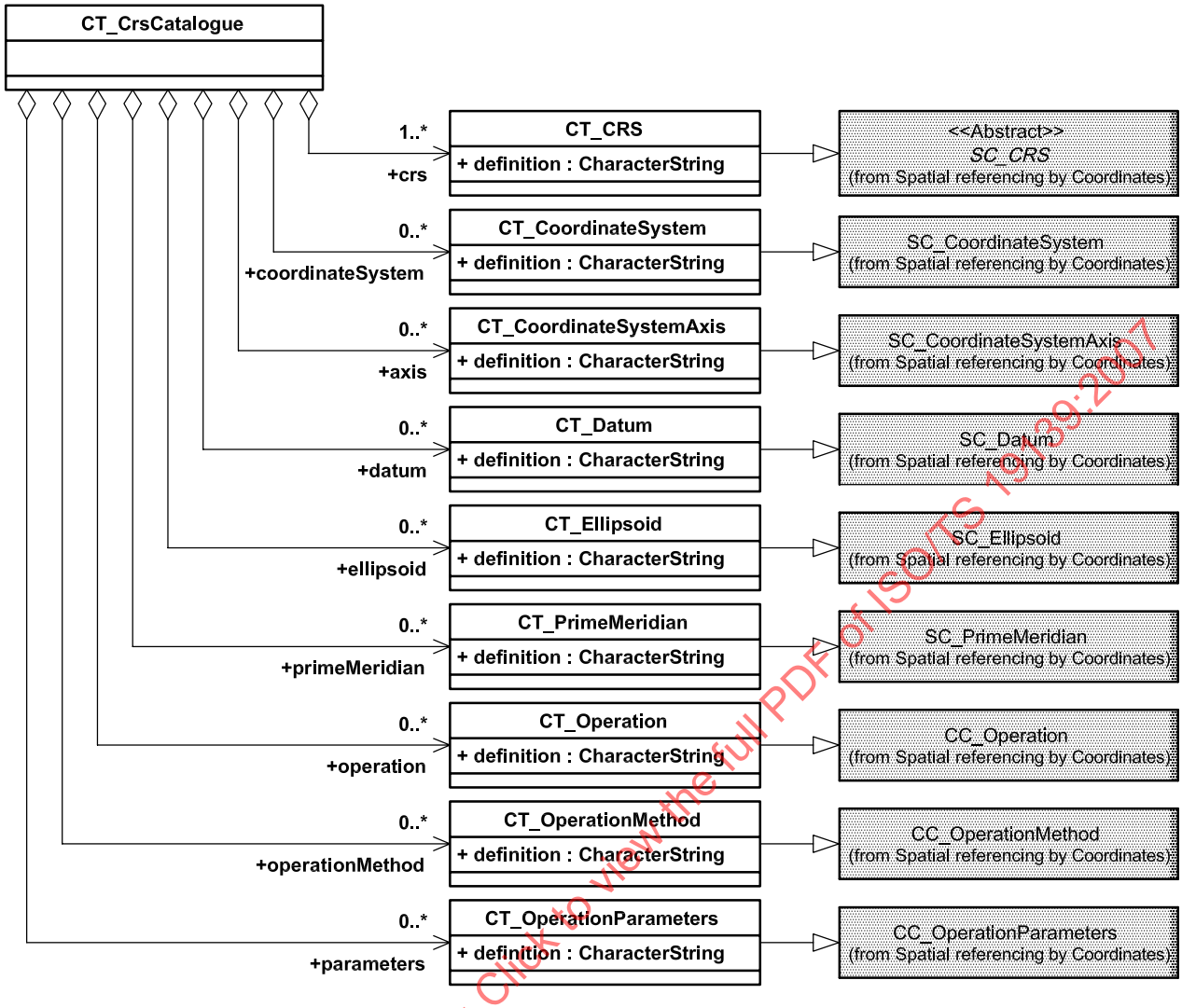

Figure 12 — CRS catalogue

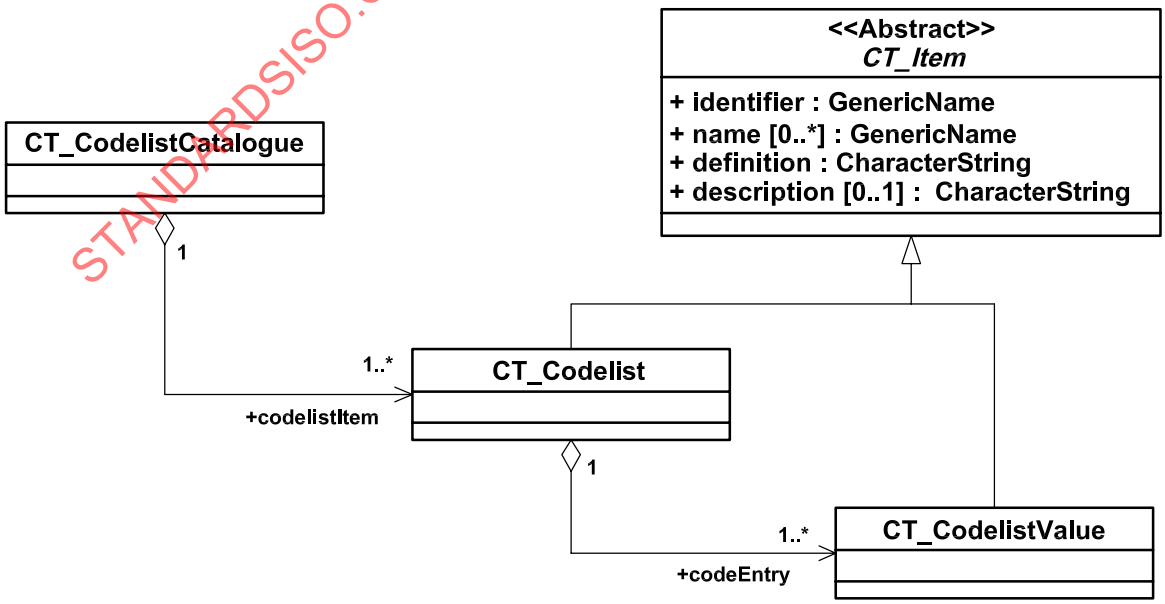

Figure 13 — Codelist catalogue

The concept of catalogue item (CT_Item) is introduced to simplify the specification of the components of a codelist catalogue, but its use is highly recommended for the definition of the components of any new type of catalogue.

8 Encoding rules

8.1 Introduction to encoding rules

General rules for transforming UML to XML schema are described in 8.2 to 8.7 and are in accordance with the rules defined in ISO 19118. In some cases ISO 19118 allows for multiple methods of transforming UML to XML schema, and these serve to clarify which method was selected for this implementation specification. Background on classes and how they are the building blocks for encoding all data exchange (and in this case metadata exchange) is purposefully absent from this Technical Specification since ISO 19118 covers this in great detail. This Technical Specification is based on ISO 19118's encoding rules and a familiarity with ISO 19118 will greatly enhance comprehension of the topics described throughout Clause 8.

A default rule is applied to every element of the specified type in the UML profile unless there are special rules for that element as documented in Clause 9. Clause 8 also shows the details of the XML specific packaging of gmd and its associated namespaces while the generic rules for packaging are described in 8.6 and 8.7.

NOTE The way the encoding rules are described does not prevent adopting the best practices in generating the XML schemas.

8.2 Default XML Class Type encoding

ISO 19118 states that the fundamental modelling concept in UML is the class. As a result, the fundamental encoding rules focus on the encoding of a UML class and build from there. It is important to recall from ISO 19118 that a property represents a name-value pair. It can represent an attribute, association, aggregation or composition (see ISO 19118). A class is made up of one or more properties. For example, in Figure 14, the *Class1* class has three properties: *attr1*, *attr2* and *role1*. For the sake of encoding into XML schema it is important to understand that there is no distinction between properties that are UML attributes, associations, aggregations or compositions.

ISO 19118 also describes the need to have identifiers in XML schema and prescribes using identifiers (ids) and universal unique identifiers (uuids) to accomplish this. There is a special XML schema type in the gco namespace, gco: AbstractObject_Type (described in detail in 9.7), which provides the necessary identifiers. It is mentioned here because it is part of the default XML class type encoding.

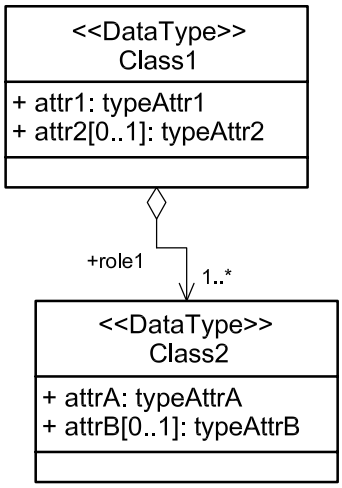

Figure 14 — Sample UML

The following list details the steps for encoding a default class.

1) A UML class is encoded into XML schema as an XML complex type: xs:complexType. This XML complex type is hereafter referred to as the XML Class Type (XCT). In XML schema each XCT has a name attribute whose value is the class name with the suffix _Type.

EXAMPLE 1 Step 1) of building the Class1 class shown in Figure 14 is:

```
<xs:complexType name="Class1_Type">
        (...)
</xs:complexType>
```

2) All UML classes following the default encoding rules will have complex content, and to provide this capability, the xs:complexType element contains an xs:complexContent element.

EXAMPLE 2 Step 2) of building the Class1 class shown in Figure 14 is:

```
<xs:complexType name="Class1_Type">
  <xs:complexContent>
        (...)
  </xs:complexContent>
</xs:complexType>
```

3) All UML classes following the default encoding rules will extend the gco:AbstractObject_Type which is done by adding an *xs:extension* element with the base attribute equal to gco:AbstractObject_Type

EXAMPLE 3 Step 3) of building the Class1 class shown in Figure 14 is:

```
<xs:complexType name="Class1_Type">
  <xs:complexContent>
    <xs:extension base="gco:AbstractObject_Type">
       (...)
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

4) All UML classes following the default encoding rules will have a sequence containing all the properties of the class. This is accomplished by adding an *xs:sequence* element, containing xs:element elements for each property of the class. The attributes of the *xs:element* element will be:

i) the *name* attribute is equal to the name of the property;

ii) the *type* attribute is equal to the name of the XCPT corresponding to the UML class specified as the type of the particular property. Subclause 8.4 explains that by default this is the class name plus "_PropertyType" and 8.5.7 and 8.5.8.2 define the only exceptions to this XCPT naming convention. The name of the XCPT used as the value for this type of attribute will also be properly prefixed with the appropriate namespace as described in 8.6;

iii) the *minOccurs* and *maxOccurs* attributes have the values described in Table A.5 of ISO 19118:2005. Additionally, if a property of the class happens to be an attribute that uses the 'set' or 'sequence' structure then the *minOccurs* attribute is "0" for optional attributes and "1" for mandatory attributes and the *maxOccurs* is "unbounded".

EXAMPLE 4 Step 4) of building the Class1 class shown in Figure 14 is:

```
<xs:complexType name="Class1_Type">
  <xs:complexContent>
    <xs:extension base="gco:AbstractObject_Type">
      <xs:sequence>
        <xs:element name="attr1" type="ns1:typeAttr1_PropertyType"/>
        <xs:element name="attr2" type="ns1:typeAttr2_PropertyType"
minOccurs="0" />
         <xs:element name="role1" type="ns1:Class2_PropertyType"
minOccurs="1"     maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

NOTE Throughout the examples in this Technical Specification there is a namespace prefix, "ns1" that is not always shown in the UML. This is a fictitious namespace used to illustrate when a namespace prefix should be present but it does not correspond to any of the namespaces or corresponding prefixes defined in this Technical Specification.

The order of the xs:elements within the xs:sequence is based on the order of a property's corresponding entry in the data dictionary (Annex B) of ISO 19115:2003. Properties are listed in the sequence starting from those with the lowest row number (shown in the first column of all tables in ISO 19115, Annex B) followed by those with a higher row number. If a property of a class is absent from ISO 19115:2003, Annex B, then it occurs following the properties present in ISO 19115:2003, Annex B and based on the order of their presence in the data dictionary of this Technical Specification (Annex B). This ordering rule applies to default encodings as well as all special case encodings for XCTs.

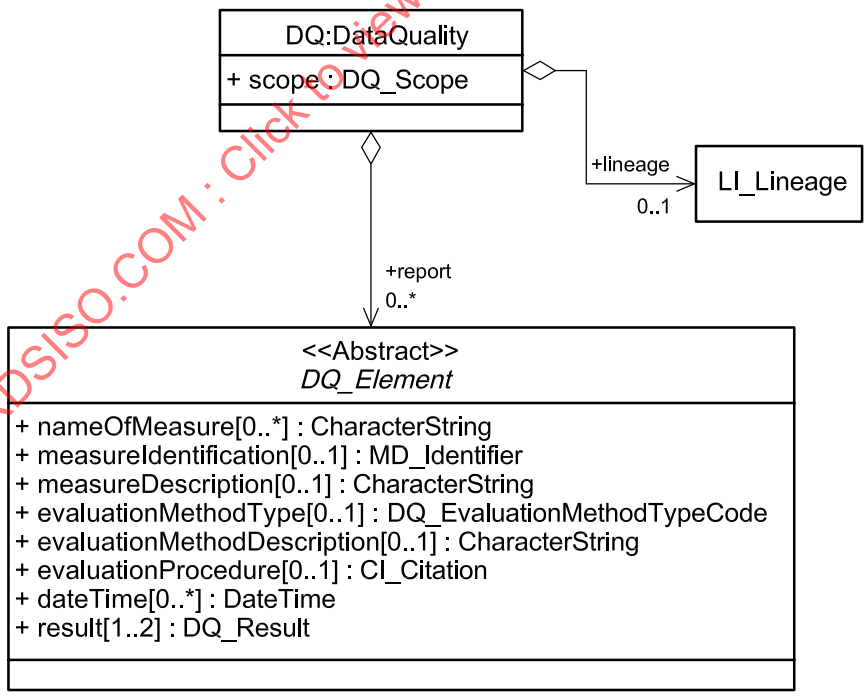

Figure 15 — Partial data quality information UML from ISO 19115:2003, A.2.4.1

The XCT corresponding to Figure 15:

```
<xs:complexType name="DQ_DataQuality_Type">
  <xs:complexContent>
    <xs:extension base="gco:AbstractObject_Type">
      <xs:sequence>
        <xs:element name="scope" type="gmd:DQ_Scope_PropertyType"/>
        <xs:element name="lineage" type="gmd:LI_Lineage_PropertyType"
minOccurs="0"/>
        <xs:element name="report" type="gmd:_DQ_Element_PropertyType"
minOccurs="0"     maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType
```

8.3 XML Class Global Element encoding

Subsequently, a UML class is also encoded as a global element with a name attribute equal to the name of the UML class and the *type* equal to the name of the XCT described in 8.2 (with the appropriate namespace prefix included). Hereafter, this global element is referred to as the XML Class Global Element (XCGE).

EXAMPLE 1 The XCGE of the Class1 class shown in Figure 15 is:

```
<!-- ..................................................... -->
<xs:element name="Class1" type="ns1:Class1_Type"/>
 <!-- ..................................................... -->
```

EXAMPLE 2 The XCGE corresponding to DQ_DataQuality shown in Figure 15 is:

```
<!-- ..................................................... -->
<xs:element name="DQ_DataQuality" type="gmd:DQ_DataQuality_Type"/>
 <!-- ..................................................... -->
```

8.4 XML Class Property Type encoding

8.4.1 Property type encoding for complexType XCTs

To support the property concept described in 8.2 and the potential for a given class to be a property type for a set of container classes, each UML class is also defined as another XSD xs:complexType which is hereafter referred to as the XML Class Property Type (XCPT). Recognise that the containment of a property is managed through the XCPT of its data type. By default both “by Value” and “by Ref” containment are allowed by the XCPT. The following list details the steps for encoding an XCPT for a default class.

1) To distinguish the XCPTs from the XCTs, the xs:complexType element has a name attribute equal to the class name suffixed with _PropertyType.

EXAMPLE 1 Step 1) of building the Class1 class XCPT shown in Figure 14 is:

```
<xs:complexType name="Class1_PropertyType">
    (...)
</xs:complextype>
```

2) To provide “by Value” containment in a manner that supports the polymorphism requirements described in 6.7 the XCPT has an optional xs:sequence element that contains one xs:element element with a ref attribute equal to the XCT of the UML. The xs:sequence element is made optional in order to allow for the possibility that the property will be implemented “by Ref”.

EXAMPLE 2 Step 2) of building the Class1 class XCPT shown in Figure 14 is:

```
<xs:complexType name="Class1_PropertyType">
  <xs:sequence minOccurs="0">
    <xs:element ref="ns1:Class1"/>
  </xs:sequence>
</xs:complexType>
```

3) To provide "by Ref" containment the XCPT has an xs:attributeGroup element with a ref attribute equal to "gco:ObjectReference". The IM_ObjectReference attribute Group is introduced in ISO 19118 and further detailed in 9.7.

EXAMPLE 3 Step 3) of building the Class1 class XCPT shown in Figure 14 is:

```
<xs:complexType name="Class1_PropertyType">
  <xs:sequence minOccurs="0">
    <xs:element ref="ns1:Class1"/>
  </xs:sequence>
  <xs:attributeGroup ref="gco:ObjectReference"/>
</xs:complexType>
```

4) To provide null value explanations when necessary, the XCPT has an xs:attribute element with a ref attribute equal to "gco:nilReason". The nilReason attribute is described in detail in 9.7.3.4.

EXAMPLE 4 Step 4) of building the Class1 class XCPT shown in Figure 14 is:

```
<xs:complexType name="Class1_PropertyType">
  <xs:sequence minOccurs="0">
    <xs:element ref="ns1:Class1"/>
  </xs:sequence>
  <xs:attributeGroup ref="gco:ObjectReference"/>
  <xs:attribute ref="gco:nilReason">
</xs:complexType>
```

EXAMPLE 5 Sample XCPT corresponding to DQ_DataQuality shown in Figure 15.

```
<xs:complexType name="DQ_DataQuality_PropertyType">
  <xs:sequence minOccurs="0">
    <xs:element ref="gmd:DQ_DataQuality"/>
  </xs:sequence>
  <xs:attributeGroup ref="gco:ObjectReference"/>
  <xs:attribute ref="gco:nilReason"/>
</xs:complexType>
```

Notice that there is no specific constraint introduced in the XCPT to force the containment information of a property defined in UML. The UML containment instructions shall be managed by external testing applied to the XML metadata. This issue is discussed in more detail in Annex A.

8.4.2 Property type encoding for simpleType XCTs

There are some cases of specific XCT encodings described in 8.5.4 and throughout Clause 9 where the XCT is an *xs:simpleType* instead of an *xs:complexType* as described in 8.2. When an XCT is *simple* the corresponding XCPT does not include the "by Ref" capabilities described in 8.4.1. Preventing the use of referencing on simple types avoids a potential complexity that would exist if all property types could be implemented by reference.

In 8.5.4 the first simple XCT is introduced for the encoding of classes stereotyped *Enumeration*. The XCPT corresponding to simpleType XCTs, such as for an enumeration, does not include the gco:ObjectReference attributeGroup.

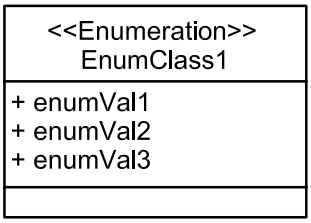

Figure 16 — Sample UML for class stereotyped enumeration (a simpleType)

EXAMPLE The XCPT for the EnumClass1 class shown in Figure 16 is:

```
<xs:complexType name="EnumClass1_PropertyType">
    <xs:sequence minOccurs="0">
        <xs:element ref="ns1:EnumClass1"/>
    </xs:sequence>
    <xs:attribute ref="gco:nilReason" />
</xs:complexType>
```

Notice that even in the case of simpleTypes there is no specific constraint introduced in the XCPT to force the use of a byVal or a nilReason of a property. This containment has to be managed by external testing applied to the XML metadata. This issue is discussed in more detail in Annex A.

8.5 Special case encodings

8.5.1 Introduction to special case encodings

While the encoding mechanisms described in 8.2 to 8.4 cover the majority of classes found in the UML of the ISO 19100 series of International Standards, there are some special cases where different encoding rules are applied. The exceptions are generally based on special stereotypes applied to classes. A class without a stereotype or a class with the stereotype *Type* or *DataType* will follow the default encodings. Other classes will follow their corresponding encoding rules in 8.5.2 to 8.5.6.

8.5.2 Abstract classes

For implementation purposes it is desirable to better identify abstract class types so that implementers realize that there cannot be elements of such types. Abstract classes in the UML of the ISO 19100 series of International Standards are encoded by:

a) prefixing the word 'Abstract' to the abstract class name of the corresponding XCT;

b) adding the *abstract* attribute with a value of "true" to the *xs:complexType* element of the XCT;

c) prefixing the word 'Abstract' to the abstract class name of the XCGE;

d) adding the *abstract* attribute with a value of "true" to the *xs:element* element of the XCGE;

e) using the appropriate prefixed class name in the *ref* attribute of the *xs:element* in the corresponding XCPT.

NOTE The word 'Abstract' is not prefixed to the name attribute of the XCPT because a property type will not be abstract in XML schema).

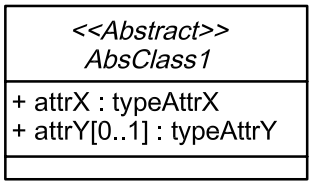

Figure 17 — Sample UML for an abstract class

EXAMPLE 1 The XCT corresponding to AbsClass1 as shown in Figure 17:

```
<xs:complexType name="AbstractAbsClass1_Type" abstract="true">
  <xs:complexContent>
    <xs:extension base="gco:AbstractObject_Type">
      <xs:sequence>
        <xs:element name="attrX" type="ns1:typeAttrX_PropertyType"/>
       <xs:element name="attrY" type="ns1:typeAttrY_PropertyType"
minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

EXAMPLE 2 The XCGE corresponding to AbsClass1 as shown in Figure 17:

```
<xs:element name="AbstractAbsClass1"
type="ns1:AbstractAbsClass1_Type" abstract="true"/>
```

EXAMPLE 3 The XCPT corresponding to AbsClass1 as shown in Figure 17:

```
<xs:complexType name="AbsClass1_PropertyType">
  <xs:sequence minOccurs="0">
    <xs:element ref="ns1:AbstractAbsClass1"/>
  </xs:sequence>
  <xs:attributeGroup ref="gco:ObjectReference"/>
  <xs:attribute ref="gco:nilReason"/>
 </xs:complexType>
```

EXAMPLE 4 Sample XCPT using actual ISO 19115 example.

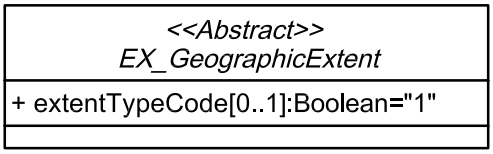

Figure 18 — The UML of the abstract class EX_GeographicExtent from ISO 19115

The XCT corresponding to EX_GeographicExtent class shown in Figure 18 is:

```
<xs:complexType name="Abstract EX_GeographicExtent_Type"
abstract="true">
  <xs:complexContent>
    <xs:extension base="gco:AbstractObject_Type">
      <xs:sequence>
        <xs:element name="extentTypeCode"
type="gco:Boolean_PropertyType" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The XCGE corresponding to EX_GeographicExtent class shown in Figure 18 is:

```
<xs:element name="AbstractEX_GeographicExtent"
type="gmd:AbstractEX_GeographicExtent_Type" abstract="true"/>
```

The XCPT corresponding to EX_GeographicExtent class shown in Figure 18 is:

```
<xs:complexType name="EX_GeographicExtent_PropertyType">
  <xs:sequence minOccurs="0">
    <xs:element ref="gmd: AbstractEX_GeographicExtent"/>
  </xs:sequence>
  <xs:attributeGroup ref="gco:ObjectReference"/>
  <xs:attribute ref="gco:nilReason"/>
 </xs:complexType>
```

8.5.3 Inheritance and subclass encodings

The concept of inheritance is implemented through generalizations that are considered simple extensions. Admittedly this is a simplification of the concept of inheritance and subclassing since it neglects cases of generalization relationships that provide restriction or cases of generalization relationships that address multiple-inheritance. This extension-only characteristic of generalizations is by design, and the implementation of restrictive generalizations is discussed in A.4. As far as XML schema is concerned, multiple-inheritance is not supported and can only be simulated. Since ISO 19115 has no cases of multiple-inheritance that need to be implemented, mechanisms for the simulation of multiple-inheritance are omitted from these encoding rules.

As with the default UML classes, those that are subclasses have corresponding XCT, XCGE and XCPT in gmd. The XCT of a subclass differs from a default class in the following ways.

a) Within the xs:complexContent element of the XCT there is an *xs:extension* element whose *base* attribute is equal to the namespace qualified XCT of the base (or super) class. (This differs from the XCT of a default UML class *xs:extension* element whose *base* attribute is a gco:AbstractObject_Type to provide id and uuid attributes. Those necessary attributes still exist through the extension of the base class). It is important to note that if the super class is an abstract class, then its corresponding XCT name will be prefixed with the word “Abstract” as described in 8.5.2.

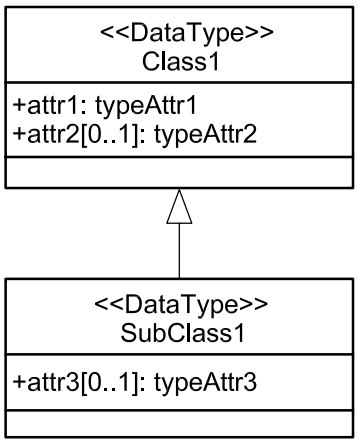

Figure 19 — Sample UML for subclass

EXAMPLE 1 Step a) of building the SubClass1 class shown in Figure 19 is:

```
<xs:complexType name="SubClass1_Type">
    <xs:complexContent>
        <xs:extension base="ns1:Class1_Type">
              (...)
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

b) The *xs:extension* element contains an *xs:sequence* that contains all the properties of the class as described for a default UML class in Step 4) of 8.2. Note that this sequence only contains the properties that are specific to the subclass.

EXAMPLE 2 Step a) of building the SubClass1 class shown in Figure 19 is:

```
<xs:complexType name="SubClass1_Type">
  <xs:complexContent>
    <xs:extension base="ns1:Class1_Type">
      <xs:sequence>
        <xs:element name="attr3" type="ns1:typeAttr3_PropertyType"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The only difference between the XCGE of the subclass and the XCGE of a default UML class is that it includes the additional *substitutionGroup* attribute which is equal to the namespace qualified XCGE of the base class.

EXAMPLE 3 The encoding of the XCGE for the SubClass1 class shown in Figure 19 is:

```
<xs:element name="SubClass1" type="ns1:SubClass1_Type"
substitutionGroup="ns1:Class1"/>
```

The XCPT of a subclass is encoded identically to the XCPT of a default UML class.

EXAMPLE 4 The encoding of the XCPT for the SubClass1 class shown in Figure 19 is:

```
<xs:complexType name="SubClass1_PropertyType">
    <xs:sequence minOccurs="0">
        <xs:element ref="ns1:SubClass1"/>
    </xs:sequence>
    <xs:attributeGroup ref="gco:ObjectReference"/>
    <xs:attribute ref="gco:nilReason"/>
</xs:complexType>
```

EXAMPLE 5 Sample SubClass in XML schema using actual ISO 19115 example.

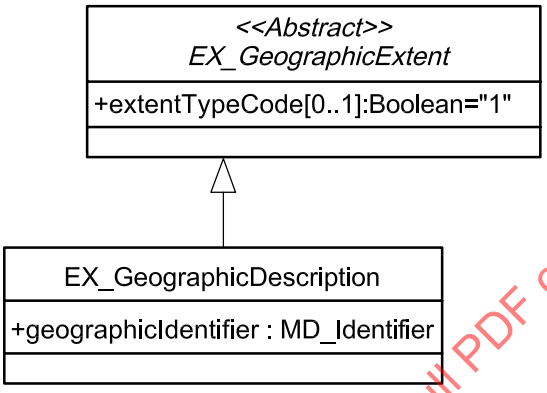

Figure 20 — UML of the classes EX_GeographicExtent and EX_GeographicDescription from ISO 19115

The XCT corresponding to EX_GeographicDescription class shown in Figure 20 is:

```
xs:complexType name="EX_GeographicDescription_Type">
    <xs:complexContent>
        <xs:extension base="gmd:AbstractEX_GeographicExtent_Type">
            <xs:sequence>
                <xs:element name="geographicIdentifier"
type="gmd:MD_Identifier_PropertyType"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
 </xs:complexType
```

The XCGE corresponding to EX_GeographicDescription class shown in Figure 20 is:

```
<xs:element name="EX_GeographicDescription"
type="gmd:EX_GeographicDescription_Type"
substitutionGroup="gmd:AbstractEX_GeographicExtent"/>
```

The XCPT corresponding to EX_GeographicDescription class shown in Figure 20 is:

```
<xs:complexType name="EX_GeographicDescription_PropertyType">
    <xs:sequence minOccurs="0">
        <xs:element ref="gmd:EX_GeographicDescription"/>
    </xs:sequence>
    <xs:attributeGroup ref="gco:ObjectReference"/>
    <xs:attribute ref="gco:nilReason"/>
</xs:complexType
```

8.5.4 Enumeration encodings

The conceptual schema language of the ISO 19100 series of International Standards supports two enumerated types whose "declaration defines a list of valid mnemonic identifiers (see ISO/TS 19103)". The two enumerated types are *Enumeration* and *CodeList*, and it is important to recognise the difference between these two types. A class that is stereotyped *Enumeration* "can contain only simple attributes which represent the enumeration values (see ISO/TS 19103)". An <<Enumeration>> class is used only in the case where no extension to the list of values is necessary, otherwise a <<CodeList>> class is used.

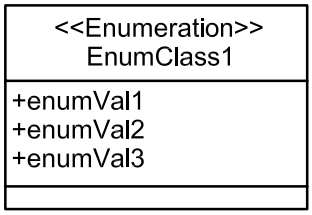

Figure 21 — Sample UML for class stereotyped Enumeration

a) The encoding of an <<Enumeration>> class begins with the creation of an XCT starting with an *xs:simpleType* element with a *name* attribute equal to the class name in the UML with the addition of the _Type suffix.

EXAMPLE 1 Step a) of building the EnumClass1 class shown in Figure 21 is:

```
<xs:simpleType name="EnumClass1_Type">
        (...)
</xs:simpleType>
```

b) The xs:simpleType element contains an xs:restriction element with a *base* attribute equal to an xs:string.

EXAMPLE 2 Step b) of building the EnumClass1 class shown in Figure 21 is

```
<xs:simpleType name="EnumClass1_Type">
  <xs:restriction base="xs:string">
     (...)
  </xs:restriction>
</xs:simpleType>
```

c) Each *xs:restriction* element contains a series of *xs:enumeration* elements whose value attributes equal the attribute values shown in the UML diagram for the enumeration.

EXAMPLE 3 Step c) of building the EnumClass1 class shown in Figure 21 is:

```
<xs:simpleType name="EnumClass1_Type">
    <xs:restriction base="xs:string">
        <xs:enumeration value="enumVal1"/>
        <xs:enumeration value="enumVal2"/>
        <xs:enumeration value="enumVal3"/>
    </xs:restriction>
</xs:simpleType>
```

The XCGE of an <<Enumeration>> class follows the default XCGE rules with the addition of a *substitutionGroup* attribute equal to the "gco:CharacterString" XCGE.

EXAMPLE 4 The XCGE of the EnumClass1 class shown in Figure 21 is:

```
<xs:element name="EnumClass1" type="ns1:EnumClass1_Type"
substitutionGroup="gco:CharacterString"/>
```

The XCPT of an <<Enumeration>> class follows the default encoding described for simple type XCTs in 8.4.2.

EXAMPLE 5 The XCPT of the EnumClass1 class shown in Figure 21 is:

```
<xs:complexType name="EnumClass1_PropertyType">
    <xs:sequence minOccurs="0">
        <xs:element ref="ns1:EnumClass1"/>
    </xs:sequence>
    <xs:attribute ref="gco:nilReason"/>
</xs:complexType>
```

EXAMPLE 6 Sample enumeration class using actual ISO 19115 example.

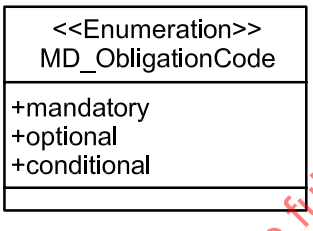

Figure 22 — MD_ObligationCode class from ISO 19115

The XCT corresponding to MD_ObligationCode class shown in Figure 22 is:

```
<xs:simpleType name="MD_ObligationCode_Type">
    <xs:restriction base="xs:string">
        <xs:enumeration value="mandatory"/>
        <xs:enumeration value="optional"/>
        <xs:enumeration value="conditional"/>
    </xs:restriction>
</xs:simpleType>
```

The XCGE corresponding to MD_ObligationCode class shown in Figure 22 is:

```
<xs:element name="MD_ObligationCode"
type="gmd:MD_ObligationCode_Type"
substitutionGroup="gco:CharacterString"/>
```

The XCPT corresponding to MD_ObligationCode class shown in Figure 22 is:

```
<xs:complexType name="MD_ObligationCode_PropertyType">
    <xs:sequence minOccurs="0">
        <xs:element ref="gmd:MD_ObligationCode"/>
    </xs:sequence>
<xs:attribute ref="gco:nilReason"/>
</xs:complexType>
```

8.5.5 CodeList encodings

8.5.5.1 Building blocks of the CodeList encoding

A class that is stereotyped CodeList is an enumerated type like a class that is stereotyped *Enumeration*. The difference is that the <<CodeList>> class is extensible. All <<CodeList>> classes defined in ISO 19115:2003, Annex B, are described in tables with three columns: Name, DomainCode and Definition.

Table 1 — CI_DateTypeCode <<CodeList>> table from ISO 19115

	Name	Domain code	Definition
1.	CI_DateTypeCode	DateTypCd	identification of when a given event occurred
2.	creation	0001	date identifies when the resource was brought into existence
3.	publication	0002	date identifies when the resource was issued
4.	revision	0003	date identifies when the resource was examined or re-examined and improved or amended

Each row in a CodeList table is intended to describe a unique concept or option that supports the intent of the CodeList's definition. The specific encoding of the CodeList stereotype is meant to support some characteristics intended by the creators of ISO 19115. These characteristics are given below.

a) CodeLists and their associated definitions are controlled in registers. ISO 19115:2003, Annex B is the base concept register of the ISO 19115 CodeLists and is the base source for the establishment of user communities' registers.

b) The Name column of the CodeList tables in ISO 19115:2003, Annex B, contains a value that all software will use to recognise the unique concept or option defined in a row of a CodeList table. As a result, this value is not a proper name in any language, even English.

c) User communities need to express the unique concepts and options of a CodeList using one or many natural languages, dialects or idioms identified here as code spaces. In each code space, a name and possibly a definition can be provided; this name and definition are expressed in a given locale (language, country and character set) when the code space corresponds to a natural language. (This would be like adding additional columns to a CodeList defined in ISO 19115:2003, Annex B.)

d) User communities may need to add new concepts or options that were not considered by the creators of ISO 19115 and it is necessary to control these extensions in user community registers. (This would be like adding additional rows to a CodeList defined in ISO 19115:2003, Annex B.)

To satisfy the intended characteristics of a CodeList, a special "CodeListValue_Type" XCT is created with three attributes:

```
<xs:complexType name="CodeListValue_Type">
    <xs:simpleContent>
        <xs:extension base="xs:string">
            <xs:attribute name="codeList" type="xs:anyURI"  use="required"/>
            <xs:attribute name="codeListValue" type="xs:anyURI"
use="required"/>
            <xs:attribute name="codeSpace" type="xs:anyURI"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
```

The definition of the special XCT makes any CodeList substitutable by a CharacterString. An XCGE based on this special XCT and an XCPT is defined for each CodeList as described in 8.5.5.2. Consequently, any CodeList instance is an XML element. Its name is the name of its XCGE. It handles a value and the three XML attributes defined by the special XCT

The *codeList* attribute contains a URL that references a codeList definition within a registry or a codeList catalogue.

EXAMPLE 1 If Table 1 was contained in a catalogue of codeLists on the ISO/TC 211 web site then a possible value for the *codeList* attribute might be:

codeList = "http://www.tc211.org/ISO19139/resources/codeList.xml#CI_DateTypeCode"

The *codeListValue* attribute carries the identifier of the codeList value definition. This identifier is the value expressed in the Name column of the tables in Annex B of ISO 19115:2003. The codeList catalogue (or registry) is expected to contain an explicit name and definition of the value in the default language of the metadata, as well as alternative expressions in different code spaces, some of them corresponding to the different locales supported by the metadata.

EXAMPLE 2 If it was intended for an instance document to indicate a creation date for a piece of metadata then the value for the codeListValue based on Table 1 would be:

codeListValue="creation"

Each of these alternative expressions is associated to a given codeSpace. The *codeSpace* attribute is an optional identifier (URI); when present it refers to the alternative expression of the codelist value definition effectively expressed as the value of the element. The codeSpace URI for the domain code is **domainCode**.

EXAMPLE 3 If it was intended for an instance document to indicate a creation date for a piece of metadata but the implemented wanted to express this value using domain codes, then the value for the codeSpace based on Table 1 would be:

codeSpace="domainCode"

The content of the element is either the name of the codelist value in the given code space or the name corresponding to the codelist value in the default language of the metadata when the *codeSpace* attribute is not present.

EXAMPLE 4 Here is a full instance of CI_DateTypeCode expressed in the domaineCode code space:

<CI_DateTypeCode codeList= "http://www.tc211.org/ISO19139/resources/codeList.xml#CI_DateTypeCode" codeListValue="creation" codeSpace="domainCode">0001</CI_DateTypeCode>

EXAMPLE 5 Here is a full instance of CI_DateTypeCode expressed in the default language of the metadata (e.g. English):

*<CI_DateTypeCode codeList="http://www.tc211.org/ISO19139/resources/codeList.xml#CI_DateTypeCode" codeListValue="creation">**Creation**</CI_DateTypeCode>*

The XML element value ensures the user of access to a valid default expression of the actual value of the CodeList, while the three attributes ensure the application the ability access the full definition of the CodeList and its values, typically for customization of the user interface (creation of a drop down box providing the registered or catalogued value of the codelist, multilingual and multicultural management).

8.5.5.2 Details of the CodeList encoding

The CodeListValue_Type was introduced in 8.5.5.1, and that complexType serves as the XCT for all <<CodeList>> classes. The XCGE of a <<CodeList>> class is encoded as a global element with a *name* attribute equal to the name of the UML class and the *type* equal to the CodeListValue_Type. Like an <<Enumeration>> class this XCGE also contains a *substitutionGroup* attribute equal to the "gco:CharacterString" XCGE.

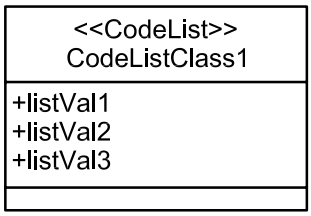

Figure 23 — Sample UML for class stereotyped CodeList

EXAMPLE 1 The XCGE defined for the CodeListClass1 class shown in Figure 23 is:

```
<xs:element name="CodeListClass1" type="gco:CodeListValue_Type"
substitutionGroup="gco:CharacterString"/>
```

The XCPT for a <<CodeList>> class follows the encoding rules defined in 8.4.2 for simple type encodings.

EXAMPLE 2 The XCPT defined for the CodeListClass1 class shown in Figure 23 is:

```
<xs:complexType name="CodeListClass1_PropertyType">
   <xs:sequence minOccurs="0">
        <xs:element ref="gmd:CodeListClass1"/>
    </xs:sequence>
<xs:attribute ref="gco:nilReason"/>
</xs:complexType>
```

EXAMPLE 3 Sample CodeList class encoding using actual ISO 19115 example.

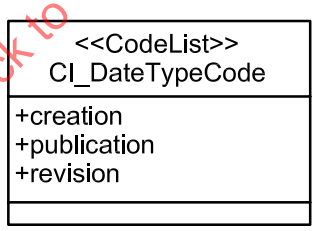

Figure 24 — CI_DateTypeCode class from ISO 19115

The XCGE corresponding to CI_DateTypeCode class shown in Figure 24 is:

```
<xs:element name="CI_DateTypeCode" type="gco:CodeListValue_Type"
substitutionGroup="gco:CharacterString"/>
```

The XCPT corresponding to CI_DateTypeCode class shown in Figure 24 is:

```
<xs:complexType name="CI_DateTypeCode_PropertyType">
   <xs:sequence minOccurs="0">
        <xs:element ref="gmd:CI_DateTypeCode"/>
    </xs:sequence>
<xs:attribute ref="gco:nilReason"/>
</xs:complexType>
```

8.5.6 Union encodings

All classes with the stereotype *Union* are transformed into an XCT via an *xs:complexType* with the *name* attribute equal to the class name in the UML with the addition of the _Type suffix. Within the *xs:complexType* element is an *xs:choice* element that contains one *xs:element* for each attribute of the UML class. The *name* attribute of each xs:element is equal to the name of the attribute in the UML class and the *type* attribute is equal to the type of the attribute prefixed with the proper namespace and suffixed with "_PropertyType".

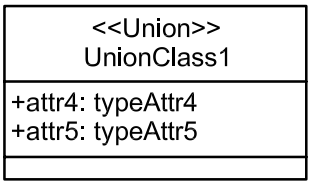

Figure 25 — Sample UML for class stereotyped union

EXAMPLE 1 The XCT defined for the UnionClass1 class shown in Figure 25 is:

```
<xs:complexType name="UnionClass1_Type">
    <xs:choice>
        <xs:element name="attr4" type="ns1:typeAttr4_PropertyType"/>
        <xs:element name="attr5" type="ns1:typeAttr5_PropertyType"/>
    </xs:choice>
</xs:complexType>
```

The XCGE of a <<Union>> class follows the default encoding rules described in 8.3. Since a union is another case where referencing is undesirable, the XCPT follows the default encoding rules for simple types as described in 8.4.2.

EXAMPLE 2 The XCGE defined for the UnionClass1 class shown in Figure 25 is:

```
<xs:element name="UnionClass1" type="ns1:UnionClass1_Type"/>
```

EXAMPLE 3 The XCPT defined for the UnionClass1 class shown in Figure 25 is:

```
xs:complexType name="UnionClass1_PropertyType">
    <xs:sequence minOccurs="0">
        <xs:element ref="gmd:UnionClass1"/>
    </xs:sequence>
    <xs:attribute ref="gco:nilReason"/>
</xs:complexType>
```

EXAMPLE 4 Sample union class encoding using actual ISO 19115 example.

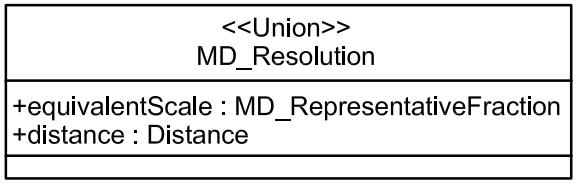

Figure 26 — The MD_Resolution class from ISO 19115

The XCT corresponding to the MD_Resolution class shown in Figure 26 is:

```
<xs:complexType name="MD_Resolution_Type">
    <xs:choice>
        <xs:element name="equivalentScale"
type="gmd:MD_RepresentativeFraction_PropertyType"/>
        <xs:element name="distance"
type="gco:Distance_PropertyType"/>
    </xs:choice>
</xs:complexType
```

The XCGE corresponding to the MD_Resolution class shown in Figure 26 is:

```
<xs:element name="MD_Resolution" type="gmd:MD_Resolution_Type"/>
```

The XCPT corresponding to MD_Resolution:

```
<xs:complexType name="MD_Resolution_PropertyType">
    <xs:sequence minOccurs="0">
      <xs:element ref="gmd:MD_Resolution"/>
    </xs:sequence>
    <xs:attribute ref="gco:nilReason"/>
</xs:complexType>
```

8.5.7 Encoding of MetaClasses

While ISO 19115 does not define any classes that are stereotyped MetaClass, it does contain some properties whose corresponding classes are stereotyped MetaClass. In the case of such a property, the corresponding class can only be instantiated through one of its realizations. The encoding and use of realizations is discussed in greater detail in 8.5.8.

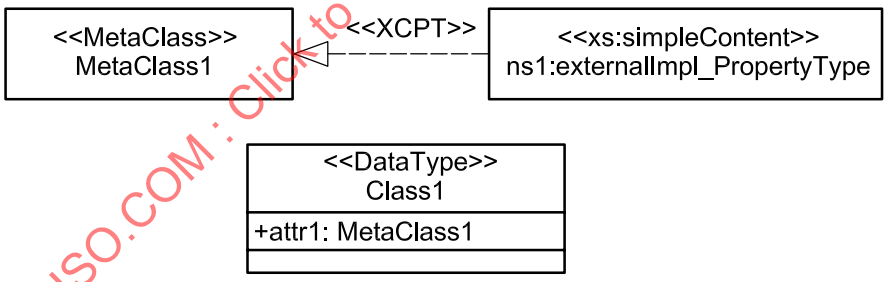

Figure 27 — Sample UML for class stereotyped MetaClass

The realization of a <<MetaClass>> is used when determining the type attribute for an element in the sequence of a default encoded XCT. In other words, in the XCT of Class1 from Figure 27, instead of the type for the "attr1" element being a "MetaClass1_PropertyType" it would be a "ns1:externalImpl_PropertyType" because of the realization shown between MetaClass1 and ns1:externalImpl_PropertyType.

EXAMPLE The XCT corresponding to the Class1 class shown in Figure 27 is:

```
<xs:complexType name="Class1_Type">
    <xs:complexContent>
        <xs:extension base="gco:AbstractObject_Type">
            <xs:sequence>
                <xs:element name="attr1" type="ns1:externalImpl_PropertyType"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

8.5.8 Encoding of externally identified implementations

8.5.8.1 Introduction to externally identified implementations

The use of externally identified implementations was introduced in 6.5 as a way of improving the interoperability and usability of this Technical Specification. It is possible to use external implementations to take advantage of implementations of the ISO 19100 series of International Standards that already exist or to use external implementations that are specific to a particular encoding technology. The UML notation for *realization* is used to indicate, in UML, where the XML schemas defined in other ISO 19100 International Standards are being used.

There are three ways of encapsulating external implementations into concepts depicted in the ISO 19100 series of UML models. The names from the ISO 19100 series UML classes are preserved as an entry point into the implementation schema based on the stereotype of the *realization* as described in 8.5.8.2 and 8.5.8.3.

8.5.8.2 Encoding through XCPT

The simplest case of utilizing external encodings occurs when the existing implementation provides class types, global elements and class property types for the specified ISO 19100 series UML classes that match the default encoding rules described in 8.2 to 8.5. This is indicated in UML by the existence of an XCPT stereotype on a realization relationship as shown in Figure 28.

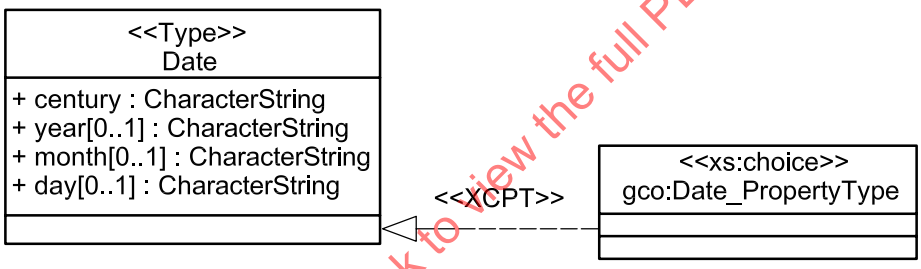

Figure 28 — ISO/TS 19103 date realized by gco:Date_PropertyType

In this situation, no XCTs, XCGEs or XCPTs are created for the target class. When the target class is the type for a property of an XCT, then the name of the source class is used as the value for the *type* attribute. In most cases the value for the *type* attribute will follow the default encoding rule that uses the target class name plus "_PropertyType", but this is not always the situation.

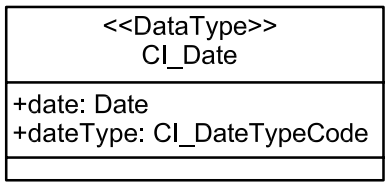

Figure 29 — CI_Date class from ISO 19115

EXAMPLE The XCT corresponding to CI_Date class shown in Figure 29 is:

```
<xs:complexType name="CI_Date_Type">
    <xs:complexContent>
        <xs:extension base="gco:AbstractObject_Type">
            <xs:sequence>
                <xs:element name="date" type="gco:Date_PropertyType"/>
                <xs:element name="dateType"
type="gmd:CI_DateTypeCode_PropertyType"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

8.5.8.3 Encoding through XCGE

The next case of utilizing external encodings occurs when the existing implementation provides class types and global elements that can be utilized but where the inheritance trees of the external implementation should be preserved. In this case an XCPT is created using the class name from the ISO 19100 series UML models. This XCPT serves as an entry point into the implementation schema through the external implementation's XCGE. This is indicated in UML by the existence of an XCGE stereotype on a realization relationship as shown in Figure 30.

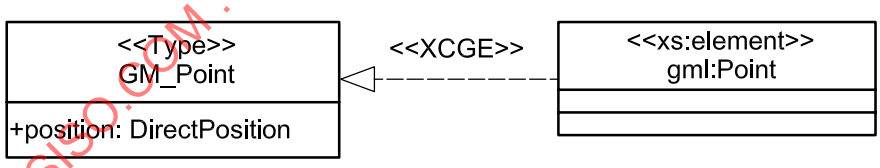

Figure 30 — ISO 19107 GM_Point realized by gml:Point

When the realization has an XCGE stereotype, the XCPT for the target class follows the encoding rules described in 8.4 for the encoding of the XCPT and uses the name of the realization's source class as the value of the *ref* attribute in the *xs:element* tag.

EXAMPLE 1 The XCPT for GM_Point based on Figure 30:

```
<xs:complexType name="GM_Point_PropertyType">
    <xs:sequence minOccurs="0">
        <xs:element ref="gml:Point"/>
    </xs:sequence>
    <xs:attributeGroup ref="gco:ObjectReference"/>
<xs:attribute ref="gco:nilReason"/>
</xs:complexType>
```

In certain cases the UML representing the external implementation may contain properties of the class stereotyped xs:element. If this is the case then there will be an XCGE corresponding to the class shown in the UML that resides in the namespace indicated by the namespace prefix used in the class name. That XCGE will be defined using an xs:element with the *name* attribute equal to the name of the XML class shown in the UML minus the namespace prefix. The *type* attribute will be equal to the only named property shown in the UML. Any other attributes of the xs:element are indicated as properties of the XML class shown in UML as the type of the property and having a value sent to any default value displayed in the UML.

EXAMPLE 2 Sample XCGE based on an xs:element in the gco namespace.

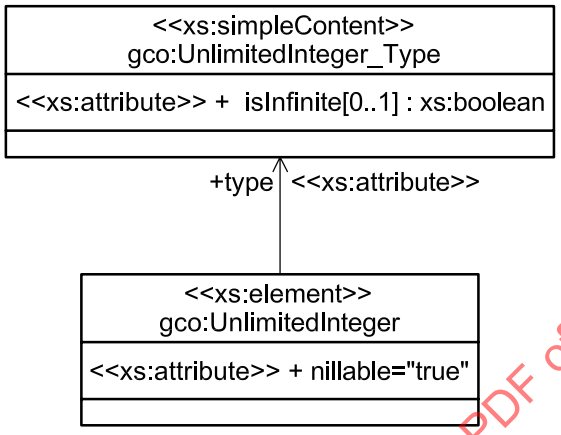

Figure 31 — XCGE for gco:UnlimitedInteger shown as xs:element stereotyped class

The XCGE corresponding to UnlimitedInteger as shown in Figure 31 is:

```
<xs:element name="UnlimitedInteger" type="gco:UnlimitedInteger_Type"
nillable="true"/>
```

8.5.8.4 Encoding through XCT

8.5.8.4.1 General

Often, only an XCT is provided in the external implementation. This is the case when XML schema *simpleTypes* are utilized. When this is the situation, the realization from the ISO 19100 series class to the XML schema object carries an XCT stereotype.

8.5.8.4.2 The xs:simpleType stereotype

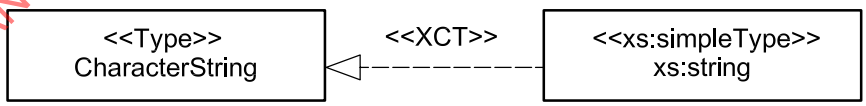

Figure 32 — Realization of CharacterString with xs:string

When the realization has an XCT stereotype and the source class has an *xs:simpleType* stereotype, the XCGE follows the encoding rules described in 8.3 for the encoding of the XCGE and uses the name of the corresponding XCT as the value of the *type* attribute.

EXAMPLE The XCGE for CharacterString based on Figure 32:

```
<xs:element name="CharacterString" type="xs:string"/>
```

8.5.8.4.3 The xs:simpleContent stereotype

When the realization has an XCT stereotype and the source class has an *xs:simpleContent* stereotype, an XCT is created corresponding to the source class.

The XCT follows the following guidelines.

a) There is an xs:complexType created with a *name* attribute equal to the name of the xml class minus any namespace prefix. If the xml class happens to be abstract then the xs:complexType will contain an *abstract* attribute with the value set to “true”.

b) The xs:complexType contains an xs:simpleContent.

c) The xs:simpleContent contains an xs:extension with a *base* attribute set to the name of the external implementation super class.

d) The xs:simpleContent will contain xs:attributes or xs:attributeGroups depending on the properties and their respective stereotypes shown in the UML representing the XML class. If there is a type value present in the UML property then the xs:attribute or xs:attributeGroup will have a *name* attribute equal to the name of the property in the UML and a *type* attribute equal to the type shown in the UML. If there is no type present for a property in the UML then the xs:attribute or xs:attributeGroup will contain a *ref* attribute with a value equal to the name of the property in the UML. Just as with default XCT encodings from 8.2, the minOccurs and maxOccurs values are based on ISO 19118:2005, Table A.5.

EXAMPLE Sample XCT realization on an xs:simpleContent in the gco namespace.

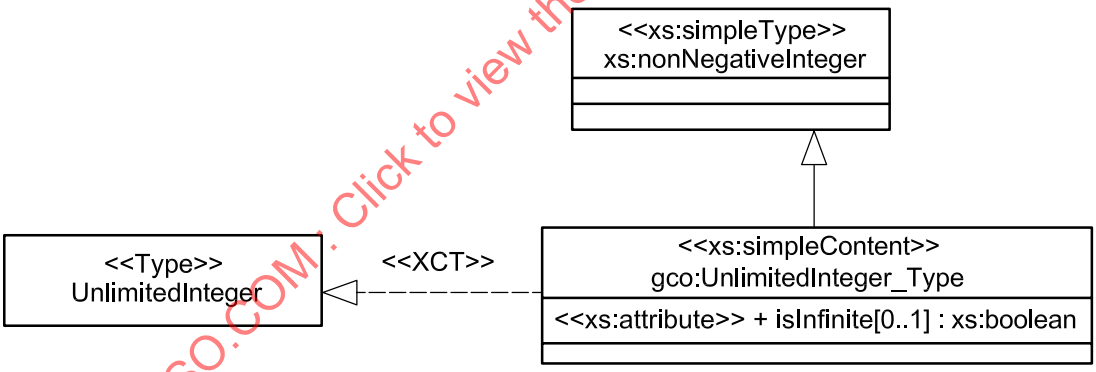

Figure 33 — XCT Realize for UnlimitedInteger_Type with xs:simpleContent stereotype

The XCT corresponding to UnlimitedInteger as shown in Figure 33 is:

```
<xs:complexType name="UnlimitedInteger_Type">
   <xs:simpleContent>
      <xs:extension base="xs:nonNegativeInteger">
         <xs:attribute name="isInfinite" type="xs:boolean"/>
      </xs:extension>
   </xs:simpleContent>
</xs:complexType>
```

8.5.8.4.4 The xs:complexType stereotype

When the realization has an XCT stereotype and the source class has an *xs:complexType* stereotype an XCT is created corresponding to the source class.

The XCT follows the following guidelines.

a) There is an xs:complexType created with a *name* attribute equal to the name of the XML class minus any namespace prefix. If the XML class happens to be abstract then the xs:complexType will contain an *abstract* attribute with the value set to “true”.

b) The xs:complexType contains an xs:complexContent.

c) The xs:complexContent contains an xs:extension with a *base* attribute set to the name of the external implementation super class. If there is no external implementation super class then the *base* attribute is set to gco:AbstractObject.

d) The xs:extension contains an xs:sequence composed of xs:elements based on the properties of the XML class which are stereotyped xs:element. Any property with the xs:element stereotype indicates the presence of an xs:element in the xs:sequence with a *name* attribute equal to the name of the property and the *type* attribute equal to the property type. Just as with default XCT encodings from 8.2, the minOccurs and maxOccurs values are based on ISO 19118:2005, Table A.5.

e) After the xs:sequence element, there may also be xs:attributes or xs:attributeGroups based on any properties of the XML class that are stereotyped xs:attribute or xs:attributeGroup. If there is a type value present in the UML element then the xs:attribute or xs:attributeGroup will have a *name* attribute equal to the name of the element in the UML and a *type* attribute equal to the type shown in the UML. If there is no type present for an element in the UML then the xs:attribute or xs:attributeGroup will contain a *ref* attribute with a value equal to the name of the element in the UML. Just as with default XCT encodings from 8.2, the minOccurs and maxOccurs values are based on Table A.5 of ISO 19118:2005.

EXAMPLE Sample XCT realization on an xs:complexType in the gmd namespace.

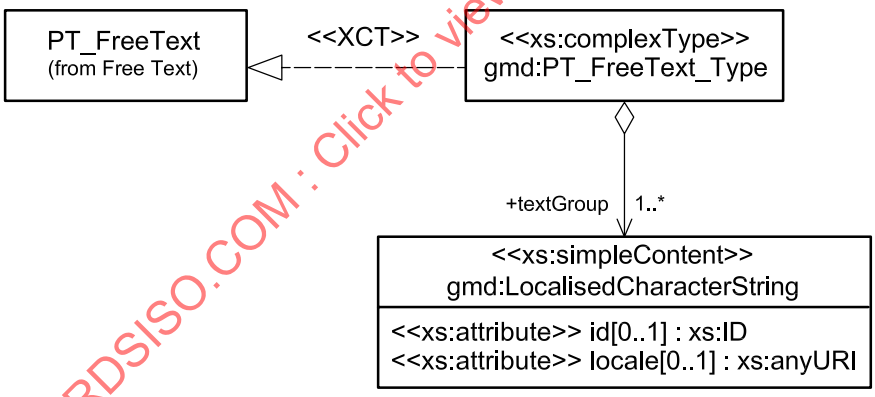

Figure 34 — XCT Realize for gmd:PT_FreeText_Type with xs:complexType stereotype

The XCT corresponding to PT_FreeText as shown in Figure 34 is:

```
<xs:complexType name="PT_FreeText_Type">
    <xs:complexContent>
       <xs:extension base="gco:AbstractObject_Type">
         <xs:sequence>
            <xs:element name="textGroup" type="
gmd:LocalisedCharacterString_PropertyType" maxOccurs="unbounded"/>
         </xs:sequence>
       </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

8.5.8.4.5 The xs:union stereotype

When a class has an xs:union stereotype an XCT is created corresponding to the UML class. The XCT is an xs:simpleType with the *name* attribute equal to the name of the UML class. Within the xs:simpleType is one xs:union with a *member* attribute equal to the names of the elements in the UML class separated by spaces.

EXAMPLE Sample XCT for an xs:union in the gco namespace.

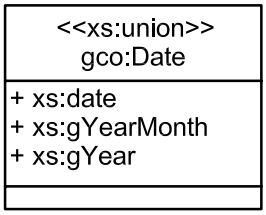

Figure 35 — gco:Date XCT with stereotype xs:union

The XCT corresponding to Date as shown in Figure 35 is:

```
<xs:simpleType name="Date_Type">
     <xs:union memberTypes="xs:date xs:gYearMonth xs:gYear"/>
  </xs:simpleType>
```

8.5.8.4.6 Creating the XCGE when encoding through the XCT

An XCGE is created for any external implementation XCTs. The XCGEs follow the default encoding rules described in 8.3, 8.5.2 (for the case of abstract classes) and 8.5.3 (for the case of subclasses).

8.5.8.4.7 Creating the XCPT

An XCPT is also created following the encoding rules in 8.4.1 if the source class of the realization is stereotyped *xs:complexType* or following the encoding rules in 8.4.2 if the source class of the realization is stereotyped *xs:simpleType* or *xs:simpleContent*.

EXAMPLE 1 The XCPT for CharacterString based on Figure 32 is:

```
<xs:complexType name="CharacterString_PropertyType">
    <xs:sequence minOccurs="0">
        <xs:element ref="gco:CharacterString"/>
    </xs:sequence>
  <xs:attribute ref="gco:nilReason"/>
</xs:complexType>
```

If a class is stereotyped *xs:choice* then the XCPT is slightly modified from the default encoding rules. It is assumed that a class stereotyped *xs:choice* represents a unique XCPT and as such its name should already include the suffix, "_PropertyType". This name will be used as the *name* attribute in the xs:complexType element for the XCPT which will also contain an *xs:choice* with a *minOccurs* attribute set to "0". Within the xs:choice will be an *xs:element* corresponding to each property of the UML class. Each xs:element will contain one *ref* attribute containing the name of the XCGE specified as the property of the UML class. After the *xs:choice* is closed, there is one gco:nilReason and then the XCPT is closed.

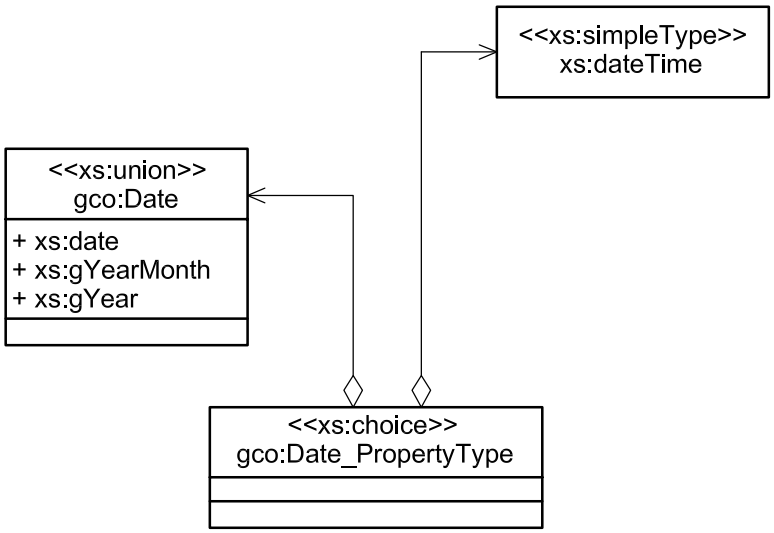

Figure 36 — Sample UML for class stereotyped xs:choice

EXAMPLE 2 The XCPT for gco:Date_PropertyType based on Figure 36 is:

```
<xs:complexType name="Date_PropertyType">
  <xs:choice minOccurs="0">
        <xs:element ref="gco:Date"/>
        <xs:element ref="gco:DateTime"/>
    </xs:choice>
    <xs:attribute ref="gco:nilReason"/>
</xs:complexType>
```

8.6 XML namespace package encoding

The stereotype *xmlNamespace* was introduced in 5.4 as a set of XML objects grouped within the same namespace. A UML package can have dependencies that are defined as a relationship between two modelling elements, in which a change to one modelling element will affect the other modelling element (see ISO/TS 19103). In the case of an <<xmlNamespace>> package there are two types of dependency relationships to consider. The stereotype *implement* on a dependency has a source representing the XML namespace (or namespace prefix) that implements the abstract concepts defined in the target. An introduction to gmd is provided in 6.1 and Figure 37 shows in UML what that introductory text says in words using an <<xmlNamespace>> package and an <<implement>> dependency.

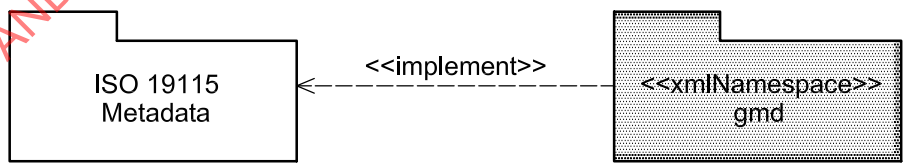

Figure 37 — xmlNamespace and implement

The stereotype *import* on a dependency has a source representing objects grouped within one namespace which depend on objects grouped within another namespace. An <<import>> dependency will exist between two <<xmlNamespace>> packages if there is a dependency relationship between the two abstract packages that are the target of <<xmlNamespace>> package <<implement>> dependencies. In Figure 38 an <<import>> dependency is shown between the gmd package and the gss package.

Figure 38 — xmlNamespace and import

The <<xmlNamespace>> package is not directly encoded into XML but it will have a corresponding <<xmlSchema>> package that utilizes the <<implement>> and <<import>> dependency relationships to generate appropriate XML schema.

8.7 XML schema package encoding

According to 5.4, a package stereotyped xmlSchema is a package that represents an XML schema. For any <<xmlSchema>> package in the UML shown in Clause 9 there exists an XML schema file with the same name as the package. An <<xmlSchema>> package can utilize an <<implement>> dependency exactly as described in 8.6 for an <<xmlNamespace>> and it can also utilize an <<include>> dependency. Each <<xmlSchema>> package is one of two types.

The first type is a root <<xmlSchema>> package that corresponds directly to an <<xmlNamespace>> package and has the same name as the <<xmlNamespace>> package with the addition of the .xsd extension. This means that because there is a gmd <<xmlNamespace>> package shown in Figure 38 there is also a corresponding gmd.xsd package with the stereotype *xmlSchema* and accordingly a gmd.xsd file that is part of this Technical Specification.

Each XML schema file starts with: *<?xml version="1.0" encoding="utf-8"?>*. The root element in the same file is an xs:schema with the attributes shown in Table 2.

Table 2 — Attributes of xs:schema element for a root <<xmlSchema>> package

Occurances of attribute	Namespace of attribute	Name of attribute	Value of attribute	EXAMPLE attribute-value pair
One	N/A	targetNamespace	http://www.isotc211.org/2005/ + *packageName*	targetNamespace= "http://www.isotc211.org/2005/gmd"
One	xmlns	*packageName*	http://www.isotc211.org/2005/ + *packageName*	xmlns:gmd= "http://www.isotc211.org/2005/gmd"
Zero to Many	xmlns	*packageName of target of <<import>> dependency*	http://www.isotc211.org/2005/ + *packageName of target of <<import>> dependency*	xmlns:gco= "http://www.isotc211.org/2005/gco"
One	xmlns	xs	http://www.w3.org/2001/XMLSchema	xmlns:xs="http://www.w3.org/2001/ XMLSchema"
One	N/A	version	1.0	version="1.0"
NOTE packageName refers to the name of the corresponding <<xmlNamespace>> package of a root <<xmlSchema>> package.				

EXAMPLE The UML shown in Figure 38 would result in the existence of a file named gmd.xsd with the following declaration of xs:schema:

< xs:schema targetNamespace="http://www.isotc211.org/2005/gmd" xmlns:gmd="http://www.isotc211.org/2005/gmd" xmlns:gss="http://www.isotc211.org/2005/gss" xmlns:xs="http://www.w3.org/2001/XMLSchema" version="1.0" />

The second type of <<xmlSchema>> package is referred to as a regular <<xmlSchema>> package and it corresponds directly to an existing package from the ISO 19100 series, which is indicated by a dependency with the stereotype *implement*. The example in Figure 39 shows a regular <<xmlSchema>> package and its target ISO 19115 package.

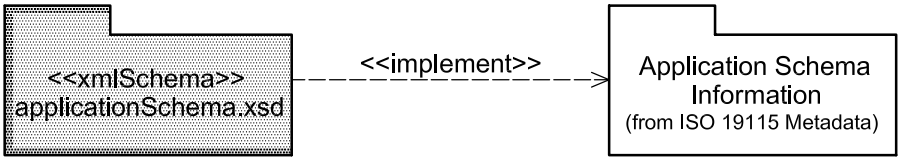

Figure 39 — Regular <<xmlSchema>> package

All regular <<xmlSchema>> packages have exactly one corresponding root <<xmlSchema>> package which is either the direct source of an <<include>> dependency between the regular <<xmlSchema>> package or is the root <<xmlSchema>> of a regular <<xmlSchema>> that is the direct source of an <<include>> dependency to the regular <<xmlSchema>> of interest. In Figure 40 the root <<xmlSchema>> is root.xsd and the regular <<xmlSchema>> are schema1.xsd and schema2.xsd. The root.xsd <<xmlSchema>> is the root of the two regular <<xmlSchema>>.

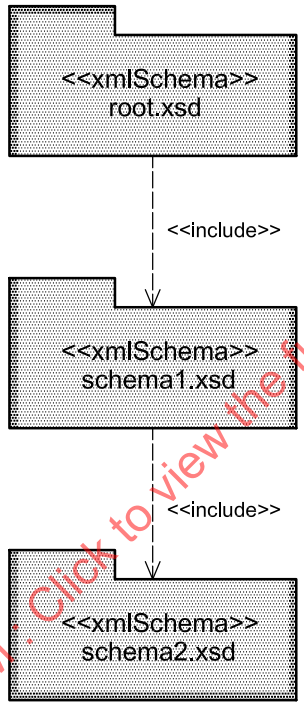

Figure 40 — Hierarchy of <<xmlSchema>> packages

The first XML schema element within each XML schema file that is created based on the existence of a regular <<xmlSchema>> package is *<?xml version="1.0" encoding="utf-8"?>*. The second element in the same file is an xs:schema with the attributes shown in Table 3.

Table 3 — Attributes of xs:schema element for a regular <<xmlSchema>> package

Occurances of attribute	Namespace of attribute	Name of attribute	Value of attribute	EXAMPLE attribute-value pair
One	N/A	targetNamespace	http://www.isotc211.org/2005/ + *rootPackageName*	targetNamespace= "http://www.isotc211.org/2005/gmd"
One	xmlns	*rootPackageName*	http://www.isotc211.org/2005/ + *rootPackageName*	xmlns:gmd= "http://www.isotc211.org/2005/gmd"
Zero to Many	xmlns	*rootPackageName of target of <<import>> dependency*	http://www.isotc211.org/2005/ + *rootPackageName of target of <<import>> dependency*	xmlns:gco= "http://www.isotc211.org/2005/gco"
One	xmlns	xs	http://www.w3.org/2001/XMLSchema	xmlns:xs="http://www.w3.org/2001/ XMLSchema"
One	xmlns	xsi	http://www.w3.org/2001/XMLSchema-instance	xmlns:xsi="http://www.w3.org/2001/ XMLSchema-instance"
One	N/A	version	1.0	version="1.0"
NOTE	*rootPackageName* refers to the name of the corresponding <<xmlNamespace>> package of the root <<xmlSchema>> package.			

9 Encoding descriptions

9.1 Introduction to the encoding descriptions

The implementation of ISO 19115 and related standards follows the encoding rules stated in Clause 8. The exceptions and the implementations based on external types are detailed in this clause. This clause uses the UML notation commonly used in the ISO 19100 series of International Standards, plus the realization concept and the implementation stereotypes defined in 5.4.

9.2 XML namespaces

Figure 41 shows the different namespaces used to implement ISO 19115 (grey boxes) along with the relationships between these namespaces and the ISO 19100 series packages (white boxes).

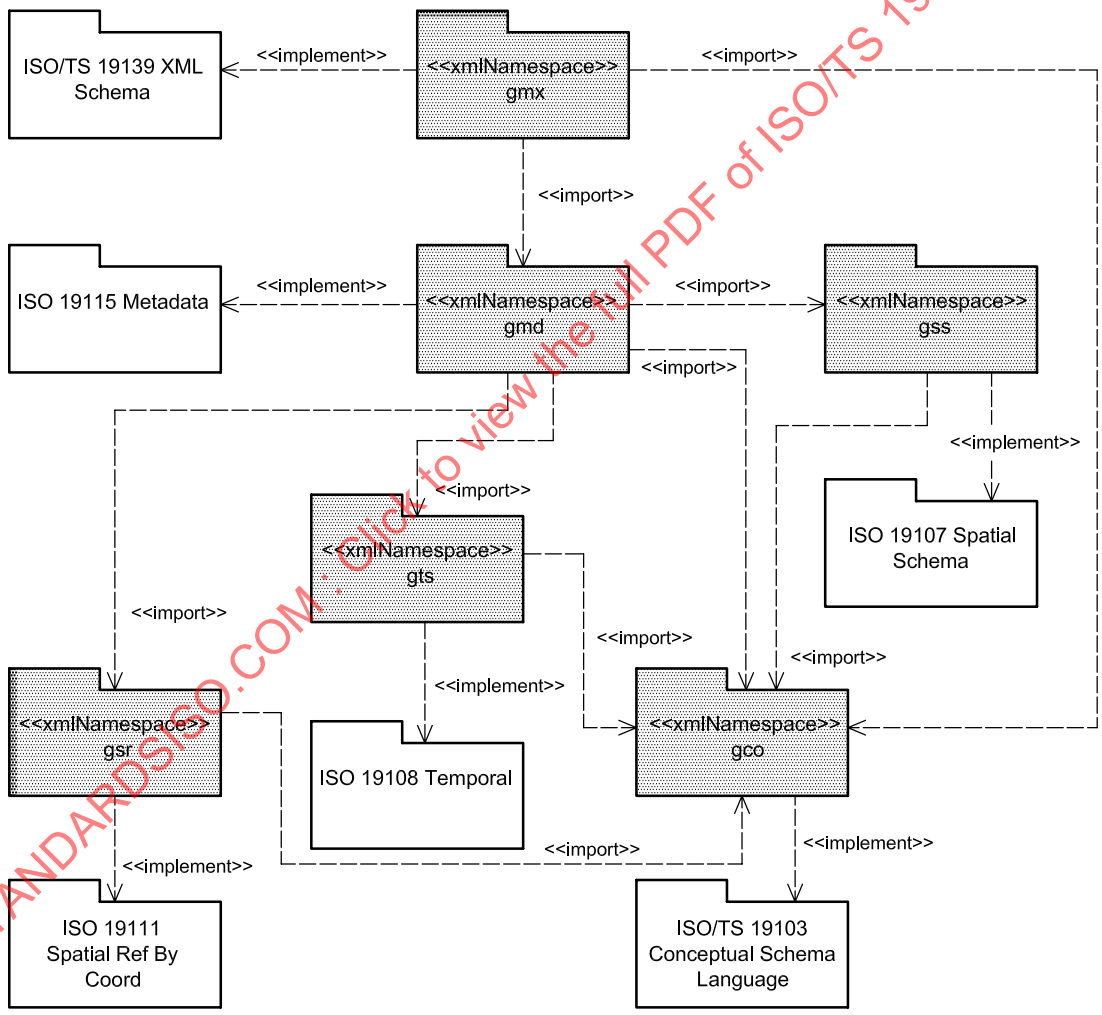

Figure 41 — XML packaging

9.3 gmd namespace

9.3.1 Organization of the gmd namespace

This namespace contains the implementation of ISO 19115. The root of this namespace is gmd.xsd. Figure 42 details the oganization of the gmd namespace.

9.3.2 gmd.xsd

This XML schema includes (directly or indirectly) all the implemented concepts of the gmd namespace, but it does not contain the declaration of any types.

9.3.3 metadataApplication.xsd

This XML schema implements the UML conceptual schema defined in 6.2 of ISO 19115:2003. It contains the implementation of the following classes: DS_Aggregate, DS_Dataset, DS_OtherAggregate, DS_Series, DS_Initiative, DS_Platform, DS_Sensor, DS_ProductionSeries, DS_StereoMate.

The classes implemented in this XML schema follow the encoding rules defined in Clause 8.

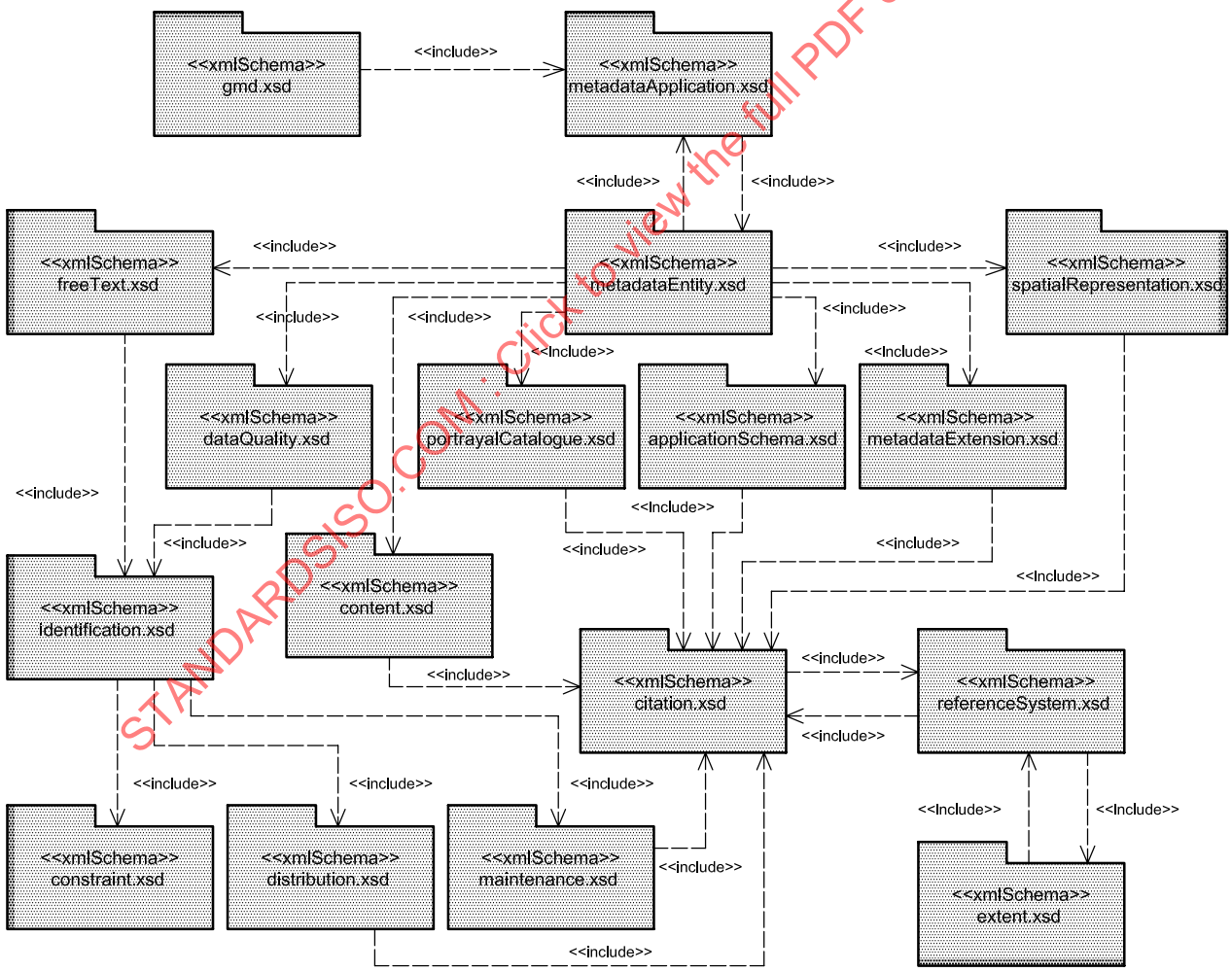

Figure 42 — Organization of the gmd namespace

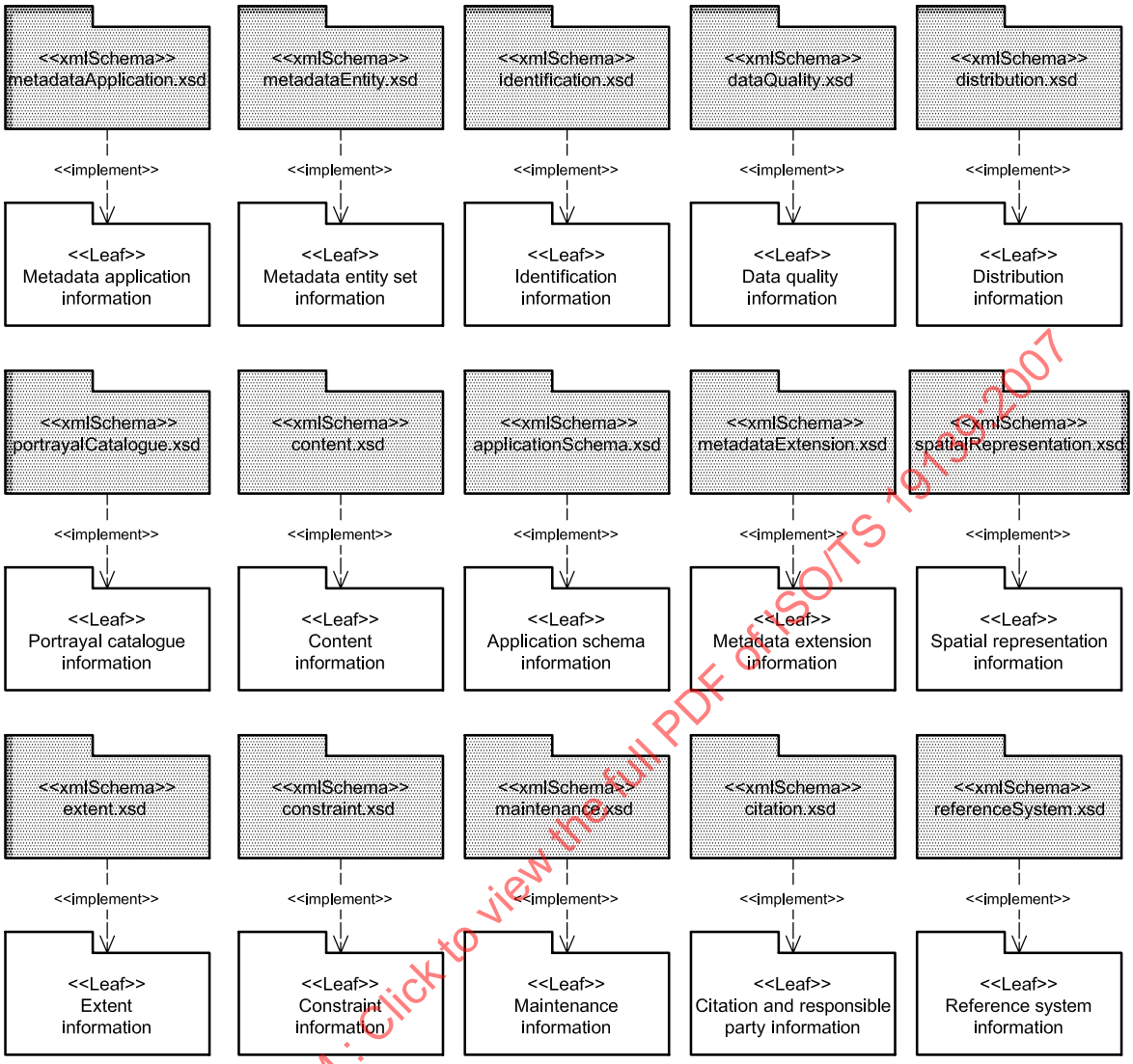

Figure 42 — Organization of gmd namespace (*continued*)

9.3.4 metadataEntity.xsd

This XML schema implements the UML conceptual schema defined in A.2.1 of ISO 19115:2003. It contains the implementation of the class MD_Metadata.

In 7.3, extension elements were introduced that provide cultural and linguistic adaptability. In addition to the discussion in that subclause, another aspect of cultural and linguistic adaptability is enabling multilingual values in XML documents. In order to express multilingual values it is necessary to declare the locales in which the alternative values can be expressed and in order to do that properly in XML the MD_Metadata class defined in ISO 19115:2003, A.2.1 shall be transformed. The additionally optional property, “locale”, of type PT_Locale (see 7.3) is added to MD_Metadata as shown in Figure 4.

The ISO/TS 19139 MD_Metadata class follows the encoding rules described in Clause 8.

9.3.5 identification.xsd

This XML schema implements the UML conceptual schema defined in A.2.2 of ISO 19115:2003. It contains the implementation of the following classes: MD_Identification, MD_BrowseGraphic, MD_DataIdentification, MD_ServiceIdentification, MD_RepresentativeFraction, MD_Usage, MD_Keywords, DS_Association,

MD_AggregateInformation, MD_CharacterSetCode, MD_SpatialRepresentationTypeCode, MD_TopicCategoryCode, MD_ProgressCode, MD_KeywordTypeCode, DS_AssociationTypeCode, DS_InitiativeTypeCode, MD_ResolutionType.

The classes implemented in this XML schema follow the encoding rules described in Clause 8.

9.3.6 constraints.xsd

This XML schema implements the UML conceptual schema defined in A.2.3 of ISO 19115:2003. It contains the implementation of the following classes: MD_Constraints, MD_LegalConstraints, MD_SecurityConstraints, MD_ClassificationCode, MD_RestrictionCode.

The classes implemented in this XML schema follow the encoding rules described in Clause 8.

9.3.7 dataQuality.xsd

This XML schema implements the UML conceptual schema defined in A.2.4 of ISO 19115:2003. It contains the implementation of the following classes: LI_ProcessStep, LI_Source, LI_Lineage, DQ_ConformanceResult, DQ_QuantitativeResult, DQ_Result, DQ_TemporalValidity, DQ_AccuracyOfATimeMeasurement, DQ_QuantitativeAttributeAccuracy, DQ_NonQuantitativeAttributeAccuracy, DQ_ThematicClassificationCorrectness, DQ_RelativeInternalPositionalAccuracy, DQ_GriddedDataPositionalAccuracy, DQ_AbsoluteExternalPositionalAccuracy, DQ_TopologicalConsistency, DQ_FormatConsistency, DQ_DomainConsistency, DQ_ConceptualConsistency, DQ_CompletenessOmission, DQ_CompletenessCommission, DQ_TemporalAccuracy, DQ_ThematicAccuracy, DQ_PositionalAccuracy, DQ_LogicalConsistency, DQ_Completeness, DQ_Element, DQ_DataQuality.

The classes implemented in this XML schema follow the encoding rules described in Clause 8.

9.3.8 maintenance.xsd

This XML schema implements the UML conceptual schema defined in A.2.5 of ISO 19115:2003. It contains the implementation of the following classes: MD_MaintenanceInformation, MD_MaintenanceFrequencyCode, MD_ScopeCode, MD_ScopeDescription.

The classes implemented in this XML schema follow the encoding rules described in Clause 8.

9.3.9 spatialRepresentation.xsd

This XML schema implements the UML conceptual schema defined in A.2.6 of ISO 19115:2003. It contains the implementation of the following classes: MD_GridSpatialRepresentation, MD_VectorSpatialRepresentation, MD_SpatialRepresentation, MD_Georeferenceable, MD_Dimension, MD_Georectified, MD_GeometricObjects, MD_TopologyLevelCode, MD_GeometricObjectTypeCode, MD_CellGeometryCode, MD_DimensionNameTypeCode, MD_PixelOrientationCode.

The classes implemented in this XML schema follow the encoding rules described in Clause 8.

9.3.10 referenceSystem.xsd

This schema implements the UML conceptual schema defined in A.2.7 of ISO 19115:2003 and ISO 19115:2003/Cor. 1:2006. It contains the implementation of the following classes: RS_Identifier, MD_ReferenceSystem, MD_Identifier and RS_Reference System.

The classes implemented in this XML schema follow the encoding rules described in Clause 8.

9.3.11 content.xsd

This schema implements the UML conceptual schema defined in ISO 19115:2003, A.2.8. It contains the implementation of the following classes: MD_FeatureCatalogueDescription, MD_CoverageDescription, MD_ImageDescription, MD_ContentInformation, MD_RangeDimension, MD_Band, MD_CoverageContentTypeCode, MD_ImagingConditionCode.

The classes implemented in this XML schema follow the encoding rules described in Clause 8.

9.3.12 portrayalCatalogue.xsd

This schema implements the UML conceptual schema defined in A.2.9 of ISO 19115:2003. It contains the implementation of the class MD_PortrayalCatalogueReference.

The classes implemented in this XML schema follow the encoding rules described in Clause 8.

9.3.13 distribution.xsd

This schema implements the UML conceptual schema defined in A.2.10 of ISO 19115:2003. It contains the implementation of the following classes: MD_Medium, MD_DigitalTransferOptions, MD_StandardOrderProcess, MD_Distributor, MD_Distribution, MD_Format, MD_MediumFormatCode, MD_MediumNameCode.

The classes implemented in this XML schema follow the encoding rules described in Clause 8.

9.3.14 metadataExtension.xsd

This schema implements the UML conceptual schema defined in A.2.11 of ISO 19115:2003. It contains the implementation of the following classes: MD_ExtendedElementInformation, MD_MetadataExtensionInformation, MD_ObligationCode, MD_DatatypeCode.

The classes implemented in this XML schema follow the encoding rules described in Clause 8.

9.3.15 applicationSchema.xsd

This schema implements the UML conceptual schema defined in A.2.12 of ISO 19115:2003. It contains the implementation of the class: MD_ApplicationSchemaInformation.

The classes implemented in this XML schema follow the encoding rules described in Clause 8.

9.3.16 extent.xsd

This schema implements the UML conceptual schema defined in A.3.1 of ISO 19115:2003 and the associated corrigendum. It contains the implementation of the following classes: EX_TemporalExtent, EX_VerticalExtent, EX_BoundingPolygon, EX_Extent, EX_GeographicExtent, EX_GeographicBoundingBox, EX_SpatialTemporalExtent, EX_GeographicDescription.

The classes implemented in this XML schema follow the encoding rules described in Clause 8.

9.3.17 citation.xsd

This schema implements the UML conceptual schema defined in A.3.2 of ISO 19115:2003. It contains the implementation of the following classes: CI_ResponsibleParty, CI_Citation, CI_Address, CI_OnlineResource, CI_Contact, CI_Telephone, URL, CI_Date, CI_Series, CI_RoleCode, CI_PresentationFormCode, CI_OnLineFunctionCode, CI_DateTypeCode.

The classes implemented in this XML schema follow the encoding rules described in Clause 8 with the exception of the URL class, which is implemented as an XML simple type mapped to xs:anyURI as shown in Figure 43.

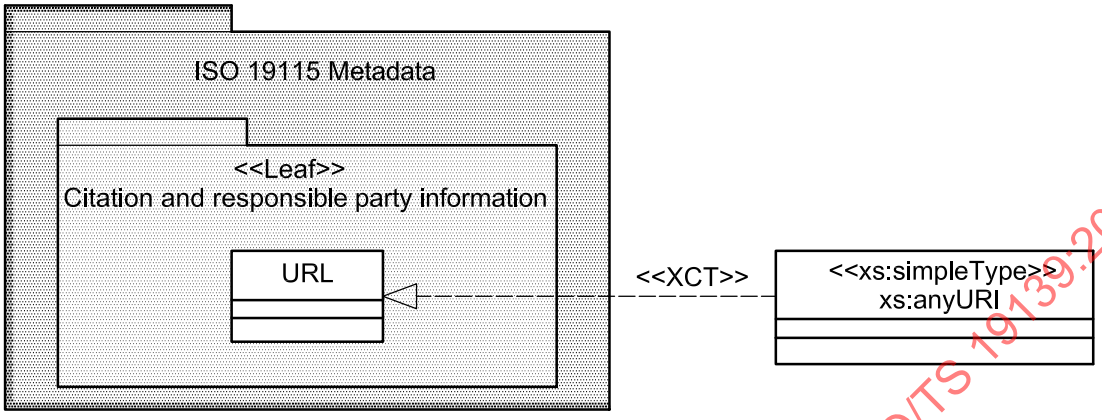

Figure 43 — Implementation of URL

9.3.18 freeText.xsd

9.3.18.1 Introduction

This schema implements cultural and linguistic adaptability extensions defined in 7.3. This extension essentially formalizes the free text concept described in Annex J of ISO 19115:2003. For this reason, and in order to simplify the organization of overall geographic metadata XML schema, this schema has been included as part of the gmd namespace instead of the gmx namespace.

The following classes are encoded using the encoding rules described in Clause 8: PT_Locale and PT_LocaleContainer.

9.3.18.2 XML implementation of localised character string

The LocalisedCharacterString requires a customized XCT and XCPT since its XCGE is a simple content element that must be referenceable:

— its XCPT inherits the gco:ObjectReference attribute group from the gco_ObjectReference_PropertyType;

— its XCT has an *id* attribute which serves to identify the LocalisedCharacterString XCGE when it is referenced.

Furthermore, the locale role of LocalisedCharacterString is limited to a by-reference XML implementation through the *locale* XML attribute of the LocalisedCharacterString XCT. The XML implementation of the LocalisedCharacterString class is shown in Figure 44.

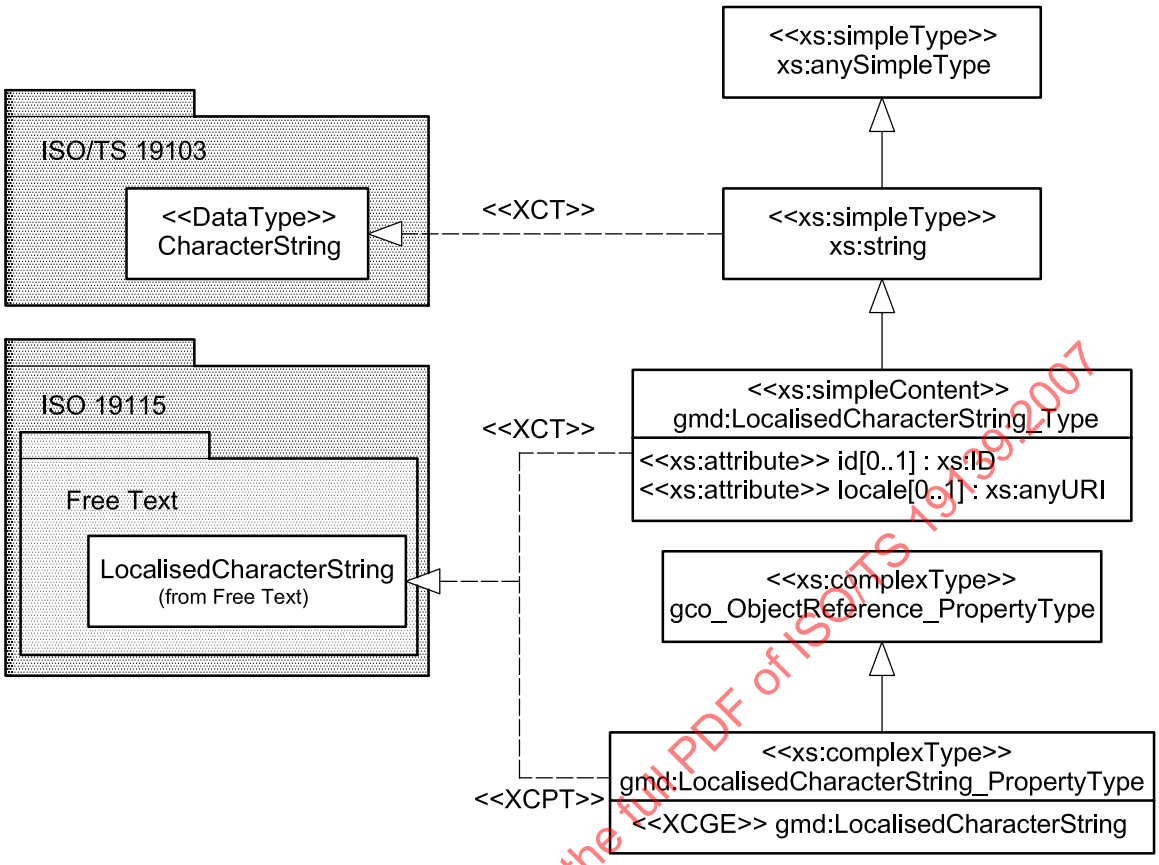

Figure 44 — XML implementation of localized character string

9.3.18.3 XML implementation of free text

PT_FreeText is implemented through a complex content XCPT even though it inherits from CharacterString, which is implemented through a simple type XCT. As an XML complex type can neither extend nor restrict an XML simple type:

— the PT_FreeText XCT is implemented using the default encoding rule except that it does not extend nor restrict the CharacterString XCT;

— the inheritance between PT_FreeText and CharacterString implies the definition of a customised XCPT. This XCPT inherits from the default CharacterString XCPT the gco:CharacterString XCGE. It extends the CharacterString XCPT with the default PT_FreeText XCGE, which handles the localized character strings.

The definition of those customized XCT and XCPT is shown in Figure 45.

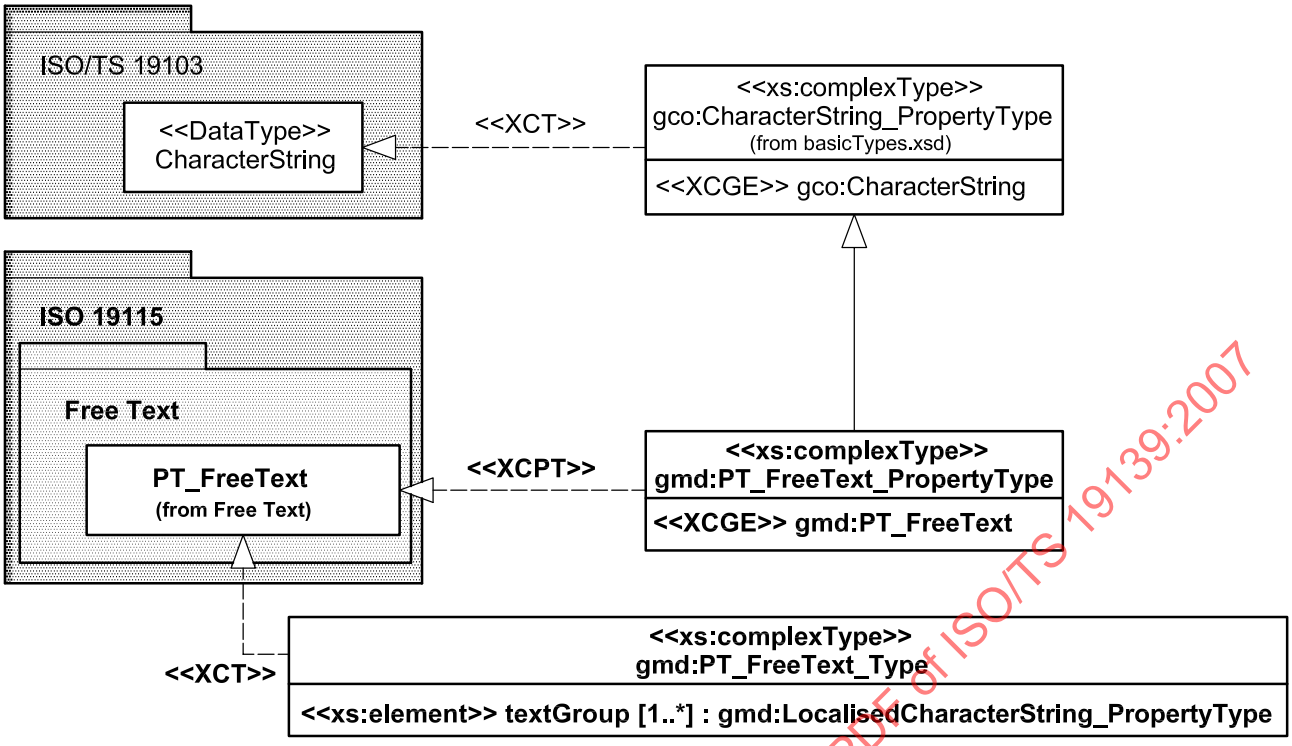

Figure 45 — XML implementation of free text

EXAMPLE Example of use:

XML schema fragment	See the “abstract” property of MD_Identification (identification.xsd) `<xs:element name="abstract" type="gco:CharacterString_PropertyType"/>`

XML instance example	`<abstract xsi:type="PT_FreeText_PropertyType">`
	`  <gco:CharacterString>Brief narrative summary of the content of the resource</gco:CharacterString>`
	`<!--== Alternative values ==-->`
	`  <PT_FreeText>`
	`  <textGroup><LocalisedCharacterString locale="locale-fr">Résumé succint du contenu du jeu de données</LocalisedCharacterString></textGroup>`
	`  <textGroup><LocalisedCharacterString locale="locale-anyLocale"><!--Translation in the specific locale--></LocalisedCharacterString></textGroup>`
	`  </PT_FreeText>`
	`</abstract>`

9.4 gss namespace

9.4.1 Organization of the gss namespace

The implementation of ISO 19115 requires the encoding of several elements from ISO 19107. As these elements are not specific to geographic metadata, a separate namespace is created to contain the XML schemas for the elements from ISO 19107. That namespace is http://www.isotc211.org/2005/gss and the common prefix used to refer to that namespace is gss, which stands for geographic spatial schema. The root of this namespace is gss.xsd. Figure 46 shows the organization of the gss namespace.

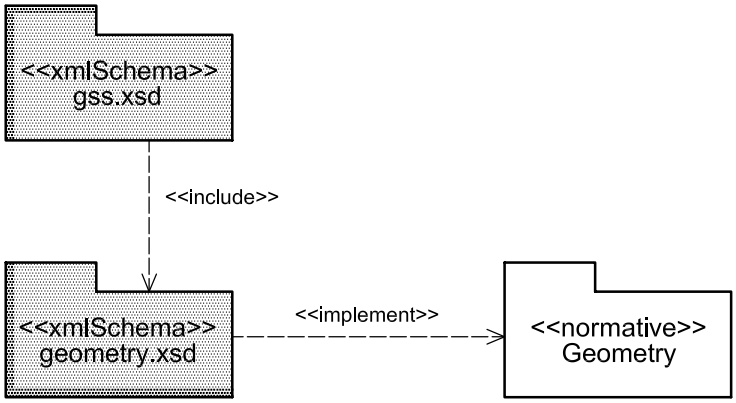

Figure 46 — Organization of the gss namespace

9.4.2 gss.xsd

This XML schema includes (directly or indirectly) all the implemented concepts of the gss namespace, but it does not contain the declaration of any types.

9.4.3 geometry.xsd

This XML schema contains the implementation of GM_Object and GM_Point. The encoding of these classes is mapped to ISO 19136 geometric types. Figure 47 illustrates the mapping.

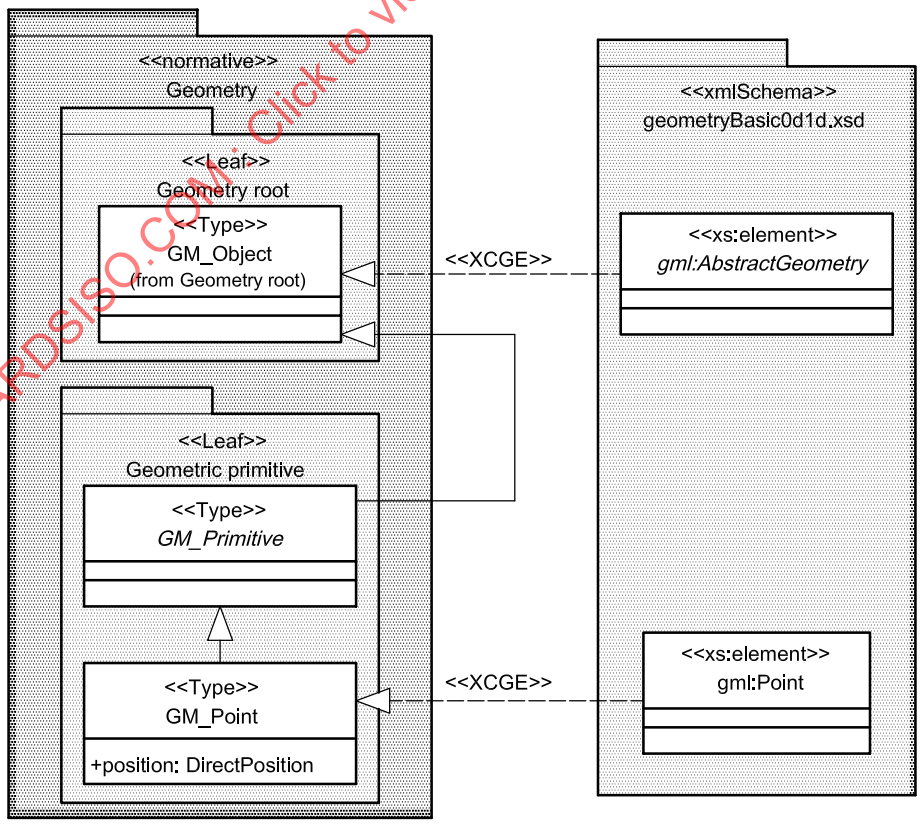

Figure 47 — ISO 19107 profile to ISO 19136

9.5 gts namespace

9.5.1 Organization of the gts namespace

The implementation of ISO 19115 requires the encoding of a few elements from ISO 19108. As these elements are not specific to geographic metadata, a separate namespace is created to contain the XML schemas for the elements in ISO 19108. That namespace is http://www.isotc211.org/2005/gts and the common prefix used to refer to that namespace is gts which stands for geographic temporal schema. The root of this namespace is gts.xsd. Figure 48 shows the organization of the gts namespace.

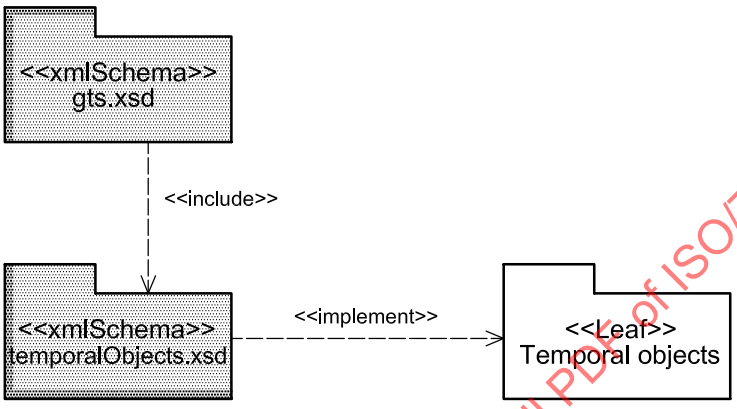

Figure 48 — Organization of the gts namespace

9.5.2 gts.xsd

This XML schema includes (directly or indirectly) all the implemented concepts of the gts namespace, but it does not contain the declaration of any types.

9.5.3 temporalObjects.xsd

This schema contains the XML implementation of TM_Object, TM_Primitive and TM_PeriodDuration. The encoding of these classes is mapped to ISO 19136 temporal types and W3C built-in types. Figure 49 illustrates the mapping to the ISO 19136 types. Figure 50 illustrates the mapping to the W3C buit-in types.

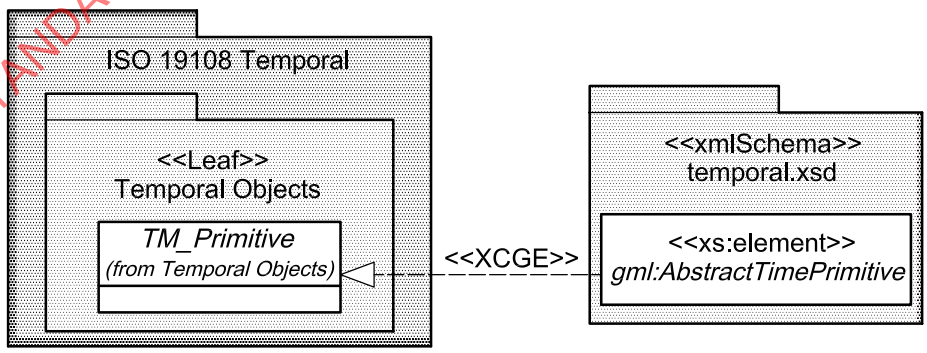

Figure 49 — ISO 19108 profile to ISO 19136

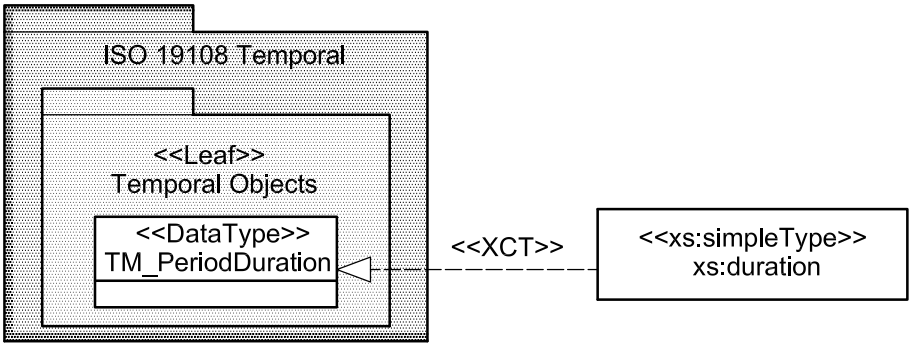

Figure 50 — ISO 19108 profile to the W3C simple type schema

9.6 gsr namespace

9.6.1 Organization of the gsr namespace

The implementation of ISO 19115 requires the encoding of a few elements from ISO 19111. As these elements are not specific to geographic metadata, a separate namespace is created to contain the XML schemas for the elements in ISO 19111. That namespace is http://www.isotc211.org/2005/gsr and the common prefix used to refer to that namespace is gsr which stands for geographic spatial referencing. The root of this namespace is gsr.xsd. Figure 51 shows the organization of the gsr namespace.

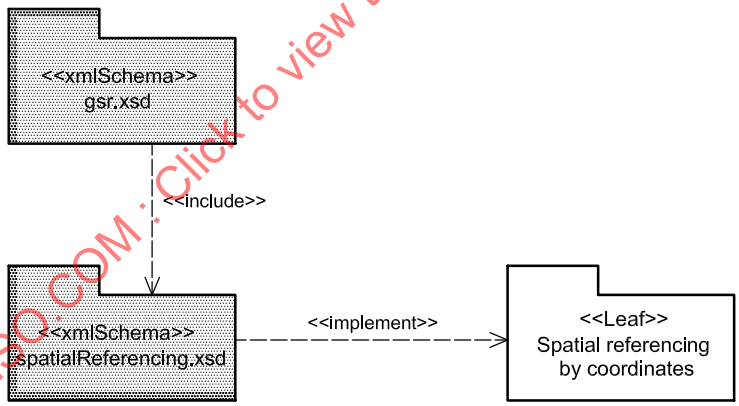

Figure 51 — Organization of the gsr namespace

9.6.2 gsr.xsd

This XML schema includes (directly or indirectly) all the implemented concepts of the gsr namespace, but it does not contain the declaration of any types.

9.6.3 spatialReferencing.xsd

This XML schema contains the implementation of SC_CRS. The encoding of this class is mapped to an ISO 19136 XML type. Figure 52 illustrates the mapping.

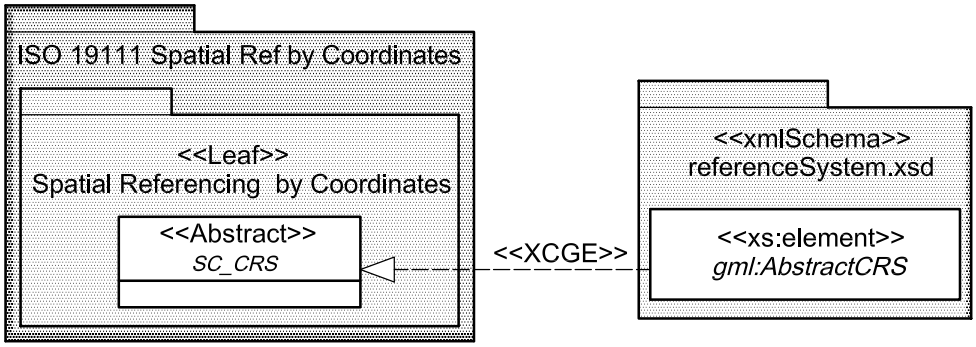

Figure 52 — ISO 19111 profile to ISO 19136

9.7 gco namespace

9.7.1 Organization of the gco namespace

The implementation of ISO 19115 requires the encoding of required basic types from ISO/TS 19103 and conceptual elements from ISO 19118. As these elements are not specific to geographic metadata, a separate namespace is created to contain the XML schemas. That namespace is http://www.isotc211.org/2005/gco and the common prefix used to refer to that namespace is gco which stands for geographic common. The root of this namespace is gco.xsd. Figure 53 shows the organization of the gco namespace.

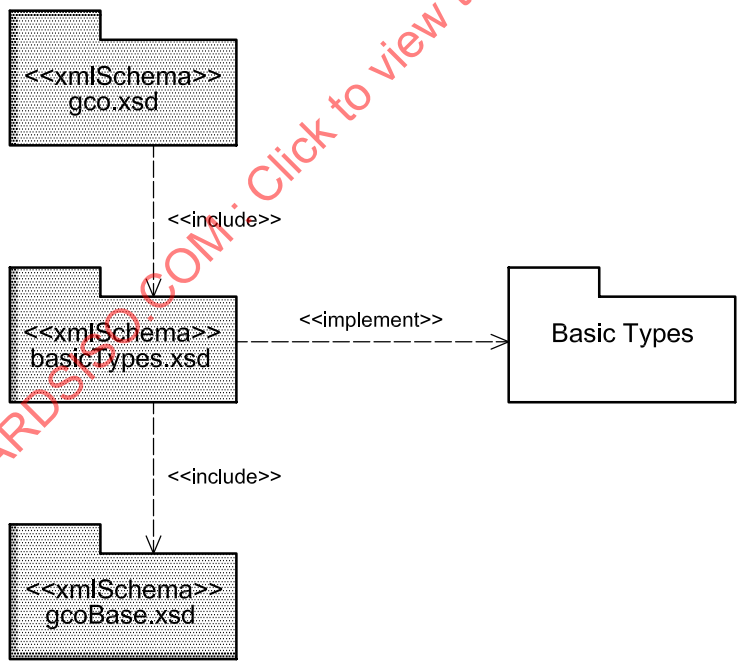

Figure 53 — Organization of the gco namespace

9.7.2 gco.xsd

This XML schema includes (directly or indirectly) all the implemented concepts of the gco namespace, but it does not contain the declaration of any types.

9.7.3 gcoBase.xsd

9.7.3.1 Introduction to gcoBase.xsd

This XML schema contains elements closely related to the common elements found in ISO/TS 19103 and serving a similar purpose for an XML implementation that ISO/TS 19103 elements serve for UML modelling. As a result, the elements contained in gcoBase.xsd and described in 9.7.3.2 to 9.7.3.4 are created in the http://www.isotc211.org/2005/gco namespace.

9.7.3.2 Elements identified for inclusion from ISO 19118

ISO 19118 defines core types used for object identification and object reference. These objects are defined in the gco namespace in the gcoBase.xsd schema file and are used by all objects in the namespaces that follow the encoding rules defined in this Technical Specification.

The gcoBase.xsd schema file contains the implementation of the following.

a) The XML element gco:AbstractObject_Type represents the ISO 19118 IM_Object. This element acts as the absolute root of the schemas, meaning all the types must be derived directly or indirectly from gco:AbstractObject_Type. The XML element gco:AbstractObject_Type contains a reference to gco:ObjectIdentification (see bullet c below) and is used to support resource identification. The XML schema fragment corresponding to gco:AbstractObject_Type is shown below:

XML type (XCT)	<xs:complexType name="AbstractObject_Type" abstract="true"> <xs:sequence/> <xs:attributeGroup ref="gco:ObjectIdentification"/> </xs:complexType>

b) The XML attributeGroup gco:ObjectReference represents the ISO 19118 IM_ObjectReference. This attribute group is used by the default XCPT pattern (described in 8.4) to reference to a remote resource. Note that the content model of this ObjectReference is slightly modified from that in ISO 19118 in order to be more consistent with ISO 19136. The gco:ObjectReference attributeGroup contains a reference to the xlink:simpleLink attributeGroup, plus the definition of an XML attribute named uuidref of type xs:string. The latter is used to support referencing to universal unique identifiers as defined in ISO 19118 (see 8.2.4.2). Moreover, an XCPT is provided in order to replace any property type which must be represented by a reference and whose content (i.e. the value content between two XML tags) must be empty. The corresponding XML schema fragments are shown below:

XML attributeGroup	<xs:attributeGroup name="ObjectReference"> <xs:attributeGroup ref="xlink:simpleLink"/> <xs:attribute name="uuidref" type="xs:string"/> </xs:attributeGroup> <!--================== NULL ====================--> <xs:attribute name="nilReason" type="gml:NilReasonType"/>
XML property type (XCPT)	<xs:complexType name="ObjectReference_PropertyType"> <xs:sequence/> <xs:attributeGroup ref="gco:ObjectReference"/> <xs:attribute ref="gco:nilReason"/> </xs:complexType>

c) The XML attributeGroup gco:ObjectIdentification represents the ISO 19118 IM_ObjectIdentification. This attribute group is used to identified a resource, it contains two elements: id of type xs:anyURI and uuid (universal unique identifier) of type xs:string. The corresponding XML schema fragment is shown below:

XML attributeGroup	<xs:attributeGroup name="ObjectIdentification"> <xs:attribute name="id" type="xs:ID"/> <xs:attribute name="uuid" type="xs:string"/> </xs:attributeGroup>

9.7.3.3 Basic elements used to manage references to registered resources

The gco:CodeListValue_Type is created to refer to a specific codelist value in a register.

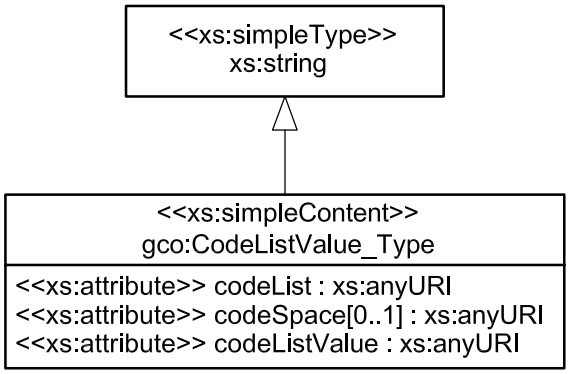

Figure 54 — CodeListValue_Type represented in XML schema

The instantiation and usage of a property with stereotype <<CodeList>> in an XML instance document is fully described in 8.5.5. This clause also explains the usage of each XML attribute of gco:CodeListValue_Type shown in Figure 54.

9.7.3.4 A basic element used to manage Null values in an XML instance document

The gco:nilReason XML attribute manages null values in an XML instance document. At the property level, this attribute allows a reason (explaining why the actual value cannot be provided) to exist in place of an actual value. It is used in the default XCPT pattern defined in 8.4. The corresponding XML schema fragment is shown below:

XML attribute	`<xs:attribute name="nilReason" type="gml:NilReasonType"/>`

The gml:NilReasonType is fully described in ISO 19136 and is an enumerated XML type allowing the values:

- “inapplicable”
- “missing”
- “template”
- “unknown”
- “withheld”

9.7.4 basicTypes.xsd

9.7.4.1 Elements identified for inclusion from ISO/TS 19103

9.7.4.1.1 ISO/TS 19103 classes following default encoding rules

The classes defined in ISO/TS 19103 that are included in the basicTypes.xsd schema file are Multiplicity, MultiplicityRange, MemberName and TypeName. Their XML schema definition follows the default encoding rules described in Clause 8.

9.7.4.1.2 ISO/TS 19103 classes represented in XML schema by W3C built-in types or other simple content types

The ISO/TS 19103 CharacterString class is represented in XML by an xs:string as shown in Figure 55.

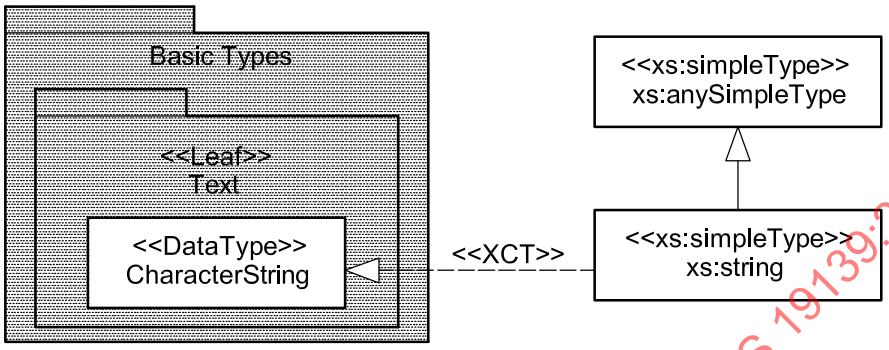

Figure 55 — CharacterString represented in XML schema

The ISO/TS 19103 Integer, Decimal and Real classes are respectively represented in XML by xs:integer, xs:decimal and xs:double as shown in Figure 56.

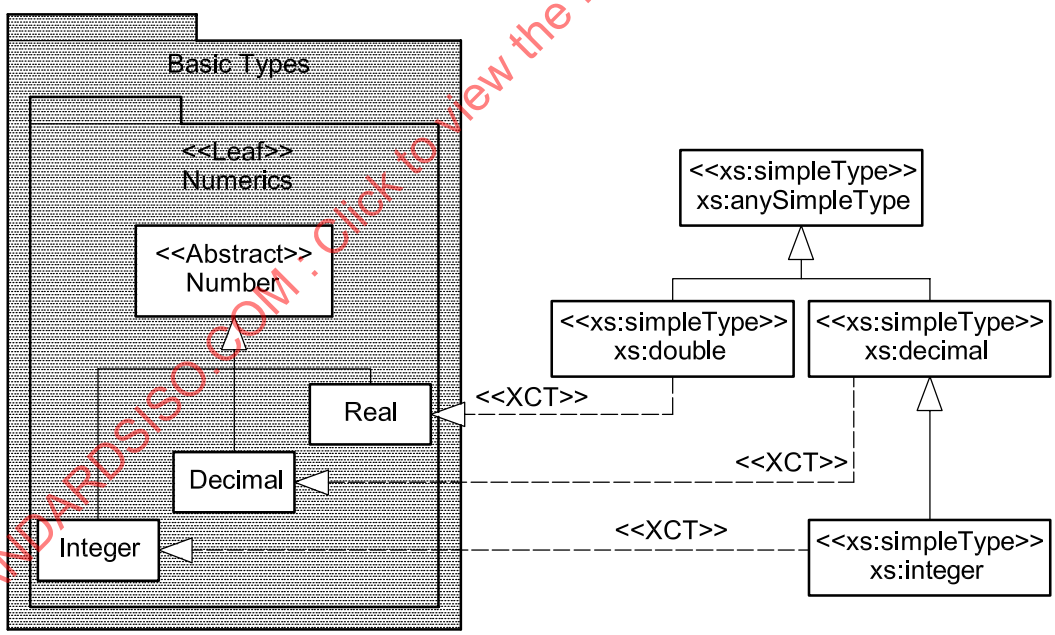

Figure 56 — Numerical data types represented in XML schema

The ISO/TS 19103 Boolean class is represented in XML schema by xs:boolean as shown in Figure 57.

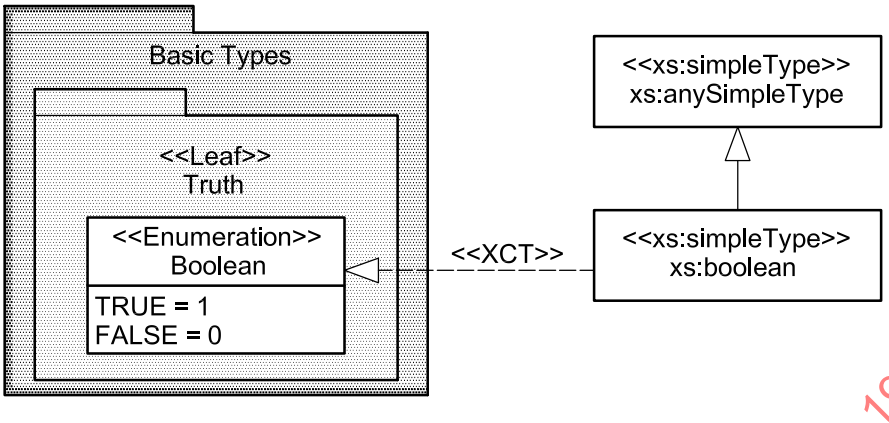

Figure 57 — Boolean data type represented in XML schema

For the sake of readability binary data is not directly embedded in XML documents. As a result, the ISO/TS 19103 Binary class is encoded as a reference to an external file containing the binary information as shown in Figure 58.

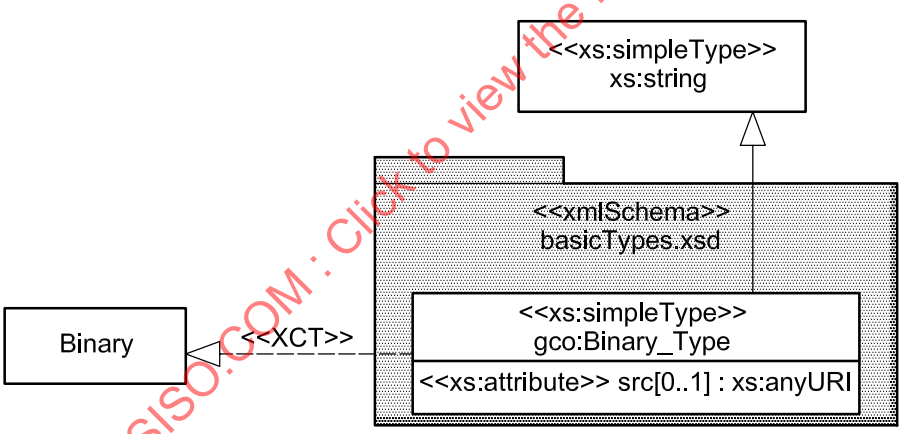

Figure 58 — Binary data type represented in XML schema

9.7.4.1.3 ISO/TS 19103 classes represented in XML schema by ISO 19136 XML types

The ISO/TS 19103 GenericName, LocalName and ScopedName classes are represented in XML schema by gml:CodeType as shown in Figure 59.

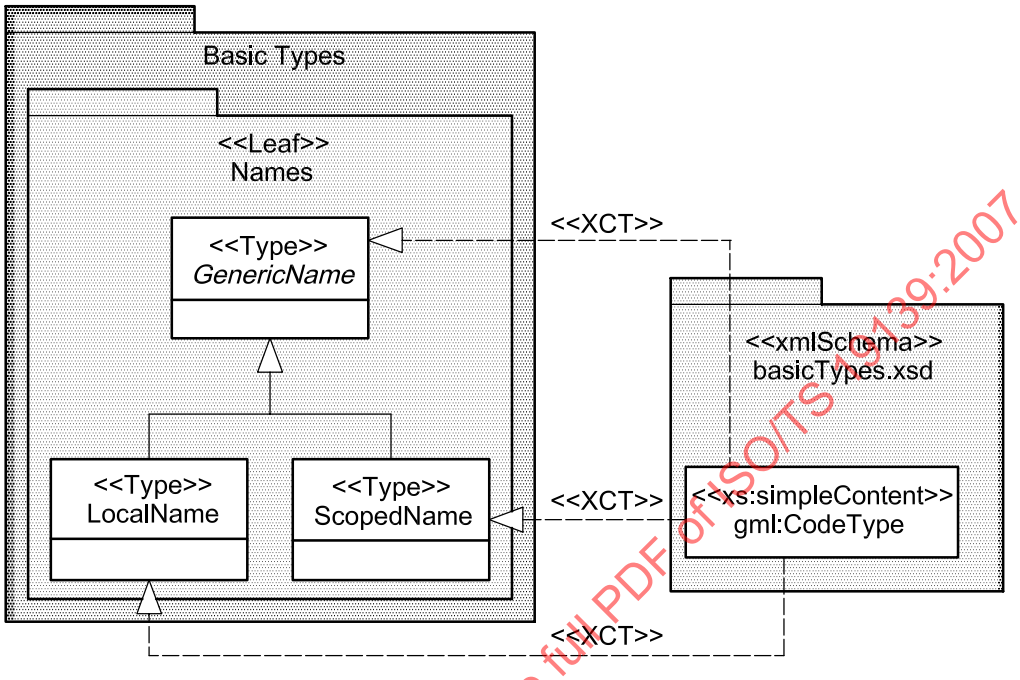

Figure 59 — Name data types implemented using ISO 19136 XML types

The ISO/TS 19103 Measure type and its subtypes (Length, Scale, Angle and Distance) are represented in XML schema by the corresponding ISO 19136 XML types as shown in Figure 60.

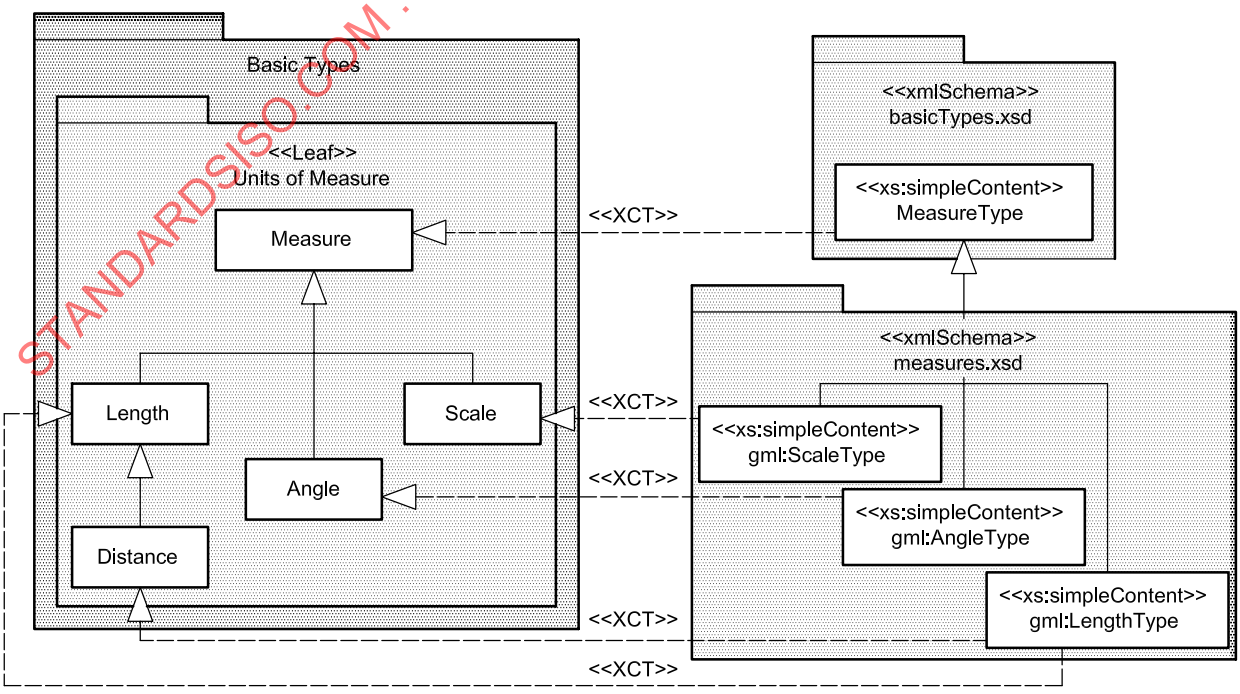

Figure 60 — Measure data types implemented using ISO 19136 XML types

The ISO/TS 19103 UnitOfMeasure type and its subtypes (UomLength, UomScale, UomAngle, UomArea, UomTime, UomVelocity and UomVolume) are represented in XML schema by a single XML type from ISO 19136 as shown in Figure 61.

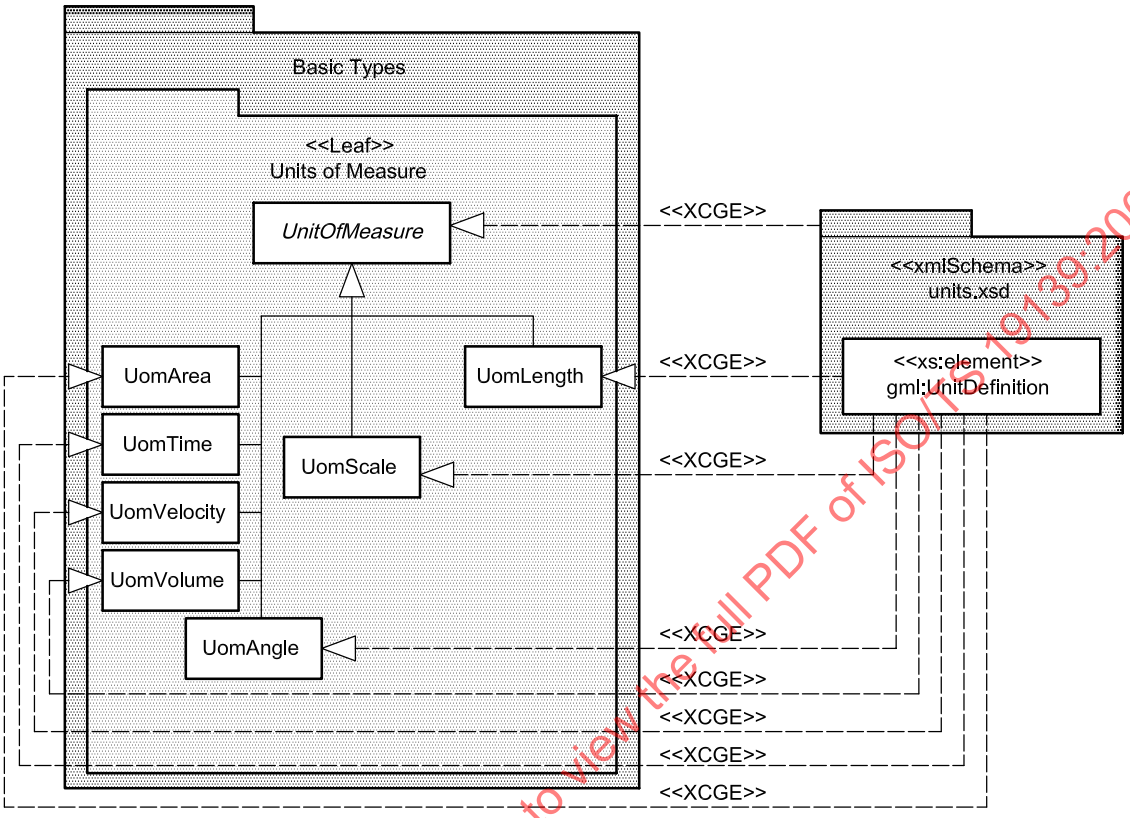

Figure 61 — Unit of measure data types implemented using ISO 19136 XML types

9.7.4.1.4 ISO/TS 19103 classes not represented using existing XML types, external XML types or default encoding rules

a) Number:

According to ISO/TS 19103, the Number abstract class is the root of all the numerical types. For the sake of consistency with the ISO/TS 19103 UML representation, Number is only represented in XML schema by an XCPT (Number_PropertyType). This XCPT is built as a choice block containing references to gco:Integer, gco:Decimal and gco:Real as shown in Figure 62.

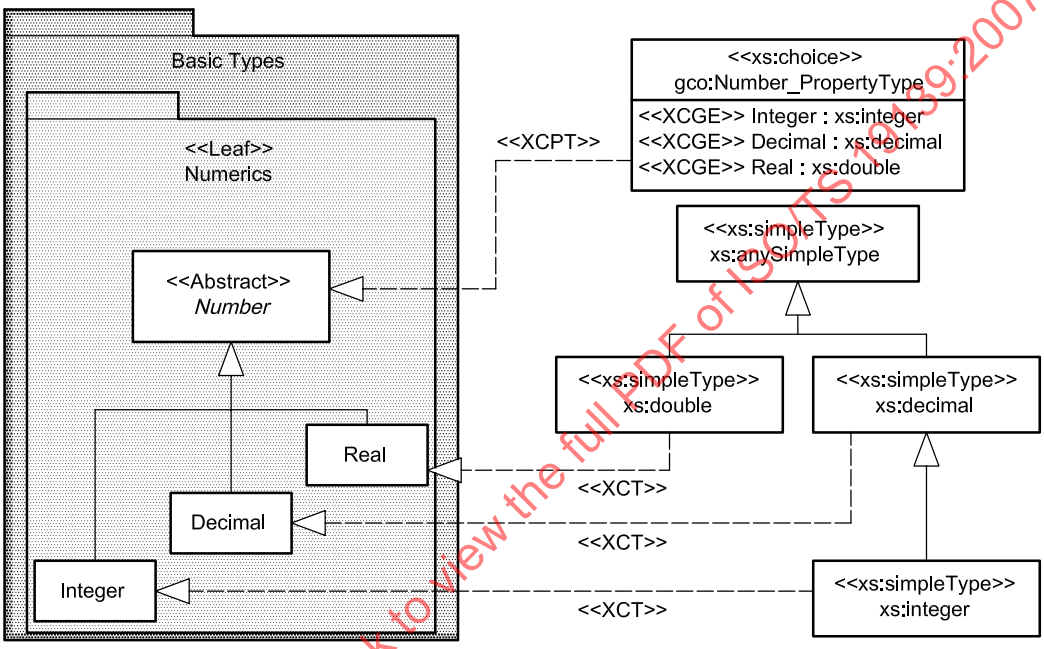

Figure 62 — Number data type represented in XML schema

b) UnlimitedInteger:

An automatic encoding of the ISO/TS 19103 UnlimitedInteger class provides an XML schema fragment which is not completely compliant with the ISO/TS 19103 definition of that class. First, it may be surprising to find the value "*" instead of an integer value in an XML instance document and, according to ISO/TS 19103, if the isInfinite property is true the value property must be empty. As a result, the encoding mechanism used for UnlimitedInteger is based on the usage of XCPT (and thus of XCGE) in XML documents and the xs:nillable attribute as shown in Figure 63.

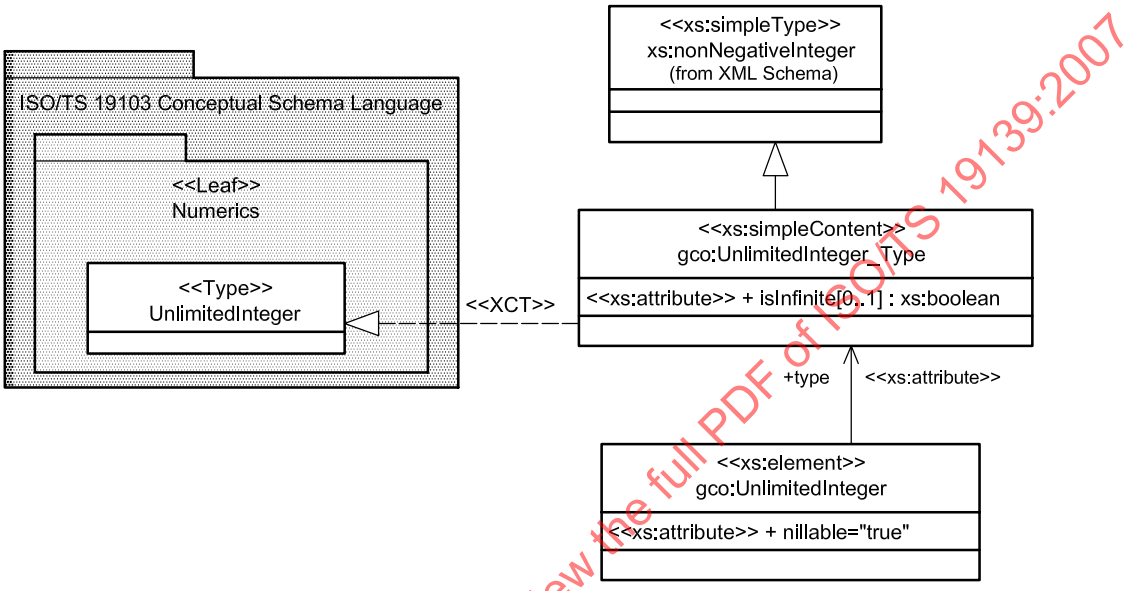

Figure 63 — UnlimitedInteger represented in XML schema

EXAMPLE Example of use:

XML schema usage	`<xs:complexType name="SAMPLE">`
	`  <xs:sequence>`
	`  <xs:element name="myUnlimitedInteger" type="gco:UnlimitedInteger_PropertyType" maxOccurs="unbounded"/>`
	`  </xs:sequence>`
	`</xs:complexType>`
	`<!--.....-->`
	`<xs:element name="sample" type="gcXML:SAMPLE"/>`

XML instance example	`<sample>`
	`  <!--=== bounded value: a positive integer ===-->`
	`  <myUnlimitedInteger>`
	`  <UnlimitedInteger>233332</UnlimitedInteger>`
	`  </myUnlimitedInteger>`
	`  <!--=== infinite value (without content) ===-->`
	`  <myUnlimitedInteger>`
	`  <UnlimitedInteger isInfinite="true" xsi:nil="true"/>`
	`  </myUnlimitedInteger>`
	`</sample>`
	NOTE xsi:nil="true" means that the content of the element is empty.

c) Date and DateTime:

ISO 19118 directs XML encodings to use the XML schema types for Date and DateTime as defined in ISO/TS 19103, stating that both types have a canonical encoding according to ISO 8601. However, there is a fundamental difference between the Date and DateTime described in ISO/TS 19103 and the xs:date and xs:dateTime of XML schema. In ISO/TS 19103 DateTime is a subtype of Date, which means that it inherits all the attributes of Date and can be used as an alternative data type wherever Date is the specified data type for an attribute. The XML schema implementation of xs:date and xs:dateTime does not have an equivalent model and as such xs:dateTime can NOT be used as the data type of attributes (or XML elements) defined with the data type of date.

It is necessary to deviate from the recommendations of ISO 19118 to properly encode in XML schema the concepts captured in ISO/TS 19103 for Date and DateTime. The specialized encoding of Date and DateTime includes an XCGE and XCPT for DateTime that matches the encoding rules stated in Clause 8. Date is encoded by: an XCT that uses an xs:union with memberTypes of xs:date, xs:gYearMonth and xs:gYear; an XCGE that is a type of XCT; and an XCPT that has a choice element containing reference elements to either the XCGE of the Date or the XCGE of the DateTime. This encoding is illustrated in Figure 64.

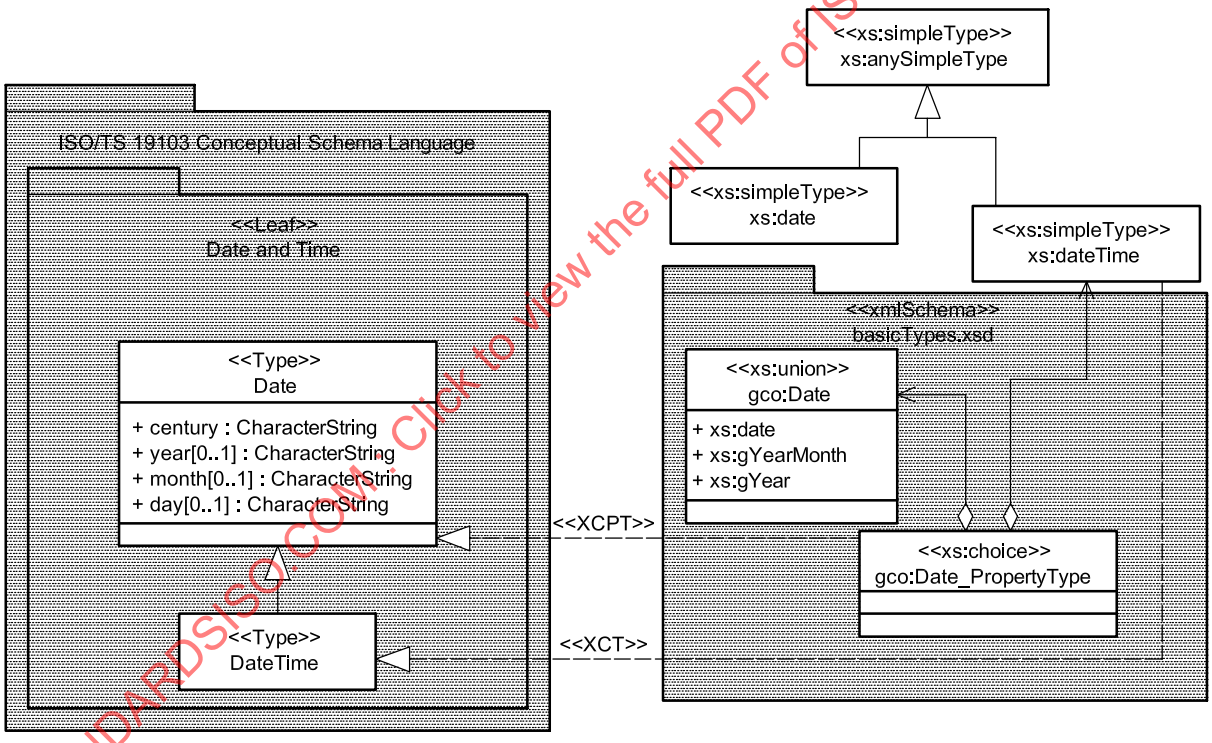

Figure 64 — Date and DateTime represented in XML schema

d) Record and RecordType:

It is not appropriate to include a full description of Record and RecordType in this Technical Specification but it is important to recognize a few characteristics of these classes. A RecordType is the physical expression of a semantic definition (typically a feature type). A Record physically expresses an instance of the semantic definition corresponding to its RecordType. The attribute MemberQualifier enforces the fact that a Record and its corresponding RecordType have the same semantic definition. The XML encoding that properly represents the Record and RecordTypes without creating the complex tag sequences or cumbersome XML instance documents that would result from following the rules in Clause 8 is shown in Figure 65.

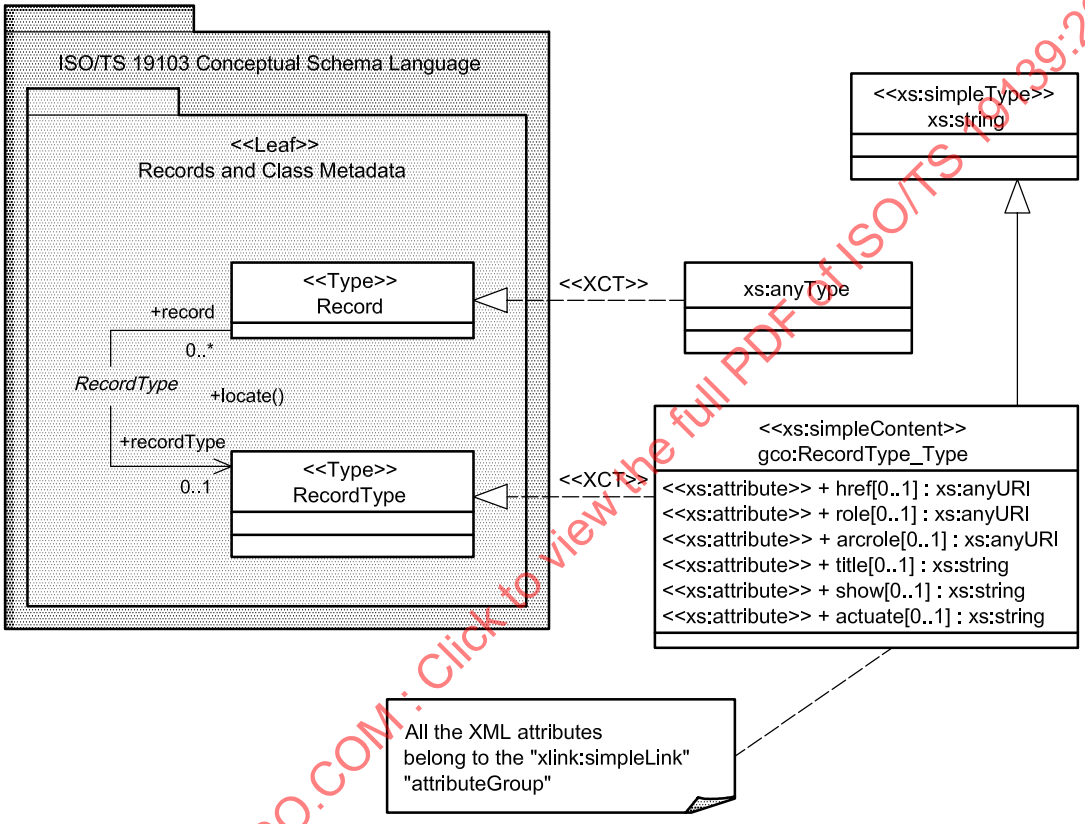

Figure 65 — Record and RecordType represented in XML schema

EXAMPLE Examples of use:

Record	
XML schema usage	`<xsd:element name="value" type="gco:Record_PropertyType"/>`

XML instance examples	`<value>`
	`  <gco:Record xsi:type="gml:PointType">`
	`  <gml:pos/>`
	`  (...)`
	`  <gco:Record>`
	`</value>`

	`<value xlink:href="aXMLFile.xml#myPointId"/>`

RecordType	
XML schema usage	`<xs:element name="valueType" type="gco:RecordType_PropertyType"/>`
XML instance examples	`<valueType> <gco:RecordType xlink:href="myXMLSchema.xsd#xpointer(//complextype [@name='myType'])">myTypeName</gco:RecordType> </valueType>`

9.8 gmx namespace

9.8.1 Organization of the gmx namespace

The namespace for the extension elements described in Clause 7 is http://www.isotc211.org/2005/gmx and the common prefix used to refer to that namespace is gmx, which stands for geographic metadata XML schema. It contains the declaration of XML types needed to create and handle XML metadata files (e.g. Registers) and extended types (e.g. FileName). The root of this namespace is gmx.xsd and the organization of the namespace is shown in Figure 66.

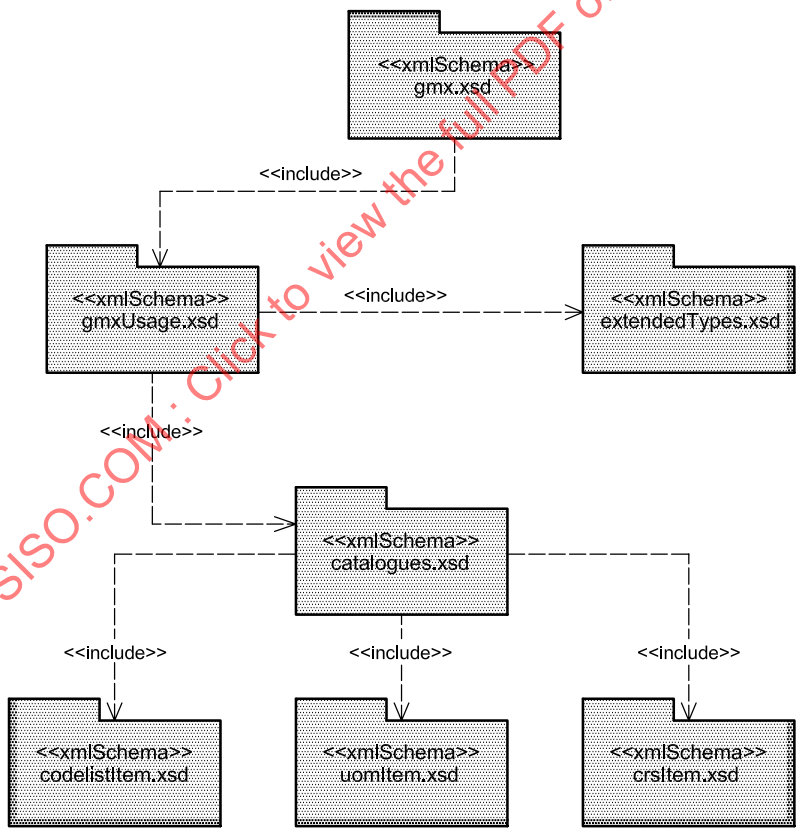

Figure 66 — Organization of gmx namespace

9.8.2 gmx.xsd

This XML schema includes all the implemented concepts of the gmx namespace, but does not contain the declaration of any types. It is the root of the gmx namespace.

9.8.3 extendedTypes.xsd

This XML schema contains the definition of FileName, Anchor and MimeFileType. These classes are fully described in 7.2, and their XML implementation is represented in Figure 67.

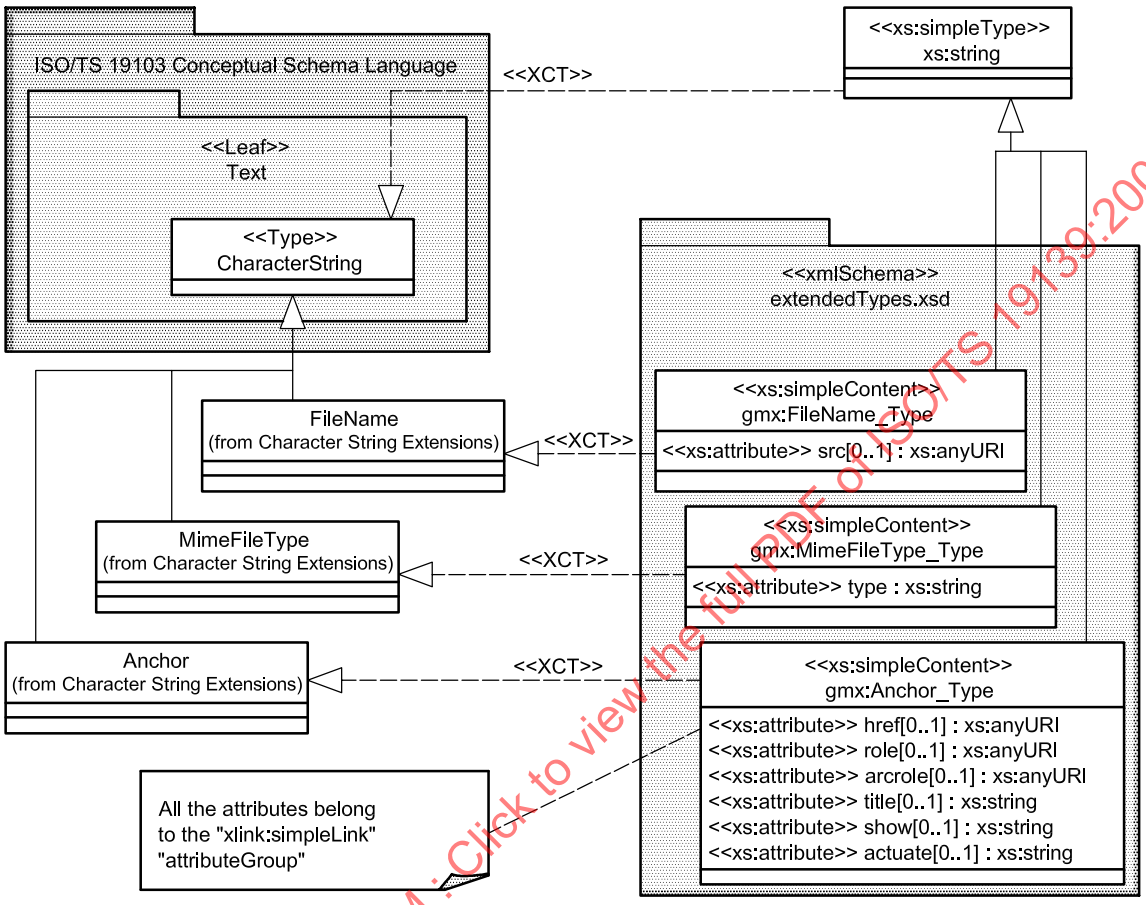

Figure 67 — FileName, Anchor and MimeFileType data types implemented in XML schema

9.8.4 gmxUsage.xsd

This XML schema implements the UML conceptual schema defined in 7.4.1. It contains the implementation of the following classes: MX_Dataset, MX_Aggregate, MX_DataFile and MX_ScopeCode.

The classes implemented in this XML schema follow the encoding rules described in Clause 8.

9.8.5 catalogues.xsd

This XML schema implements the UML conceptual schema defined in 7.4.4.1. It contains the implementation of CT_Catalogue, CT_CodelistCatalogue, CT_UomCatalogue and CT_CrsCatalogue.

The classes implemented in this XML schema follow the encoding rules described in Clause 8.

9.8.6 Multilingual capabilities

Multilingual capabilities are supported by gmx catalogues. XML types for multilingual catalogue items are defined in the following XML schemas: CodeListItem.xsd, UomItem.xsd and CrsItem.xsd. This subclause aims at presenting the basics of the extension of regular data models (and more specifically ISO 19136 dictionaries and derived dictionaries) to create new types for multilingual support.

To support cultural and linguistic adaptability (CLA), a multilingual structure must be created in parallel to the main register data structure, concretely:

a) each leaf element derived from a *gml:Definition* has to be extended (XML derivation by extension) to create a new concept, with the same definition plus an *alternativeExpression* property that carries the description of alternative (multilingual) values;

NOTE 1 Only the non-abstract components are extended.

b) the type of *alternativeExpression* property is defined as follows: it derives by extension from *gml:Definition* (or one of its subtypes) and holds a *locale* property of type *gmd:PT_Locale*; this locale property is designed to contain a reference to the specific locale in which the alternative expression is expressed.

NOTE 2 *alternativeExpression* is a support mechanism for multilingual items, thus, except for the property *locale*, *alternativeExpression* instances shall only contain the translation of textual fields whose domain is "freeText".

9.8.7 codelistItem.xsd

This XML schema implements the UML conceptual schema defined in 7.4.4.4. It contains the implementation of CT_Codelist and CT_CodelistValue.

The encoding of CT_Codelist and CT_CodelistValue is mapped to XML types for implementation also declared in this schema (gmx:CodeDefinition, gmx:CodeListDictionary). These implementation types are derived from ISO 19136 dictionary types and complete ISO 19136 schemas which do not define any content model for codelists. Figure 68 illustrates the encoding.

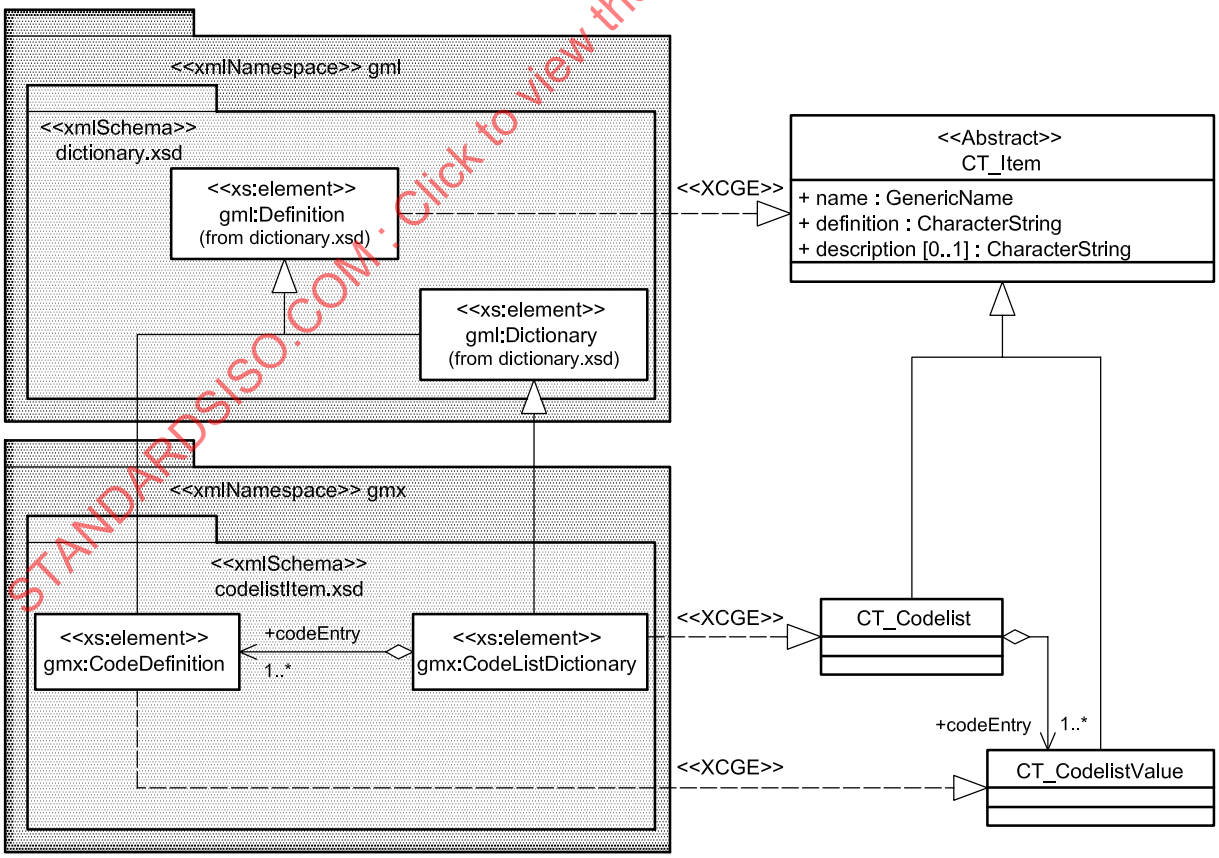

Figure 68 — Codelist items — XML implementation

gmx:CodeDefinition and gmx:CodeListDictionary are extended to support multilingual items. The new multilingual types gmx:ML_CodeDefinition and gmx:ML_CodeListDictionary follow the regular extension mechanism described in 9.8.6. The codelistItem.xsd schema also contains the declaration of gmx:CIAlternativeExpression and gmx:CodeAlternativeExpression, which specialize the type of the “alternativeExpression” property. Figure 69 illustrates the encoding.

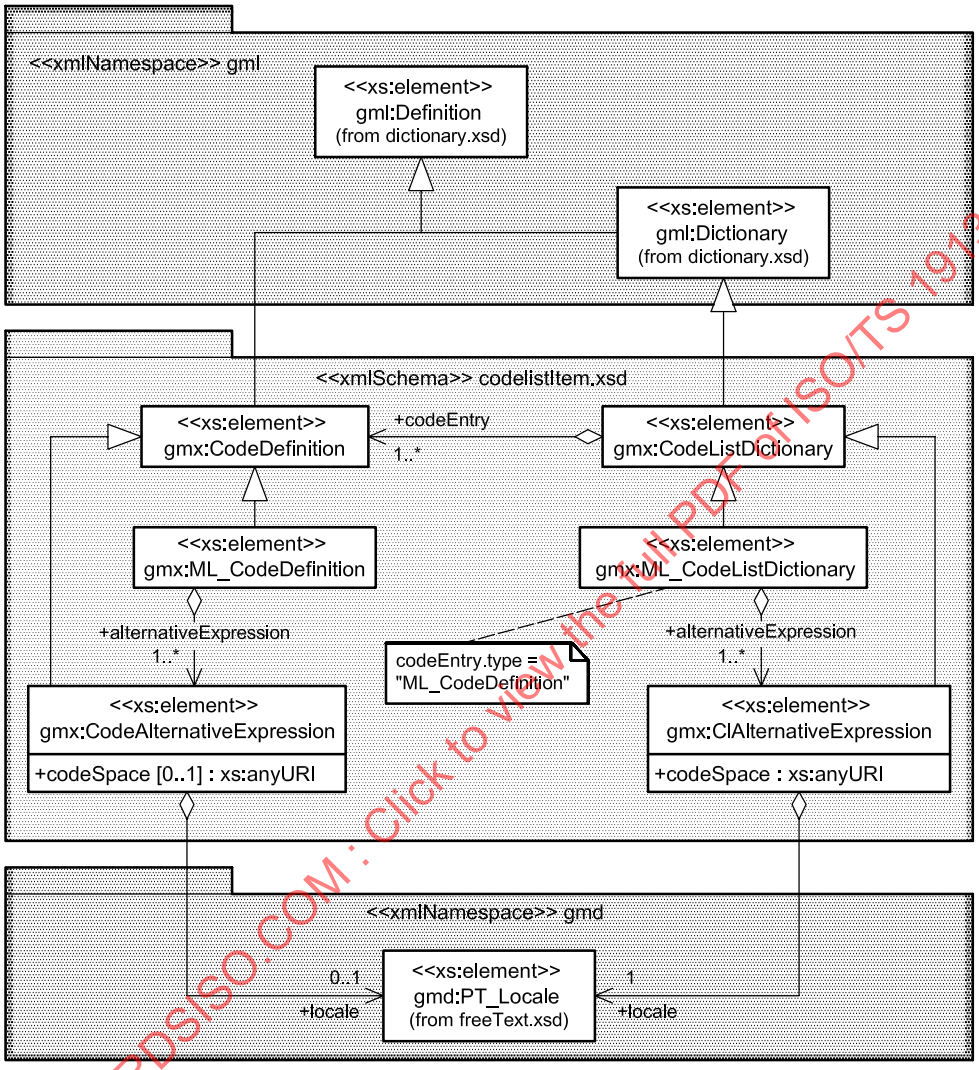

Figure 69 — Codelist items — Multilingual support

9.8.8 uomItem.xsd

This XML schema implements the UML conceptual schema defined in 7.4.4.2. It contains the implementation of the UnitDefinition class. Figure 70 illustrates the encoding.

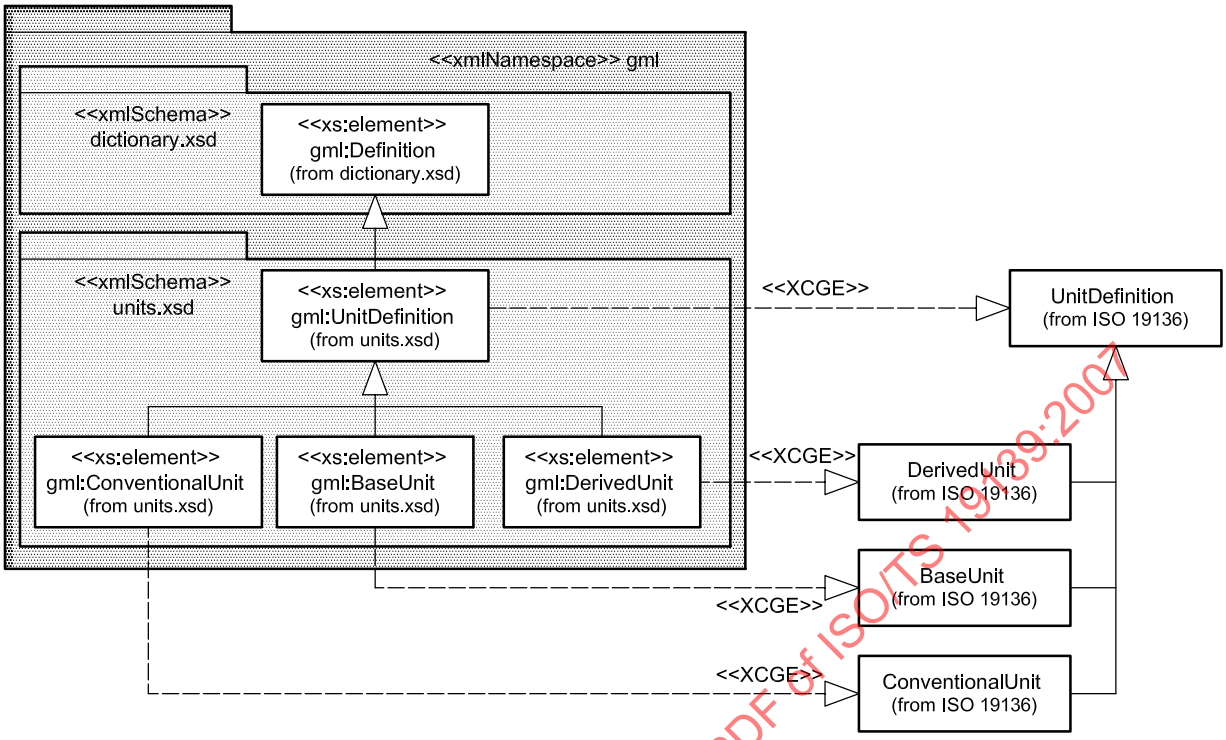

Figure 70 — UoM catalogue — XML implementation

gml:UnitDefinition, gml:BaseUnit, gml:DerivedUnit and gml:ConventionalUnit are extended to support multilingual items. The new multilingual types gmx:ML_CodeDefinition and gmx:ML_CodeListDictionary follow the regular extension mechanism described in 9.8.6. The uomItem.xsd schema also contains the declaration of gmx:UomAlternativeExpression, which specializes the type of the “alternativeExpression” property. Figure 71 illustrates the encoding.

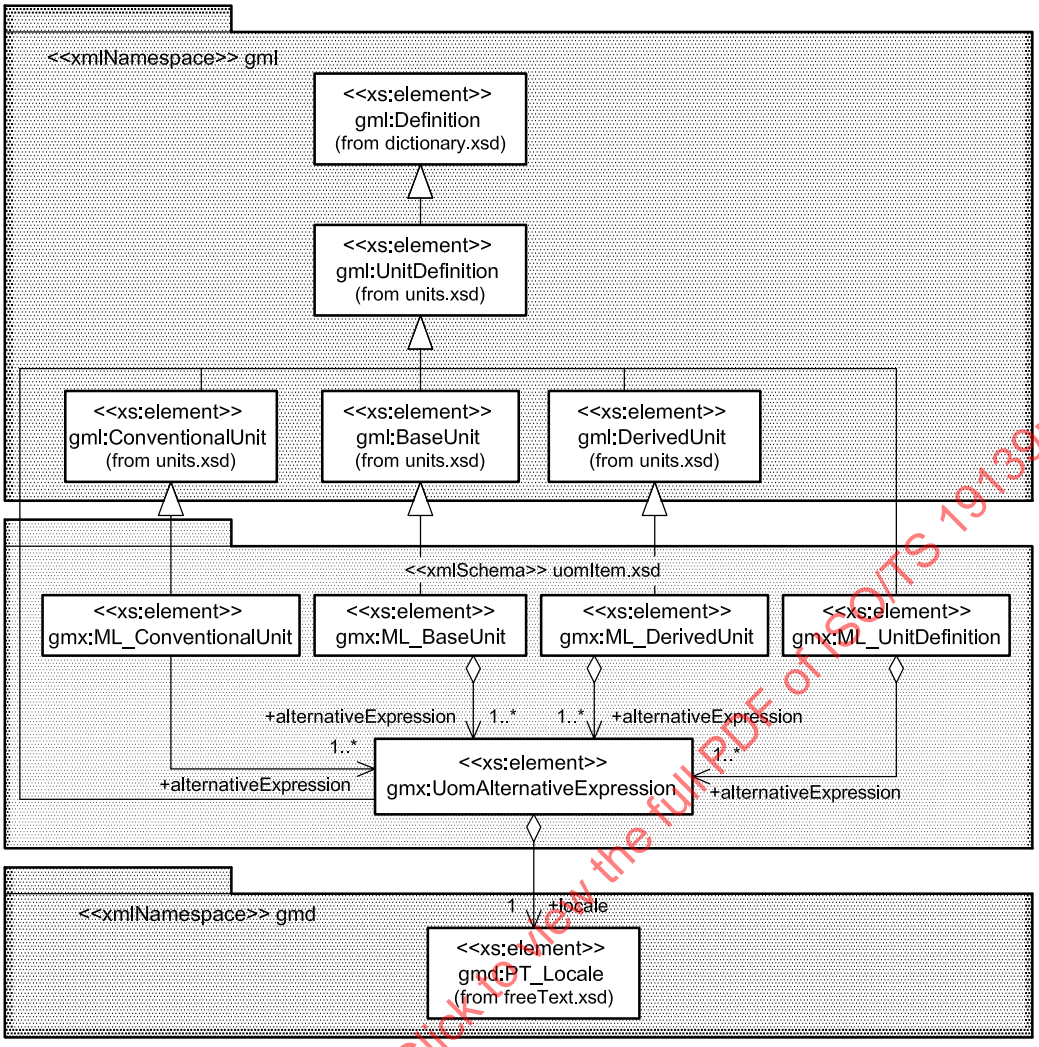

Figure 71 — Uom items — Multilingual support

9.8.9 crsItem.xsd

This XML schema implements the UML conceptual schema defined in 7.4.4.3. It contains the implementation of CT_CRS, CT_CoordinateSystem, CT_CoordinateSystemAxis, CT_Datum, CT_Ellipsoid, CT_PrimeMeridian, CT_Operation, CT_OperationMethod and CT_OperationParameters.

Table 4 presents the XML implementation of each of these classes:

— its basic implementation as an ISO 19136 XCGE;

— the ISO 19136 concrete forms of its ISO 19136 XCGE, i.e. the ISO 19136 substitute concrete global elements of its ISO 19136 XCGE;

— the multilingual subtype of each of its concrete forms;

— the alternative expressions of the multilingual subtypes.

Table 4 — gmx:ML_xxx elements in crsItem.xsd

Group	ISO 19136 XCGE	ISO 19136 concrete forms	Multilingual subtype	Alternative expression
CT_CRS	gml:AbstractCRS	gml:CompoundCRS	gmx:ML_CompoundCRS	gmx:CrsAlt
		gml:EngineeringCRS	gmx:ML_EngineeringCRS	
		gml:VerticalCRS	gmx:ML_VerticalCRS	
		gml:GeodeticCRS	gmx:ML_GeodeticCRS	
		gml:TemporalCRS	gmx:ML_TemporalCRS	
		gml:ImageCRS	gmx:ML_ImageCRS	
		gml:ProjectedCRS	gmx:ML_ProjectedCRS	
		gml:DerivedCRS	gmx:ML_DerivedCRS	
CT_Coordinate System	gml:AbstractCoordinate System	gml:EllipsoidalCS	gmx:ML_EllipsoidalCS	gmx:CoordinateSystemAlt
		gml:CartesianCS	gmx:ML_CartesianCS	
		gml:AffineCS	gmx:ML_AffineCS	
		gml:UserDefinedCS	gmx:ML_UserDefinedCS	
		gml:VerticalCS	gmx:ML_VerticalCS	
		gml:TimeCS	gmx:ML_TimeCS	
		gml:CylindricalCS	gmx:ML_CylindricalCS	
		gml:SphericalCS	gmx:ML_SphericalCS	
		gml:PolarCS	gmx:ML_PolarCS	
		gml:LinearCS	gmx:ML_LinearCS	
CT_Coordinate SystemAxis	gml:CoordinateSystemAxis	gml:CoordinateSystemAxis	gmx:ML_CoordinateSystemAxis	gmx:CoordinateSystem AxisAlt
CT_Datum	gml:AbstractDatum	gml:TemporalDatum	gmx:ML_TemporalDatum	gmx:DatumAlt
		gml:VerticalDatum	gmx:ML_VerticalDatum	
		gml:ImageDatum	gmx:ML_ImageDatum	
		gml:EngineeringDatum	gmx:ML_EngineeringDatum	
		gml:GeodeticDatum	gmx:ML_GeodeticDatum	
CT_Ellipsoid	gml:Ellipsoid	gml:Ellipsoid	gmx:ML_Ellipsoid	gmx:EllipsoidAlt
CT_Prime Meridian	gml:PrimeMeridian	gml:PrimeMeridian	gmx:ML_PrimeMeridian	gmx:PrimeMeridianAlt
CT_Operation	gml:AbstractCoordinate Operation	gml:Concatenated Operation	gmx:ML_Concatenated Operation	gmx:OperationAlt
		gml:PassThroughOperation	gmx:ML_PassThroughOperation	
		gml:Transformation	gmx:ML_Transformation	
		gml:Conversion	gmx:ML_Conversion	
CT_Operation Method	gml:OperationMethod	gml:OperationMethod	gmx:ML_OperationMethod	gmx:OperationMethodAlt
CT_Operation Parameter	gml:AbstractGeneral OperationParameter	gml:OperationParameter Group	gmx:ML_OperationParameter Group	gmx:OperationParameter Alt
		gml:OperationParameter	gmx:ML_OperationParameter	

9.9 From the conceptual schema to XML file instances

9.9.1 Introduction

Due to the envisioned usage of the geographic metadata XML schema, it is fundamental to keep the organization of the data, its associated metadata and the related information in very flexible files. It is very important to understand that the MD_Metadata XCGE will rarely be the root element of an XML file, but depending on the context it may appear one or many times in a single XML file describing one or many different types of resources.

It is even possible to have an XML file containing a metadata set without containing a single MD_Metadata XCGE element. This is a consequence of polymorphism, which may imply that an XCGE of an MD_Metadata subclass, potentially defined in a user community profile, occurs instead of the MD_Metadata XCGE element. This is true for MD_Metadata as well as for any of the concepts defined in the ISO 19100 series of International Standards. In order to accommodate this characteristic and ensure the understanding of user-profiled metadata sets, a specific requirement has been expressed in A.3. The XCT of any new metadata element shall support a mandatory XML attribute called isoType that is expected to contain the name of the ISO class from which it derives directly or indirectly.

9.9.2 In the context of a *gml* document or data access service using *gml* as an exchange standard

ISO 19136 defines an XML type named `gml:AbstractMetadataPropertyType` which is dedicated to metadata and quality-related attributes. Consider a Feature Type named **Road** having a metadata attribute named **metadata** of type **MD_Metadata**. This attribute has been defined as an XML element locally to the ISO 19136 definition of the feature type.

```
<xs:element name="metadata">
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="gml:AbstractMetadataPropertyType">
         <xs:sequence minOccurs="0">
           <xs:element ref="gmd:MD_Metadata"/>
         </xs:sequence>
      </xs:extension>
      <attributeGroup ref="gml:AssociationAttributeGroup"/>
    </xs:complexContent>
  </xs:complexType>
<xs:element>
```

Here is a resulting sample XML file:

```
<myAs:Road>
   <!-- other Road Properties -->
  <myAs:metadata>
    <gmd:MD_Metadata>
      <!-- a full set of ISO/TS 19139 metadata elements -->
    </gmd:MD_Metadata>
  </myAs:metadata>
  <!-- other Road Properties -->
</myAs:Road>
```

9.9.3 In the context of a catalogue service

When the data being passed through a cataloguing service is XML encoded, the catalogue service interface defines the different XML schemas to be used as a response to the user queries. When the geographic metadata XML schema is used, there should be one or many MD_Metadata instances in the returned XML file.

9.9.4 In the context of the standard interchange by transfer

The transfer aggregate and transfer dataset concepts are the two major components of an interchange by transfer. There may be one or many XML files composing the interchange, but the root element of at least one of the files is an XML instance of MX_Dataset, MX_Aggregate or one of their extensions. From such an element, the parsing of the interchange is model-driven and it follows the principles described in 7.4.

Annex A
(normative)

Abstract test suite

A.1 Overview of conformance test tools

Minimum conformance with this Technical Specification requires that geographic metadata instance (XML) documents can be validated without error against the XML schemas defined by the encoding rules in Clause 8 and described in detail in Clause 9. While many tools are available to test validation of XML instance documents against provided XML schemas, it is important to understand that not all validation tools implement the full W3C XML schema recommendation and not all validation tools interpret the W3C XML schema recommendation in the same manner. It is recommended that a tool with strict interpretation of XML schema and full support for the W3C XML schema recommendation be used to ensure conformance.

A.2 Conformance requirements — Constraints

A.2.1 By-value or by-reference or gco:nilReason

Validation of XML instance documents against the schemas described in this Technical Specification is not all that is required for conformance. As stated in 8.4, a property element following the default XCPT pattern is designed to have content (by-value) or attributes (by-reference or NULL with reason). However, because of the design of the XCPT, the property element may have no content or attributes, or it may have both content and attributes and still be XML-schema-valid. It is not possible to constrain the co-occurrence of content or attributes. Some mechanism in addition to an XML schema validation (e.g. Schematron, XSL transformations) shall be used to restrict a property to be exclusively by-value, or by-reference, or expressing a NULL reason.

A.2.2 Co-constraints

XML 1.0 does not support the enforcement of certain types of constraints. For example, co-constraints such as the requirement that an 'extent' in the form of an 'EX_GeographicBoundingBox' or 'EX_GeographicDescription' be used in the 'MD_DataIdentification' object when the 'hierarchyLevel' of 'MD_Metadata' is equal to "dataset" cannot be enforced with an XML schema. As a result, it is imperative that implementers heed the anchor notes shown in the Figures in Annex A of ISO 19115:2003 and ISO 19115:2003/Cor.1:2006. Table A.1 shows the affected elements and the anchor notes and annotations. Due to the inheritance design of XML schema some inheritance principles are not enforced but need to be properly recognised. These are also listed in Table A.1.

Table A.1 — Conformance rules not enforceable with XML schema

Affected class	Conformance rule
MD_Metadata	language: documented if not defined by the encoding standard
MD_Metadata	characterSet: documented if ISO/IEC 10646 not used and not defined by the encoding standard
MD_DataIdentification	characterSet: documented if ISO/IEC 10646 is not used
MD_DataIdentification	MD_Metadata.hierarchyLevel = “dataset” implies count (extent.geographicElement.EX_GeographicBoundingBox) + count (extent.geographicElement.EX_GeographicDescription) >=1
MD_DataIdentification	MD_Metadata.hierarchyLevel notEqual “dataset” implies topicCategory is not mandatory
MD_AggregateInformation	Either “aggregateDataSetName” or “aggregateDataSetIdentifier” must be documented
MD_LegalConstraints	otherConstraints: documented if accessConstraints or useConstraints = “otherRestrictions”
DQ_DataQuality	“report” or “lineage” role is mandatory if scope.DQ_Scope.level = ‘dataset’
DQ_Scope	“levelDescription” is mandatory if “level” notEqual ‘dataset’ or ‘series’
LI_Lineage	If (count(source) + count(processStep) =0) and (DQ_DataQuality.scope.level = ‘dataset’ or ‘series’) then statement is mandatory
LI_Lineage	“source” role is mandatory if LI_Lineage.statement and “processStep” role are not documented
LI_Lineage	“processStep” role is mandatory if LI_Lineage.statement and “source” role are not documented
LI_Source	“description” is mandatory if “sourceExtent” is not documented
LI_Source	“sourceExtent” is mandatory if “description” is not documented
MD_Georectified	“checkPointDescription” is mandatory if “checkPointAvailability” = 1
MD_Band	“units” is mandatory if “maxValue” or “minValue” are provided
MD_Medium	“densityUnits” is mandatory if “density” is provided
MD_Distribution / MD_Format	count (distributionFormat + distributorFormat) > 0"
MD_ExtendedElementInformation	if “dataType” notEqual ‘codelist’, ‘enumeration’ or ‘codelistElement’ then “obligation”, "maximumOccurence" and “domainValue” are mandatory
MD_ExtendedElementInformation	if “obligation” = ‘conditional’ then “condition” is mandatory
MD_ExtendedElementInformation	if “dataType” = ‘codelistElement’ then “domainCode” is mandatory
MD_ExtendedElementInformation	if “dataType” notEqual ‘codelistElement’ then “shortName” is mandatory
EX_Extent	count (description + geographicElement + temporalElement + verticalElement) > 0")
CI_ResponsibleParty	count of (individualName + organisationName + positionName) > 0")
Distance	The UoM element of the Distance Type must be instantiated using the UomLength_PropertyType
Length	The UoM element of the Length Type must be instantiated using the UomLength_PropertyType
Scale	The UoM element of the Scale Type must be instantiated using the UomScale_PropertyType
Angle	The UoM element of the Angle Type must be instantiated using the UomAngle_PropertyType

A.3 Conformance requirements — Extensions

User profiles and extensions are described for metadata in Annex C of ISO 19115:2003. The following rules define conformance for the different types of extensions described in C.2 of ISO 19115:2003.

1) Adding a new metadata section: any new metadata sections should be added in their own namespace following the encoding rules described in Clause 8.

2) Creating a new metadata codelist to replace the domain of an existing metadata element that has "free text" listed as its domain value: new codelists can be implemented by following the encoding rules described in 8.5.5. By following these guidelines, the new codeList will be a substitution group for an existing metadata element with "free text" listed as its domain value and CharacterString as its data type.

3) Creating new metadata codelist elements (expanding a codelist): adding new elements in the codelist registry, as described in 9.8.7, provides conformance for this type of extension.

4) Adding a new metadata element: new elements may not be added directly to the XML schemas defined by this Technical Specification. Any new elements can be added via subclassing existing ISO 19100 series classes and following the guidelines described in 8.5.3 for encoding subclasses. In addition, the XCT of the extended classes shall handle an additional mandatory attribute *isoType* of type xs:string defined in gcoBase.xsd. This attribute is expected to contain the name of the ISO class it extends directly or indirectly. It provides an efficient means to parse any user community metadata XML file, looking for either ISO/TS 19139 elements or elements whose *isoType* attribute contains an ISO class name. The resulting schema will not reside in one of the namespaces defined in this Technical Specification.

5) Adding a new metadata entity: any new metadata entities should be added in their own namespace following the encoding rules described in Clause 8.

6) Imposing a more stringent obligation on an existing metadata element: restriction conformance is defined in A.4.

7) Imposing a more restrictive domain on an existing metadata entity: restriction conformance is defined in A.4.

In addition to following the guidelines above it is also necessary to follow the conformance rules described for metadata in 19115:2003, Annex C, related to the proper analysis of the requirement for the extension and the resulting documentation to describe the extension.

A.4 Conformance requirements — Restrictions

There are certain cases when it is desirable to restrict an existing XML schema. One case is when utilizing external implementations as described in 8.5.8 and illustrated in various figures in Clause 9 where the details of the encodings reside. An external implementation might be perfectly suitable to achieve the requirements of this Technical Specification with the exception of needing stricter restriction. This is the case when using the gml:DefinitionType as the super class of gmx:AlternativeExpression. As shown in a gml:DefinitionType the *id* attribute is optional, but in gmx:AlternativeExpression it is desirable to make this attribute mandatory. This is not done by modifying the gml:DefinitionType, but instead by adding an annotation to the UML such as the note, "XML attribute: "id" is Mandatory" and attaching the annotation to the gmx:AlternativeExpression class. Some tool in addition to an XML schema validator must be used to enforce the mandatory condition on the *id* attribute of a gmx:AlternativeExpression.

Just as restriction of external implementations is done through annotation in UML and enforced via a tool other than an XML schema validator in the namespaces defined in this Technical Specification, any user profile restrictions should also use this restriction mechanism.

Annex B
(normative)

Data dictionary for extensions

B.1 Data dictionary overview

B.1.1 Introduction

This data dictionary describes the characteristics of the extensions of ISO 19115 defined in Clause 7. The dictionary is specified in a hierarchy to establish relationships and organization for information. The dictionary is categorized into sections corresponding to a type of extension: web environment, cultural and linguistic adaptability and standard interchange of geospatial information by transfer.

The clause titles of several of the tables have been expanded to reflect class specification within the respective diagram. Each UML model class equates to a data dictionary entity. Each UML model class attribute equates to a data dictionary element. The shaded rows define entities. The entities and elements within the data dictionary are defined by seven attributes (those attributes are listed in B.1.2 to B.1.7 and are based on those specified in ISO/IEC 11179-3 for the description of data element concepts, i.e. data elements without representation).

B.1.2 Name/role name

This is a label assigned to a metadata entity or to a metadata element. Metadata entity names start with an upper case letter. Spaces do not appear in a metadata entity name. Instead, multiple words are concatenated, with each new sub-word starting with a capital letter (example: XnnnYmmm). Metadata entity names are unique within the entire data dictionary of this Technical Specification. Metadata element names are unique within a metadata entity, not the entire data dictionary of this Technical Specification. Metadata element names are made unique, within an application, by the combination of the metadata entity and metadata element names (example: MD_Metadata.characterSet). Role names are used to identify metadata abstract model associations and are preceded by “Role name:” to distinguish them from other metadata elements. Names and role names may be in a language other than that used in this Technical Specification.

B.1.3 Definition

This is the metadata entity/element description.

B.1.4 Obligation/condition

B.1.4.1 General

This is a descriptor indicating whether a metadata entity or metadata element shall always be documented in the metadata or sometimes be documented [i.e. contains value(s)]. This descriptor may have the following values: M (mandatory), C (conditional) or O (optional).

B.1.4.2 Mandatory (M)

The metadata entity or metadata element shall be documented.

B.1.4.3 Conditional (C)

This specifies an electronically manageable condition under which at least one metadata entity or a metadata element is mandatory. "Conditional" is used for one of the three following possibilities:

— expressing a choice between two or more options; at least one option is mandatory and must be documented;

— documenting a metadata entity or a metadata element if another element has been documented;

— documenting a metadata element if a specific value for another metadata element has been documented; to facilitate reading by humans, the specific value is used in plain text; however, the code shall be used to verify the condition in an electronical user interface.

If the answer to the condition is positive, then the metadata entity or the metadata element shall be mandatory.

B.1.4.4 Optional (O)

The metadata entity or the metadata element may or may not be documented. Optional metadata entities and optional metadata elements have been defined to provide a guide to those looking to fully document their data. (Use of this common set of defined elements will help promote interoperability among geographic data users and producers world-wide.) If an optional entity is not used, the elements contained within that entity (including mandatory elements) will also not be used. Optional entities may have mandatory elements; those elements only become mandatory if the optional entity is used.

B.1.5 Maximum occurrence

This specifies the maximum number of instances the metadata entity or the metadata element may have. Single occurrences are shown by “1”; repeating occurrences are represented by “N”. Fixed number occurrences other than one are allowed, and will be represented by the corresponding number (i.e. “2”, “3”…etc).

B.1.6 Data type

This specifies a set of distinct values for representing the metadata elements; e.g. Integer, Real, String, DateTime and Boolean. The data type attribute is also used to define metadata entities, stereotypes and metadata associations.

B.1.7 Domain

For an entity, the domain indicates the line numbers covered by that entity.

For a metadata element, the domain specifies the values allowed or the use of free text. “Free text” indicates that no restrictions are placed on the content of the field. Integer-based codes shall be used to represent values for domains containing codelists.

B.2 Metadata extension data dictionaries

B.2.1 Web environment extensions

B.2.1.1 Anchor

	Name/role name	Definition	Obligation/ condition	Maximum occurrence	Data type	Domain
1	Anchor	Supports hyper-linking capabilities and ensures a web-like implementation of CharacterStrings	Use obligation from referencing object	Use maximum occurrence from referencing object	Class	Line 2
2	href	Supplies the data that allow an XLink application to find a remote resource (or resource fragment) [W3C XLINK]	M	1	URI	

B.2.1.2 File name

	Name/role name	Definition	Obligation/ condition	Maximum occurrence	Data type	Domain
3	FileName	Supports explicitly referencing an external file corresponding to a property containing the name of the file	Use obligation from referencing object	Use maximum occurrence from referencing object	Class	Line 4
4	src	Provides a machine-readable path to the location of a corresponding file	M	1	URI	

B.2.1.3 Mime file type

	Name/role name	Definition	Obligation/ condition	Maximum occurrence	Data type	Domain
5	MimeFile Type	Supports identification of the file type using the mime media type name and subtype name	Use obligation from referencing object	Use maximum occurrence from referencing object	Class	Line 6
6	fileType	Provides the mime media type name and subtype name	M	1	CharacterString	

B.2.2 Cultural and linguistic adaptability extensions

B.2.2.1 Free text

	Name/role name	Definition	Obligation/ condition	Maximum occurrence	Data type	Domain
7	PT_Free Text	Multi-language free text data type. A metadata element whose data type is CharacterString and domain is free text can be alternatively expressed using the PT_FreeText subtype of CharacterString. A free text instance acts as a normal character string except that it handles complementary translations of the character string value in different locales.	Use obligation from referencing object	Use maximum occurrence from referencing object	Specified class (Character String)	Line 8
8	*Role name*: textGroup	Provides the list of localized character strings each expressing the free text value (sequence of characters) in a given locale	M	N	Association	Localised Character String
9	Localised Character String	Expression of a free text in a given locale	Use obligation from referencing object	Use maximum occurrence from referencing object	Specified class (Character String)	Line 10
10	*Role name:* locale	Defines the locale in which the value (sequence of characters) of the localized character string is expressed	M	1	Class	PT_Locale
11	PT_Locale	Description of a locale	Use obligation from referencing object	Use maximum occurrence from referencing object	Class	Lines 12-14
12	language	Designation of the locale language	M	1	Class	LanguageCode <<Codelist>> (ISO 639-2 3-alphabetic digits code)
13	country	Designation of the specific country of the locale language	O	1	Class	CountryCode <<Codelist>> (ISO 3166-1, other parts may be used)
14	Character Encoding	Designation of the character set to be used to encode the textual value of the locale	M	1	Class	MD_Character SetCode <<Codelist>> (See ISO 19115:2003, B.5.10)

B.2.2.2 Locale container

	Name/role name	Definition	Obligation/ condition	Maximum occurrence	Data type	Domain
15	PT_Locale Container	Container of localized character strings. It provides a mean to isolate the localised strings related to a given locale.	O	N	Class	Lines 16-20
16	description	Designation of the locale language	M	1	CharacterString	Free text
17	locale	Locale in which the localized strings of the container are expressed	M	1	PT_Locale	
18	date	Date of creation or revision of the locale container	M	N	CI_Date	
19	Responsible Party	Responsible parties of the locale container	M	N	CI_Responsible Party	
20	*Role name*: localised String	Provides the list of localized character strings expressing the linguistic translation of a set of textual information in a given locale	M	1	Association	LocalisedString

B.2.3 Standard interchange of geospatial information by transfer extension

B.2.3.1 Transfer aggregate

	Name/role name	Definition	Obligation/ condition	Maximum occurrence	Data type	Domain
21	MD_ Aggregate	Direct or indirect aggregation of datasets	Use obligation from referencing object	Use maximum occurrence from referencing object	Class	Lines 22-25
22	*Role name:* series Metadata	Provides the list of metadata sets related to the aggregate	M	N	Association	MD_Metadata
23	*Role name:* subset	Provides the list of subset aggregates composing the aggregate	O	N	Association	MD_Metadata
24	*Role name:* superset	Provides the list of superset aggregates to which the aggregate pertains	O	N	Association	MD_Metadata
25	*Role name:* composedOf	Provides the list of datasets directly composing the aggregate. The datasets composing an aggregate are not part of the list of datasets composing one of its superset aggregates	M	N	Association	MD_Metadata
26	MX_ Aggregate	Description of a transfer aggregate	Use obligation from referencing object	Use maximum occurrence from referencing object	Specified class (MD_Aggregate)	Lines 27-28
27	*Role name*: aggregate Catalogue	Provides the list of catalogues related to the transfer aggregate	O	N	Association	CT_Catalogue
28	*Role name*: aggregate File	Provides the list of support files related to the aggregate. The support files of the datasets composing the aggregate are not part of this list	O	N	Association	MX_SupportFile

B.2.3.2 Transfer dataset

	Name/role name	Definition	Obligation/ condition	Maximum occurrence	Data type	Domain
29	MD_Dataset	Description of an identifiable collection of data	Use obligation from referencing object	Use maximum occurrence from referencing object	Class	Line 30
30	*Role name*: has	Provides the list of metadata sets related to the dataset and its data	M	N	Association	MD_Metadata
31	MX_Dataset	Description of a transfer dataset	Use obligation from referencing object	Use maximum occurrence from referencing object	Specified class (MD_Dataset)	Lines 32-34
32	*Role name*: dataset Catalogue	Provides the list of catalogues related to the transfer dataset	O	N	Association	CT_Catalogue
33	*Role name*: dataFile	Provides the list of data files in which the data are stored	M	N	Association	MX_DataFile
34	*Role name*: supportFile	Provides the list of support files related to the dataset	O	N	Association	MX_SupportFile

B.2.3.3 Transfer file

	Name/role name	Definition	Obligation/ condition	Maximum occurrence	Data type	Domain
35	MX_File	Description of a transfer file	Use obligation from referencing object	Use maximum occurrence from referencing object	Class	Lines 36-38
36	fileName	Name of the transfer data file. This name is associated with a path to the physical file.	M	1	CharacterString	FileName
37	File Description	General description of the transfer data file	M	1	CharacterString	Free text
38	fileType	Type of the transfer data file. The textual description of the file type is associated with an indication of the MIME Type.	M	1	CharacterString	MimeFileType
39	MX_Data File	Description of a transfer data file	Use obligation from referencing object	Use maximum occurrence from referencing object	Specified class (MX_File)	Lines 40-41
40	featureType	Provides the list of feature types concerned by the transfer data file. Depending on the transfer choices, a data file may contain data related to one or many feature types. This attribute may be omitted when the dataset is composed of a single file and/or the data does not relate to a feature catalogue.	O	N	CharacterString	LocalName
41	*Role name*: fileFormat	Defines the format of the transfer data file	M	1	Class	MD_Format
42	MX_Support File	Description of a transfer support file	Use obligation from referencing object	Use maximum occurrence from referencing object	Specified class (MX_File)	

B.2.3.4 Catalogue

	Name/role name	Definition	Obligation/ condition	Maximum occurrence	Data type	Domain
43	CT_ Catalogue	General description of a catalogue	Use obligation from referencing object	Use maximum occurrence from referencing object	Class	Lines 44-52
44	name	Name of the catalogue	M	1	CharacterString	Free text
45	scope	Subject domain(s) of the catalogue content	M	N	CharacterString	Free text
46	fieldOf Application	Description of kind(s) of use to which the catalogue may be put	O	N	CharacterString	Free text
47	Version Number	Version Number of this catalogue, which may include both a major Version Number or letter and a sequence of minor release numbers or letters, such as "3.2.4a". The format of this attribute may differ between cataloguing authorities.	M	1	CharacterString	
48	versionDate	Effective date of this catalogue	M	1	Date	
49	language	Default language of the textual information contained in the catalogue	O	1	Class	LanguageCode <<Codelist>> (ISO 639-2 3-alphabetic digits code)
50	characterSet	Default character coding standard used for the catalogue	O	1	Class	MD_CharacterSet Code <<Codelist>> (See ISO 19115:2003, B.5.10)
51	locale	Provides the list of locale in which the free text content of the catalogue has been translated.	C/Mandatory when the textual content of the catalogue is provided as free text	N	Class	PT_Locale
52	subCatalogue	Catalogue which is an integral part of the current catalogue NOTE The intent is not to aggregate all the catalogues whose content is referred to by the content of the current catalogue.	O	N	Class	CT_Catalogue
53	CT_Crs Catalogue	Description of a catalogue of Coordinate Reference Systems (CRS)	Use obligation from referencing object	Use maximum occurrence from referencing object	Specified class (CT_Catalogue)	Lines 54-62
54	*Role name:* crs	Provides the list of CRS described in the CRS catalogue	M	N	Association	CT_CRS
55	*Role name:* coordinate System	Provides the list of coordinate systems described in the CRS catalogue	O	N	Association	CT_Coordinate System
56	*Role name:* axis	Provides the list of coordinate system axis described in the CRS catalogue	O	N	Association	CT_Coordinate SystemAxis
57	*Role name:* datum	Provides the list of data described in the CRS catalogue	O	N	Association	CT_Datum

	Name/role name	Definition	Obligation/ condition	Maximum occurrence	Data type	Domain
58	*Role name:* ellipsoid	Provides the list of ellipsoids described in the CRS Catalogue	O	N	Association	CT_Ellipsoid
59	*Role name:* prime Meridian	Provides the list of prime meridians described in the CRS Catalogue	O	N	Association	CT_Prime Meridian
60	*Role name:* operation	Provides the list of operations described in the CRS Catalogue	O	N	Association	CT_Operation
61	*Role name:* operation Method	Provides the list of operationMethods described in the CRS Catalogue	O	N	Association	CT_Operation Method
62	*Role name:* operation Parameters	Provides the list of operation parameters described in the CRS Catalogue	O	N	Association	CT_Operation Parameters
63	CT_UoM Catalogue	Description of a catalogue of units of measure	Use obligation from referencing object	Use maximum occurrence from referencing object	Specified class (CT_Catalogue)	Line 64
64	*Role name:* uomItem	Provides the list of unit of measure items described in the catalogue	M	N	Association	CT_UnitOf Measure
65	CT_Codelist Catalogue	Description of a catalogue of codelists	Use obligation from referencing object	Use maximum occurrence from referencing object	Specified class (CT_Catalogue)	Line 66
66	*Role name:* codelistItem	Provides the list of codelist items described in the catalogue	M	N	Association	CT_Codelist

B.2.3.5 Catalogue items

B.2.3.5.1 Unit of measure and codelist items

	Name/role name	Definition	Obligation/ condition	Maximum occurrence	Data type	Domain
67	CT_Item	Description of an abstract item of a catalogue	Use obligation from referencing object	Use maximum occurrence from referencing object	<<Abstract>> class	Lines 68-71
68	identifier	Identifier of the item	M	1	GenericName	–
69	name	Name of the item	O	N	GenericName	–
70	definition	Definition of the item	M	1	CharacterString	Free text
71	description	Complementary description or remarks related to the item	O	1	CharacterString	Free text
72	CT_Codelist	Description of codelist item	Use obligation from referencing object	Use maximum occurrence from referencing object	Specified class (CT_Item)	Line 73
73	*Role name*: codeEntry	Provides the list of code value entries composing the codelist. Each entry is a codelist value item.	M	N	Association	CT_CodelistValue
74	CT_Codelist Value	Description of codelist value item	Use obligation from referencing object	Use maximum occurrence from referencing object	Specified class (CT_Item)	–

B.2.3.5.2 CRS catalogue items

	Name/role name	Definition	Obligation/ condition	Maximum occurrence	Data type	Domain
75	CT_CRS	Description of a catalogued CRS	Use obligation from referencing object	Use maximum occurrence from referencing object	Specified class (SC_CRS)	Line 76
76	definition	Definition of the catalogued CRS	M	1	CharacterString	Free text
77	CT_ Coordinate System	Description of a catalogued coordinate system	Use obligation from referencing object	Use maximum occurrence from referencing object	Specified class (SC_CoordinateSystem)	Line 78
78	definition	Definition of the catalogued coordinate system	M	1	CharacterString	Free text
79	CT_ Coordinate SystemAxis	Description of a catalogued coordinate system axis	Use obligation from referencing object	Use maximum occurrence from referencing object	Specified class (SC_CoordinateSystemAxis)	Line 80
80	definition	Definition of the catalogued coordinate system axis	M	1	CharacterString	Free text
81	CT_Datum	Description of a catalogued datum	Use obligation from referencing object	Use maximum occurrence from referencing object	Specified class (SC_Datum)	Line 82
82	definition	Definition of the catalogued datum	M	1	CharacterString	Free text
83	CT_Ellipsoid	Description of a catalogued ellipsoid	Use obligation from referencing object	Use maximum occurrence from referencing object	Specified class (SC_Ellipsoid)	Line 84
84	definition	Definition of the catalogued ellisoid	M	1	CharacterString	Free text
85	CT_Prime Meridian	Description of a catalogued prime meridian	Use obligation from referencing object	Use maximum occurrence from referencing object	Specified class (SC_Prime Meridian)	Line 86
86	definition	Definition of the catalogued prime meridian	M	1	CharacterString	Free text
87	CT_ Operation	Description of a catalogued operation	Use obligation from referencing object	Use maximum occurrence from referencing object	Specified class (CC_Operation)	Line 88
88	definition	Definition of the catalogued operation	M	1	CharacterString	Free text
89	CT_ Operation Method	Description of a catalogued operation method	Use obligation from referencing object	Use maximum occurrence from referencing object	Specified class (CC_Operation Method)	Line 90
90	definition	Definition of the catalogued operation method	M	1	CharacterString	Free text
91	CT_ Operation Parameters	Description of catalogued operation parameters	Use obligation from referencing object	Use maximum occurrence from referencing object	Specified class (CC_Operation Parameters)	Line 92
92	definition	Definition of the catalogued operation parameters	M	1	CharacterString	Free text

B.3 Codelist and enumerations

B.3.1 MX_ScopeCode

	Name	Definition
1	MX_ScopeCode (extends MD_ScopeCode)	Class of information to which the referencing entity applies in the context of a transfer
2	**initiative**	The referencing entity applies to a transfer aggregate which was originally identified as an initiative (DS_Initiative)
3	**stereoMate**	The referencing entity applies to a transfer aggregate which was originally identified as a stereo mate (DS_StereoMate)
4	**sensor**	The referencing entity applies to a transfer aggregate which was originally identified as a sensor (DS_Sensor)
5	**platformSeries**	The referencing entity applies to a transfer aggregate which was originally identified as a platform series (DS_PlatformSeries)
6	**sensorSeries**	The referencing entity applies to a transfer aggregate which was originally identified as a sensor series (DS_SensorSeries)
7	**productionSeries**	The referencing entity applies to a transfer aggregate which was originally identified as a production series (DS_ProductionSeries)
8	**transferAggregate**	The referencing entity applies to a transfer aggregate which has no existence outside of the transfer context
9	**otherAggregate**	The referencing entity applies to a transfer aggregate which has an existence outside of the transfer context, but which does not pertain to a specific aggregate type.

Annex C
(informative)

Geographic Metadata XML resources

C.1 XML schemas defined in this Technical Specification

This Technical Specification defines the content of six XML namespaces commonly identified using the following prefixes: gco, gmd, gmx, gsr, gss, and gts. Each of these namespace prefixes is appended to http://www.isotc211.org/2005/ to make a complete namespace identifier. The XML schemas associated with each of these namespaces can be found at http://www.iso.org/ittf/ISO_19139_Schemas with the following directory and file structure.

The files that make up the Geographic Common extensible markup language or http://www.isotc211.org/2005/gco are found at http://www.iso.org/ittf/ISO_19139_Schemas in the "gco" directory and are: basicTypes.xsd, gco.xsd and gcoBase.xsd.

The files that make up the Geographic MetaData extensible markup language or http://www.isotc211.org/2005/gmd are found at http://www.iso.org/ittf/ISO_19139_Schemas in the "gmd" directory and are: applicationSchema.xsd, avantfreeText.xsd, citation.xsd, constraints.xsd, content.xsd, dataQuality.xsd, distribution.xsd, extent.xsd, freeText.xsd, gmd.xsd, identification.xsd, maintenance.xsd, metadataApplication.xsd, metadataEntity.xsd, metadataExtension.xsd, portrayalCatalogue.xsd, referenceSystem.xsd, and spatialRepresentation.xsd.

The files that make up the Geographic Metadata XML Schema or http://www.isotc211.org/2005/gmx are found at http://www.iso.org/ittf/ISO_19139_Schemas in the "gmx" directory and are: catalogues.xsd, codelistItem.xsd, crsItem.xsd, extendedTypes.xsd, gmx.xsd, gmxUsage.xsd, and uomItem.xsd.

The files that make up the Geographic Spatial Referencing extensible markup language or http://www.isotc211.org/2005/gsr are found at http://www.iso.org/ittf/ISO_19139_Schemas in the "gsr" directory and are: gsr.xsd and spatialReferencing.xsd.

The files that make up the Geographic Spatial Schema extensible markup language or http://www.isotc211.org/2005/gss are found at http://www.iso.org/ittf/ISO_19139_Schemas in the "gss" directory and are: geometry.xsd and gss.xsd.

The files that make up the Geographic Temporal Schema extensible markup language or http://www.isotc211.org/2005/gts are found at http://www.iso.org/ittf/ISO_19139_Schemas in the "gts" directory and are: gts.xsd and temporalObjects.xsd.

C.2 XML schemas defined outside this Technical Specification

In addition to those namespaces listed above, this Technical Specification makes use of the *Geography Markup Language* or http://www.opengis.net/gml namespace. To locate the authoritative XML schemas associated with this namespace please refer to ISO 19136. This Technical Specification also makes use of the xml linking language or http://www.w3.org/1999/xlink namespace. The XML schemas associated with this namespace and used by this Technical Specification are referenced in ISO 19136.

C.3 Additional resources

To ease the use of this Technical Specification, several XML files are available for download in the "resources" directory. They are organized into the following categories of support: Codelists, Coordinate Reference Systems and Units of Measure.

The XML files related to the utilization of codelists that are available for download are found in the “Codelist” directory of the “resources”. Those files are ML_gmxCodelists.xml (multi-lingual codelists) and gmxCodelists.xml (standard codelists based on ISO 19115).

The XML files related to the utilization of coordinate reference systems that are available for download are found in the “CRS” directory of the “resources”. Those files are ML_gmxCrs.xml (multi-lingual CRSs) and gmxCrs.xml (standard CRSs).

The XML files related to the utilization of units of measure that are available for download are found in the “uom” directory of the “resources”. Those files are ML_gmxUom.xml (multi-lingual UnitsOfMeasure) and gmxUom.xml (standard UnitsOfMeasures).

Additionally there are XML metadata example files contained in the “examples” directory within the “resources” directory.