INTERNATIONAL STANDARD

ISO 19110

Second edition
2016-12-01

Geographic information — Methodology for feature cataloguing

Information géographique — Méthodologie de catalogage des entités

Reference number
ISO 19110:2016(E)

International Organization for Standardization

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

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.

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

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

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

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see the following URL: www.iso.org/iso/foreword.html.

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

This second edition cancels and replaces the first edition (ISO 19110:2005), which has been technically revised. It also replaces ISO 19110:2005/Amd1:2011. Annex G explains how to transform feature catalogues from the first edition to this revised version.

Introduction

Geographic features are real-world phenomena associated with a location relative to the Earth, about which data are collected, maintained, and disseminated. Feature catalogues defining the types of features, their operations, attributes, and associations represented in geographic data are indispensable to turning the data into usable information. Such feature catalogues promote the dissemination, sharing, and use of geographic data through providing a better understanding of the content and meaning of the data. Unless suppliers and users of geographic data have a shared understanding of the kinds of real-world phenomena represented by the data, users will be unable to judge whether the data supplied are fit for their purpose.

The availability of standard feature catalogues that can be used multiple times will reduce costs of data acquisition and simplify the process of product specification for geographic datasets.

This document provides a standard framework for organizing and reporting the classification of real-world phenomena in a set of geographic data. Any set of geographic data is a greatly simplified and reduced abstraction of a complex and diverse world. A catalogue of feature types can never capture the richness of geographic reality. However, such a feature catalogue should present the particular abstraction represented in a given dataset clearly, precisely, and in a form readily understandable and accessible to users of the data.

Geographic features occur at two levels: instances and types. At the instance level, a geographic feature is represented as a discrete phenomenon that is associated with its geographic and temporal coordinates and may be portrayed by a particular graphic symbol. These individual feature instances are grouped into classes with common characteristics: feature types. It is recognized that geographic information is subjectively perceived and that its content depends on the needs of particular applications. The needs of particular applications determine the way instances are grouped into types within a particular classification scheme. ISO 19109 specifies how data shall be organized to reflect the particular needs of applications with similar data requirements.

NOTE The full description of the contents and structure of a geographic dataset is given by the application schema developed in compliance with ISO 19109. The feature catalogue defines the meaning of the feature types and their associated feature attributes, feature operations, and feature associations contained in the application schema.

This document enables the multilingual description of application schemas compliant with ISO 19109. It goes further to provide a mechanism enabling a single global description of some properties occurring many times in an application schema and a binding of those global properties to the corresponding feature types.

The collection criteria used to identify individual real-world phenomena and to represent them as feature instances in a dataset are not specified in this document. Because they are not included in the standards, collection criteria should be included separately in the product specification for each dataset.

A standard way of organizing feature catalogue information will not automatically result in harmonization or interoperability between applications. In situations where classifications of features differ, this document may at least serve to clarify the differences and thereby help to avoid the errors that would result from ignoring them. It may also be used as a standard framework within which to harmonize existing feature catalogues that have overlapping domains.

This revision of ISO 19110 addresses issues related to the multilingual management of feature catalogues and applies the changes documented in a previous amendment. In addition to removing minor inconsistencies in the conceptual schemas, the amendment enhanced the mechanism ensuring the management of global properties. The amendment also provided an XML schema implementation of the feature catalogue conceptual schema and a management of feature catalogue registers. If the initial conceptual schema is not a subset of the amended conceptual schema, it is possible to transform legacy instances.

Geographic information — Methodology for feature cataloguing

1 Scope

This document defines the methodology for cataloguing feature types. This document specifies how feature types can be organized into a feature catalogue and presented to the users of a set of geographic data. This document is applicable to creating catalogues of feature types in previously uncatalogued domains and to revising existing feature catalogues to comply with standard practice. This document applies to the cataloguing of feature types that are represented in digital form. Its principles can be extended to the cataloguing of other forms of geographic data. Feature catalogues are independent of feature concept dictionaries defined in ISO 19126 and can be specified without having to use or create a Feature Concept Dictionary.

This document is applicable to the definition of geographic features at the type level. This document is not applicable to the representation of individual instances of each type. This document excludes portrayal schemas as specified in ISO 19117.

This document may be used as a basis for defining the universe of discourse being modelled in a particular application, or to standardize general aspects of real world features being modelled in more than one application.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements 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 19103, *Geographic information — Conceptual schema language*

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

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

ISO/TS 19115-3:2016, *Geographic information — Metadata — Part 3: XML schema implementation for fundamental concepts*

ISO 19135-1:2015, *Geographic information — Procedures for item registration — Part 1: Fundamentals*

ISO/TS 19139:2007, *Geographic information — Metadata — XML schema implementation*

3 Terms and definitions

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

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— IEC Electropedia: available at http://www.electropedia.org/

— ISO Online browsing platform: available at http://www.iso.org/obp

3.1
designation
designator
representation of a concept by a sign which denotes it

Note 1 to entry: In terminology work, three types of designations are distinguished: symbols, appellations and terms.

[SOURCE: ISO 1087-1:2000, 3.4.1]

3.2
feature
abstraction of real-world phenomena

EXAMPLE The phenomenon named "Eiffel Tower" may be classified with other similar phenomena into a feature type "tower."

Note 1 to entry: A feature may occur as a type or an instance. Feature type or feature instance should be used when only one is meant.

[SOURCE: ISO 19101-1:2014, 4.1.11]

3.3
feature association
relationship that links instances of one *feature* (3.2) type with instances of the same or a different feature type

3.4
feature attribute
characteristic of a *feature* (3.2)

EXAMPLE 1 A feature attribute named "colour" may have an attribute value "green" which belongs to the data type "text".

EXAMPLE 2 A feature attribute named "length" may have an attribute value "82,4" which belongs to the data type "real".

Note 1 to entry: A feature attribute has a name, a data type, and a value domain associated to it. A feature attribute for a feature instance also has an attribute value taken from the value domain.

[SOURCE: ISO 19101-1:2014, 4.1.12]

3.5
feature catalogue
catalogue containing definitions and descriptions of the *feature* (3.2) types, *feature attributes* (3.4) and feature relationships occurring in one or more sets of geographic data, together with any *feature operations* (3.7) that can be applied

Note 1 to entry: Feature relationships include *feature inheritances* (3.6) and *feature associations* (3.3).

[SOURCE: ISO 19101-1:2014, 4.1.13]

3.6
feature inheritance
mechanism by which more specific *features* (3.2) incorporate structure and behaviour of more general features related by behaviour

3.7
feature operation
operation that every instance of a *feature* (3.2) type may perform

EXAMPLE A feature operation upon a "dam" is to raise the dam. The results of this operation are to raise the height of the "dam" and the level of water in a "reservoir".

Note 1 to entry: Sometimes, feature operations provide a basis for feature type definition.

3.8
functional language
language in which *feature operations* (3.7) are formally specified

Note 1 to entry: In a functional language, feature types may be represented as abstract data types.

3.9
signature
text string that specifies the name and parameters required to invoke an operation

Note 1 to entry: It may contain optional returned parameters. This signature is usually derived from the formal definition. This is the equivalent of the UML signature.

4 Conformance

4.1 Conformance classes

The methodology for cataloguing feature types is defined through a set of requirements for the description of feature types. A single conformance class is defined for models meeting all the conceptual requirements. This document presents a conceptual model for a representation of feature type descriptions as a set of UML diagrams that satisfy this conformance class. Annex A presents the abstract test suits for conformance classes.

A second set of requirements for XML implementation of the conceptual model are the basis for a conformance class for XML implementation of the UML model for representation of feature types in a feature catalogue. This implementation is based on rules defined in ISO/TS 19139 and ISO/TS 19115-3.

Annex D defines a conceptual model for a registered feature catalogue, but no corresponding XML implementation is specified by this document.

Table 1 — Conformance classes defined by this specification

Conformance class URI[a]	Standardization target	Conformance class name (implemented clause)
/conf/conceptual-model	Conceptual model	Conceptual model for a feature catalogue
/conf/feature-catalogue-xml	XML implementation	XML implementation of feature catalogue conceptual model
/conf/feature-catalogue-xml-instance	XML instance document	Valid XML instance document for interchange of feature catalogue content

[a] All Conformance Class URIs are HTTP URIs, prefix 'http://standards.iso.org/iso/19110' to the paths in the table cell to get the complete URI.

5 Abbreviated terms

GFC	Geographic Feature Cataloguing
GFM	General Feature Model
HTTP	Hyper Text Transfer Protocol
IHO	International Hydrographic Organization
TS	Technical Specification

UML	Unified Modeling Language
URI	Uniform Resource Identifier
XML	eXtensible Markup Language

6 Requirements

6.1 General

Clause 6 specifies general and specific requirements for feature catalogue information elements.

6.2 Conceptual requirements

Tables 2 to 10, 12, and 13 summarize requirements for the conceptual model for describing feature types and their attributes, operations, and associations in a feature catalogue. Requirements are grouped into requirements classes; each requirements class has a URI, and each requirement has a URI that is based on the URI for the requirements class to which it is included. The dependencies column provides the URI for any requirements class that must be met as a precondition for the requirements class.

In Tables 2 to 10 and 12 to 13, the following conventions are followed.

— The term "entity" is used to refer to model elements representing instantiable information objects in the conceptual model. Depending on the modelling paradigm used, these may have other labels, e.g. "object," "class," "element," or "feature".

— Inclusion of a model entity name in a requirements statement implies that entity or any subtype is derived from that entity in a model instance.

— Property names are provided in single quotes (' '), and are all in lower case.

Table 2 — Requirements class for catalogue

Identifier	http://standards.iso.org/iso/19110/1.1/req/catalogue
Target type	Conceptual model
Name	Core conceptual catalogue requirements for ISO 19110 feature catalogue
Dependency	ISO 19115-1:2014, 6.6.2 ISO/TS 19139:2007, 7.4.4
Requirement	**/req/catalogue/representation** A feature catalogue shall document an abstraction of reality representing one or more geographic features. NOTE A feature catalogue may comply with the specifications of this document independent of any existing set of geographic data.
Requirement	**/req/catalogue/abstraction** The feature type shall be the basic level of abstraction in a feature catalogue.
Requirement	**/req/catalogue/electronic-form** A feature catalogue shall be available in electronic form.
Requirement	**/req/catalogue/inheritance** A feature catalogue shall inherit all the properties and relationships defined on the abstract CT_Catalogue class in ISO/TS 19139:2007, 7.4.4.
Requirement	**/req/catalogue/identification** The feature catalogue shall include identification information that includes a 'name', a 'versionNumber', a 'versionDate'.

Table 2 *(continued)*

Requirement	**/req/catalogue/producer** The feature catalogue entity shall include a mandatory 'producer' property with exactly one value that is an entity conforming to the content of the CI_Responsibility entity defined by ISO 19115-1:2014, 6.6.2.
Requirement	**/req/catalogue/functional-language** If a functional language is used to formally define feature operations, the feature catalogue entity shall have a 'functional language' property with a text value that specifies the language used.
Requirement	**/req/catalogue/identifier** If a globally unique identifier is included as a property of a feature catalogue, it shall be named 'identifier' and shall have a value that is an entity conforming to the content of the MD_Identifier entity defined by ISO 19115-1:2014, 6.6.2.
Recommendation	**/rec/catalogue/schema-language** To maximize the usefulness of a feature catalogue across different applications, the use of a conceptual schema language to model feature catalogue information is recommended. NOTE Natural-language definitions, feature-type aliases, criteria for the creation and withdrawal of feature instances, and other semantic elements of the feature catalogue may be included in a conceptual schema as structured comments or as attributes.

Table 3 — Requirements class for base content

Identifier	http://standards.iso.org/iso/19110/1.1/req/base-content
Target type	Conceptual model
Name	Core conceptual base content requirements for ISO 19110 feature catalogue
Dependency	http://standards.iso.org/iso/19110/req/catalogue
Requirement	**/req/base-content/minimum** A feature catalogue shall describe at least one feature type.
Requirement	**/req/base-content/feature-type-names** A feature type shall be identified by exactly one 'type name' that is unique in the scope of the containing feature catalogue. The 'type name' value shall consist of an optional 'codespace' property that specifies a namespace, and a string value that provides the name.
Requirement	**/req/base-content/feature-type-is-abstract** A feature type shall have a mandatory 'is abstract' property with a Boolean value.
Requirement	**/req/base-content/feature-properties** Properties of a feature type shall be linked to the feature type with the role 'carrierOfCharacteristics'. Properties of a feature type shall be categorized as one of feature attribute, feature operation, or association role.
Requirement	**/req/base-content/property-type-names** All property types (attribute, operation, or association role) shall be identified by a single 'type name' that is unique within the scope of the feature type (local properties) or feature catalogue (global properties) that contains the property definition. The 'type name' value shall consist of an optional 'codespace' property that specifies a namespace, and a string value that provides the name.
Requirement	**/req/base-content/all-type-definitions** The feature catalogue shall include definitions of all feature types and property types (attribute, operation, or association role) included in the model.

Table 3 *(continued)*

Requirement	**/req/base-content/constraints** If the model includes constraints on feature type or property type entities, they shall be represented by a constraints entity that has a description property with a string value. The feature type or property type entity shall be linked to the constraints entity via a 'constrainedBy' role.
Recommendation	**/rec/base-content/multiple-definition** If the same term is defined in both a referenced definition source and in a feature catalogue element (feature type, property, association, or listed value), the definition in the feature catalogue element should take precedence.
Recommendation	**/rec/base-content/only-elements** To ensure predictability and comparability of feature catalogue content, it is recommended that the model includes only the elements specified in the UML conceptual model included in this document (see Annex B).
Recommendation	**/rec/base-content/functional-language** The use of functional language specifications to help define feature types is recommended.
Identifier	http://standards.iso.org/iso/19110/1.1/req/base-content
Target type	Conceptual model
Name	Core conceptual base content requirements for ISO 19110 feature catalogue
Dependency	http://standards.iso.org/iso/19110/req/catalogue
Recommendation	**/rec/base-content/abstract-property-type** Specification of feature properties should be implemented through an abstract property type class from which attribute, operation, and association role types are derived.
Recommendation	**/rec/base-content/property-closure** A feature catalogue should include, for each feature type, specifications of any properties bound to the feature type, and specifications of any operations that affect feature properties defined in the catalogue.
Informative	Each feature attribute, listed value, feature association, and feature type may have a 'code' property that has an alphanumeric value intended as an identifier.

Table 4 — Requirements class for attribute

Identifier	http://standards.iso.org/iso/19110/1.1/req/attribute
Target type	Conceptual model
Name	Core conceptual attribute requirements for ISO 19110 feature catalogue
Dependency	http://standards.iso.org/iso/19110/req/base-content
Requirement	**/req/attribute/inheritance** The feature attribute entity shall inherit all the properties and associations of the property type entity.
Requirement	A feature attribute shall have a property named 'valueType' with a value that is a string referencing a data or object type. The 'valueType' shall be specified either directly as part of the attribute definition, or indirectly in a binding (see /req/global, below).

Table 4 *(continued)*

Requirement	**/req/attribute/attribute-cardinality** A feature attribute shall have a cardinality property that specifies the lower and upper limit on the number of instances of the target value type that may occur in a valid data instance. Default value is 1.
Requirement	**/req/attribute/measurement-unit** If a feature attribute entity has a property specifying the measurement units associated with an attribute value, it shall be named 'valueMeasurementUnit' and have a value that is a string that identifies a unit of measure.
Requirement	**/req/attribute/code** If a feature attribute entity has a property providing an additional identifier for the attribute, it shall be named 'code'.

Table 5 — Requirements class for association

Identifier	http://standards.iso.org/iso/19110/1.1/req/association
Target type	Conceptual model
Name	Core conceptual association requirements for ISO 19110 feature catalogue
Dependency	http://standards.iso.org/iso/19110/req/base-content
Requirement	**/req/association/inheritance** The feature association entity shall inherit all the properties and associations of the feature type entity.
Requirement	**/req/association/association-participation** A feature association shall have at least two association role instances linked through the 'roleName' role.
Requirement	**/req/association/role-cardinality** The association role entity shall have a 'cardinality' property that specifies the lower and upper limit on the number of instances of the target feature type that may occur in a valid data instance. Default value is "0..*".
Requirement	**/req/association/association-role-inheritance** The association role entity shall inherit all the properties and associations of the property type entity.
Requirement	**/req/association/role-relation** An association role entity shall be bound to exactly one feature association entity through a 'relation' role.
Requirement	**/req/association/role-player** An association role entity shall be bound to at least one feature type through the 'rolePlayer' role. NOTE This binding may be directly from the association role, in which case the cardinality is 1, or indirectly via a binding and bound association role, in which case multiple feature types may be bound to the association role.

Table 5 *(continued)*

Requirement	**/req/association/role-type** An association role entity shall have a 'type' property with a value specified by an enumeration consisting of the following terms {ordinary, aggregation, composition}.
Requirement	**/req/association/role-is-ordered** An association role entity shall have an 'isOrdered' property with a Boolean value that indicates if the instances of this association role within the containing feature instance are ordered or not, with FALSE= "not ordered" and TRUE = "ordered".
Requirement	**/req/association/role-is-navigable** An association role entity shall have an 'is navigable property with a Boolean value that indicates whether this role is navigable from the source feature to the target feature of the association. TRUE = 'navigable'.

Table 6 — Requirements class for inheritance

Identifier	http://standards.iso.org/iso/19110/1.1/req/inheritance
Target type	Conceptual model
Name	Core conceptual inheritance requirements for ISO 19110 feature catalogue
Dependency	http://standards.iso.org/iso/19110/req/base-content
Requirement	**/req/inheritance/class** Inheritance relationships in a feature catalogue shall be represented using an inheritance relation class that associates one feature type in the 'subtype' role with a different feature type in the 'supertype' role.
Requirement	**/req/inheritance/description** An inheritance relation shall have a 'description' property.
Requirement	**/req/inheritance/unique-instance** An inheritance relation shall have a 'uniqueInstance' property that is a Boolean value, true if an instance of the 'supertype' can be an instance of at most one of its 'subtype' feature types.

Table 7 — Requirements class for global

Identifier	http://standards.iso.org/iso/19110/1.1/req/global
Target type	Conceptual model
Name	Core conceptual global properties requirements for ISO 19110 feature catalogue
Dependency	http://standards.iso.org/iso/19110/req/base-content
Requirement	**/req/global/global-property** A globalProperty type shall be associated to one feature catalogue through the 'featureCatalogue' role and shall not be associated to any feature type via a 'featureType' role.
Requirement	**/req/global/binding** A binding shall associate exactly one property type in a 'globalProperty' role to exactly one feature type in either a 'featureType' role, or a 'rolePlayer' role, but not in both roles.
Requirement	**/req/global/binding-description** If a binding has a property used to describe the binding instance, it shall be named 'description' and have a string value.
Requirement	**/req/global/global-xor-local** A property type that fills a 'globalProperty' role on a binding shall not also fill a 'carrierOfCharacteristics' role on a feature type and shall be associated to one feature catalogue through the 'featureCatalogue' role.

Table 7 *(continued)*

Requirement	**/req/global/bound-association-role** If the model allows a 'rolePlayer' association between an association role in a 'globalProperty' role and a feature type, then that binding shall be through a bound association role entity that inherits all the properties and associations of the binding entity.
Identifier	http://standards.iso.org/iso/19110/1.1/req/global
Target type	Conceptual model
Name	Core conceptual global properties requirements for ISO 19110 feature catalogue
Dependency	http://standards.iso.org/iso/19110/req/base-content
Requirement	**/req/global/bound-feature-attribute** A feature attribute that fills a 'globalProperty' role shall specify a 'valueType' for the attribute if and only if there is no valid 'valueType' specified in the feature attribute definition. A binding between a feature attribute and a feature type that specifies a 'valueType' shall be through a bound feature attribute entity that inherits all the properties and associations of the binding entity.
Requirement	**/req/global/binding-constraints** If the model includes constraints on binding entities, they shall be represented by a Constraints entity that has a description property with a string value. The binding entity shall be linked to the Constraints entity via a 'constrainedBy' role.

Table 8 — Requirements class for operation

Identifier	http://standards.iso.org/iso/19110/1.1/req/operation
Target type	Conceptual model
Name	Core conceptual operation requirements for ISO 19110 feature catalogue
Dependency	http://standards.iso.org/iso/19110/req/base-content
Requirement	**/req/operation/inheritance** The feature operation class shall inherit all the properties and associations of the property type class.
Requirement	**/req/operation/affected-features** If the feature operation is contained inside a feature type, this containment shall be specified through the 'featureType' role. NOTE A feature operations may be defined as a standalone class for operations that affect property values of other features but are not considered a part of any feature type.
Requirement	**/req/operation/operation-attributes** Only feature attributes shall be associated with feature operations via 'observesValueOf', 'affectsValuesOf', or 'triggeredByValuesOf' association roles.
Requirement	**/req/operation/signature** Each feature operation entity shall have exactly one 'signature' property that is unique in the scope of the feature catalogue. NOTE The signature specifies the operation name and required parameter names used to invoke the operation.
Requirement	**/req/operation/operation-cardinality** A feature operation entity shall have a cardinality property that specifies the number of return values possible. Default value is 1.
Identifier	http://standards.iso.org/iso/19110/1.1/req/operation
Target type	Conceptual model
Name	Core conceptual operation requirements for ISO 19110 feature catalogue

Table 8 *(continued)*

Dependency	http://standards.iso.org/iso/19110/req/base-content
Requirement	**/req/operation/formal-definition** If an operation is formally specified using a functional language; the text of this specification shall be provided in a feature operation entity property named 'formal definition.'
Requirement	**/req/operation/functional-language** If a feature operation entity includes a 'formalDefinition' property, then the containing feature catalogue entity shall have a 'functional language' property. NOTE The intention is that formal definitions utilize the functional language specified for the catalogue, but this can only be verified with feature catalogue instances.

Table 9 — Requirements class for value list

Identifier	http://standards.iso.org/iso/19110/1.1/req/value-list
Target type	Conceptual model
Name	Core conceptual value list requirements for ISO 19110 feature catalogue
Dependency	http://standards.iso.org/iso/19110/req/base-content
Requirement	**/req/value-list/class** If the domain of a feature attribute is an enumeration or controlled vocabulary, as indicated by the 'valueType' property, that list shall be specified using a listed value entity that fills the 'listed value' role on the feature attribute.
Requirement	**/req/value-list/label** Each listed value instance shall have exactly one text 'label' property value.
Requirement	**/req/value-list/definition** If a listed value entity has a property providing a definition of the value, it shall be named 'definition'.
Requirement	**/req/value-list/code** If a listed value entity has a property providing an additional identifier for a value, it shall be named 'code'.

Table 10 — Requirements class for conceptual model

Identifier	http://standards.iso.org/iso/19110/1.1/req/conceptual-model
Target type	Conceptual model
Name	Conceptual model for ISO 19110 feature catalogue
Dependency	http://standards.iso.org/iso/19110/req/catalogue http://standards.iso.org/iso/19110/req/base-content http://standards.iso.org/iso/19110/req/attributes http://standards.iso.org/iso/19110/req/association http://standards.iso.org/iso/19110/req/inheritance http://standards.iso.org/iso/19110/req/global http://standards.iso.org/iso/19110/req/operation http://standards.iso.org/iso/19110/req/value-list
Requirement	**/req/conceptual-model/all** A conceptual model that conforms to the ISO 19110 methodology for representing a feature catalogue shall meet all the requirements enumerated in the catalogue, base-content, attributes, association, inheritance, global, operation and value-list requirements classes.

In order to enable XML implementation of a model conforming to the above requirements, this document includes as Annex B a UML model conforming to the requirements enumerated above. The UML model adds some additional, optional properties, associations, and classes that are not mandated by the requirements classes in Table 1, but were determined to be useful by the editing committee. The UML model in Annex B is considered normative for the definitions, properties, and cardinalities of these elements and associations.

6.3 XML implementation requirements

The XML implementation of the methodology for feature cataloguing is based on the UML model presented in Annex B. The implementation requires two namespaces. The namespace http://standards.iso.org/iso/19110/gfc/1.1 (standard abbreviation is gfc) implements the classes, properties and associations included in the UML model. The namespace http://standards.iso.org/iso/19110/fcc/1.0 (standard abbreviation is fcc) includes elements that implement abstract classes necessary to enable other ISO XML implementations to link to elements of this document without committing to a particular version of the XML implementation. This pattern is based on the implementation rule in ISO/TS 19115-3:2016, Clause 8. Table 11 summarizes the abstract classes implemented in the fcc namespace.

Table 11 — Abstract classes defined in the fcc namespace to allow loose coupling to other ISO/TC 211 XML implementations

Abstract class	Namespace defining concrete implementation
_FeatureCatalogue	feature catalogue (gfc)
_FeatureType	feature catalogue (gfc)

ISO/TS 19139:2007, Clauses 7, 8 and 9, and ISO/TS 19115-3:2016, 8.3 describe the details of encoding the UML conceptual schema into a set of XML schemas. The XML schema implementation of the ISO 19110 conceptual model provided in Annex B follows the rules and patterns described in those clauses, applying them to the UML model for XML implementation.

Because XML schema validation is insufficient to test all of the enumerated requirements, some conformance tests require other validation procedures. For instance, as stated in ISO/TS 19139:2007, 8.4, a property element following the default XML Class property type (XCPT) pattern may have exactly one of

1) inline content (by-value) that is an XML Class,

2) an xlink:href attribute (by-reference value), or

3) a gco:nilReason attribute (nil value).

Because XML schema cannot constrain the co-occurrence of content or attributes, some mechanism in addition to XML schema validation shall be used to restrict a property to be exclusively by-value or by-reference or a nil value.

Rules implementing these constraints are included in the appropriate requirements class for the XML document instances. The ISO 19110 XML implementation package includes a Schematron rule set for testing conformance with these requirements. If a tool for Schematron validation is not available, conformance to these requirements may need to be tested by inspection.

Table 12 — Requirements class for XML implementation

Identifier	http://standards.iso.org/iso/19110/req/1.1/xml-implementation
Target type	XML schema and Schematron rule documents
Name	Core requirements for implementation of feature catalogue conceptual model using XML.

Table 12 *(continued)*

Dependency	http://standards.iso.org/iso/19139/spec#7 http://standards.iso.org/iso/19139/spec#8 http://standards.iso.org/iso/19139/spec#9 http://standards.iso.org/iso/19115-3/1.0/spec#8.4 http://standards.iso.org/iso/19110/1.1/spec/annex-b
Requirement	**/req/xml-implementation/rule-based** XML schema and Schematron rules implementing the UML model for feature catalogue defined in this specification shall follow the rules and patterns defined in ISO/TS 19139 and ISO/TS 19115-3.
Requirement	**/req/xml-implementation/base-data-types** Base data types shall be implemented according to rules set forth in ISO/TS 19139.

6.4 XML instance document requirements

Various conformance tests for this document require that metadata instance (XML) documents can be validated without error against the XML schemas defined in this document. While many tools are available to test the 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 that implements a strict interpretation of and full support for the W3C XML Schema Recommendation be used when validating XML instance documents to test conformance.

Conformance classes for requirements related to XML instance documents (the conformance target) are tested by XML schema validation, the use of Schematron rule sets, and by inspection of instance documents. Conformance class requirements and tests are presented in Tables A.1 through A.2.

Table 13 — Requirements class for XML instance

Identifier	http://standards.iso.org/iso/19110/req/xml-instance
Target type	XML instance document
Name	Core requirements for feature catalogue instance documents
Dependency	http://standards.iso.org/iso/19139/spec#8.4.1 http://standards.iso.org/iso/19110/1.1/req/xml-implementation
Requirement	**/req/xml-instance/instance-validation** XML instance documents shall be well formed and valid according to XML schema and Schematron rules associated with feature catalogue namespace (gfc.xsd, gfc.sch).
Requirement	**/req/xml-instance/property-type-content** A property element instance shall have exactly one of 1) inline content (by-value) that is a schema-valid XML Class instance, or 2) an xlink:href attribute (by-reference value), or 3) a gco:nilReason attribute (nil value).
Requirement	**/req/xml-instance/unique-type-name** The values of the 'typeName' property for each FC_FeatureType shall be unique within a FC_FeatureCatalogue instance.
Requirement	**/req/xml-instance/unique-code-value** Values assigned to 'code' properties on FC_FeatureAttribute, FC_ListedValue, FC_FeatureAssociation, and FC_FeatureType instances shall be unique within the scope of a FC_FeatureCatalogue instance.

Table 13 *(continued)*

Requirement	**/req/xml-instance/unique-property-name** If an FC_PropertyType (attribute, operation, or association role) is the value for a 'globalProperty' XML property, then the 'memberName' of the property shall be unique within the containing FC_FeatureCatalogue instance; otherwise, the 'memberName' of the property shall be unique within the containing FC_FeatureType. If the identifiers are not globally unique URIs, then their scope of uniqueness should be specified in the catalogue scope statement.
Requirement	**/req/xml-instance/definition-language** All 'definition' XML properties shall have values provided in a natural language. NOTE The concrete XML classes that have definition properties are FC_FeatureType, FC_FeatureAssociation, FC_FeatureAttribute, FC_FeatureOperation, FC_AssociationRole, and FC_ListedValue.
Requirement	**/req/xml-instance/unique-subtype-instance** If the 'uniqueInstance' property of FC_InheritanceRelation is true, then an instance of the 'supertype' shall be an instance of at most one of its 'subtype' featureTypes.
Requirement	**/req/xml-instance/functional-language-specified** If any 'formalDefinition' property of FC_FeatureOperation has a non-null value, then the 'functionalLanguage' property of FC_FeatureCatalogue shall have a valid value.
Requirement	**/req/xml-instance/definition-source** Definitions for featureTypes, feature properties (attributes, operations, or association role), or listed values shall either be included as text values for 'definition' property instances, or referenced to a separate definition source via 'definitionReference' properties populated by FC_DefinitionReference elements.
Identifier	http://standards.iso.org/iso/19110/req/xml-instance
Target type	XML instance document
Name	Core requirements for feature catalogue instance documents
Dependency	http://standards.iso.org/iso/19139/spec#8.4.1 http://standards.iso.org/iso/19110/1.1/req/xml-implementation
Requirement	**/req/xml-instance/value-list-label-unique** Each 'label' value in an FC_ListedValue element shall be unique within scope of the FC_FeatureAttribute to which it is bound.
Requirement	**/req/xml-instance/role-is-ordered** If the 'isOrdered' property of an FC_AssociationRole is TRUE, then the 'definition' property shall be present and shall contain an explanation of the meaning of the order (Table B.10).
Recommendation	**/rec/xml-instance/functional-language-consistency** The 'functionalLanguage' property value in the FC_FeatureCatalogue element should actually be the language used in the 'formalDefinition' values of FC_FeatureOperation elements, and all formal operations definitions provided in a feature catalogue should use the same functional language. NOTE This is a recommendation because testing of this condition is considered to be outside the scope of expected test facilities.
Recommendation	**/rec/xml-instance/all-feature-types** It is recommended that FC_FeatureCatalogue instances that are bound to specific datasets document all the featureTypes found in those datasets.

Annex A
(normative)

Abstract test suite

A.1 Introduction

This annex presents the abstract test suite for evaluating conformance to this document. This abstract test suite defines test procedures for the requirements enumerated in Clause 6. Three conformance classes are defined that aggregate these tests to specify validation criteria for conceptual models conforming to this document, XML schema and Schematron rules implementing the conceptual model, and XML instance documents that encode feature catalogue content.

A.2 Conceptual model conformance class

Most of the specified requirements constrain the conceptual model for a feature catalogue. The single requirements class specified for a conformant conceptual model aggregates all the requirements classes defined in Table 2. Tests for these requirements are enumerated in Table A.1, grouped for each requirements class aggregated in the conformance class.

In these tables, the following conventions are followed.

— The term "entity" is used to refer to model elements representing instantiable information objects in the conceptual model. Depending on the modelling paradigm used, these may have other labels, e.g. "object," "entity," "class," "element," or "feature".

— Inclusion of a model entity name in a condition or test implies that entity or any subtype is derived from that entity in a model instance.

— Property names are provided in single quotes (' '), and are all in lower case.

Table A.1 — Conformance class: Conceptual model

Identifier	http://standards.iso.org/iso/19110/1.1/conf/conceptual-model	
Standardization target	Conceptual model	
Name	Conceptual model for a feature catalogue	
Requirements	http://standards.iso.org/iso/19110/1.1/req/conceptual-model	
Dependent requirements class	http://standards.iso.org/iso/19110/1.1/req/catalogue	
Test	Identifier	**/conf/conceptual-model/catalogue/representation**
	Requirement	/req/catalogue/representation
	Test method	Inspect the model to verify that the model documents representation of geographic features.
Identifier	http://standards.iso.org/iso/19110/1.1/conf/conceptual-model	
Standardization target	Conceptual model	
Name	Conceptual model for a feature catalogue	
Requirements	http://standards.iso.org/iso/19110/1.1/req/conceptual-model	

Table A.1 *(continued)*

Test	Identifier	**/conf/conceptual-model/catalogue/abstraction**
	Requirement	/req/catalogue/abstraction
	Test method	Inspect the model to verify that a feature catalogue is represented as a collection of feature type definitions, and that any other child elements that are members of the catalogue describe properties, associations, or definition sources related to a FeatureType.
Test	Identifier	**/conf/conceptual-model/catalogue/electronic-form**
	Requirement	/req/catalogue/electronic-form
	Test method	Verify that the model can be implemented for representation of a feature catalogue in electronic form.
Test	Identifier	**/conf/conceptual-modelf/catalogue/inheritance**
	Requirement	/req/catalogue/inheritance
	Test method	Verify that the root feature catalogue element in the model has all the properties and relationships defined on the abstract CT_Catalogue class in ISO/TS 19139:2007, 7.4.4.
Test	Identifier	**/conf/conceptual-model/catalogue/identification**
	Requirement	/req/catalogue/identification
	Test method	Verify that the root feature catalogue element in the model has a 'name', a 'versionNumber', a 'versionDate', and that each of these properties is required to have a valid value. NOTE These properties are inherited from the abstract CT_Catalogue class.
Test	Identifier	**/conf/conceptual-model/catalogue/producer**
	Requirement	/req/catalogue/producer
	Test method	Verify that the root feature catalogue element in the model has a 'producer' property, that is required to have a single value that is an entity conforming to the CI_Responsibility entity defined in ISO 19115-1:2014, 6.6.2.
Test	Identifier	**/conf/conceptual-model/catalogue/functional-language**
	Requirement	/req/catalogue/functional-language
	Test method	Verify that if the root feature catalogue element in the model has a property intended to specify a functional language used to define operations in the catalogue, the property is named 'functional language', and has a text value.
Identifier	http://standards.iso.org/iso/19110/1.1/conf/conceptual-model	
Standardization target	Conceptual model	
Name	Conceptual model for a feature catalogue	
Requirements	http://standards.iso.org/iso/19110/1.1/req/conceptual-model	
Test	Identifier	**/conf/conceptual-model/catalogue/identifier**
	Requirement	/req/catalogue/identifier
	Test method	Verify that if the root feature catalogue element in the model has a property intended to provide a unique identifier for the catalogue instance that the property is named 'identifier', and has a value that is an entity conforming to the MD_Identifier entity defined in ISO 19115-1:2014, 6.6.2.
Dependent requirements class	http://standards.iso.org/iso/19110/1.1/req/base-content	
Test	Identifier	**/conf/conceptual-model/base-content/minimum**
	Requirement	/req/base-content/minimum
	Test method	Inspect the model to verify that at least one feature type instance is mandatory in a feature catalogue.

Table A.1 *(continued)*

Test	Identifier	**/conf/conceptual-model/base-content/feature-type-names**
	Requirement	/req/base-content/feature-type-names
	Test method	Inspect the model to verify that feature type entities have a 'name' property that is required to have exactly one value, and the value is an object that allows a local name string value and a codespace string value that specifies the scope of the local name.
Test	Identifier	**/conf/conceptual-model/base-content/feature-type-is-abstract**
	Requirement	/req/base-content/feature-type-is-abstract
	Test method	Inspect the model to verify that feature type entities have an 'is abstract' property that is required to have exactly one Boolean value.
Test	Identifier	**/conf/conceptual-model/base-content/feature-properties**
	Requirement	/req/base-content/feature-properties
	Test method	Inspect the model to verify that feature type entities are associated to property type entities via a 'carrierOfCharacteristics' role. This association may be directly to the property type, or indirectly through a binding entity and a 'globalProperty' role. Inspect the model to verify that all property type entities are categorized as one of feature attribute, feature operation, or association role.
Test	Identifier	**/conf/conceptual-model/base-content/property-type-names**
	Requirement	/req/base-content/property-type-names
	Test method	Inspect the model to verify that property type entities have a mandatory, single-valued 'member name' property exactly one value, and the value is an object that allows a local name string value and a 'codespace' property string value that specifies the namespace of the local name.
Identifier	http://standards.iso.org/iso/19110/1.1/conf/conceptual-model	
Standardization target	Conceptual model	
Name	Conceptual model for a feature catalogue	
Requirements	http://standards.iso.org/iso/19110/1.1/req/conceptual-model	
Test	Identifier	**/conf/conceptual-model/base-content/all-type-definitions**
	Requirement	/req/base-content/all-type-definitions
	Test method	Inspect the model to verify that every defined feature type entity and every property type has a 'definition' property or a 'definitionReference' property and a constraint that at least one of these has a valid value. NOTE The intention is that the definition is either provided by a text property on the Feature or property type entity, or by reference to an source external to the feature catalogue.
Test	Identifier	**/conf/conceptual-model/base-content/constraints**
	Requirement	/req/base-content/constraints
	Test method	Inspect the model to verify that if the model includes constraints on Feature or property types, the representation is via a 'constrainedBy' role to a Constraints entity that has a 'description' property with a string value.
Dependent requirements class	http://standards.iso.org/iso/19110/1.1/req/attribute	
Test	Identifier	**conf/conceptual-model/attribute/inheritance**
	Requirement	/req/attribute/inheritance
	Test method	Inspect the model to verify that feature attribute entities have all of the properties and associations that are defined on the property type class.

Table A.1 *(continued)*

Test	Identifier	**/conf/conceptual-model/attribute/attribute-datatype**
	Requirement	/req/attribute/attribute-datatype
	Test method	Inspect the model to verify that a 'valueType' must be specified for each feature attribute either as a property contained in the attribute definition or as part of a binding between a globalProperty and a feature type.
Test	Identifier	**/conf/conceptual-model/attribute/attribute-cardinality**
	Requirement	/req/attribute/attribute-cardinality
	Test method	Inspect the model to verify that feature attribute entities have a mandatory 'cardinality' property with a default value of '1'
Test	Identifier	**/conf/conceptual-model/attribute/measurement-unit**
	Requirement	/req/attribute/measurement-unit
	Test method	Inspect the model to verify that if a feature attribute entity has a property specifying the measurement units associated with an attribute value, it is named 'valueMeasurementUnit' and has a value type that allows a string value identifying a unit of measure.
Identifier	http://standards.iso.org/iso/19110/1.1/conf/conceptual-model	
Standardization target	Conceptual model	
Name	Conceptual model for a feature catalogue	
Requirements	http://standards.iso.org/iso/19110/1.1/req/conceptual-model	
Test	Identifier	**/conf/conceptual-model/attribute/code**
	Requirement	/req/attribute/code
	Test method	Inspect the model to verify that if a feature attribute entity has a property providing an additional identifier for the attribute, it is named 'code' and has a string value.
Dependent requirements class	http://standards.iso.org/iso/19110/1.1/req/association	
Test	Identifier	**/conf/conceptual-model/association/inheritance**
	Requirement	/req/association/inheritance
	Test method	Inspect the model to verify that feature association entities have all of the properties and associations that are defined on the property type entity.
Test	Identifier	**/conf/conceptual-model/association/association-participation**
	Requirement	/req/association/association-participation
	Test method	Inspect the model to verify that feature association entities are required to have at least two associated association role instances via a 'roleName' role.
Test	Identifier	**/conf/conceptual-model/association/role-cardinality**
	Requirement	/req/association/role-cardinality
	Test method	Inspect the model to verify that association role classes have a cardinality property.
Test	Identifier	**/conf/conceptual-model/association/association-role-inheritance**
	Requirement	/req/association/association-role-inheritance
	Test method	Inspect the model to verify that association role entities have all of the properties and associations that are defined on the property type class.
Test	Identifier	**/conf/conceptual-model/association/role-relation**
	Requirement	/req/association/role-relation
	Test method	Inspect the model to verify that association role entities all have a mandatory association to exactly one feature association through a 'relation' role.

Table A.1 *(continued)*

Identifier	http://standards.iso.org/iso/19110/1.1/conf/conceptual-model	
Standardization target	Conceptual model	
Name	Conceptual model for a feature catalogue	
Requirements	http://standards.iso.org/iso/19110/1.1/req/conceptual-model	
Test	Identifier	**/conf/conceptual-model/association/role-player**
	Requirement	/req/association/role-player
	Test method	Inspect the model to verify that association role entities have a mandatory association to one feature type entities through a 'rolePlayer' role. This association may be either directly from the association role to FeatureType, or indirectly through a binding entity that has a 'globalProperty' role association to the association role entity.
Test	Identifier	**/conf/conceptual-model/association/role-type**
	Requirement	/req/association/role-type
	Test method	Inspect the model to verify that association role entities have a mandatory 'type' property with a value restricted to the enumeration {ordinary, aggregation, composition}.
Test	Identifier	**/conf/conceptual-model/association/role-is-ordered**
	Requirement	/req/association/role-is-ordered
	Test method	Inspect the model to verify that association role entities have a mandatory 'isOrdered' property with a Boolean value.
Test	Identifier	**/conf/conceptual-model/association/role-is-navigable**
	Requirement	/req/association/role-is-navigable
	Test method	Inspect the model to verify that association role entities have a mandatory 'isNavigable' property with a Boolean value.
Dependent requirements class	http://standards.iso.org/iso/19110/1.1/req/inheritance	
Test	Identifier	**/conf/conceptual-model/inheritance/class**
	Requirement	/req/inheritance/class
	Test method	Inspect the model to verify that inheritance relationships between feature types are represented by an inheritance relation entity, and that inheritance relation entities associate one feature type in a 'subtype' role with a feature type in a 'supertype' role.
Test	Identifier	**/conf/conceptual-model/inheritance/description**
	Requirement	/req/inheritance/description
	Test method	Inspect the model to verify that inheritance relation entities have a mandatory 'description' property.
Identifier	http://standards.iso.org/iso/19110/1.1/conf/conceptual-model	
Standardization target	Conceptual model	
Name	Conceptual model for a feature catalogue	
Requirements	http://standards.iso.org/iso/19110/1.1/req/conceptual-model	
Test	Identifier	**/conf/conceptual-model/inheritance/unique-instance**
	Requirement	/req/inheritance/unique-instance
	Test method	Inspect the model to verify that inheritance relation entities have a mandatory, Boolean 'uniqueInstance' property.
Dependent requirements class	http://standards.iso.org/iso/19110/1.1/req/global	

Table A.1 *(continued)*

Test	Identifier	**/conf/conceptual-model/global/global-property**
	Requirement	/req/global/global-property
	Test method	Inspect the model to verify that there is a constraint that the same property type instance may not fill both a 'featureCatalogue' role and a 'featureType' role.
Test	Identifier	**/conf/conceptual-model/global/binding**
	Requirement	/req/global/binding
	Test method	Inspect the model to verify that a binding entity associates a property type entity in a 'globalProperty' role to a feature type entity in a 'featureType' role.
Test	Identifier	**/conf/conceptual-model/global/binding-description**
	Requirement	/req/global/binding-description
	Test method	Inspect the model to verify that if a binding entity has a property intended to provide a description of the binding, it is named 'description' and has a string value.
Test	Identifier	**/conf/conceptual-model/global/global-xor-local**
	Requirement	/req/global/global-xor-local
	Test method	Inspect the model to verify that there is a constraint that a property type that plays the 'globalProperty' role on a binding entity may not also play a 'carrierOfCharacteristics' role on a feature type. Inspect the model to verify that there is a constraint that a property type that plays the 'globalProperty' role on a binding entity is required to be associated to a feature catalogue entity through a 'featureCatalogue' role.
Identifier	http://standards.iso.org/iso/19110/1.1/conf/conceptual-model	
Standardization target	Conceptual model	
Name	Conceptual model for a feature catalogue	
Requirements	http://standards.iso.org/iso/19110/1.1/req/conceptual-model	
Test	Identifier	**/conf/conceptual-model/global/bound-association-role**
	Requirement	/req/global/bound-association-role
	Test method	Inspect the model to verify that if there is a bound association role entity it has a mandatory association to a feature type via a 'rolePlayer' role and a mandatory association to an association role via a 'globalProperty' role, and that it inherits all properties and associations from the binding entity. Inspect the model to verify that there is a constraint that if an association role instance fills the 'globalProperty' role from a bound association role instance, it may not also have a direct association to a feature type via a 'rolePlayer' role.
Test	Identifier	**/conf/conceptual-model/global/bound-feature-attribute**
	Requirement	/req/global/bound-feature-attribute
	Test method	Inspect the model to verify that if there is a bound feature attribute entity, it has a mandatory association to a feature attribute via a 'globalProperty' role and a mandatory association to a feature type via a 'rolePlayer' role, and that it inherits all properties and associations from the binding entity.
Test	Identifier	**/conf/conceptual-model/global/binding-constraints**
	Requirement	/req/global/binding-constraints
	Test method	Inspect the model to verify that if the model includes constraints on binding entities, the representation is via a 'constrainedBy' role to a Constraints entity that has a 'description' property with a string value.

Table A.1 *(continued)*

Dependent requirements class	http://standards.iso.org/iso/19110/1.1/req/operation	
Test	Identifier	**/conf/conceptual-model/operation/inheritance**
	Requirement	/req/operation/inheritance
	Test method	Inspect the model to verify that Operation entities have all of the properties and associations that are defined on the property type class.
Test	Identifier	**/conf/conceptual-model/operation/affected-features**
	Requirement	/req/operation/affected-features
	Test method	Inspect the model to verify that Operation entities have an optional association to a feature type entity with the role 'featureType' to link the operation to feature types that it affects.
Identifier	http://standards.iso.org/iso/19110/1.1/conf/conceptual-model	
Standardization target	Conceptual model	
Name	Conceptual model for a feature catalogue	
Requirements	http://standards.iso.org/iso/19110/1.1/req/conceptual-model	
Test	Identifier	**/conf/conceptual-model/operation/operation-attributes**
	Requirement	/req/operation/operation-attributes
	Test method	Inspect the model to verify that if an Operation entity is associated with another entity via 'observesValueOf', 'affectsValuesOf' or 'triggeredByValuesOf' association roles, the target of that association is a feature attribute or a binding entity that has a 'globalProperty' association to a feature attribute.
Test	Identifier	**/conf/conceptual-model/operation/signature**
	Requirement	/req/operation/signature
	Test method	Inspect the model to verify that Operation entities have a 'signature' property.
Test	Identifier	**/conf/conceptual-model/operation/operation-cardinality**
	Requirement	/req/operation/operation-cardinality
	Test method	Inspect the model to verify that Operation entities have a 'cardinality' property, and that the default value for this property is '1'.
Test	Identifier	**/conf/conceptual-model/operation/formal-definition**
	Requirement	/req/operation/formal-definition
	Test method	Inspect the model to verify that if an Operation entities has a property defined with the intention of providing a formal textual definition of the operation, then that property is named 'formalDefinition'.
Test	Identifier	**/conf/conceptual-model/operation/functional-language**
	Requirement	/req/operation/functional-language
	Test method	Inspect the model to verify that if any Operation entity has a 'formalDefinition' property, the feature catalogue entity that contains the operation has a 'functional language' property.
Dependent requirements class	http://standards.iso.org/iso/19110/1.1/req/value-list	
Test	Identifier	**/conf/conceptual-model/value-list/class**
	Requirement	/req/value-list/class
	Test method	Inspect the model to verify that if a feature attribute entity is modelled to have enumerated values as a domain, the list of values is modelled as a listed value entity that fills a 'listed value' role on the feature attribute entity.
Identifier	http://standards.iso.org/iso/19110/1.1/conf/conceptual-model	
Standardization target	Conceptual model	

Table A.1 *(continued)*

Name	Conceptual model for a feature catalogue	
Requirements	http://standards.iso.org/iso/19110/1.1/req/conceptual-model	
Test	Identifier	**/conf/conceptual-model/value-list/label**
	Requirement	/req/value-list/label
	Test method	Inspect the model to verify that if a listed value entity is included, it has a mandatory 'label' property with a string value.
Test	Identifier	**/conf/conceptual-model/value-list/definition**
	Requirement	/req/value-list/definition
	Test method	Inspect the model to verify that if a listed value entity includes a property intended to provide a definition of the value that property is named 'definition'.
Test	Identifier	**/conf/conceptual-model/value-list/code**
	Requirement	/req/value-list/code
	Test method	Inspect the model to verify that if a listed value entity includes a property intended to provide an additional identifier for the value, the property is named 'code' and has a text value.

A.3 XML schema conformance class

Because the XML schema is generated by a rule-based process from the UML model presented in Annex B, the schema so generated is considered a part of this specification and is used for tests to validate XML instance documents. Because the schema is presented as part of this document, no conformance tests are necessary. XML schema generation is described in detail in Annex C.

A.4 XML instance document conformance class

Table A.2 — Conformance class: XML instance document

Identifier	http://standards.iso.org/iso/19110/1.1/conf/xml-instance	
Standardization target	XML instance document	
Name	XML instance document for feature catalogue interchange	
Requirements	http://standards.iso.org/iso/19110/1.1/req/xml-instance	
Test	Identifier	**/conf/xml-instance/validation**
	Requirement	/req/xml-instance/validation
	Test method	Validate document using gfc.xsd and gfc.sch.
Identifier	http://standards.iso.org/iso/19110/1.1/conf/xml-instance	
Standardization target	XML instance document	
Name	XML instance document for feature catalogue interchange	
Requirements	http://standards.iso.org/iso/19110/1.1/req/xml-instance	
Test	Identifier	**/conf/xml-instance/property-type-content**
	Requirement	/req/xml-instance/property-type-content
	Test method	Validate document using gfc.sch.
Test	Identifier	**/conf/xml-instance/unique-type-name**
	Requirement	/req/xml-instance/unique-type-name
	Test method	Validate document using gfc.sch.

Table A.2 *(continued)*

Test	Identifier	**/conf/xml-instance/unique-code-value**
	Requirement	/req/xml-instance/unique-code-value
	Test method	Validate document using gfc.sch.
Test	Identifier	**/conf/xml-instance/unique-property-name**
	Requirement	/req/xml-instance/unique-property-name
	Test method	Validate document using gfc.sch.
Test	Identifier	**/conf/xml-instance/definition-language**
	Requirement	/req/xml-instance/definition-language
	Test method	Inspect instance document to determine that all definitions are expressed in a natural language.
Test	Identifier	**/conf/xml-instance/unique-subtype-instance**
	Requirement	/req/xml-instance/unique-subtype-instance
	Test method	Validate document using gfc.sch.
Test	Identifier	**/conf/xml-instance/functional-language-specified**
	Requirement	/req/xml-instance/functional-language-specified
	Test method	Inspect the instance document to determine that the functional language specified in FC_FeatureCatalogue.functionalLanguage is the language used to populate all FC_FeatureOperation.formalDefinition properties that are not null or absent.
Test	Identifier	**/conf/xml-instance/definition-source**
	Requirement	/req/xml-instance/definition-source
	Test method	Validate document using gfc.sch
Test	Identifier	**/conf/xml-instance/value-list-label-unique**
	Requirement	/req/xml-instance/value-list-label-unique
	Test method	Validate document using gfc.sch
Identifier	http://standards.iso.org/iso/19110/1.1/conf/xml-instance	
Standardization target	XML instance document	
Name	XML instance document for feature catalogue interchange	
Requirements	http://standards.iso.org/iso/19110/1.1/req/xml-instance	
Test	Identifier	**/conf/xml-instance/role-is-ordered**
	Requirement	/req/xml-instance/role-is-ordered
	Test method	Validate document using gfc.sch; inspect the 'definition' property value for all FC_AssociationRole elements to determine that the text explains the meaning of the order.

Annex B
(normative)

Feature catalogue conceptual schema and data dictionary

B.1 Introduction

This annex presents the conceptual and data dictionary for the organization of feature catalogue information according to this document. It is composed of

— a conceptual schema described in B.2 as a set of figures (Figure B.1 to B.7), and

— a data dictionary described in B.3 as a set of tables (Tables B.1 through B.18). The table rows describe the feature catalogue information elements. For each element, the following information is provided.

  — **No** is a unique number associated with the element.

  — ***Element Type*/Label** defines the designation of the element preceded by the element type (Class, Attribute, Role).

  — **Definition** is a synthetic description of the element.

  — **Obligation/Condition** defines the obligation and conditions of presence of properties (roles and attributes) with respect to their related classes.

    M – The element is mandatory; it shall be included in the feature catalogue. If a minimum number of occurrences is expected, it is noted (default is 1).

    C – The element is conditional; the condition is stated as a question. If the answer to the question is yes, the element shall be included in the feature catalogue.

    O – The element is optional; if an element is included in the feature catalogue, mandatory sub-elements of the element shall also be included.

  — **Maximum occurrence** is 1 when the property cannot be repeated and N when it can be repeated. It is not applicable to classes.

  — **Data Type** defines the type of a property. It is not applicable to classes.

  — **Domain** expresses

    — the domain of value of attributes and/or their initial values; free text enables the instantiation of a CharacterString property as free text, i.e. enables the provision of translations of the character string in different locales,

    — whether a role is or not involved in a composition or an aggregation, and

    — additional information about Classes.

Figures B.1 to B.3 illustrate the feature catalogue conceptual schema in the form of a Unified Modeling Language (UML) package (ISO 19103). Figures B.4 to B.7 illustrate how the structure of a standard feature catalogue conforms to the General Feature Model (ISO 19109). Examples of feature cataloguing is given in Annex E. Feature cataloguing concepts are described in Annex F.

Both the data dictionary and the conceptual schema are normative, but in case of inconsistency, the conceptual schema takes precedence.

B.2 Conceptual schema

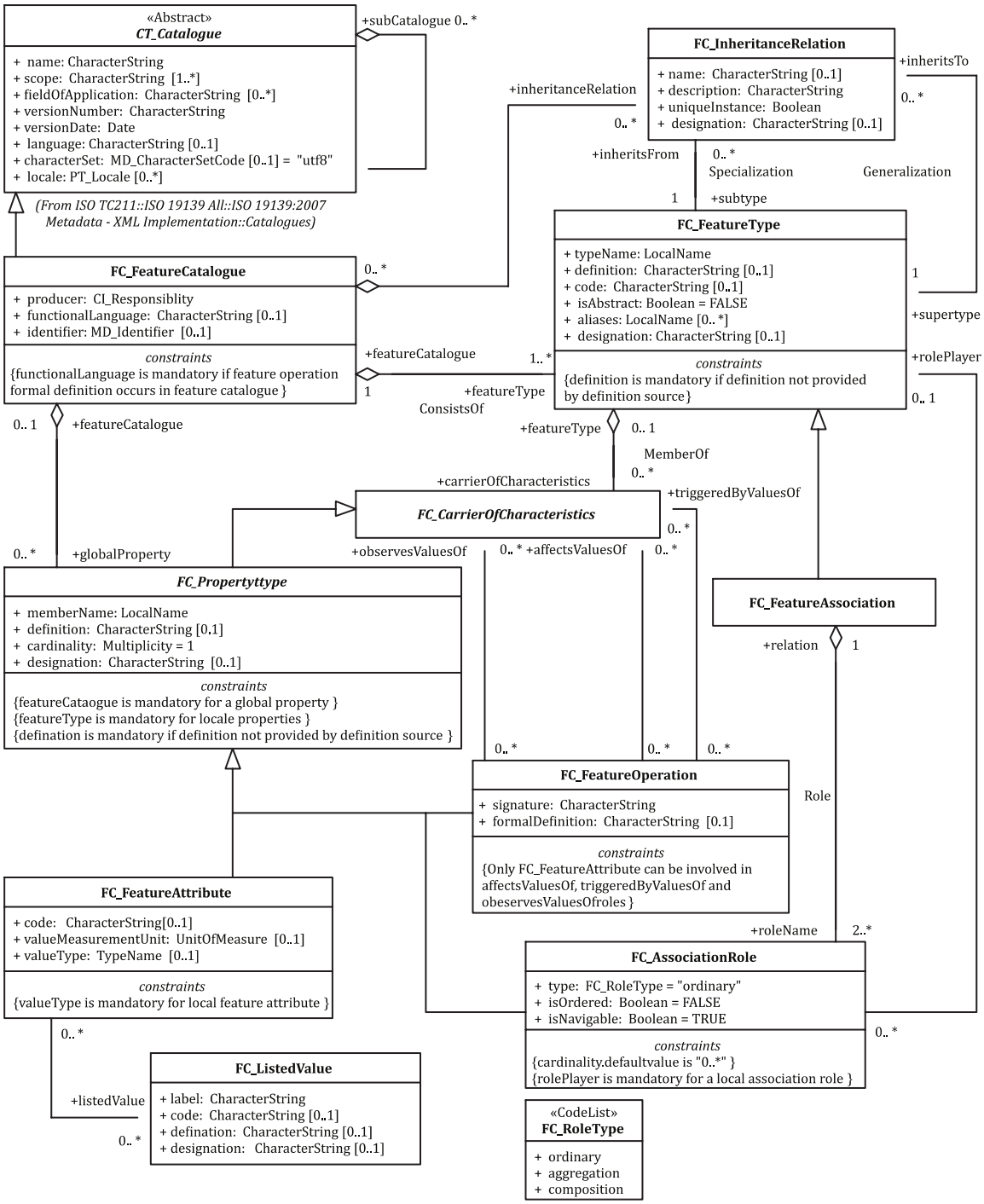

Figure B.1 — Conceptual model of a feature catalogue

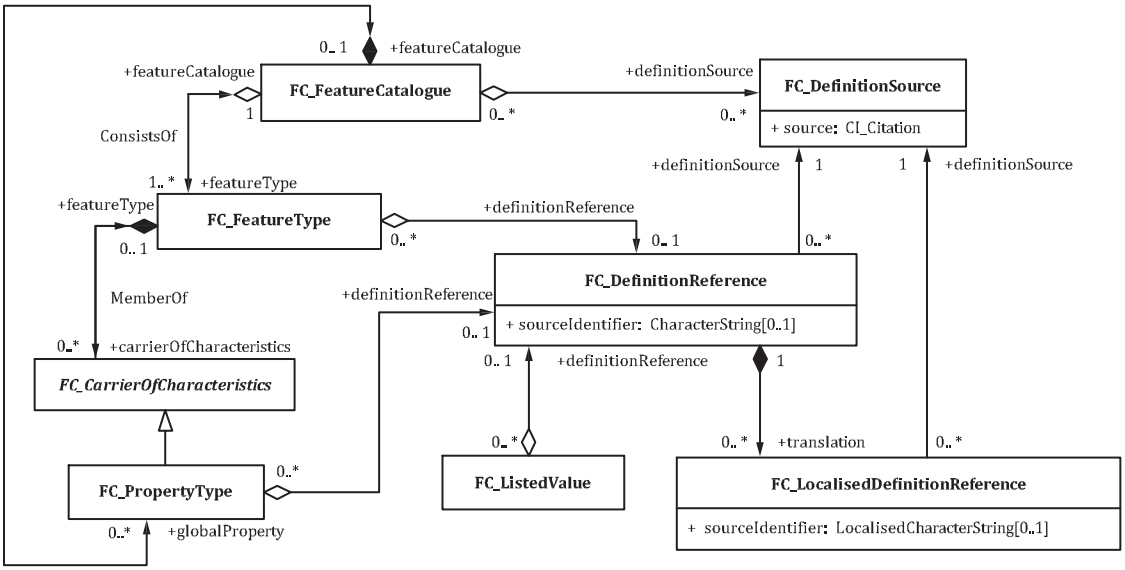

Figure B.2 — Definition source and reference

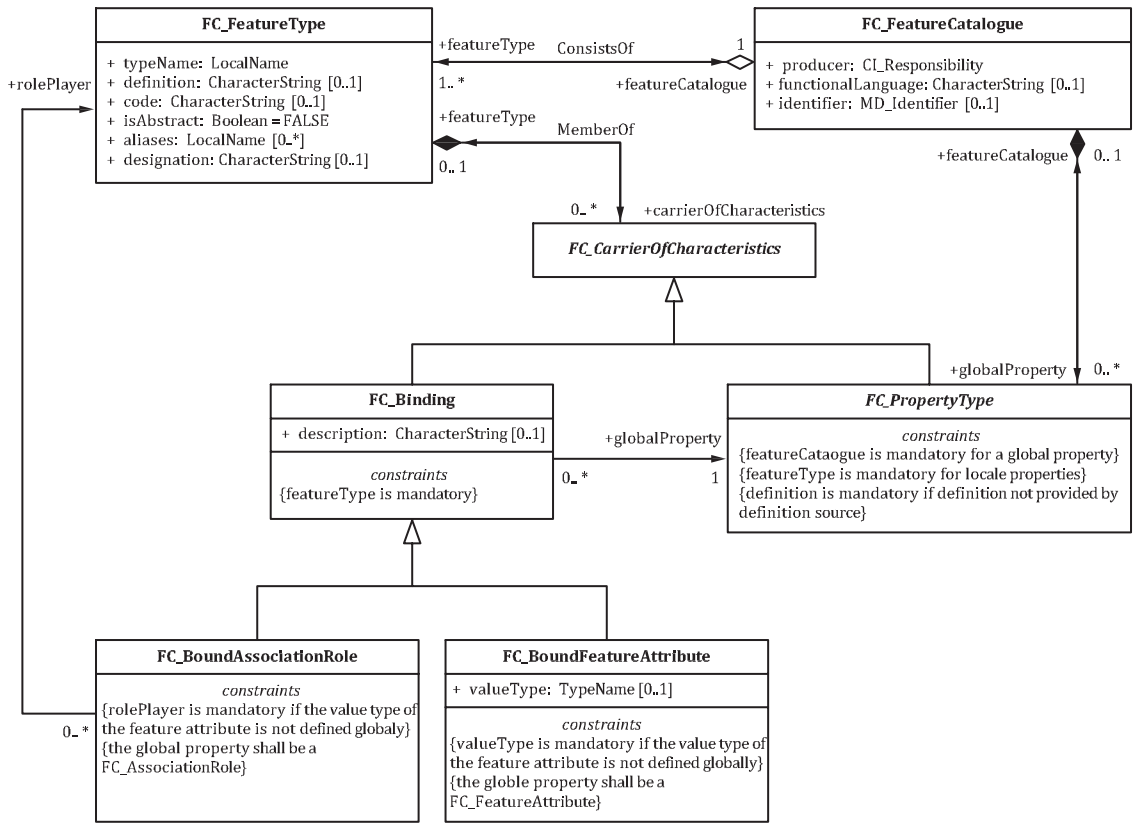

Figure B.3 — Global properties

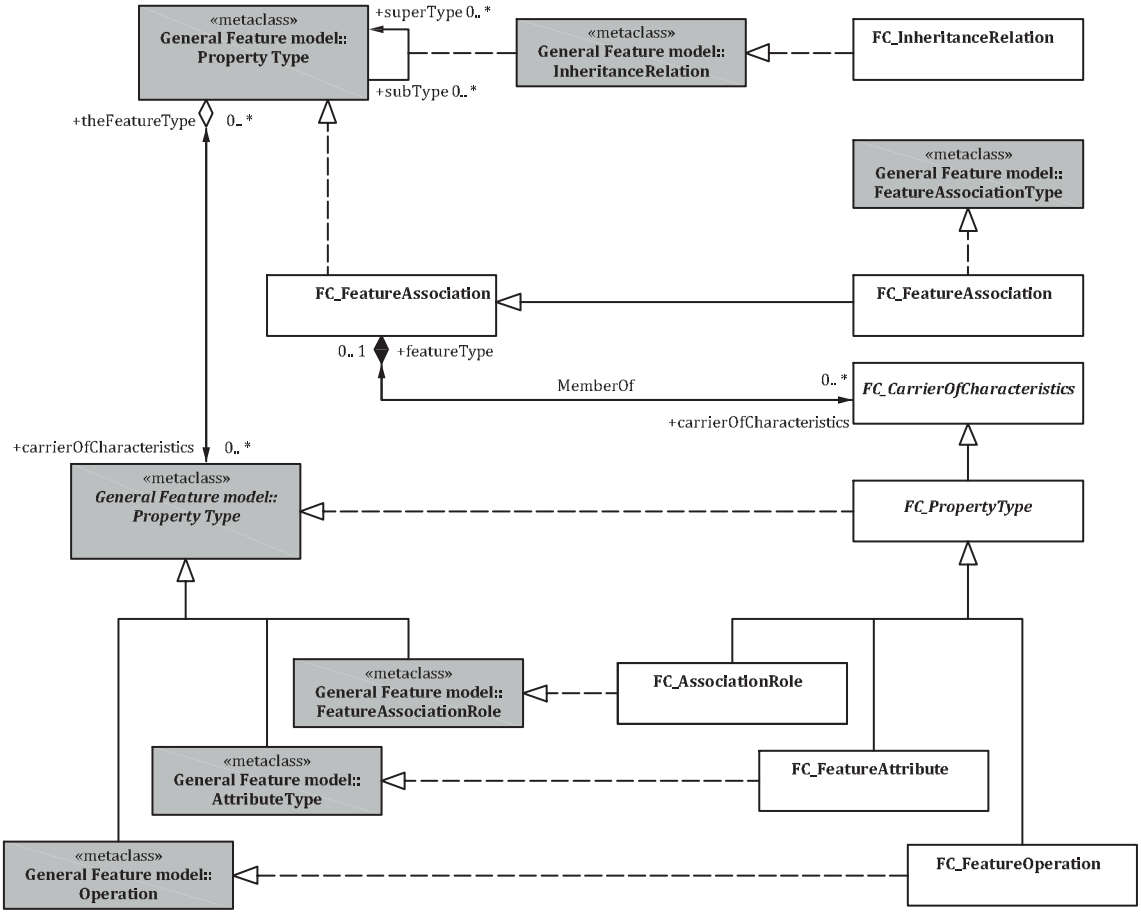

Figure B.4 — Feature cataloguing classes as realizations of General Feature Model metaclasses

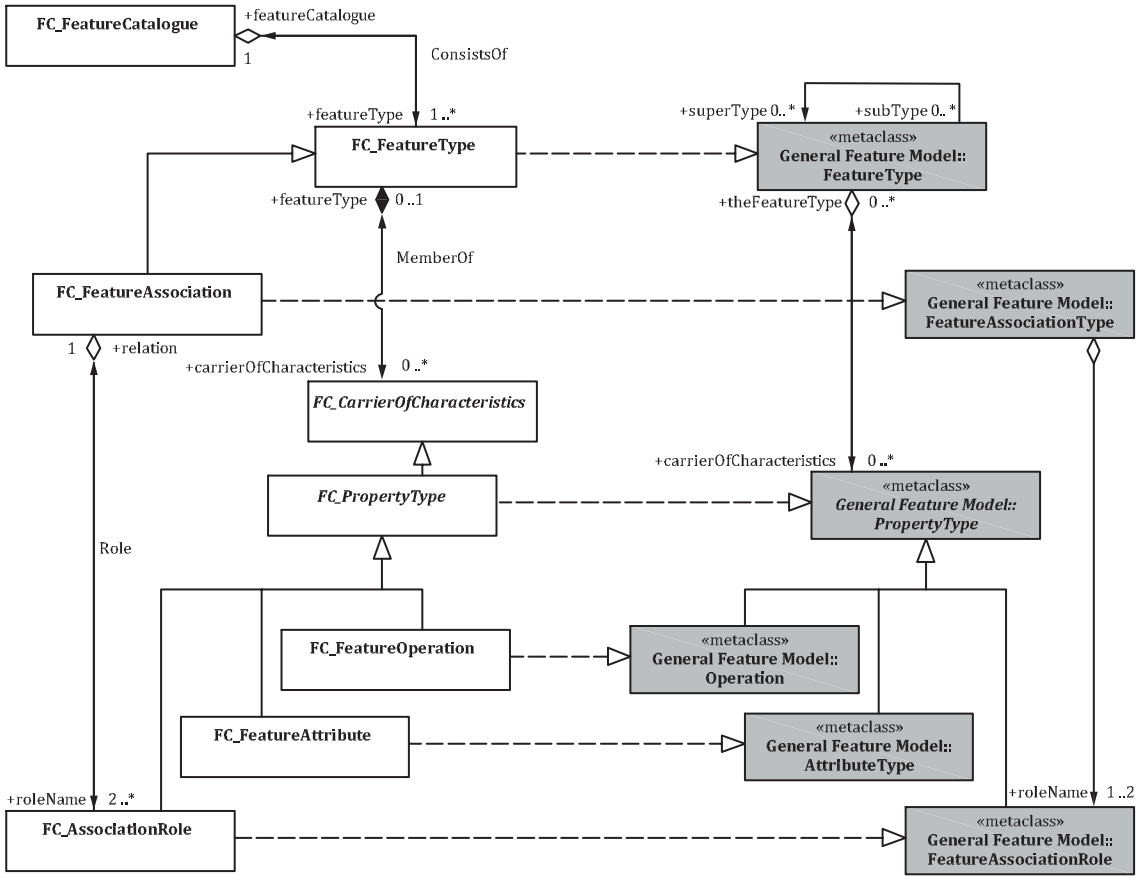

Figure B.5 — Derivation of FC_FeatureType, FC_FeatureAttribute, FC_FeatureAssociation and FC_AssociationRole from GFM metaclasses

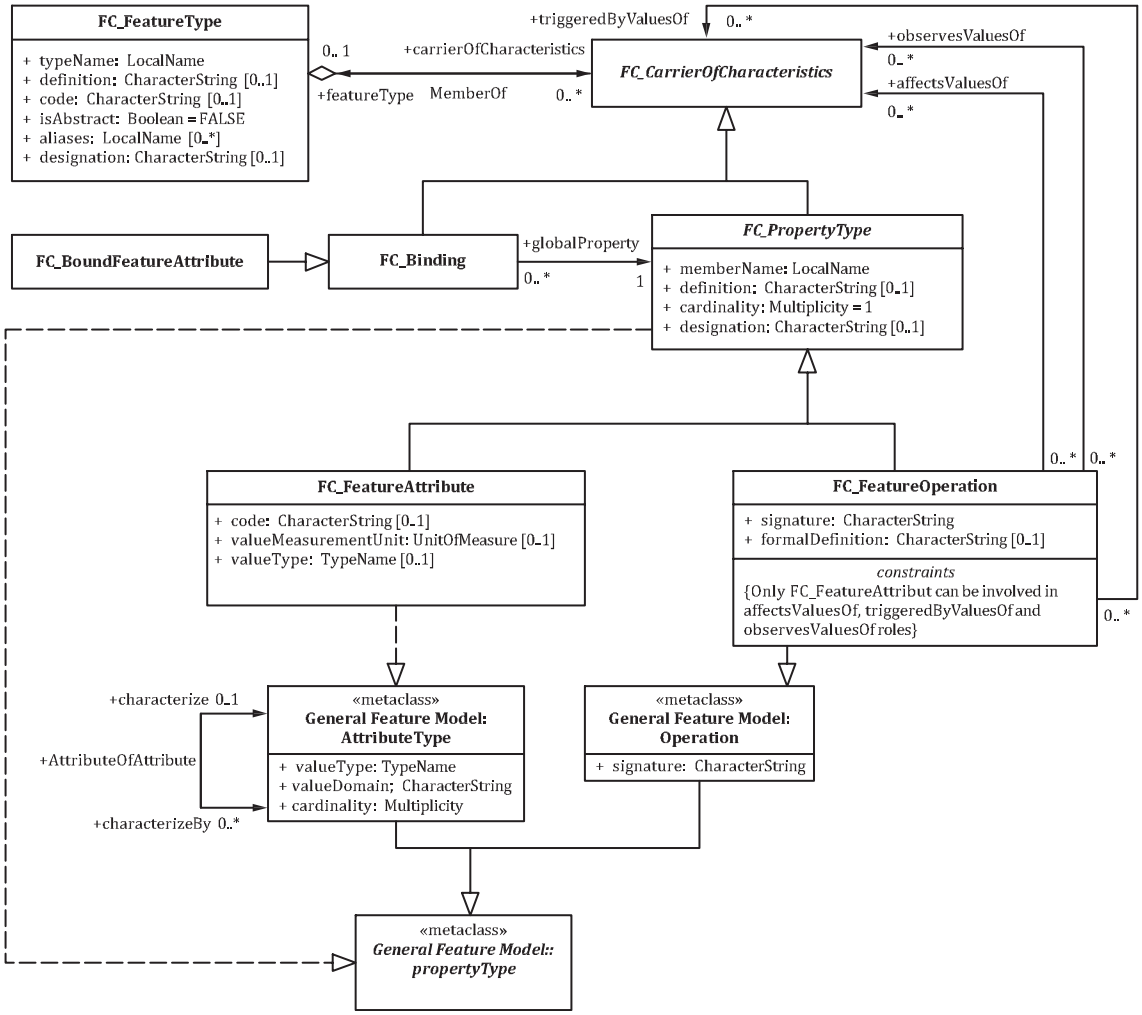

Figure B.6 — Derivation of FC_FeatureOperation from the GF_Operation metaclass

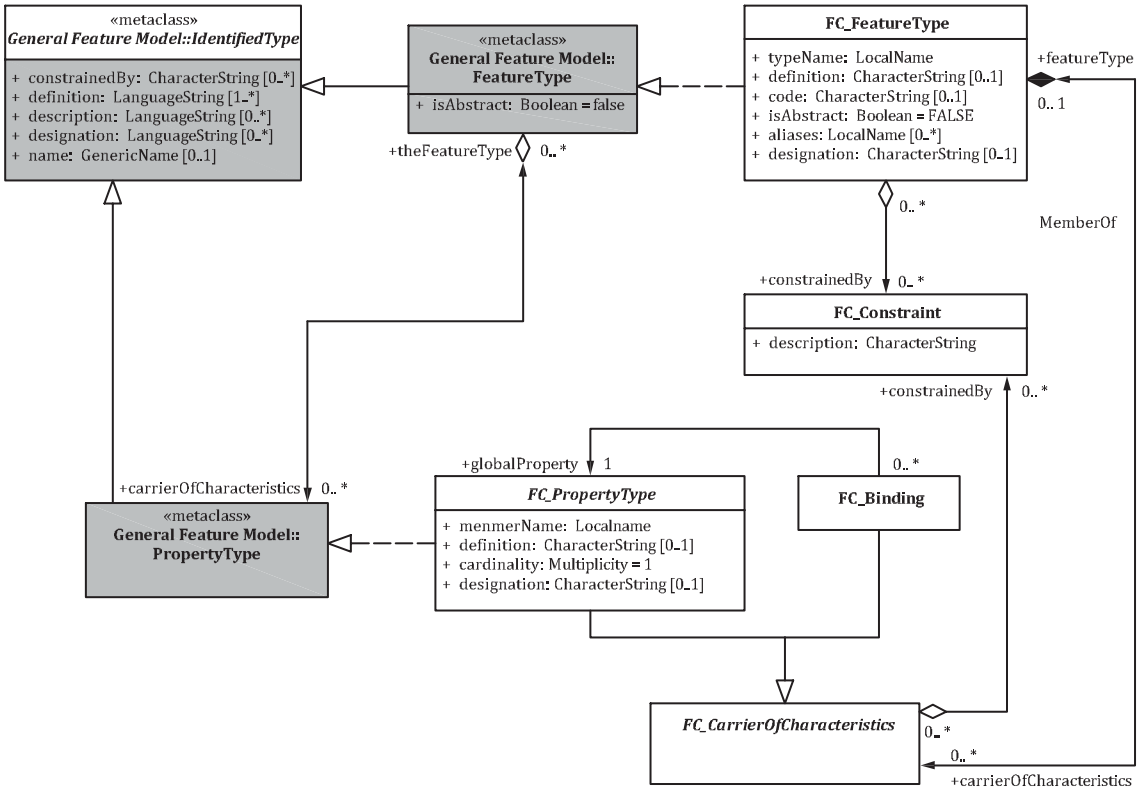

Figure B.7 — Derivation of FC_Constraint from the GF_Constraint metaclass

B.3 Data dictionary

Table B.1 describes the properties of the FC_FeatureCatalogue class as described in Figures B.1 and B.2.

Table B.1 — Feature catalogue

No.	*Element type*/ Label	Definition	Obligation/condition	Maximum occurrence	Data type	Domain
1	*Class* FC_FeatureCatalogue	feature catalogue containing the definition of some number feature types with other information necessary for their definitions	—	—	—	—
	Subtype of CT_Catalogue	See ISO/TS 19139 [Inherits properties and relations from CT_Catalogue]	—	—	—	—
1.1	*Attribute* producer	name, address, country, and telecommunications address of person or organization having primary responsibility for the intellectual content of this feature catalogue	M	1	ISO 19115-1 Metadata fundamentals:CI_Responsibility	—

Table B.1 *(continued)*

No.	*Element type*/ Label	Definition	Obligation/condition	Maximum occurrence	Data type	Domain
1.2	*Attribute* functionalLanguage	formal functional language in which the feature operation formal definition occurs in this feature catalogue	C/Mandatory if feature operation formal definition occurs in the feature catalogue.	1	CharacterString	Free text
1.3	*Attribute* identifier	identifier of the feature catalogue	O	1	ISO 19115-1 Metadata fundamentals::MD_Identifier	—
1.4	*Role* featureType	role that links this feature catalogue to the feature types that it contains	O	N	FC_FeatureType	Aggregation
1.5	*Role* definitionSource	role that links this feature catalogue to the sources of definitions of feature types, property types, and listed values that it contains	O	N	FC_DefinitionSource	Aggregation
1.6	*Role* inheritanceRelation	role that links this feature catalogue to the inheritance relationships that it contains	O	N	FC_InheritanceRelation	Aggregation
1.7	*Role* globalProperty	role that links this feature catalogue to the global feature properties, i.e. the feature properties which may bound to many feature types	O	N	FC_PropertyType	Composition

Table B.2 describes the properties of the FC_FeatureType class as described in Figures B.1 to B.3.

Table B.2 — FeatureType

No.	*Element type*/ Label	Definition	Obligation/ condition	Maximum occurrence	Data type	Domain
2	*Class* FC_FeatureType	class of real-world phenomena with common properties	—	—	—	typeName realizes GF_FeatureType::typeName; isAbstract realizes GF_FeatureType::isAbstract; constrainedBy realizes GF_FeatureType::-constrainedBy
2.1	*Attribute* typeName	text string that uniquely identifies this feature type within the feature catalogue that contains this feature type	M	1	LocalName	—
2.2	*Attribute* definition	definition of the feature type in a natural language. This attribute is required if the definition is not provided by FC_FeatureCatalogue::definitionSource. If not provided, the definitionReference should specify a citation where the definition may be found, and any additional information as to which definition is to be used.	C/Mandatory if definition not provided by definition source.	1	CharacterString	Free text
2.3	*Attribute* code	code that uniquely identifies this feature type within the feature catalogue that contains this feature type	O	1	CharacterString	—
2.4	*Attribute* isAbstract	indicates if the feature type is abstract or not	M	1	Boolean	Initial value = FALSE
2.5	*Attribute* aliases	equivalent name(s) of this feature type	O	N	LocalName	—
2.6	*Attribute* designation	designation of the feature type in a natural language	O	1	CharacterString	Free text
2.7	*Role* inheritsFrom	role that links this feature type to a set of superclasses from which it inherits operations, associations, and properties	O	N	FC_Inheritance Relation	—
2.8	*Role* inheritsTo	role that links this feature type to a set of subclasses which inherit its operations, associations, and properties	O	N	FC_Inheritance Relation	—

Table B.2 *(continued)*

No.	*Element type*/ Label	Definition	Obligation/ condition	Maximum occurrence	Data type	Domain
2.9	*Role* featureCatalogue	role that links this feature type to the feature catalogue that contains it	M	1	FC_Feature Catalogue	—
2.10	*Role* carrierOf Characteristics	role that links this feature type to the property types that it contains	O	N	FC_CarrierOf Characteristics	Composition
2.11	*Role* constrainedBy	role that links this feature type to the constraints placed upon it	O	N	FC_Constraint	—
2.12	*Role* definitionReference	role that links this feature type to the source of its definition	O	1	FC_Definition Reference	Aggregation

Table B.3 describes the properties of the FC_InheritanceRelation class as described in Figure B.1.

Table B.3 — Inheritance relation

No.	*Element type*/ Label	Definition	Obligation/ condition	Maximum occurrence	Data type	Domain
3	*Class* FC_Inheritance Relation	FC_InheritanceRelation realizes GF_InheritanceRelation	—	—	—	FC_Inheritance Relation always assumes that its GF_Inheritance Relation::uniqueInstance is TRUE.
3.1	*Attribute* name	text string that uniquely identifies this inheritance relation within the feature catalogue that contains this inheritance relation	O	1	CharacterString	—
3.2	*Attribute* description	natural language description of this inheritance relation	M	1	CharacterString	Free text
3.3	*Attribute* uniqueInstance	indicates if an instance of the supertype can be an instance of at most one of its subtypes	M	1	Boolean	—

Table B.3 *(continued)*

No.	*Element type*/ Label	Definition	Obligation/ condition	Maximum occurrence	Data type	Domain
3.4	*Attribute* designation	designation of the inheritance relation in a natural language	O	1	CharacterString	Free text
3.5	*Role* subtype	identifies one feature type to which the associated superclass feature type supplies inherited properties, associations, and operation	M	1	FC_FeatureType	—
3.6	*Role* supertype	identifies one feature type to which the associated subtype class inherits properties, associations, and operations	M	1	FC_FeatureType	—

Table B.4 describes the properties of the FC_PropertyType class as described in Figures B.1 and B.2.

Table B.4 — Property type

No.	*Element type*/ Label	Definition	Obligation/ condition	Maximum occurrence	Data type	Domain
4	*Class* FC_PropertyType	abstract class for local and global feature properties	—	—	—	—
	Subtype of FC_ CarrierOf Characteristics	Table B.5	—	—	—	—
4.1	Attribute memberName	member name that locates this member within a feature type for a local property or within the feature catalogue for a globalProperty	M	1	LocalName	—
4.2	*Attribute* definition	definition of the member in a natural language: This attribute is required if the definition is not provided by FC_FeatureCatalogue::-definitionSource. If not provided, the definitionReference should specify a citation where the definition may be found, and any additional information as to which definition is to be used.	C/Mandatory if not provided by definition source	1	CharacterString	Free text

Table B.4 *(continued)*

No.	*Element type*/ Label	Definition	Obligation/ condition	Maximum occurrence	Data type	Domain
4.3	*Attribute* cardinality	cardinality of the member in the feature class. If this is an attribute or operation, the default cardinality is 1. If this is an association role, then the default cardinality is 0..*. For operations, this is the number of return values possible. This is an elaboration of the GFM to allow for complete specifications for various programming and data definition languages.	M	1	Multiplicity	Initial value = 1
4.4	*Attribute* designation	designation of the feature property in a natural language	O	1	CharacterString	Free text
4.5	*Role* definitionReference	role that links this instance to the source of its definition	O	1	FC_Definition Reference	Aggregation
4.6	*Role* featureCatalogue	feature catalogue to which a globalProperty pertains	C/Mandatory for a globalProperty	1	FC_FeatureCatalogue	—

Table B.5 describes the properties of the FC_CarrierOfCharacteristics class as described in Figures B.1 and B.7.

Table B.5 — Carrier of characteristics

No.	*Element type*/ Label	Definition	Obligation/ condition	Maximum occurrence	Data type	Domain
5	*Class* FC_CarrierOf Characteristics	abstract class for local feature properties and bound global properties of a feature type	—	—	—	—
5.1	*Role* featureType	role that links the local and bound properties with the feature type that contain them	C/Mandatory for local properties	1	FC_FeatureType	—
5.2	*Role* constrainedBy	role that links a carrier of characteristics to the constraints placed upon it	O	N	FC_Constraint	Aggregation

Table B.6 describes the properties of the FC_FeatureOperation class as described in Figures B.1 and B.6.

Table B.6 — Feature operation

No.	*Element type*/ Label	Definition	Obligation/ condition	Maximum occurrence	Data type	Domain
6	*Class* FC_FeatureOperation	operation that every instance of an associated feature type must implement	—	—	—	triggeredByValuesOf realizes GF_Operation::triggeredByValuesOf; observesValuesOf realizes GF_Operation::observesValuesOf; affectsValuesOf realizes GF_Operation::affectsValuesOf
	Subtype of FC_PropertyType	Table B.4	—	—	—	—
6.1	*Attribute* signature	name and parameters for this operation. It may contain optional returned parameters. This signature is usually derived from the formalDefinition. The signature of an operation must be unique. This is the equivalent of the UML signature.	M	1	CharacterString	—
6.2	*Attribute* formalDefinition	formal description of the behaviour of the member, expressed in the symbol set defined by FC_FeatureCatalogue::functionalLanguage; involves operational parameters, and interactions with other members of the feature type	O	1	CharacterString	—

Table B.6 *(continued)*

No.	*Element type*/ Label	Definition	Obligation/ condition	Maximum occurrence	Data type	Domain
6.3	*Role* triggeredByValuesOf	specifies attribute which may trigger an operation	O	N	FC_CarrierOf Characteristics	Shall be instantiated as FC_FeatureAttribute, FC_BoundFeature Attribute or on of their derived classes.
6.4	*Role* observesValuesOf	specifies attribute that may be used as input to perform an operation	O	N	FC_CarrierOf-Characteristics	Shall be instantiated as FC_FeatureAttribute, FC_BoundFeature Attribute or on of their derived classes.
6.5	*Role* affectsValuesOf	specifies attribute that will be affected by an operation	O	N	FC_CarrierOf-Characteristics	Shall be instantiated as FC_FeatureAttribute, FC_BoundFeature Attribute or on of their derived classes.

Table B.7 describes the properties of the FC_Binding class as described in Figure B.3.

Table B.7 — Binding

No.	*Element type*/ Label	Definition	Obligation/ condition	Maximum occurrence	Data type	Domain
7	*Class* FC_Binding	class that is used to describe the specifics of how a property type is bound to a particular feature type	—	—	—	—
	Subtype of FC_CarrierOf Characteristics	Table B.5	—	—	—	—
7.1	*Attribute* description	description of how a property type is bound to a particular feature type	O	1	CharacterString	Free text
7.2	*Role* globalProperty	role that links to the bound globalProperty	M	1	FC_PropertyType	

Table B.8 describes the properties of the FC_Constraint class as described in Figure B.7.

Table B.8 — Constraint

No.	*Element type*/ Label	Definition	Obligation/ condition	Maximum occurrence	Data type	Domain
8	*Class* FC_Constraint	class for defining constraints for types	—	—	—	—
8.1	*Attribute* description	description of the constraint that is being applied	M	1	CharacterString	Free text

Table B.9 describes the properties of the FC_FeatureAttribute class as described in Figure B.1.

Table B.9 — Feature attribute

No.	*Element type*/ Label	Definition	Obligation/ condition	Maximum occurrence	Data type	Domain
9	*Class* FC_FeatureAttribute	characteristic of a feature type	—	—	—	—
	Subtype of FC_PropertyType	Table B.4	—	—	—	—
9.1	*Attribute* code	numeric or alphanumeric code that uniquely identifies the feature attribute within the feature catalogue	O	1	CharacterString	—
9.2	*Attribute* valueMeasurement Unit	unit of measure used for values of this feature attribute	O	1	UnitOfMeasure	—
9.3	*Attribute* valueType	type of the value of this feature attribute; a name from some namespace	C/Mandatory for local feature attribute	1	TypeName	—
9.4	*Role* listedValue	defines the permissible values of this feature attribute as a restriction of the attribute valueType	O	1	FC_ListedValue	—

Table B.10 describes the properties of the FC_AssociationRole class as described in Figure B.1.

Table B.10 — Association role

No.	*Element type*/ Label	Definition	Obligation/ condition	Maximum occurrence	Data type	Domain
10	*Class* FC_AssociationRole	role of the feature association FC_AssociationRole:: relation	—	—	—	roleName = FC_PropertyType:: memberName;
	Subtype of FC_PropertyType	Table B.4 — Property type	—	—	—	—
10.1	*Attribute* type	type of association role, indicating whether this role acts as a “is part of” or “is a member of” semantics	M	1	FC_RoleType	Initial value = 1 (“ordinary”)

Table B.10 *(continued)*

No.	*Element type*/ Label	Definition	Obligation/ condition	Maximum occurrence	Data type	Domain
10.2	*Attribute* isOrdered	indicates if the instances of this association role within the containing feature instance are ordered or not, with FALSE = not ordered" and TRUE = "ordered". If TRUE, the FC_PropertyType::definition shall contain an explanation of the meaning of the order	M	1	Boolean	Initial value = FALSE
10.3	*Attribute* isNavigable	indicates whether this role is navigable from the source feature to the target feature of the association	M	1	Boolean	Initial value = TRUE
10.4	*Role* relation	relation of which this association role is a part	M	1	FC_Feature Association	—
10.5	*Role* rolePlayer	type of the target value of this association role	C/ Mandatory for a local association role	1	FC_Feature Type	—

Table B.11 describes the values of FC_RoleType codelist as described in Figure B.1.

Table B.11 — Role type code list

No.	Concept name (English)	Code	Definition
11	Class FC_RoleType	—	code list for the classification of roles
11.1	Ordinary	ordinary	indicates an ordinary association
11.2	Aggregation	aggregation	indicates a UML aggregation (part role)
11.3	Composition	composition	indicates a UML composition (member role)

Table B.12 describes the properties of the FC_ListedValue class as described in Figure B.1.

Table B.12 — Listed value

No.	*Element type*/ Label	Definition	Obligation/ condition	Maximum occurrence	Data type	Domain
12	*Class* FC_ListedValue	value for an enumerated feature attribute domain, including its codes and interpretation	—	—	—	—
12.1	*Attribute* label	descriptive label that uniquely identifies one value of the feature attribute	M	1	CharacterString	—
12.2	*Attribute* code	numeric or alphanumeric code (such as a country code) that uniquely identifies this value of the feature attribute	O	1	CharacterString	—

Table B.12 *(continued)*

No.	*Element type*/ Label	Definition	Obligation/ condition	Maximum occurrence	Data type	Domain
12.3	*Attribute* definition	definition of the attribute value in a natural language. If not provided, the definitionReference may specify a citation where the definition may be found, and any additional information as to which definition is to be used.	O	1	CharacterString	Free text
12.4	*Attribute* designation	designation of the attribute value in a natural language	O	1	CharacterString	Free text
12.5	*Role* definitionReference	role that links this instance to the source of its definition	O	1	FC_DefinitionReference	Aggregation

Table B.13 describes the properties of the FC_FeatureAssociation class as described in Figure B.1.

Table B.13 — Feature association

No.	*Element type*/ Label	Definition	Obligation/ condition	Maximum occurrence	Data type	Domain
13	*Class* FC_Feature Association	relationship that links instances of this feature type with instances of the same or of a different feature type The memberOf-link-Between association in the General Feature Model is not directly implemented here since it can be easily derived from combining the Role and MemberOf associations.	—	—	—	—
	Subtype of FC_FeatureType	Table B.2	—	—	—	—
13.1	*Role* roleName	roles that are a part of this association	M (minimum occurrence: 2)	N	FC_AssociationRole	Composition

Table B.14 describes the properties of the FC_DefinitionSource class as described in Figure B.2.

Table B.14 — Definition source

No.	*Element type*/ Label	Definition	Obligation/ condition	Maximum occurrence	Data type	Domain
14	*Class* FC_DefinitionSource	class that specifies the source of a definition	—	—	—	—
14.1	*Attribute* source	actual citation of the source, sufficient to identify the document and how to obtain it	M	1	ISO 19115-1 Metadata fundamentals:: CI_Citation	—

Table B.15 describes the properties of the FC_DefinitionReference class as described in Figure B.2.

Table B.15 — Definition reference

No.	*Element type*/ Label	Definition	Obligation/ condition	Maximum occurrence	Data type	Domain
15	*Class* FC_Definition Reference	class that links a data instance to the source of its definition	—	—	—	—
15.1	*Attribute* sourceIdentifier	additional information to help locate the definition in the source document. The format of this information is specific to the structure of the source document.	O	1	CharacterString	—
15.2	*Role* definitionSource	role that links this definition reference to the citation for the source document	M	1	FC_DefinitionSource	—
15.3	*Role* translation	role that links this definition reference to translations	O	N	FC_Localised DefinitionReference	—

Table B.16 describes the properties of the FC_LocalisedDefinitionReference class as described in Figure B.2.

Table B.16 — Localised definition reference

No.	*Element type*/ Label	Definition	Obligation/ condition	Maximum occurrence	Data type	Domain
16	*Class* FC_Localised DefinitionReference	class that links a definition reference to a translation of the definition reference in an alternate source reference	—	—	—	—
16.1	*Attribute* sourceIdentifier	additional information to help locate the definition in the alternate source document This information is expressed in one of the feature catalogue locales.	O	1	LocalisedCharacter String	—
16.2	*Role* definitionSource	role that links this localised definition reference to the citation for the source document	M	1	FC_DefinitionSource	—

Table B.17 describes the properties of the FC_BoundFeatureAttribute class as described in Figure B.3.

Table B.17 — Bound feature attribute

No.	*Element type*/ Label	Definition	Obligation/ condition	Maximum occur-rence	Data type	Domain
17	*Class* FC_BoundFeature Attribute	class that is used to describe the specifics of how a global feature attribute is bound to a particular feature type	—	—	—	—
	Subtype of FC_ Binding	Table B.7	—	—	—	—
17.1	*Attribute* valueType	type of the value of this fea-ture attribute; a name from some namespace	C/Manda-tory if the value type of the feature attribute is not defined globally	1	TypeName	—

Table B.18 describes the properties of the FC_BoundAssociationRole class as described in Figure B.3.

Table B.18 — Bound association role

No.	*Element type*/ Label	Definition	Obligation/ condition	Maximum occur-rence	Data type	Domain
18	*Class* FC_BoundAssocia-tionRole	class that is used to describe the specifics of how a global association role is bound to a particular feature type	—	—	—	—
	Subtype of FC_Bind-ing	Table B.7	—	—	—	—
18.1	*Role* rolePlayer	target feature type of this association role	C/Man-datory if the target featureType this associ-ation role is not defined globally	1	FC_Feature-Type	—

Annex C
(normative)

Encoding description

C.1 Introduction

This XML schema implementation of this document follows the encoding rules stated in ISO/TS 19139:2007, Clause 8, using the patterns for decoupling XML namespaces outlined in ISO/TS 19115-3:2016, Clause 8 for better modularization of the ISO/TC 211 harmonized model. This is a revision of the XML schema implementation of ISO 19110 documented in a previous amendment.

C.2 XML namespaces

The XML schema definitions resulting from this XML schema implementation of this document shall pertain to the following namespace: http://standards.iso.org/iso/19110/gfc/1.1. This namespace is abbreviated **gfc**, which stands for **G**eographic **F**eature **C**ataloguing.

NOTE The XML schema implementation of ISO 19110 documented in a previous amendment pertained to http://www.isotc211.org/2005/gfc namespace which corresponded implicitly to version 1.0 of gfc. The elements of gfc 1.0 can be directly moved to gfc 1.1 namespace: gfc 1.1 is backward compatible. It provides additional elements for multilingual support.

Figure C.1 shows the different namespaces used to implement this document (grey boxes) along with the relationships between these namespaces and the ISO 19100 series packages (white boxes).

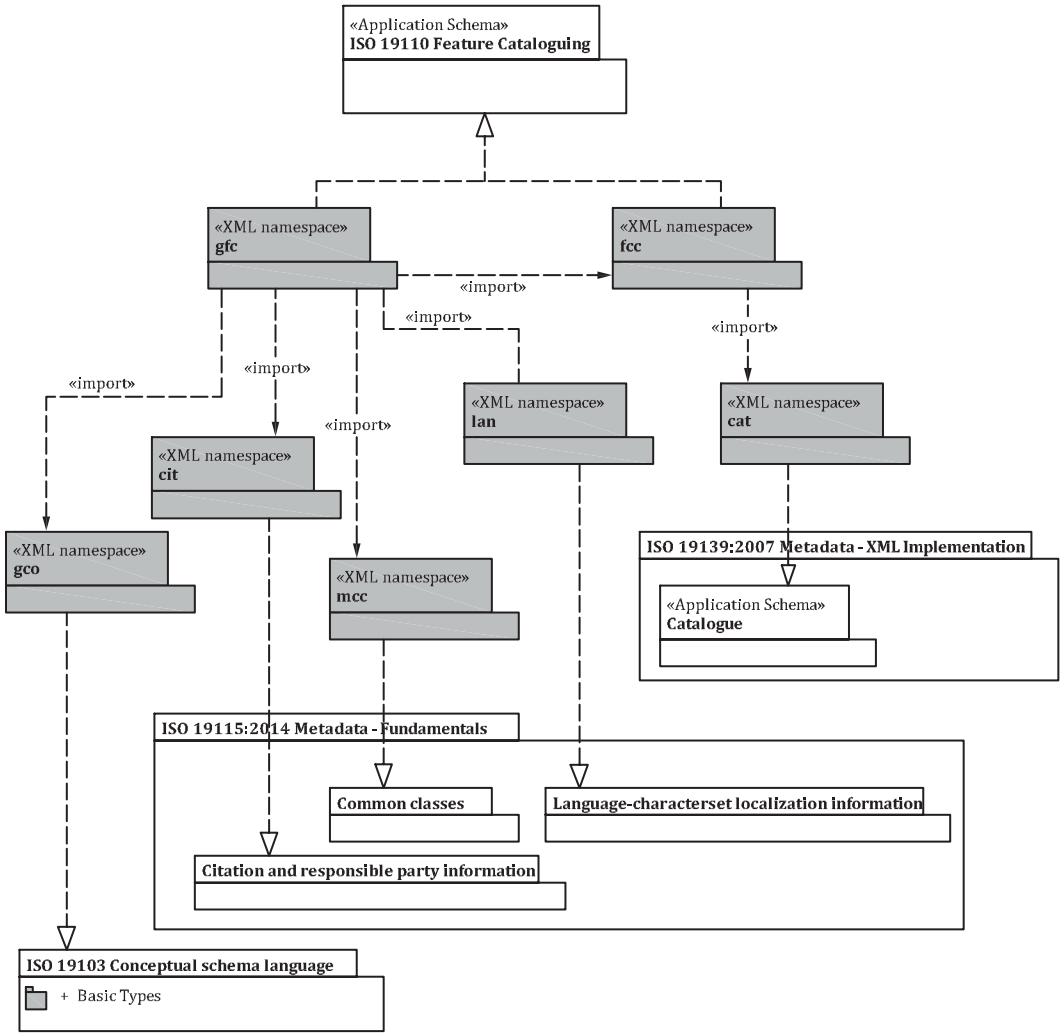

Figure C.1 — XML packaging

C.3 gfc namespace

C.3.1 Organization of the gfc namespace

The root and only XML Schema document providing the XML Schema definitions of the gfc namespace shall be gfc.xsd. This XML schema implements all the feature cataloguing concepts of this document as illustrated in Figure C.2.

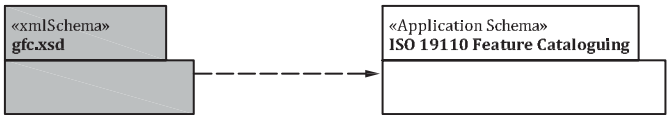

Figure C.2 — Organization of the gfc namespace

C.3.2 gfc.xsd

As described in Figure C.1, this XML schema document shall import the gco, gmd and gmx namespaces using their respective root XML schema document as defined in ISO/TS 19139. It is highly recommended to express the location of these root XML schema documents using an absolute URL.

This XML schema shall implement all the UML classes defined in the feature cataloguing package of this document: FC_FeatureCatalogue, FC_FeatureType, FC_InheritanceRelation, FC_PropertyType, FC_CarrierOfCharacteristics, FC_FeatureOperation, FC_Binding, FC_Constraint, FC_FeatureAttribute, FC_AssociationRole, FC_RoleType, FC_ListedValue, FC_FeatureAssociation, FC_DefinitionSource, FC_DefinitionReference, FC_LocalisedDefinitionReference, FC_BoundFeatureAttribute and FC_BoundAssociationRole.

Annex D
(normative)

Management of feature catalogue registers

D.1 Introduction

ISO 19135-1 specifies procedures for managing a register of items of geographic information, as well as a set of content elements common to all such registers. The content of an instance of RE_Register consists of a set of instances of RE_RegisterItem, which belong to item classes described by instances of RE_ItemClass.

This document does not prescribe the use of a specific register structure. Instead, it defines concepts that permit the management of feature catalogues either in a multi-part register or in a hierarchical register containing multi-part subregisters. Feature catalogues may indeed be managed as registers, i.e. as instances of RE_Register (ISO 19135-1), which contain feature types as registered items. They may also be managed as registered items, i.e. as instances of RE_RegisterItem, in a multi-part register.

Annex D includes a schema for feature catalogues as a realization of RE_Register (D.2) as well as a schema for feature catalogues, feature types, feature associations, feature attributes, operations and association roles, inheritance relations, listed values and definition sources as realizations of RE_RegisterItem (D.3) and specifies a set of instances of RE_ItemClass that describe feature catalogues and their content as item classes (D.4).

D.2 Feature catalogue register as a realization of RE_Register

D.2.1 Introduction

This document specifies a realization of RE_Register (Figure D.1) that represents the description of a feature catalogue. Because the content of this class is in its equivalent class in the feature catalogue schema, it serves only as a pointer from the register to the catalogue, and need not be implemented explicitly.

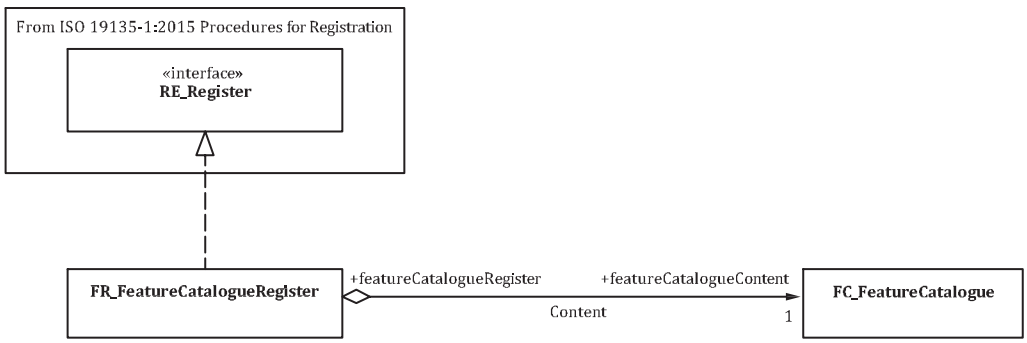

Figure D.1 — Feature catalogue register

D.2.2 FR_FeatureCatalogueRegister

D.2.2.1 Introduction

The class FR_FeatureCatalogueRegister shall represent an instance of FC_FeatureCatalogue managed as a register. It implements all the attributes and associations of RE_Register as specified in ISO 19135-1, and has one additional association.

D.2.2.2 Content

The aggregation association *Content* shall connect the instance of FR_FeatureCatalogueRegister to one and only one instance of FC_FeatureCatalogue. The association shall be navigable from featureCatalogueRegister to featureCatalogueContent, but need not be navigable in the opposite direction.

D.3 Realizations of RE_RegisterItem for registering the content of a feature catalogue

D.3.1 Introduction

This document specifies nine realizations of RE_RegisterItem (Figure D.2) that represent the content of a feature catalogue managed within an instance of RE_Register. These realizations belong to a single partition of the realization of RE_RegisterItem which is identified by the discriminator feature catalogue. The generalization relationship carries the constraint {incomplete} because it can be extended to include many different classes of items of geographic information. Because the only content of each of these classes is its equivalent class in the feature catalogue schema, they serve only as pointers from the register to the catalogue, and need not be implemented explicitly.

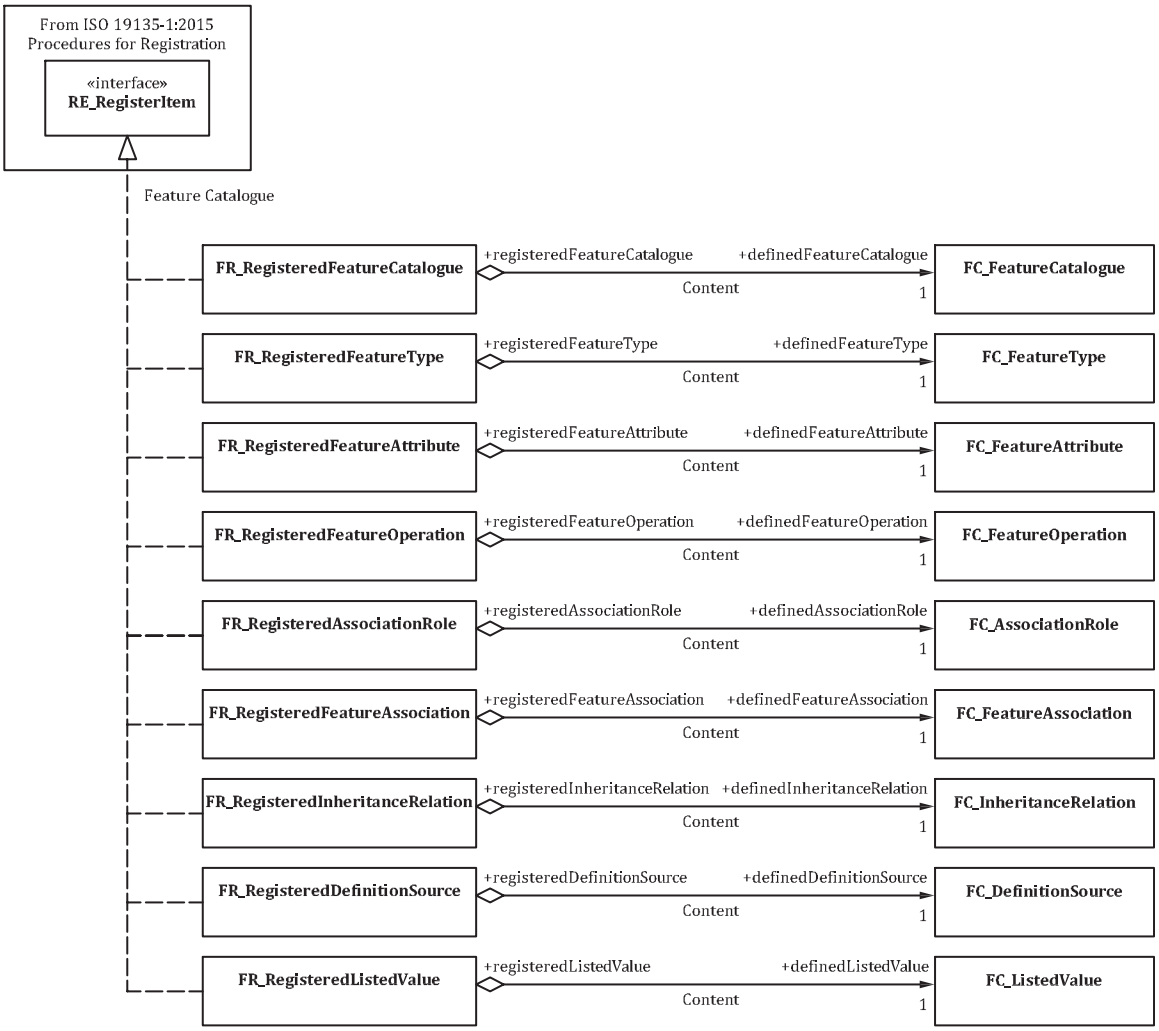

Figure D.2 — Feature catalogue register items

D.3.2 FR_RegisteredFeatureCatalogue

D.3.2.1 Introduction

The class FR_RegisteredFeatureCatalogue shall represent an instance of FC_FeatureCatalogue managed as a registered item. It implements all the attributes and associations of RE_RegisterItem as specified in ISO 19135-1, and has one additional association.

D.3.2.2 Content

The aggregation association *Content* shall connect the instance of FR_RegisteredFeatureCatalogue to one and only one instance of FC_FeatureCatalogue. The association shall be navigable from registeredFeatureCatalogue to definedFeatureCatalogue, but need not be navigable in the opposite direction.

D.3.3 FR_RegisteredFeatureType

D.3.3.1 Introduction

The class FR_RegisteredFeatureType shall represent an instance of FC_FeatureType managed as a register item. It implements all the attributes and associations of RE_RegisterItem as specified in ISO 19135-1, and has one additional association.

D.3.3.2 Content

The aggregation association *Content* shall connect the instance of FR_RegisteredFeatureType to one and only one instance of FC_FeatureType. The association shall be navigable from registeredFeatureType to definedFeatureType, but need not be navigable in the opposite direction.

D.3.4 FR_RegisteredFeatureAttribute

D.3.4.1 Introduction

The class FR_RegisteredFeatureAttribute shall represent an instance of FC_FeatureAttribute managed as a registered item. It implements all the attributes and associations of RE_RegisterItem as specified in ISO 19135-1, and has one additional association.

D.3.4.2 Content

The aggregation association *Content* shall connect the instance of FR_RegisteredFeatureAttribute to one and only one instance of FC_FeatureAttribute. The association shall be navigable from registeredFeatureAttribute to definedFeatureAttribute, but need not be navigable in the opposite direction.

D.3.5 FR_RegisteredFeatureOperation

D.3.5.1 Introduction

The class FR_RegisteredFeatureOperation shall represent an instance of FC_FeatureOperation managed as a registered item. It implements all the attributes and associations of RE_RegisterItem as specified in ISO 19135-1, and has one additional association.

D.3.5.2 Content

The aggregation association *Content* shall connect the instance of FR_RegisteredFeatureOperation to one and only one instance of FC_FeatureOperation. The association shall be navigable from registeredFeatureOperation to definedFeatureOperation, but need not be navigable in the opposite direction.

D.3.6 FR_RegisteredAssociationRole

D.3.6.1 Introduction

The class FR_RegisteredAssociationRole shall represent an instance of FC_AssociationRole managed as a registered item. It implements all the attributes and associations of RE_RegisterItem as specified in ISO 19135-1, and has one additional association.

D.3.6.2 Content

The aggregation association *Content* shall connect the instance of FR_RegisteredAssociationRole to one and only one instance of FC_AssociationRole. The association shall be navigable from registeredAssociationRole to definedAssociationRole, but need not be navigable in the opposite direction.

D.3.7 FR_RegisteredFeatureAssociation

D.3.7.1 Introduction

The class FR_RegisteredFeatureAssociation shall represent an instance of FC_FeatureAssociation managed as a registered item. It implements all the attributes and associations of RE_RegisterItem as specified in ISO 19135-1, and has one additional association.

D.3.7.2 Content

The aggregation association *Content* shall connect the instance of FR_RegisteredFeatureAssociation to one and only one instance of FC_FeatureAssociation. The association shall be navigable from registeredFeatureAssociation to definedFeatureAssociation, but need not be navigable in the opposite direction.

D.3.8 FR_RegisteredInheritanceRelation

D.3.8.1 Introduction

The class FR_RegisteredInheritanceRelation shall represent an instance of FC_InheritanceRelation managed as a registered item. It implements all the attributes and associations of RE_RegisterItem as specified in ISO 19135-1, and has one additional association.

D.3.8.2 Content

The aggregation association *Content* shall connect the instance of FR_RegisteredInheritanceRelation to one and only one instance of FC_InheritanceRelation. The association shall be navigable from registeredInheritanceRelation to definedInheritanceRelation, but need not be navigable in the opposite direction.

D.3.9 FR_RegisteredDefinitionSource

D.3.9.1 Introduction

The class FR_RegisteredDefinitionSource shall represent an instance of FC_DefinitionSource managed as a registered item. It implements all the attributes and associations of RE_RegisterItem as specified in ISO 19135-1, and has one additional association.

D.3.9.2 Content

The aggregation association *Content* shall connect the instance of FR_RegisteredDefinitionSource to one and only one instance of FC_DefinitionSource. The association shall be navigable from registeredDefinitionSource to definedDefinitionSource, but need not be navigable in the opposite direction.

D.3.10 FR_RegisteredListedValue

D.3.10.1 Introduction

The class FR_RegisteredListedValue shall represent an instance of FC_ListedValue managed as a registered item. It implements all the attributes and associations of RE_RegisterItem as specified in ISO 19135-1, and has one additional association.

D.3.10.2 Content

The aggregation association *Content* shall connect the instance of FR_RegisteredListedValue to one and only one instance of FC_ListedValue. The association shall be navigable from registeredListedValue to definedListedValue, but need not be navigable in the opposite direction.

D.4 Item classes for feature-related concepts

D.4.1 Introduction

RE_Register has a mandatory association to a set of instances of RE_ItemClass, each of which describes one class of items held in the register. RE_RegisterItem also has a mandatory association to the instance of RE_ItemClass that describes the item class to which it belongs. D.4 specifies the instances of RE_ ItemClass that describe the types of feature-related concepts to be held as registered items in a register used to manage a feature catalogue.

There are nine such item class instances, each of which corresponds to one of the types of feature-related concepts. They differ only in the value of the *name* attribute of RE_ItemClass.

D.4.2 Item class for feature catalogue registers

The item class for feature catalogues managed as registers or subregisters shall be an instance of RE_ ItemClass (ISO 19135-1) that shall be assigned the following attribute values.

— The value of the attribute name:CharacterString shall be “feature catalogue”.

— The value of the attribute technicalStandard:CI_Citation shall be

a) title:CharacterString = “ISO 19110:2016, Geographic information — Methodology for feature cataloguing”,

b) alternateTitle:CharacterString = “ISO 19110:2016”, and

c) date:CI_Date:

   1) date:Date = 2016;

   2) dateType:CI_DateTypeCode = “publication”.

D.4.3 Item class for feature types

The item class for feature types shall be an instance of RE_ItemClass (ISO 19135-1) that shall be assigned the following attribute values.

— The value of the attribute name:CharacterString shall be “feature type”.

— The value of the attribute technicalStandard:CI_Citation shall be

a) title:CharacterString = “ISO 19110:2016, Geographic information — Methodology for feature cataloguing”,

b) alternateTitle:CharacterString = “ISO 19110:2016”, and

c) date:CI_Date:

   1) date:Date = 2016;

   2) dateType:CI_DateTypeCode = “publication”.

D.4.4 Item class for feature attributes

The item class for feature attributes shall be an instance of RE_ItemClass (ISO 19135-1) that shall be assigned the following attribute values.

— The value of the attribute name:CharacterString shall be “feature attribute”.

— The value of the attribute technicalStandard:CI_Citation shall be

a) title:CharacterString = “ISO 19110:2016, Geographic information – Methodology for feature cataloguing”,

b) alternateTitle:CharacterString = “ISO 19110:2016”, and

c) date:CI_Date:

   1) date:Date = 2016;

   2) dateType:CI_DateTypeCode = “publication”.

D.4.5 Item class for feature operations

The item class for feature operations shall be an instance of RE_ItemClass (ISO 19135-1) that shall be assigned the following attribute values.

— The value of the attribute name:CharacterString shall be “feature operation”.

— The value of the attribute technicalStandard:CI_Citation shall be

a) title:CharacterString = “ISO 19110:2016, Geographic information – Methodology for feature cataloguing”,

b) alternateTitle:CharacterString = “ISO 19110:2016”, and

c) date:CI_Date:

   1) date:Date = 2016;

   2) dateType:CI_DateTypeCode = “publication”.

D.4.6 Item class for association roles

The item class for association roles shall be an instance of RE_ItemClass (ISO 19135-1) that shall be assigned the following attribute values.

— The value of the attribute name:CharacterString shall be “association role”.

— The value of the attribute technicalStandard:CI_Citation shall be

a) title:CharacterString = “ISO 19110:2016, Geographic information – Methodology for feature cataloguing”,

b) alternateTitle:CharacterString = “ISO 19110:2016”, and

c) date:CI_Date:

   1) date:Date = 2016;

   2) dateType:CI_DateTypeCode = “publication”.

D.4.7 Item class for feature associations

The item class for feature associations shall be an instance of RE_ItemClass (ISO 19135-1) that shall be assigned the following attribute values.

— The value of the attribute name:CharacterString shall be “feature association”.

— The value of the attribute technicalStandard:CI_Citation shall be

a) title:CharacterString = “ISO 19110:2016, Geographic information – Methodology for feature cataloguing”,

b) alternateTitle:CharacterString = “ISO 19110:2016”, and

c) date:CI_Date:

   1) date:Date = 2016;

   2) dateType:CI_DateTypeCode = “publication”.

D.4.8 Item class for inheritance relations

The item class for inheritance relations shall be an instance of RE_ItemClass (ISO 19135-1) that shall be assigned the following attribute values.

— The value of the attribute name:CharacterString shall be “inheritance relation”.

— The value of the attribute technicalStandard:CI_Citation shall be

a) title:CharacterString = “ISO 19110:2016, Geographic information – Methodology for feature cataloguing”,

b) alternateTitle:CharacterString = “ISO 19110:2016”, and

c) date:CI_Date:

   1) date:Date = 2016;

   2) dateType:CI_DateTypeCode = “publication”.

D.4.9 Item class for definition sources

The item class for definition sources shall be an instance of RE_ItemClass (ISO 19135-1) that shall be assigned the following attribute values.

— The value of the attribute name:CharacterString shall be “Definition Source”.

— The value of the attribute technicalStandard:CI_Citation shall be

a) title:CharacterString = “ISO 19110:2016, Geographic information – Methodology for feature cataloguing”,

b) alternateTitle:CharacterString = “ISO 19110:2016”, and

c) date:CI_Date:

   1) date:Date = 2016;

   2) dateType:CI_DateTypeCode = “publication”.

D.4.10 Item class for listed values

The item class for listed values shall be an instance of RE_ItemClass (ISO 19135-1) that shall be assigned the following attribute values.

— The value of the attribute name:CharacterString shall be “listed value”.

— The value of the attribute technicalStandard:CI_Citation shall be

a) title:CharacterString = “ISO 19110:2016, Geographic information – Methodology for feature cataloguing”,

b) alternateTitle:CharacterString = “ISO 19110:2016”, and

c) date:CI_Date:

   1) date:Date = 2016;

   2) dateType:CI_DateTypeCode = “publication”.

D.5 Feature catalogue register model conformance tests

D.5.1 Test case: Conformance of a feature catalogue register

a) Test purpose: To determine the conformance of a feature catalogue register.

b) Test method: Check that

   1) the feature catalogue register is a valid instance of FR_FeatureCatalogueRegister which connects to one and only one valid instance of FC_FeatureCatalogue;

   2) the feature catalogue register supports at least one of the nine item classes representing the possible content of a feature catalogue;

   3) the supported item classes are managed through the corresponding subclasses of RE_RegisterItem.

c) Reference: D.2, D.3 and D.4.

d) Test type: Basic.

D.5.2 Test case: Conformance of a registered feature catalogue

a) Test purpose: To determine the conformance of registered feature catalogues.

b) Test method: Check that

   1) the register contains at least one instance of FR_RegisteredFeatureCatalogue;

   2) each instance is valid and connected to a valid feature catalogue.

c) Reference: D.3.2.

d) Test type: Basic.

Annex E
(informative)

Feature cataloguing examples

E.1 Introduction

This annex presents examples of the functionality of the feature catalogue conceptual schema presented in Annex B. These examples are not intended to satisfy the needs of any particular application, or to be complete or comprehensive in any other sense. They are intended merely to illustrate aspects of the form and content of an ISO-conformant feature catalogue.

In order to illustrate associations between objects, each object has been assigned an identity whose value can be used as a pointer. This mimics the common mechanism in object-oriented programming languages, the foreign key mechanism in relational databases, the XPointer mechanism in XML, and the URI mechanism in HTTP[8]. Unused optional elements are omitted from the examples.

E.2 Feature catalogue

A feature catalogue contains its identification and contact information, and definition of some number of feature types with other information necessary for those definitions. Table E.1 illustrates a populated FC_FeatureCatalogue (Table B.1). Only one of the contained feature types is illustrated; in addition, there is one feature association (see E.4) in this example feature catalogue.

Table E.2 illustrates a populated FC_DefinitionSource (Table B.14) for the example feature catalogue.

Table E.1 — Example feature catalogue

Class FC_FeatureCatalogue (identity = 1)	
Attribute FC_FeatureCatalogue.name	“A sample feature catalogue”
Attribute FC_FeatureCatalogue.scope	“Hydrography”
	“Ports and Harbours”
	“Transportation Networks”
Attribute FC_FeatureCatalogue.fieldOfApplication	“Military Engineering”
	“Marine Navigation”
Attribute FC_FeatureCatalogue.versionNumber	“2.1”
Attribute FC_FeatureCatalogue.versionDate	2000-09-30

Table E.1 *(continued)*

Attribute FC_FeatureCatalogue.producer	**Class** ISO 19115-1 Metadata fundamentals:: CI_Responsibility			
	role	pointOfContact		
	party	Class ISO 19115-1 Metadata fundamentals::CI_Organization		
	name	"US National Geospatial-Intelligence Agency (NGA)"		
	contactInfo	**Class** ISO 19115-1 Metadata fundamentals:: **CI_Contact**		
		phone	**Class** ISO 19115-1 Metadata fundamentals:: **CI_Telephone**	
			voice	"1 703 xxx xxxx"
			facsimile	"1 703 xxx xxxx"
		address	**Class** ISO 19115-1 Metadata fundamentals:: **CI_Address**	
			deliveryPoint	"12310 Sunrise Valley Drive"
			city	"Reston"
			administrativeArea	"Virginia"
			postalCode	"20191-3449"
			country	"USA"
			electronicMailAddress	"PublicJQ@nga.mil"
Role FC_FeatureCatalogue.featureType	**FC_FeatureType** (identity = 3) *(additional feature types are included in this example but not listed here for brevity)*			
Role FC_FeatureCatalogue.featureType	**FC_FeatureType** (identity = 22)			
Role FC_FeatureCatalogue.definitionSource	FC_DefinitionSource (identity = 2)			

Table E.2 — Example definition source

Class FC_DefinitionSource (identity = 2)			
Attribute FC_DefinitionSource.source	**Class** ISO 19115-1 Metadata fundamentals::**CI_Citation**		
	title	"International Hydrographic Organization (IHO) Hydrographic Dictionary, Part I, Volume I English"	
	date	**Class** ISO 19115-1 Metadata fundamentals:: **CI_Date**	
		date	1994
		dateType	02 (publication)
	edition	"Fifth"	
	citedResponsibleParty	**Class** ISO 19115-1 Metadata fundamentals:: CI_Responsibility	
		organisationName	"International Hydrographic Bureau"
		role	11 (publisher)
	otherCitationDetails	"Special publication No. 32"	

E.3 Feature types and feature attributes

E.3.1 The 'depth' of a 'mine'

Feature types are classes of real-world phenomena with common properties. The example feature catalogue contains many feature types, represented using FC_FeatureType (Table B.2). Table E.3

illustrates a 'mine' feature type; it includes a definition and a code, and is not an abstract feature type. It also has an alias.

Table E.3 — Example feature type 'mine'

Class FC_FeatureType (identity = 3)	
Attribute FC_FeatureType.typeName	"Mine"
Attribute FC_FeatureType.definition	"An excavation made in the earth for the purpose of extracting natural deposits. (See also AQ090.)"
Attribute FC_FeatureType.code	"AA010"
Attribute FC_FeatureType.isAbstract	FALSE
Attribute FC_FeatureType.aliases	"Extraction mine"
Role FC_FeatureType.featureCatalogue	**FC_FeatureCatalogue** (identity = 1)
Role FC_FeatureType.carrierOfCharacteristics	**FC_FeatureAttribute** (identity = 4)

The example feature catalogue includes the feature attribute 'depth'. Table E.4 illustrates its representation using FC_FeatureAttribute (Table B.9); it is real-valued and measured in the unit 'metre'.

Table E.4 — Example quantitative feature attribute 'depth'

Class FC_FeatureAttribute (identity = 4)	
Attribute FC_PropertyType.memberName	"Depth"
Attribute FC_PropertyType.definition	"Distance measured from the highest point at surface level to the lowest point of the feature below the surface."
Attribute FC_PropertyType.cardinality	1
Role FC_PropertyType.featureType	**FC_FeatureType** (identity = 3)
Role FC_PropertyType.constrainedBy	**FC_Constraint** (identity = 5)
Attribute FC_FeatureAttribute.code	"DEP"
Attribute FC_FeatureAttribute.valueMeasurementUnit	"Metre"
Attribute FC_FeatureAttribute.valueType	Real

The measurement of the value of the 'depth' feature attribute is constrained as to direction of measurement. Table E.5 illustrates how this information is represented using an FC_Constraint (Table B.8).

Table E.5 — Example feature attribute constraint

Class FC_Constraint (identity = 5)	
Attribute FC_Constraint.description	"Positive values represent distance below the reference point from which the measurement is made."

E.3.2 The abstract representation of a 'berthing structure'

The example feature catalogue also includes a 'berthing structure' feature type (illustrated in Table E.6) and a 'pier/wharf/quay classification' feature attribute (illustrated in Table E.7). Unlike the real-valued 'depth' feature attribute, the 'pier/wharf/quay classification' feature attribute uses listed values. These are represented using FC_ListedValue (Table B.12) and are illustrated in Tables E.8, E.9, E.11, E.13, E.15, E.16 and E.17.

Table E.6 — Example feature type 'berthing structure'

Class FC_FeatureType (identity = 7)	
Attribute FC_FeatureType.typeName	"Berthing Structure"
Attribute FC_FeatureType.definition	"A fixed (not afloat) artificial structure attached to a shore and normally used for berthing and protection of vessels."
Attribute FC_FeatureType.code	"BB999"
Attribute FC_FeatureType.isAbstract	FALSE
Role FC_FeatureType.featureCatalogue	**FC_FeatureCatalogue** (identity = 1)
Role FC_FeatureType.carrierOfCharacteristics	**FC_FeatureAttribute** (identity = 8)

Table E.7 — Example feature attribute with listed values

Class FC_FeatureAttribute (identity = 8)	
Attribute FC_PropertyType.memberName	"Pier/Wharf/Quay Classification"
Attribute FC_PropertyType.definition	"Classification of decked berthing structure, based on configuration and structure."
Attribute FC_PropertyType.cardinality	1
Role FC_PropertyType.featureType	**FC_FeatureType** (identity = 7)
Attribute FC_FeatureAttribute.code	"PWC"
Attribute FC_FeatureAttribute.listedValue	**FC_ListedValue** (identity = 9)
	FC_ListedValue (identity = 10)
	FC_ListedValue (identity = 12)
	FC_ListedValue (identity = 14)
	FC_ListedValue (identity = 16)
	FC_ListedValue (identity = 17)
	FC_ListedValue (identity = 18)

Not all of the listed values have definitions; in three cases (Tables E.9, E.11 and E.13), the definitions are located outside of the feature catalogue and are found in the source document previously specified in Table E.2. Tables E.10, E.12 and E.14 illustrate the representation, using FC_DefinitionReference (Table B.15) of the additional citation information necessary to locate each listed value definition in the source document.

Table E.8 — Example feature attribute listed value 'unknown'

Class FC_ListedValue (identity = 9)	
Attribute FC_ListedValue.label	"Unknown"
Attribute FC_ListedValue.code	"0"
Attribute FC_ListedValue.definition	"The attribute value is missing."

Table E.9 — Example feature attribute listed value 'pier'

Class FC_ListedValue (identity = 10)	
Attribute FC_ListedValue.label	"Pier"
Attribute FC_ListedValue.code	"1"
Role FC_ListedValue.definitionReference	**FC_DefinitionReference** (identity = 11)

Table E.10 — Example feature attribute listed value definition reference 'pier'

Class FC_DefinitionReference (identity = 11)	
Attribute FC_DefinitionReference.sourceIdentifier	"3833, pier"[a]
Role FC_DefinitionReference.definitionSource	**FC_DefinitionSource** (identity = 2)
[a] "A long, narrow structure extending into a water body to afford a berthing place for vessels or to serve as a promenade."	

Table E.11 — Example feature attribute listed value 'wharf'

Class FC_ListedValue (identity = 12)	
Attribute FC_ListedValue.label	"Wharf"
Attribute FC_ListedValue.code	"2"
Role FC_ListedValue.definitionReference	**FC_DefinitionReference** (identity = 13)

Table E.12 — Example feature attribute listed value definition reference 'wharf'

Class FC_DefinitionReference (identity = 13)	
Attribute FC_DefinitionReference.sourceIdentifier	"5985, wharf"[a]
Role FC_DefinitionReference.definitionSource	**FC_DefinitionSource** (identity = 2)
[a] "A structure serving as a berthing place for vessels."	

Table E.13 — Example feature attribute listed value 'quay'

Class FC_ListedValue (identity = 14)	
Attribute FC_ListedValue.label	"Quay"
Attribute FC_ListedValue.code	"3"
Role FC_ListedValue.definitionReference	**FC_DefinitionReference** (identity = 15)

Table E.14 — Example feature attribute listed value definition reference 'quay'

Class FC_DefinitionReference (identity = 15)	
Attribute FC_DefinitionReference.sourceIdentifier	"4125, quay"[a]
Role FC_DefinitionReference.definitionSource	**FC_DefinitionSource** (identity = 2)
[a] "A wharf approximately parallel to the shoreline and accommodating vessels on one side only, the other side being attached to the shore. It is usually of solid construction, as contrasted with the open pile construction usually used for piers."	

Table E.15 — Example feature attribute listed value 'unpopulated'

Class FC_ListedValue (identity = 16)	
Attribute FC_ListedValue.label	"Unpopulated"
Attribute FC_ListedValue.code	"997"
Attribute FC_ListedValue.definition	"The attribute value exists, but due to policy considerations it cannot be given."

Table E.16 — Example feature attribute listed value 'not applicable'

Class FC_ListedValue (identity = 17)	
Attribute FC_ListedValue.label	"Not applicable"
Attribute FC_ListedValue.code	"998"
Attribute FC_ListedValue.definition	"No attribute value in the range of possible attribute values is applicable."

Table E.17 — Example feature attribute listed value 'other'

Class FC_ListedValue (identity = 18)	
Attribute FC_ListedValue.label	"Other"
Attribute FC_ListedValue.code	"999"
Attribute FC_ListedValue.definition	"The attribute value cannot be given for some reason other than it is 'multiple', 'not applicable', 'unknown', or 'unpopulated'."

E.4 Feature associations and association roles

Feature associations are relationships that link instances of a feature type with instances of the same or of a different feature type. The feature types have specific roles in the association. The example feature catalogue contains the 'stacked on' association, relating the two feature types 'road' and 'bridge'. These feature types are illustrated in Tables E.18 and E.19, respectively.

Table E.18 — Example feature type 'road'

Class FC_FeatureType (identity = 20)	
Attribute FC_FeatureType.typeName	"Road"
Attribute FC_FeatureType.definition	"An open way maintained for vehicular use."
Attribute FC_FeatureType.code	"AP030"
Attribute FC_FeatureType.isAbstract	FALSE
Role FC_FeatureType.featureCatalogue	**FC_FeatureCatalogue** (identity = 1)
Role FC_FeatureType.carrierOfCharacteristics	**FC_AssociationRole** (identity = 23)

Table E.19 — Example feature type 'bridge'

Class FC_FeatureType (identity = 21)	
Attribute FC_FeatureType.typeName	"Bridge"
Attribute FC_FeatureType.definition	"A man-made structure spanning and providing passage over a body of water, depression, or other obstacles."
Attribute FC_FeatureType.code	"AQ040"
Attribute FC_FeatureType.isAbstract	FALSE
Role FC_FeatureType.featureCatalogue	**FC_FeatureCatalogue** (identity = 1)
Role FC_FeatureType.carrierOfCharacteristics	**FC_FeatureAssociationRole** (identity = 24)

Table E.20 illustrates the representation of the 'stacked on' feature association using FC_FeatureAssociation (Table B.13).

Table E.20 — Example feature association 'stacked on'

Class FC_FeatureAssociation (identity = 22)	
Attribute FC_FeatureType.typeName	"Stacked On"
Attribute FC_FeatureType.definition	"An object is over another object."

Table E.20 *(continued)*

Attribute FC_FeatureType.code	"101"
Attribute FC_FeatureType.isAbstract	FALSE
Role FC_FeatureType.featureCatalogue	**FC_FeatureCatalogue** (identity = 1)
Role FC_FeatureAssociation.role	**FC_AssociationRole** (identity = 23)
Role FC_FeatureAssociation.role	**FC_AssociationRole** (identity = 24)

The 'stacked on' feature association has two association roles, represented using FC_AssociationRole (Table B.10), illustrated in Tables E.21 and E.22. The 'over' role pertains to the 'road' feature type. The role cardinality is zero or more, and any bridges traversed by the road are ordered.

Table E.21 — Example association role 'over'

Class FC_AssociationRole (identity = 23)	
Attribute FC_PropertyType.memberName	"Over"
Attribute FC_PropertyType.definition	"Bridges which this road crosses over. Within the road, the ordering of crossings reflects the order in which the crossings would be made if one traversed the road."
Attribute FC_ PropertyType.cardinality	"0..*"
Role FC_PropertyType.featureType	**FC_FeatureType** (identity = 20)
Attribute FC_AssociationRole.type	FC_RoleType.ordinary
Attribute FC_AssociationRole.isOrdered	TRUE
Attribute FC_AssociationRole.isNavigable	TRUE
Role FC_AssociationRole.relation	**FC_FeatureAssociation** (identity = 22)
Role FC_AssociationRole.valueType	**FC_FeatureType** (identity = 21)

The 'under' role pertains to the 'bridge' feature type; since at most only a single road crosses each bridge, the road crossed by the bridge is not ordered and the role cardinality is zero or one. For both roles of the 'stacked on' feature association, its type is that of an ordinary association (FC_RoleType: Table B.11).

Table E.22 — Example association role 'under'

Class FC_AssociationRole (identity = 24)	
Attribute FC_PropertyType.memberName	"Under"
Attribute FC_PropertyType.definition	"Roads which cross this bridge."
Attribute FC_ PropertyType.cardinality	"0..1"
Role FC_PropertyType.featureType	**FC_FeatureType** (identity = 21)
Attribute FC_AssociationRole.type	FC_RoleType.ordinary
Attribute FC_AssociationRole.isOrdered	FALSE
Attribute FC_AssociationRole.isNavigable	TRUE
Role FC_AssociationRole.relation	**FC_FeatureAssociation** (identity = 22)
Role FC_AssociationRole.valueType	**FC_FeatureType** (identity = 20)

E.5 Inheritance relations

Inheritance relations are relationships that link a more generalized feature type (supertype) with a more specialized feature type (subtype). The example feature catalogue contains the 'is a' inheritance relation, relating the two feature types 'building' and 'lighthouse'. These feature types are illustrated in Tables E.23 and E.24, respectively.

Table E.23 — Example feature type 'building'

Class FC_FeatureType (identity = 25)	
Attribute FC_FeatureType.typeName	"Building"
Attribute FC_FeatureType.definition	"A relatively permanent structure, roofed and usually walled and designed for some particular use. (See also AL100)"
Attribute FC_FeatureType.code	"AL015"
Attribute FC_FeatureType.isAbstract	FALSE
Role FC_FeatureType.inheritsTo	**FC_FeatureInheritanceRelation** (identity = 27)
Role FC_FeatureType.featureCatalogue	**FC_FeatureCatalogue** (identity = 1)

Table E.24 — Example feature type 'lighthouse'

Class FC_FeatureType (identity = 26)	
Attribute FC_FeatureType.typeName	"Lighthouse"
Attribute FC_FeatureType.definition	"A distinctive structure exhibiting light(s) designed to serve as an aid to navigation. (See also BC040)"
Attribute FC_FeatureType.code	"BC050"
Attribute FC_FeatureType.isAbstract	FALSE
Role FC_FeatureType.inheritsFrom	**FC_FeatureInheritanceRelation** (identity = 27)
Role FC_FeatureType.featureCatalogue	**FC_FeatureCatalogue** (identity = 1)

An instance of a 'lighthouse' feature type is also an instance of a 'building' feature type; feature properties that apply to the 'building' feature type in the example feature catalogue also apply to the 'lighthouse' feature type. Table E.25 illustrates the representation of the 'is a' inheritance relation using FC_InheritanceRelation (Table B.3).

Table E.25 — Example inheritance relation 'is a'

Class FC_InheritanceRelation (identity = 27)	
Attribute FC_InheritanceRelation.name	"is a"
Attribute FC_InheritanceRelation.description	"An object is classified as a specialization of another object."
Attribute FC_InheritanceRelation.uniqueInstance	TRUE
Role FC_InheritanceRelation.subtype	**FC_FeatureType** (identity = 26)
Role FC_InheritanceRelation.supertype	**FC_FeatureType** (identity = 25)

E.6 Feature operations

Feature operations specify the behaviour of feature types. The example feature catalogue contains the 'raise dam' feature operation as a property of the 'dam' feature type. The semantics of the 'raise dam' feature operation are dependent on feature attributes of the 'watercourse' and 'reservoir' feature types. The representation of these feature types and their feature attributes are illustrated in the Tables E.26 to E.34.

Tables E.26, E.27, and E.28 illustrate the specification of the 'dam' feature type and its 'dam height' and 'maximum height' feature attributes, respectively. The 'dam' has two aliases, and both of its feature attributes are positive real-valued and measured in the unit 'metre'.

Table E.26 — Example feature type 'dam'

Class FC_FeatureType (identity = 28)	
Attribute FC_FeatureType.typeName	"Dam"
Attribute FC_FeatureType.definition	"Barrier constructed across a watercourse to control the level or flow of water in the watercourse or the level of water in a reservoir."
Attribute FC_FeatureType.code	"359"
Attribute FC_FeatureType.isAbstract	FALSE
Attribute FC_FeatureType.aliases	"Barrage"
	"Weir"
Role FC_FeatureType.featureCatalogue	**FC_FeatureCatalogue** (identity = 1)
Role FC_FeatureType.carrierOfCharacteristics	**FC_FeatureAttribute** (identity = 29)
Role FC_FeatureType.carrierOfCharacteristics	**FC_FeatureAttribute** (identity = 30)
Role FC_FeatureType.carrierOfCharacteristics	**FC_FeatureOperation** (identity = 36)

Table E.27 — Example feature attribute 'dam height'

Class FC_FeatureAttribute (identity = 29)	
Attribute FC_PropertyType.memberName	"Dam height"
Attribute FC_PropertyType.definition	"Vertical distance from the base of a dam to the level where water spills over its top."
Attribute FC_PropertyType.cardinality	1
Role FC_PropertyType.featureType	FC_FeatureType (identity = 28)
Attribute FC_FeatureAttribute.code	"damHeight"
Attribute FC_FeatureAttribute.valueMeasurementUnit	"Metre"
Attribute FC_FeatureAttribute.valueType	Positive real

Table E.28 — Example feature attribute 'maximum height'

Class FC_FeatureAttribute (identity = 30)	
Attribute FC_PropertyType.memberName	"Maximum height"
Attribute FC_PropertyType.definition	"Maximum possible dam height."
Attribute FC_PropertyType.cardinality	1
Role FC_PropertyType.featureType	**FC_FeatureType** (identity = 28)
Attribute FC_FeatureAttribute.code	"maxHeight"
Attribute FC_FeatureAttribute.valueMeasurementUnit	"Metre"
Attribute FC_FeatureAttribute.valueType	Positive real

Tables E.29 and E.30 illustrate the specification of the 'reservoir' feature type and its 'reservoir depth' feature attribute, respectively. The 'reservoir' has a single alias, and its feature attribute is positive real-valued and measured in the unit 'metre'.

Table E.29 — Example feature type 'reservoir'

Class FC_FeatureType (identity = 31)	
Attribute FC_FeatureType.typeName	"Reservoir"
Attribute FC_FeatureType.definition	"Natural or artificial pond or lake used for the storage and regulation of water."
Attribute FC_FeatureType.code	"765"

Table E.29 *(continued)*

Attribute FC_FeatureType.isAbstract	FALSE
Attribute FC_FeatureType.aliases	“Storage pond”
Role FC_FeatureType.featureCatalogue	**FC_FeatureCatalogue** (identity = 1)
Role FC_FeatureType.carrierOfCharacteristics	**FC_FeatureAttribute** (identity = 32)

Table E.30 — Example feature attribute ‘reservoir depth’

Class FC_FeatureAttribute (identity = 32)	
Attribute FC_PropertyType.memberName	“Reservoir depth”
Attribute FC_PropertyType.definition	“Maximum vertical distance from the water surface to the bottom of a reservoir.”
Attribute FC_PropertyType.cardinality	1
Role FC_PropertyType.featureType	**FC_FeatureType** (identity = 31)
Attribute FC_FeatureAttribute.code	“reservoirDepth”
Attribute FC_FeatureAttribute.valueMeasurementUnit	“Metre”
Attribute FC_FeatureAttribute.valueType	Positive real

Tables E.31, E.32 and E.33 illustrate the specification of the ‘watercourse’ feature type and its ‘stream depth’ and ‘stream flow’ feature attributes, respectively. The ‘reservoir’ has five aliases. While the ‘stream depth’ feature attribute is positive real-valued and measured in the unit ‘metre’, the ‘stream flow’ feature attribute is positive integer-valued and measured in the unit ‘cubic metres per second’.

Table E.31 — Example feature type ‘watercourse’

Class FC_FeatureType (identity = 33)	
Attribute FC_FeatureType.typeName	“Watercourse”
Attribute FC_FeatureType.definition	“Way or course through which water may or does flow.”
Attribute FC_FeatureType.code	“1470”
Attribute FC_FeatureType.isAbstract	FALSE
Attribute FC_FeatureType.aliases	“Brook”
	“Kill”
	“River”
	“Seaway”
	“Stream”
Role FC_FeatureType.featureCatalogue	**FC_FeatureCatalogue** (identity = 1)
Role FC_FeatureType.carrierOfCharacteristics	**FC_FeatureAttribute** (identity = 34)
Role FC_FeatureType.carrierOfCharacteristics	**FC_FeatureAttribute** (identity = 35)

Table E.32 — Example feature attribute ‘stream depth’

Class FC_FeatureAttribute (identity = 34)	
Attribute FC_PropertyType.memberName	“Stream depth”
Attribute FC_PropertyType.definition	“Maximum vertical distance from the water surface to the bottom.”
Attribute FC_PropertyType.cardinality	1
Role FC_PropertyType.featureType	**FC_FeatureType** (identity = 33)

Table E.32 *(continued)*

Attribute FC_FeatureAttribute.code	"streamDepth"
Attribute FC_FeatureAttribute.valueMeasure-mentUnit	"Metre"
Attribute FC_FeatureAttribute.valueType	Positive real

Table E.33 — Example feature attribute 'stream flow'

Class FC_FeatureAttribute (identity = 35)	
Attribute FC_PropertyType.memberName	"Stream flow"
Attribute FC_PropertyType.definition	"Quantity of water flowing per unit of time."
Attribute FC_PropertyType.cardinality	1
Role FC_PropertyType.featureType	**FC_FeatureType** (identity = 33)
Attribute FC_FeatureAttribute.code	"streamFlow"
Attribute FC_FeatureAttribute.valueMeasure-mentUnit	"Cubic metres per second"
Attribute FC_FeatureAttribute.valueType	Positive integer

Table E.26, illustrating the specification of the 'dam' feature type, identifies the presence of three property types. Two of these were feature attributes (damHeight and maxHeight); the third was the 'raise dam' feature operation and its specification, represented using FC_FeatureOperation (Table B.6), is illustrated in Table E.34.

The feature operation definition describes the semantics of 'raise dam' while the feature operation signature specifies that given a real-valued newHeight and a Dam feature, that feature is accordingly revised.

The 'raise dam' operation observes the feature attribute maxHeight (of the Dam), since the operation result is contingent on its value. Additionally, the value of the feature attribute of damHeight (of the Dam) is affected, as well as the values of the feature attributes of streamDepth and streamFlow (of the downstream Watercourse) and reservoirDepth (of the upstream Reservoir).

NOTE Values of feature attributes may be observed or affected for another feature instance only if there is a feature association between the feature types involved. The necessary feature associations between the 'dam' and the upstream 'reservoir' and between the 'dam' and the downstream 'watercourse' feature types are not illustrated in this example.

Table E.34 — Example feature operation 'raise dam'

Class FC_FeatureOperation (identity = 36)	
Attribute FC_PropertyType.memberName	"Raise dam"
Attribute FC_PropertyType.definition	"The action of raising the dam causes changes in the discharge from the dam. The rate of discharge, in turn, affects the depth and flow of water in the downstream segment of the watercourse and the depth of water in the reservoir behind the dam."
Attribute FC_PropertyType.cardinality	1
Role FC_PropertyType.featureType	**FC_FeatureType** (identity = 28)
	FC_Binding (unspecified in this example)
Attribute FC_FeatureOperation.signature	"damRaise((Dam) dam, (Real) newHeight): Dam"
Role FC_FeatureOperation.observesValuesOf	**FC_FeatureAttribute** (identity = 30)
Role FC_FeatureOperation.affectsValuesOf	**FC_FeatureAttribute** (identity = 29)

Annex F
(informative)

Feature cataloguing concepts

F.1 Introduction

A feature catalogue forms a repository for a set of definitions to classify real-world phenomena of significance to a particular universe of discourse. The catalogue provides a means for organizing the abstract representation of the data that represent these phenomena, so that the resulting information is as unambiguous, comprehensible, and useful as possible.

In the past, it has been a common practice to isolate and distinguish three separate aspects of geographic features: the definitions used to group them into feature types, the attributes associated with each feature type, and the associations among the feature types. Within this general framework, the operations of the feature types have generally been included as part of the feature definitional criteria, and have been expressed only in terms of their natural language definitions. As the examples in Annex F will show, the attributes of features and the associations among them have a much richer meaning when viewed in the context of how the features operate. Within this context, attributes provide measures of the state of a feature as it exhibits certain kinds of behaviour over time, not just static measures of the differences among features at a given instant in time. Associations can also be seen in this active sense, that one phenomenon's behaviour or condition is affected by the operation of another one.

Although, for the purpose of this document, feature operations are presented as a fourth major aspect of feature abstraction, they represent a difference in point of view as much as they do a difference in kind. In a functional specification, an operation is triggered by, returns, or affects a value (i.e. a feature attribute value) for a given type of geographic feature. If values are observed or affected for more than one feature, the operation also specifies a functional relationship between them. By including feature operations as an additional dimension of feature abstract representation, this document seeks to support the anticipated transition from current practice to a future, more rigorously functional, approach[9].

F.2 Feature operations

Feature operations are frequently included in the natural-language definitions of the feature types. They are important for several reasons. First and foremost, they are the distinguishing characteristics that are embedded in the perceptions of the human beings who distinguish one type of geographic feature from another: they have psychological and behavioural significance to the people who use geographic information. Another reason is that computer systems are increasingly able to represent geographic phenomena, not just as a static set of maps, but as a dynamic representation of events occurring in geographic space in real time. Still another reason is that interoperability is an increasingly important goal in the design of geographic information systems. Functional equivalence of features is the key to interoperability of geographic information systems in the emerging open systems environment.

Feature operations are of two kinds: observer functions and constructor functions. Observer functions return the current values of attributes. Constructor functions include actions that change those values. For example, an observer function may be used to find the height of a dam. Raising the dam is a constructor function that changes the height of the dam and also affects the attributes of the watercourse and the reservoir associated with the dam.

F.3 Feature attributes

Feature attributes are derived directly from feature operations. For example, the volume of traffic over the bridge is a measure of its behaviour. All bridges exhibit the operation of carrying traffic, making this property a part of the definition of the feature.

Other feature attributes may be indirectly derived for a feature. For example, the 'clearance' is an important attribute of a bridge because it limits the height of vessels that can pass under it. This attribute results from a different operation, the navigation of vessels in the water under the bridge. Therefore, in specifying the attributes for a feature, it is important to consider the operations that are performed **on** it as well as those that are performed **by** it.

Finally, in a feature catalogue, there may be feature attributes included for a given feature type that are unrelated to any feature operation specified in the catalogue. For example, the catalogue may define a feature 'mountain' that has no specified operations, but includes the attribute 'altitude'. There is a general operation 'air navigation' that relies on observing the altitude of a mountain, even though air navigation is not a kind of behaviour either engaged in by mountains or specified elsewhere in the feature catalogue. The feature catalogue producers have included the feature attribute in response to perceived (but unspecified) external demands for information about mountains.

F.4 Feature relationships

F.4.1 Kinds of relationships

Feature relationships may be one of two kinds: generalization or association. Associations may be specialized as aggregation or other logical relationships. Feature operations, feature attributes, and association roles are properties that are inherited through generalization relationships.

F.4.2 Generalization

In generalization, the members of one feature type are automatically members of another feature type by definition. For example, a bridge is a transportation feature if a 'bridge' is defined by the operation 'carries traffic' and a more general feature 'transportation feature' is also defined by the operation 'carries traffic'.

Generalization implies inheritance of properties, e.g. feature operations, feature attributes, and association roles, from the more general to the more specific. Many feature types have multiple operations and attributes; generalization may result from a pattern of multiple inheritance of properties. For example, the feature type 'bridge' may belong to both the general class of 'transportation feature' for road features and to the general class of 'hazards' for navigation features.

Generalization is thus an inheritance relation between feature type; it is supported in Table B.2 by the optional role 'inheritsFrom'.

F.4.3 Aggregation

Instances of feature types are grouped into different types that have different properties. For example, a 'canal lock' is composed of walls, gates, and a portion of a canal. The operation of moving vessels around a dam or rapid is not performed by the walls or gates by themselves, but only when they are aggregated to form a lock. Similarly, a 'road network' has some properties that are not inherited by the individual roads composing the network.

The aggregation association does not imply a hierarchical organization of feature types unless all the members of each constituent feature also belong to the aggregate feature. For example, not all walls are part of canal locks. It is a potential relationship to which individual instances of a feature type may or may not belong.

F.4.4 Other logical relationships

In the bridge example, there is a relationship between the watercourse and the bridge because of the operation of navigation on the watercourse, which is affected by the clearance of the bridge (e.g. 'stacked on' with role 'under'; see also E.4). The association between the feature type 'watercourse' and the feature type 'bridge' is neither a generalization nor an aggregation. A logical relationship of 'transportation related' might be specified to include bridges, watercourses, roads, and the feature type 'signs'. The operation 'carries traffic' does not apply to signs so the association 'transportation related' is not a generalization. Again, the organization of other logical relationships is not necessarily hierarchical: for example, not all signs are transportation related.

F.5 Synonyms and included terms

In existing collections of feature type specifications, there may be 'included terms' listed for 'standard terms'. The 'included terms' may be subtypes of a more general feature type.

EXAMPLE 1 Standard term: Watercourse, with included terms: Brook, Kill, River, Seaway, Stream.

The 'included terms' may be synonyms or near synonyms that have overlapping definitions with a term selected as the 'standard term'.

EXAMPLE 2 Standard term: Coastline, with included term: Coastal Shoreline.

The 'included terms' may be equivalent terms in other languages.

EXAMPLE 3 Standard term: Mine, with included terms: Grube/Zeche (German), Miniera (Italian), Mijn (Dutch).

Where the feature types are different (with regard to feature operations, feature attributes, or association roles), they should be included in feature catalogues as distinct items with their own specifications. When an included term is a functionally equivalent synonym (e.g. in another language), it may be listed as an 'alias' for the feature type.

Producers of feature catalogues should take care to ensure that the meaning of 'alias' terms is precisely equivalent (with regard to feature operations, feature attributes, and association roles) for a given purpose. The functional specification of the feature types provides an unambiguous method for evaluating equivalence.

Annex G
(informative)

Transformation of legacy feature catalogues

G.1 Transformation of a legacy catalogue

This annex explains how to move forward legacy feature catalogue implementations to conform to this document. The start point of the transformation is an instance of FC_FeatureCatalogue.

— The attributes of FC_FeatureCatalogue have not changed.

— The new inheritanceRelation role of FC_FeatureCatalogue enables the aggregation of the FC_InheritanceRelation used in the feature catalogue directly to the FC_FeatureCatalogue instance. This can be achieved when processing the inheritsFrom and inheritsTo roles of FC_FeatureType, taking care not to duplicate the different instances of FC_InheritanceRelation.

— The new globalProperty role enables the aggregation of the global properties used in the feature catalogue directly to the FC_FeatureCatalogue instance. This is also recommended to ease the management of the feature catalogue. This can be achieved when processing the feature type properties (see G.2).

— The feature type and definitionSource roles have not changed, but the feature type properties need to be processed specifically (see G.2), as well as the feature associations (see G.3).

G.2 Transformation of legacy feature type properties

The relationship between the feature types and the feature properties has changed. In the previous model, an FC_PropertyType could be aggregated to many FC_FeatureType and the FC_Binding association class was used to implement the effective binding of those properties to each individual feature type.

Properties bound to a single feature type can be directly transformed as subclasses of FC_PropertyType considering that a specific processing is necessary for association roles (see G.5) and feature attributes (see G.4).

Properties bound to multiple feature types need to be managed as global properties in the new model, which implies

— on the one hand, to aggregate them as instances of FC_PropertyType to the globalProperty of FC_FeatureCatalogue;

— on the other hand, to bind them to their respective feature type through the FC_Binding (operation), FC_BoundAssociationRole (role) or FC_BoundFeatureAttribute (attribute) class. The globalProperty role of this bound property has to relate to the corresponding globalProperty of the feature catalogue.

G.3 Transformation of legacy feature associations

The following issues have to be addressed.

— The role property has been renamed roleName.

— The cardinality of the role property is 2..* instead of 1..* which may imply the creation of a fake role if one is missing in the legacy implementation.

G.4 Transformation of legacy feature attributes

The attributed listedValue of type FC_ListedValue is a role in the new model. Since FC_ListedValue has not changed, this change should not have a real impact.

G.5 Transformation of legacy association roles

The valueType role of FC_AssociationRole has been renamed rolePlayer. rolePlayer is optional to support the requirements for global roles (where the type of the role may require to be set when it is bound), but this should not have any impact for this transformation.

Bibliography

[1] ISO 19101-1:2014, *Geographic information — Reference model — Part 1: Fundamentals*

[2] ISO 19107, *Geographic information — Spatial schema*

[3] ISO 19108, *Geographic information — Temporal schema*

[4] ISO 19117, *Geographic information — Portrayal*

[5] ISO 19126, *Geographic information — Feature concept dictionaries and registers*

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

[7] Defence Geospatial Information Working Group (DGIWG). *Digital Geographic Information Exchange Standard, Part 4: Feature and Attribute Coding Catalogue (FACC) Data Dictionary* [online]. Ed. 2.1. Washington: DGIWG, 2000. [cited July 5, 2016]. Available at: <https://www.dgiwg.org/digest/html/DIGEST_2-1_Part4.pdf >

[8] Internet Engineering Task Force (IETF), The Internet Society. *Uniform Resource Identifiers (URI): Generic Syntax.* RFC 2396. Reston (Virginia): IETF, 1998. [cited July 5, 2016]. Available at: <http://www.faqs.org/rfcs/rfc2396.html>.

[9] Rugg, R. D., Egenhofer, M. J. and Kuhn, W., *Formalizing Behavior of Geographic Feature Types, Geographical Systems*, Vol. 4, No. 2, pp. 159-179, 1997. [cited 13 December 2003]. Available at: <http://www.comp.dit.ie/pbrowne/Information%20Systems%20Research%20Practice/JGS%201997.pdf>.

ICS 35.240.70

Price based on 70 pages