INTERNATIONAL STANDARD

ISO 19110

First edition
2005-02-15

Geographic information — Methodology for feature cataloguing

Information géographique — Méthodologie de catalogage des entités

Reference number
ISO 19110:2005(E)

Contents

Foreword

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

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

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

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

ISO 19110 was prepared by Technical Committee ISO/TC 211, *Geographic information/Geomatics*.

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 International Standard 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 upon 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, *Geographic information — Rules for application schema* 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.

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 International Standard. 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 International Standard 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.

Geographic information — Methodology for feature cataloguing

1 Scope

This International Standard defines the methodology for cataloguing feature types. This International Standard specifies how the classification of feature types is organized into a feature catalogue and presented to the users of a set of geographic data. This International Standard is applicable to creating catalogues of feature types in previously uncatalogued domains and to revising existing feature catalogues to comply with standard practice. This International Standard 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.

This International Standard is applicable to the definition of geographic features at the type level. This International Standard is not applicable to the representation of individual instances of each type. This International Standard excludes spatial, temporal, and portrayal schemas as specified in ISO 19107, ISO 19108, and ISO 19117, respectively. It also excludes collection criteria for feature instances.

This International Standard 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 Conformance

Because this International Standard specifies a number of options that are not required for all feature catalogues, this clause specifies 12 conformance classes. These classes are differentiated on the basis of three criteria:

a) What elements of a feature type are required in a catalogue:

   1) feature attributes only?

   2) feature attributes and feature associations?

   3) feature attributes, feature associations, and feature operations?

b) Is there a requirement to link feature attributes, feature associations, and feature operations to only one feature type or may they be linked to multiple feature types?

c) Is there a requirement to include inheritance relationships in the feature catalogue?

Annex A specifies a test module for each of the conformance classes, as shown in Table 1.

Table 1 — Conformance classes

Attributes only	Attributes and associations	Attributes, associations and operations	Properties associated with multiple features	Inheritance relationships included	Test module
X	—	—	—	—	A.17
—	X	—	—	—	A.18
—	—	X	—	—	A.19
X	—	—	X	—	A.20
—	X	—	X	—	A.21
—	—	X	X	—	A.22
X	—	—	—	X	A.23
—	X	—	—	X	A.24
—	—	X	—	X	A.25
X	—	—	X	X	A.26
—	X	—	X	X	A.27
—	—	X	X	X	A.28

3 Normative references

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

ISO/TS 19103:—[1)], *Geographic information — Conceptual schema language*

ISO 19109:—[1)], *Geographic information — Rules for application schema*

ISO 19115:2003, *Geographic information — Metadata*

4 Terms and definitions

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

4.1
feature
abstraction of real world phenomena

[ISO 19101]

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

NOTE A feature may occur as a type or an instance. Feature type or feature instance should be used when only one is meant.

1) To be published.

4.2
feature association
relationship that links instances of one **feature** (4.1) type with instances of the same or a different **feature** type

4.3
feature attribute
characteristic of a **feature** (4.1)

[ISO 19101]

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 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.

4.4
feature catalogue
catalogue containing definitions and descriptions of the **feature** (4.1) types, **feature attributes** (4.3), and **feature associations** (4.2) occurring in one or more sets of geographic data, together with any feature operations that may be applied

4.5
feature operation
operation that every instance of a **feature** (4.1) 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 Sometimes feature operations provide a basis for feature type definition.

4.6
functional language
language in which **feature operations** are formally specified

NOTE In a functional language, feature types may be represented as abstract data types.

5 Abbreviations

DIGEST	Digital Geographic Information Exchange Standard
FACC	Feature and Attribute Coding Catalogue
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 Principal requirements

6.1 Feature catalogue

A feature catalogue shall present the abstraction of reality represented in one or more sets of geographic data as a defined classification of phenomena. The basic level of classification in a feature catalogue shall be the feature type. A feature catalogue shall be available in electronic form for any set of geographic data that contains features. A feature catalogue may also comply with the specifications of this International Standard independently of any existing set of geographic data.

6.2 Information elements

6.2.1 Introduction

The following clauses specify general and specific requirements for feature catalogue information elements. Annex B specifies detailed requirements. Annex C illustrates the application of these requirements. Annex D discusses the application of feature operations as the conceptual basis for determining feature types in a feature catalogue.

6.2.2 Completeness

A template for the representation of feature classification information is specified in Annex B. A feature catalogue prepared according to this template shall document all of the feature types found in a given set of geographic data. The feature catalogue shall include identification information as specified in Annex B. The feature catalogue shall include definitions and descriptions of all feature types contained in the data, including any feature attributes and feature associations contained in the data that are associated with each feature type, and optionally including feature operations that are supported by the data. To ensure predictability and comparability of feature catalogue content across different applications, it is recommended that the feature catalogue should include only the elements specified in Annex B. 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 birth and death of feature instances, and other semantic elements of the feature catalogue may be included in a conceptual schema as structured comments or as attributes.

6.2.3 General requirements

6.2.3.1 Form of names

All feature types, feature attributes, feature associations, association roles, and feature operations included in a feature catalogue shall be identified by a name that is unique within that feature catalogue. If the name of a feature type, feature attribute, feature association, association role, or feature operation appears more than once in that feature catalogue, the definition shall be the same for all occurrences.

6.2.3.2 Form of definitions

Definitions of feature types, feature attributes, feature attribute listed values, feature associations, association roles, and feature operations shall be given in a natural language. These definitions shall be included in the catalogue, unless the catalogue specifies a separate definition source. If the same term appears in both the definition source and the feature catalogue, the definition in the feature catalogue shall apply.

6.2.4 Requirements for feature types

Each feature type shall be identified by a name and defined in a natural language. Each feature type may also be identified by an alphanumeric code that is unique within the catalogue and it may have a set of aliases. The feature catalogue shall also include, for each feature type, its feature operations and associated feature attributes, feature associations and association roles, if any. The use of functional language specifications to help define feature types is recommended.

6.2.5 Requirements for feature operations

Feature operations, if any, shall be identified and defined for each feature type. Feature attributes involved in each feature operation shall be specified well as any feature types affected by the operation. The definition shall include a natural language definition and may be formally specified in a functional language.

6.2.6 Requirements for feature attributes

Feature attributes, if any, shall be identified and defined for each feature type. The definition shall include a natural language definition and a specified data type for values of the attribute. Each feature attribute may also be identified by an alphanumeric code that is unique within the catalogue.

6.2.7 Requirements for feature attribute listed values

Feature-attribute listed values, if any, shall be labelled for each feature attribute. The label shall be unique within the feature attribute of which it is a listed value. Each listed value may also be identified by an alphanumeric code that is unique within the feature attribute of which it is a listed value.

6.2.8 Requirements for feature associations

Feature associations, if any, shall be named and defined. Each feature association may also be identified by an alphanumeric code that is unique within the catalogue. The names and roles of the feature types that participate in the association shall be specified.

6.2.9 Requirements for association roles

Association roles, if any, shall be named and defined. The name of the feature type that holds the role and the association in which it participates shall be specified.

Annex A
(normative)

Abstract test suite

A.1 Introduction

This normative annex presents the abstract test suite for evaluating conformance to this International Standard. This abstract test suite contains fifteen test cases and twelve test modules: a test case for the existence and form of feature catalogue information (A.2); a test case for general feature catalogue requirements (A.3); test cases for each principal feature catalogue information class (A.4 through A.16); and test modules for specific subsets of the functionality of a feature catalogue (A.17 through A.28).

Test cases are based on each principal-feature catalogue information class specified in Annex B, Tables B.1 through B.15. Each class-based test case consists of examining each class element (attribute or role) and verifying that:

— the obligation/condition specification for the presence of the element is met;

— the maximum number of occurrences of the element is not exceeded;

— the type of the value of the element is correct;

NOTE Unless otherwise stated, e.g., by specifying a type in a well-known package or ISO standard, the type specifications as will be given in ISO/TS 19103 may be applied.

— the value of the element is in accordance with the element description;

— any specified constraint on the element is met.

Tests on the class as a whole are specified as class-constraints as specified in Annex B, Tables B.1 through B.15, and/or in test modules.

Test modules are based on useful subsets of the functionality of the feature catalogue template specified in Annex B. Subsets are organized starting with core functionality capable of representing feature types and feature attributes that are unique to a feature type (see A.17). The core representation functionality is extended by supporting one or more of the following

— additional feature property types (for association roles, see A.18; for both association roles and feature operations, see A.19);

— additional relationships among feature catalogue information elements (for multiple-use feature attributes, see A.20; for inheritance, see A.23; for both multiple-use feature attributes and inheritance, see A.26);

— both additional feature property types and additional relationships among feature catalogue information elements (see A.21, A.22, A.24, A.25, A.27 and A.28).

Test modules are specified by the test cases that apply. Test modules A.18 through A.28 extend the core representation functionality specified by test module A.17. These extensions are summarized in Table 1.

To check that a feature catalogue conforms to this International Standard, verify that all of the requirements in at least one test module are satisfied.

A.2 Test case for existence and form of feature catalogue information

Information for the test case is as follows:

a) test purpose: verify the existence and form of a feature catalogue;

b) test method: check whether the feature catalogue exists and can be obtained in electronic form, by obtaining a copy of the feature catalogue such as on a computer disk or through a file transfer;

c) reference: 6.1;

d) test type: basic.

A.3 Test case for general feature catalogue requirements

Information for the test case is as follows:

a) test purpose: verify that general feature catalogue requirements are met;

b) test method: check

   1) if the feature catalogue is specified as applying to a given set of geographic data, then the feature catalogue documents all of the feature types found in that set of geographic data,

   2) whether all feature types, feature attributes, feature associations, association roles, and feature operations are identified by a name that is unique within the feature catalogue,

   3) whether all feature attribute listed values are identified by a label that is unique within the feature attribute of which it is a listed value,

   4) whether all feature types, feature attributes, feature associations, association roles, and feature operations are either defined or reference a definition from another source,

   5) if any feature type, feature attribute, or feature association is identified by an alphanumeric code, that alphanumeric code is unique within the feature catalogue,

   6) if any feature attribute listed value is identified by an alphanumeric code, that alphanumeric code is unique within the feature attribute of which it is a listed value;

c) reference: 6.2;

d) test type: capability.

A.4 Test case for the feature catalogue class

Information for the test case is as follows:

a) test purpose: verify that the required information is included in objects of class feature catalogue;

b) test method: test each attribute and role listed in Table B.1 by verifying that for each the specified description, obligation/condition, maximum occurrence, type, and constraint are satisfied;

c) reference: Annex B, Table B.1;

d) test type: capability.

A.5 Test case for the feature type class

Information for the test case is as follows:

a) test purpose: verify that the required information is included in objects of class feature type;

b) test method: test each attribute and role listed in Table B.2 by verifying that for each the specified description, obligation/condition, maximum occurrence, type, and constraint are satisfied;

c) reference: Annex B, Table B.2;

d) test type: capability.

A.6 Test case for the inheritance relation class

Information for the test case is as follows:

a) test purpose: verify that the required information is included in objects of class inheritance relation;

b) test method: test each attribute and role listed in Table B.3 by verifying that for each the specified description, obligation/condition, maximum occurrence, type, and constraint are satisfied;

c) reference: Annex B, Table B.3;

d) test type: capability.

A.7 Test case for the feature operation class

Information for the test case is as follows:

a) test purpose: verify that the required information is included in objects of class feature operation;

b) test method: test each attribute and role listed in Table B.5 (and Table B.4) by verifying that for each the specified description, obligation/condition, maximum occurrence, type, and constraint are satisfied. In addition:

   1) If feature attributes are unique to a feature type (test modules A.19 and A.25), the type of the following roles shall be FC_FeatureAttribute rather than FC_BoundFeatureAttribute:

      i) FC_FeatureOperation::triggeredByValuesOf (Table B.5, line 5.3),

      ii) FC_FeatureOperation::observesValuesOf (Table B.5, line 5.4),

      iii) FC_FeatureOperation::affectsValuesOf (Table B.5, line 5.5);

   2) If the role FC_FeatureOperation::featureType (Table B.4, line 4.4) exists more than once, the association class FC_Binding for each shall satisfy test case A.8. This condition occurs in test modules A.22 and A.28;

c) reference: Annex B, Tables B.4 and B.5;

d) test type: capability.

A.8 Test case for the binding class

Information for the test case is as follows:

a) test purpose: verify that the required information is included in objects of class binding;

b) test method: test each attribute listed in Table B.6 by verifying that for each the specified description, obligation/condition, maximum occurrence, type, and constraint are satisfied;

c) reference: Annex B, Table B.6;

d) test type: capability.

A.9 Test case for the constraint class

Information for the test case is as follows:

a) test purpose: verify that the required information is included in objects of class constraint;

b) test method: test each attribute listed in Table B.7 by verifying that for each the specified description, obligation/condition, maximum occurrence, type, and constraint are satisfied;

c) reference: Annex B, Table B.7;

d) test type: capability.

A.10 Test case for the feature attribute class

Information for the test case is as follows:

a) test purpose: verify that the required information is included in objects of class feature attribute;

b) test method: test each attribute and role listed in Table B.8 (and Table B.4) by verifying that for each the specified description, obligation/condition, maximum occurrence, type, and constraint are satisfied. In addition:

   If the role FC_FeatureAttribute::featureType (Table B.4, line 4.4) exists more than once, the association class FC_Binding for each shall satisfy test case A.8. This condition occurs in test modules A.20, A.21, A.22, A.26, A.27 and A.28;

c) reference: Annex B, Tables B.4 and B.8;

d) test type: capability.

A.11 Test case for the association role class

Information for the test case is as follows:

a) test purpose: verify that the required information is included in objects of class association role;

b) test method: test each attribute and role listed in Table B.9 (and Table B.4) by verifying that for each the specified description, obligation/condition, maximum occurrence, type, and constraint are satisfied. In addition:

   1) Verify that the value of attribute type (Table B.9, line 9.2) is a domain code from Table B.10,

2) If the role FC_AssociationRole::featureType (Table B.4, line 4.4) exists more than once, the association class FC_Binding for each shall satisfy test case A.8. This condition occurs in test modules A.21, A.22, A.27 and A.28;

c) reference: Annex B, Tables B.4, B.9 and B.10;

d) test type: capability.

A.12 Test case for the listed value class

Information for the test case is as follows:

a) test purpose: verify that the required information is included in objects of class listed value;

b) test method: test each attribute and role listed in Table B.11 by verifying that for each the specified description, obligation/condition, maximum occurrence, type, and constraint are satisfied;

c) reference: Annex B, Table B.11;

d) test type: capability.

A.13 Test case for the feature association class

Information for the test case is as follows:

a) test purpose: verify that the required information is included in objects of class feature association;

b) test method: test each attribute and role listed in Table B.12 (and Table B.2) by verifying that for each the specified description, obligation/condition, maximum occurrence, type, and constraint are satisfied;

c) reference: Annex B, Tables B.2 and B.12;

d) test type: capability.

A.14 Test case for the definition source class

Information for the test case is as follows:

a) test purpose: verify that the required information is included in objects of class definition source;

b) test method: test each attribute listed in Table B.13 by verifying that for each the specified description, obligation/condition, maximum occurrence, type, and constraint are satisfied;

c) reference: Annex B, Table B.13;

d) test type: capability.

A.15 Test case for the definition reference class

Information for the test case is as follows:

a) test purpose: verify that the required information is included in objects of class definition reference;

b) test method: test each attribute and role listed in Table B.14 by verifying that for each the specified description, obligation/condition, maximum occurrence, type, and constraint are satisfied;

c) reference: Annex B, Table B.14;

d) test type: capability.

A.16 Test case for the bound feature attribute class

Information for the test case is as follows:

a) test purpose: verify that the required information is included in objects of class bound feature attribute;

b) test method: test each role listed in Table B.15 by verifying that for each the specified description, obligation/condition, maximum occurrence, type, and constraint are satisfied;

c) reference: Annex B, Table B.15;

d) test type: capability.

A.17 Test module for a catalogue with single-use feature attributes

Information for the test module is as follows:

a) test purpose: verify that the feature catalogue supports core representation functionality, comprised of feature types and feature attributes that are unique to a feature type;

b) test method: perform the following nine test cases: A.2 (existence), A.3 (general), A.4 (feature catalogue), A.5 (feature type), A.9 (constraint), A.10 (feature attribute), A.12 (listed value), A.14 (definition source) and A.15 (definition reference);

c) reference: A.2, A.3, A.4, A.5, A.9, A.10, A.12, A.14 and A.15;

d) test type: capability.

A.18 Test module for a catalogue with single-use feature attributes and association roles

Information for the test module is as follows:

a) test purpose: verify that the feature catalogue supports core representation functionality, comprised of feature types, and feature attributes and association roles that are unique to a feature type;

b) test method: perform test module A.17 and the two test cases A.11 (association role) and A.13 (feature association);

c) reference: A.11, A.13 and A.17;

d) test type: capability.

A.19 Test module for a catalogue with single-use feature attributes, association roles and operations

Information for the test module is as follows:

a) test purpose: verify that the feature catalogue supports core representation functionality, comprised of feature types, and feature attributes, association roles and feature operations that are unique to a feature type;

b) test method: perform test module A.18 and test case A.7 (feature operation), including verification that the type of the following FC_FeatureOperation roles shall be FC_FeatureAttribute rather than FC_BoundFeatureAttribute:

1) FC_FeatureOperation::triggeredByValuesOf (Table B.5, line 5.3),

2) FC_FeatureOperation::observesValuesOf (Table B.5, line 5.4), and

3) FC_FeatureOperation::affectsValuesOf (Table B.5, line 5.5);

c) reference: A.7; A.18 and Annex B, Table B.5;

d) test type: capability.

A.20 Test module for a catalogue with multiple-use feature attributes

Information for the test module is as follows:

a) test purpose: verify that the feature catalogue supports core representation functionality, comprised of feature types and feature attributes that may be members of multiple feature types;

b) test method: perform test module A.17 and in addition:

If the role FC_FeatureAttribute::featureType (Table B.4, line 4.4) exists more than once for the same FC_FeatureAttribute, the association class FC_Binding for each shall satisfy test case A.8 (binding);

c) reference: A.8; A.17 and Annex B, Table B.4;

d) test type: capability.

A.21 Test module for a catalogue with multiple-use feature attributes and association roles

Information for the test module is as follows:

a) test purpose: verify that the feature catalogue supports core representation functionality, comprised of feature types, and feature attributes and association roles that may be members of multiple feature types;

b) test method: perform test module A.18 and in addition:

1) If the role FC_FeatureAttribute::featureType (Table B.4, line 4.4) exists more than once for the same FC_FeatureAttribute the association class FC_Binding for each shall satisfy test case A.8 (binding),

2) If the role FC_AssociationRole::featureType (Table B.4, line 4.4) exists more than once for the same FC_AssociationRole the association class FC_Binding for each shall satisfy test case A.8 (binding);

c) reference: A.8, A.18 and Annex B, Table B.4;

d) test type: capability.

A.22 Test module for a catalogue with multiple-use feature attributes, associations and operations

Information for the test module is as follows:

a) test purpose: verify that the feature catalogue supports core representation functionality, comprised of feature types, and feature attributes, association roles and feature operations that may be members of multiple feature types;

b) test method: perform test module A.19 and the test case A.16 (bound feature attribute). In addition:

   1) If the role FC_FeatureAttribute::featureType (Table B.4, line 4.4) exists more than once for the same FC_FeatureAttribute, the association class FC_Binding for each shall satisfy test case A.8 (binding),

   2) If the role FC_AssociationRole::featureType (Table B.4, line 4.4) exists more than once for the same FC_AssociationRole, the association class FC_Binding for each shall satisfy test case A.8 (binding),

   3) If the role FC_FeatureOperation::featureType (Table B.4, line 4.4) exists more than once for the same FC_FeatureOperation, the association class FC_Binding for each shall satisfy test case A.8 (binding);

c) reference: A.8, A.16; A.19 and Annex B, Table B.4;

d) test type: capability.

A.23 Test module for a catalogue with single-use feature attributes and inheritance

Information for the test module is as follows:

a) test purpose: verify that the feature catalogue supports core representation functionality, comprised of feature types and feature attributes that are unique to a feature type, plus inheritance;

b) test method: perform test module A.17 and the test case A.6 (inheritance relation);

c) reference: A.6, A.17;

d) test type: capability.

A.24 Test module for a catalogue with single-use feature attributes and association roles with inheritance

Information for the test module is as follows:

a) test purpose: verify that the feature catalogue supports core representation functionality, comprised of feature types, and feature attributes and association roles that are unique to a feature type, plus inheritance;

b) test method: perform test module A.18 and the test case A.6 (inheritance relation);

c) reference: A.6, A.18;

d) test type: capability.

A.25 Test module for a catalogue with single-use feature attributes, association roles and operations with inheritance

Information for the test module is as follows:

a) test purpose: verify that the feature catalogue supports core representation functionality, comprised of feature types, and feature attributes, association roles and feature operations that are unique to a feature type, plus inheritance;

b) test method: perform test module A.19 and the test case A.6 (inheritance relation);

c) reference: A.6, A.19;

d) test type: capability.

A.26 Test module for a catalogue with multiple-use feature attributes and inheritance

Information for the test module is as follows:

a) test purpose: verify that the feature catalogue supports core representation functionality, comprised of feature types and feature attributes that may be members of multiple feature types, plus inheritance;

b) test method: perform test module A.20 and the test case A.6 (inheritance relation);

c) reference: A.6, A.20;

d) test type: capability.

A.27 Test module for a catalogue with multiple-use feature attributes and association roles with inheritance

Information for the test module is as follows:

a) test purpose: verify that the feature catalogue supports core representation functionality, comprised of feature types, and feature attributes and association roles that may be members of multiple feature types, plus inheritance;

b) test method: perform test module A.21 and the test case A.6 (inheritance relation);

c) reference: A.6, A.21;

d) test type: capability.

A.28 Test module for a catalogue with multiple-use feature attributes, association roles and operations with inheritance

Information for the test module is as follows:

a)	test purpose:	verify that the feature catalogue supports core representation functionality, comprised of feature types, and feature attributes, association roles and feature operations that may be members of multiple feature types, plus inheritance;
b)	test method:	perform test module A.22 and the test case A.6 (inheritance relation);
c)	reference:	A.6, A.22;
d)	test type:	capability.

Annex B
(normative)

Feature catalogue template

This normative annex presents the template for the organization of feature catalogue information according to this International Standard. Tables B.1 through B.15 present templates for the representation of feature catalogue contents. Feature catalogue information elements and the relationships among them are identified using the following notation for obligation and conditions:

M – The element is mandatory; it shall be included in the feature catalogue.

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.

Table B.15 and Figure B.1 illustrate the feature catalogue template in the form of a Unified Modeling Language (UML) package (ISO/TS 19103). Figures B.2 through B.5 illustrate how the structure of a standard feature catalogue conforms to the General Feature Model (ISO 19109 —[1], 7.3).

Table B.1 — Feature catalogue

No.	Name/Role Name	Description	Obligation/ Condition[a]	Maximum Occurrence[b]	Type	Constraint
1	Class FC_FeatureCatalogue	A feature catalogue contains its identification and contact information, and definition of some number of feature types with other information necessary for those definitions.	—	—	—	—
1.1	Attribute name	name for this feature catalogue	M	1	CharacterString	—
1.2	Attribute scope	subject domain(s) of feature types defined in this feature catalogue	M	N	CharacterString	—
1.3	Attribute fieldOfApplication	description of kind(s) of use to which this feature catalogue may be put	O	N	CharacterString	—
1.4	Attribute versionNumber	version number of this feature catalogue, which may include both a major version number or letter and a sequence of minor release numbers or letters, such as “3.2.4a.” The format of this attribute may differ between cataloguing authorities.	M	1	CharacterString	—
1.5	Attribute versionDate	effective date of this feature catalogue	M	1	Date	—
1.6	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 Metadata:: CI_ResponsibleParty	—
1.7	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 feature catalogue.	1	CharacterString	—
1.8	Role featureType	role that links this feature catalogue to the feature types that it contains	M	N	FC_FeatureType	Aggregation
1.9	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

[a] M = mandatory; O = optional; C = conditional.

[b] N = repeating occurrences.

Table B.2 — Feature type

No.	Name/Role Name	Description	Obligation/ Condition[a]	Maximum Occurrence[b]	Type	Constraint
2	Class FC_FeatureType	class of real world phenomena with common properties	—	—	—	name 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	—
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	Role inheritsFrom	role that links this feature type to a set of superclasses from which it inherits operations, associations and properties	O	N	FC_InheritanceRelation	—
2.7	Role inheritsTo	role that links this feature type to a set of subclasses which inherit its operations, associations and properties	O	N	FC_InheritanceRelation	—
2.8	Role featureCatalogue	role that links this feature type to the feature catalogue that contains it	M	1	FC_FeatureCatalogue	—
2.9	Role carrierOfCharacteristics	role that links this feature type to the property types that it contains The association class FC_Binding describes particular information regarding the use of this property type within this feature type.	O	N	FC_PropertyType; FC_Binding	Association Class is FC_Binding; Aggregation

Table B.2 (*continued*)

No.	Name/Role Name	Description	Obligation/ Condition [a]	Maximum Occurrence[b]	Type	Constraint
2.10	Role constrainedBy	role that links this feature type to the constraints placed upon it	O	N	FC_Constraint	—
2.11	Role definitionReference	role that links this feature type to the source of its definition	O	1	FC_DefinitionReference	—

[a] M = mandatory; O = optional; C = conditional.

[b] N = repeating occurrences.

Table B.3 — Inheritance relation

No.	Name/Role Name	Description	Obligation/ Condition[a]	Maximum Occurrence[b]	Type	Constraint
3	Class FC_InheritanceRelation	FC_InheritanceRelation realizes GF_InheritanceRelation.	—	—	—	FC_InheritanceRelation always assumes that its GF_InheritanceRelation::uniqueInstance is TRUE.
3.1	Attribute name	text string that uniquely identifies this inheritance relation within the feature catalogue that contains this inheritance relation	M	1	CharacterString	—
3.2	Attribute description	natural language description of this inheritance relation	M	1	CharacterString	—
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	—
3.4	Role subtype	identifies one feature type to which the associated superclass feature type supplies inherited properties, associations and operation	M	1	FC_FeatureType	—
3.5	Role supertype	identifies one feature type from which the associated subtype class inherits properties, associations and operations	M	1	FC_FeatureType	—

[a] M = mandatory.

Table B.4 — Property type

No.	Name/Role Name	Description	Obligation/ Condition[a]	Maximum Occurrence[b]	Type	Constraint
4	Class FC_PropertyType	abstract class for feature properties	—	—	—	—
4.1	Attribute memberName	member name that locates this member within a feature type	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 definition not provided by definition source.	1	CharacterString	—
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	Role featureType	role that links the operations, attributes and association roles with feature types that contain them The association class FC_Binding describes particular information regarding the use of this property type within this feature type. This is a “strong aggregation” or composition in the General Feature Model (ISO 19109). Here it is realized by a “weak aggregation”. This is valid since a weak aggregation can be converted to a strong aggregation by replicating the value of the target role for each use. In this case, the property type represents a value, and the realization of the GFM's property type would create a GF_PropertyType whose identity is the combination of the FC_PropertyType and the owning FC_FeatureType.	O	N	FC_FeatureType; FC_Binding	Association Class is FC_Binding; Aggregation
4.5	Role constrainedBy	role that links this property type to the constraints placed upon it	O	N	FC_Constraint	—
4.6	Role definitionReference	role that links this instance to the source of its definition	O	1	FC_DefinitionReference	—

[a] M = mandatory; O = optional; C = conditional.

[b] N = repeating occurrences.

Table B.5 — Feature operation

No.	Name/Role Name	Description	Obligation/ Condition[a]	Maximum Occurrence[b]	Type	Constraint
5	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 — Property type	—	—	—	—
5.1	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	Character String	—
5.2	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	Character String	—
5.3	Role triggeredByValuesOf	specifies attribute which may trigger an operation	O	N	FC_BoundFeatureAttribute	—
5.4	Role observesValuesOf	specifies attribute that may be used as input to perform an operation	O	N	FC_BoundFeatureAttribute	—
5.5	Role affectsValuesOf	specifies attribute that will be affected by an operation	O	N	FC_BoundFeatureAttribute	—

[a] M = mandatory; O = optional.

[b] N = repeating occurrences.

Table B.6 — Binding

No.	Name/Role Name	Description	Obligation/ Condition[a]	Maximum Occurrence	Type	Constraint
6	Class FC_Binding	class that is used to describe the specifics of how a property type is bound to a particular feature type; used as an association class for the association MemberOf between feature type and property type	—	—	—	—
6.1	Attribute description	description of how a property type is bound to a particular feature type	O	1	CharacterString	—

[a] O = optional.

Table B.7 — Constraint

No.	Name/Role Name	Description	Obligation/ Condition[a]	Maximum Occurrence	Type	Constraint
7	Class FC_Constraint	class for defining constraints for types	—	—	—	—
7.1	Attribute description	description of the constraint that is being applied	M	1	CharacterString	—
[a] M = mandatory.						

Table B.8 — Feature attribute

No.	Name/Role Name	Description	Obligation/ Condition[a]	Maximum Occurrence[b]	Type	Constraint
8	Class FC_FeatureAttribute	characteristic of a feature type	—	—	—	—
	Subtype of FC_PropertyType	Table B.4 — Property type	—	—	—	—
8.1	Attribute code	numeric or alphanumeric code that uniquely identifies the feature attribute within the feature catalogue	O	1	CharacterString	—
8.2	Attribute valueMeasurementUnit	unit of measure used for values of this feature attribute	O	1	UnitOfMeasure	—
8.3	Attribute listedValue	permissible values of this feature attribute. If present, then this feature attribute is enumerated (such as with a code list). If not present, then this feature attribute is not enumerated.	C/Mandatory if feature attribute valueType is not given.	N	FC_ListedValue	—
8.4	Attribute valueType	type of the value of this feature attribute; a name from some namespace. Implementations of this International Standard shall specify which namespace implementation is to be used. One possibility is the URI (RFC 2396)[6].	C/Mandatory if feature attribute listedValue is empty.	1	TypeName	—
[a] O = optional; C = conditional. [b] N = repeating occurrences.						

Table B.9 — Association role

No.	Name/Role Name	Description	Obligation/ Condition[a]	Maximum Occurrence	Type	Constraint
9	Class FC_AssociationRole	role of the feature association FC_AssociationRole::relation.	—	—	—	roleName = FC_PropertyType::memberName; FC_PropertyType::cardinality realizes GF_AssociationRole::cardinality
	Subtype of FC_PropertyType	Table B.4 — Property type	—	—	—	—
9.1	Attribute cardinality	number of instances of the feature type that can act in this role relative to a single instance of the target feature type of the association	M	1	Multiplicity	Initial value = 0..*
9.2	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")
9.3	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
9.4	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
9.5	Role relation	relation of which this association role is a part	M	1	FC_Feature Association	—
9.6	Role valueType	type of the target value of this association role	M	1	FC_Feature Type	—

[a] M = mandatory.

Table B.10 — Role type code list

No.	Name	Domain Code	Description
10	Class FC_RoleType	—	code list for the classification of roles
10.1	ordinary	1	indicates an ordinary association
10.2	aggregation	2	indicates a UML aggregation (part role)
10.3	composition	3	indicates a UML composition (member role)

Table B.11 — Listed value

No.	Name/Role Name	Description	Obligation/ Condition[a]	Maximum Occurrence	Type	Constraint
11	Class FC_ListedValue	value for an enumerated feature attribute domain, including its codes and interpretation	—	—	—	—
11.1	Attribute label	descriptive label that uniquely identifies one value of the feature attribute	M	1	CharacterString	—
11.2	Attribute code	numeric or alphanumeric code (such as a country code) that uniquely identifies this value of the feature attribute	O	1	CharacterString	—
11.3	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	—
11.4	Role definitionReference	role that links this instance to the source of its definition	O	1	FC_DefinitionReference	—

[a] M = mandatory; O = optional.

Table B.12 — Feature association

	Name/Role Name	Description	Obligation/ Condition[a]	Maximum Occurrence[b]	Type	Constraint
12	Class FC_FeatureAssociation	relationship that links instances of this feature type with instances of the same or of a different feature type. The memberOf-linkBetween 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 — Feature type	—	—	—	—
12.1	Role role	roles that are a part of this association	M	N	FC_AssociationRole	Aggregation

[a] M = mandatory.

[b] N = repeating occurrences.

Table B.13 — Definition source

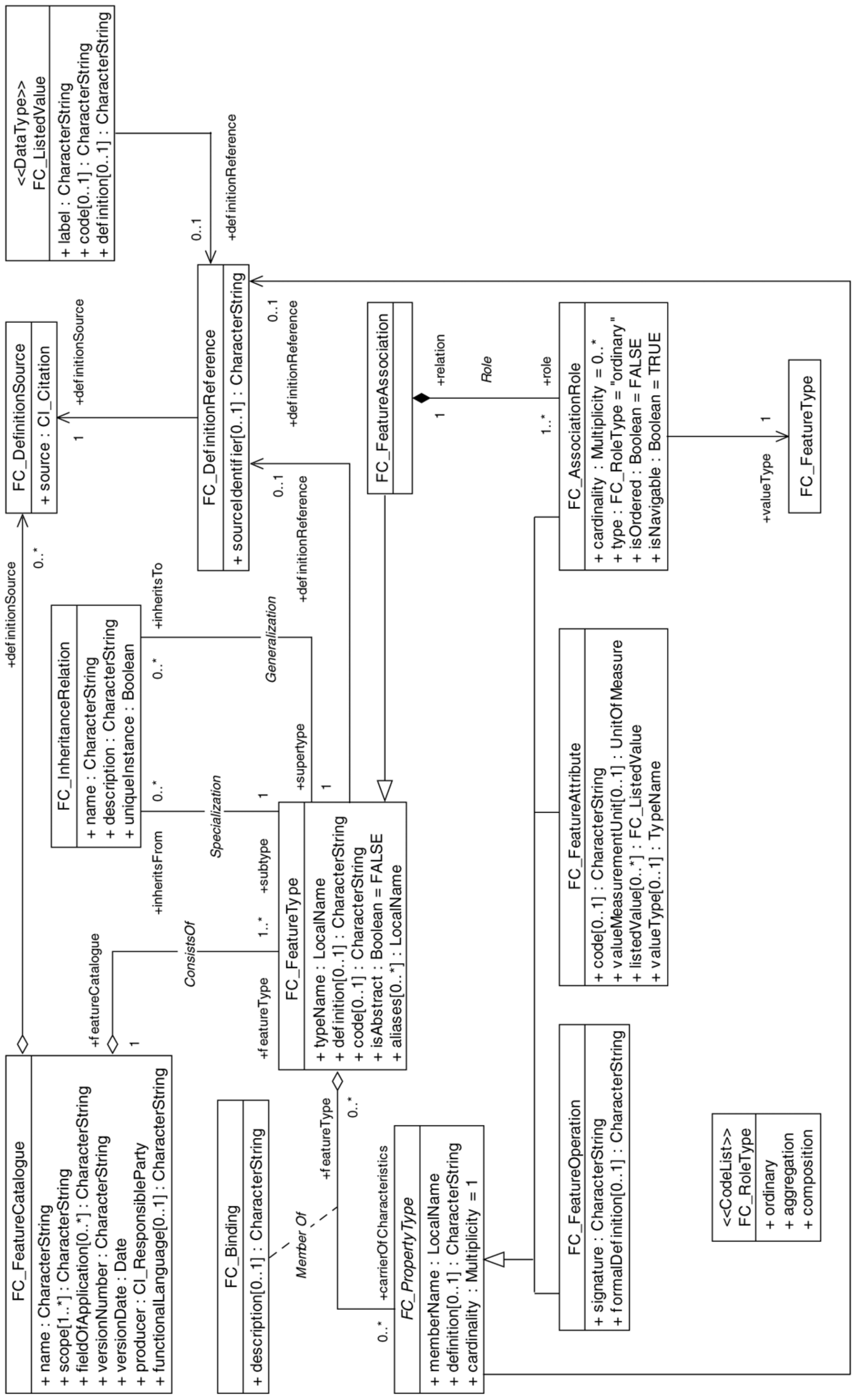
No.	Name/Role Name	Description	Obligation/ Condition[a]	Maximum Occurrence[b]	Type	Constraint
13	Class FC_DefinitionSource	class that specifies the source of a definition	—	—	—	—
13.1	Attribute source	actual citation of the source, sufficient to identify the document and how to obtain it	M	1	ISO 19115 Metadata:: CI_Citation	—
[a] M = mandatory.						

Table B.14 — Definition reference

No.	Name/Role Name	Description	Obligation/ Condition[a]	Maximum Occurrence	Type	Constraint
14	Class FC_DefinitionReference	class that links a data instance to the source of its definition	—	—	—	—
14.1	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	—
14.2	Role definitionSource	role that links this definition reference to the citation for the source document	M	1	FC_DefinitionSource	—
[a] M = mandatory; O = optional.						

Table B.15 — Bound feature attribute

No.	Name/Role Name	Description	Obligation/ Condition[a]	Maximum Occurrence	Type	Constraint
15	Class FC_BoundFeatureAttribute	class that represents an association between a particular feature type and a particular property type, in order that operational effect information may be supplied for feature operations	—	—	—	—
15.1	Role featureType	feature type involved in the binding	M	1	FC_FeatureType	—
15.2	Role attribute	property type involved in the binding	M	1	FC_FeatureAttribute	—
[a] M = mandatory.						



<<DataType>>
FC_ListedValue
+ label : CharacterString
+ code[0..1] : CharacterString
+ definition[0..1] : CharacterString
0..1
+definitionReference
FC_DefinitionSource
+ source : CI_Citation
1
+definitionSource
FC_DefinitionReference
+ sourceIdentifier[0..1] : CharacterString
0..1
+definitionReference
0..1
+definitionReference
FC_FeatureAssociation
+relation
1
Role
+role
1..*
FC_AssociationRole
+ cardinality : Multiplicity = 0..*
+ type : FC_RoleType = "ordinary"
+ isOrdered : Boolean = FALSE
+ isNavigable : Boolean = TRUE
1
+valueType
FC_FeatureType
+definitionSource
0..*
FC_InheritanceRelation
+ name : CharacterString
+ description : CharacterString
+ uniqueInstance : Boolean
+inheritsTo
0..*
Generalization
+supertype
1
0..*
Specialization
1
+inheritsFrom
+subtype
FC_FeatureAttribute
+ code[0..1] : CharacterString
+ valueMeasurementUnit[0..1] : UnitOfMeasure
+ listedValue[0..*] : FC_ListedValue
+ valueType[0..1] : TypeName
+featureCatalogue
1
ConsistsOf
1..*
+featureType
FC_FeatureType
+ typeName : LocalName
+ definition[0..1] : CharacterString
+ code[0..1] : CharacterString
+ isAbstract : Boolean = FALSE
+ aliases[0..*] : LocalName
FC_FeatureCatalogue
+ name : CharacterString
+ scope[1..*] : CharacterString
+ fieldOfApplication[0..*] : CharacterString
+ versionNumber : CharacterString
+ versionDate : Date
+ producer : CI_ResponsibleParty
+ functionalLanguage[0..1] : CharacterString
FC_Binding
+ description[0..1] : CharacterString
Member Of
+featureType
0..*
+carrierOfCharacteristics
0..*
FC_PropertyType
+ memberName : LocalName
+ definition[0..1] : CharacterString
+ cardinality : Multiplicity = 1
FC_FeatureOperation
+ signature : CharacterString
+ formalDefinition[0..1] : CharacterString
<<CodeList>>
FC_RoleType
+ ordinary
+ aggregation
+ composition

Figure B.1 — Conceptual model of a feature catalogue

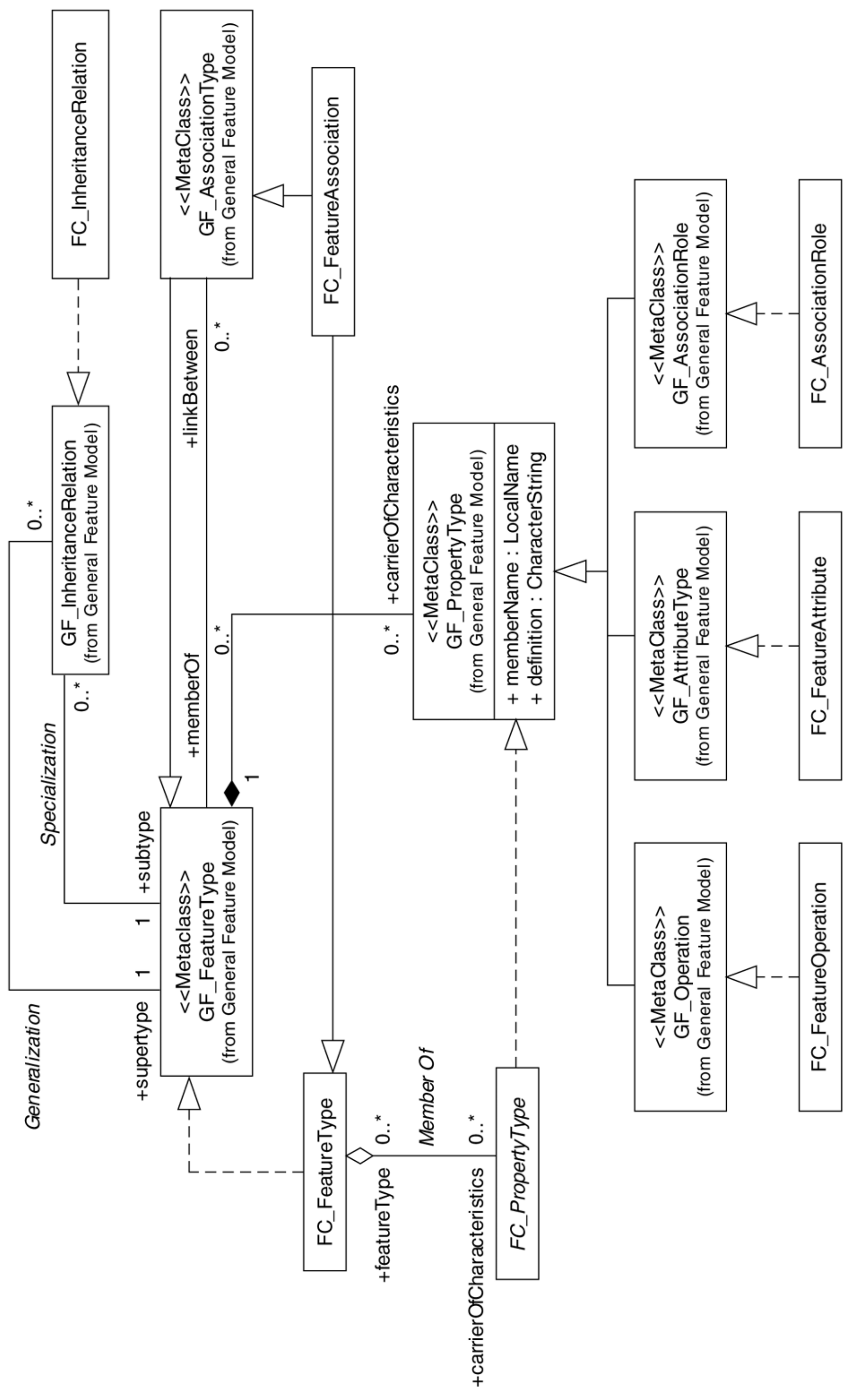
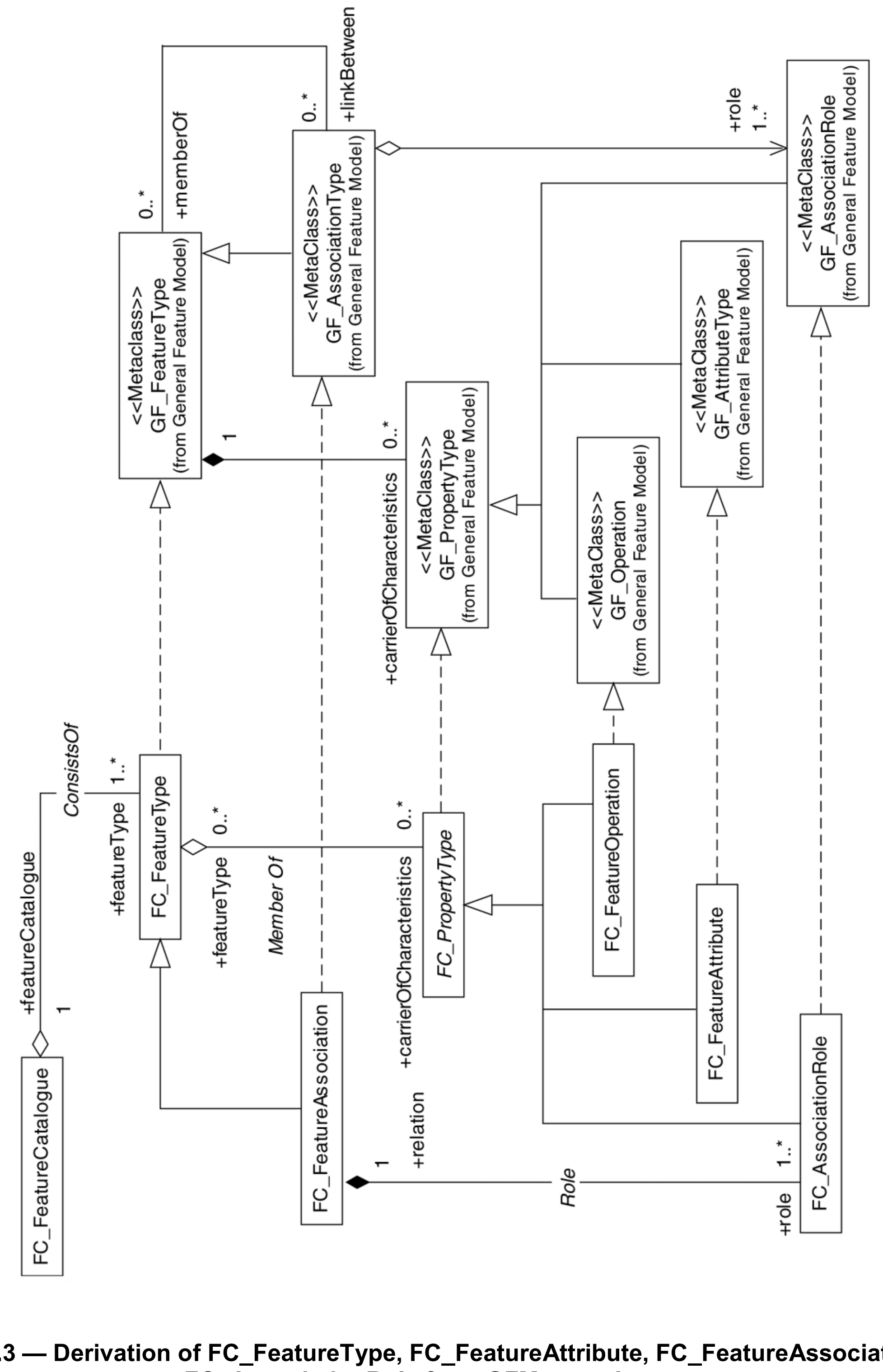


Figure B.2 — Feature cataloguing classes as realizations of General Feature Model metaclasses



<<Metaclass>> GF_FeatureType (from General Feature Model)
<<MetaClass>> GF_AssociationType (from General Feature Model)
<<MetaClass>> GF_AssociationRole (from General Feature Model)
<<MetaClass>> GF_PropertyType (from General Feature Model)
<<MetaClass>> GF_Operation (from General Feature Model)
<<MetaClass>> GF_AttributeType (from General Feature Model)
FC_FeatureCatalogue
FC_FeatureType
FC_FeatureAssociation
FC_PropertyType
FC_FeatureOperation
FC_FeatureAttribute
FC_AssociationRole
+memberOf
0..*
+linkBetween
0..*
+role
1..*
1
0..*
+carrierOfCharacteristics
+featureCatalogue
1
ConsistsOf
+featureType
1..*
+featureType
0..*
Member Of
+carrierOfCharacteristics
0..*
+relation
1
Role
+role
1..*

Figure B.3 — Derivation of FC_FeatureType, FC_FeatureAttribute, FC_FeatureAssociation and FC_AssociationRole from GFM metaclasses

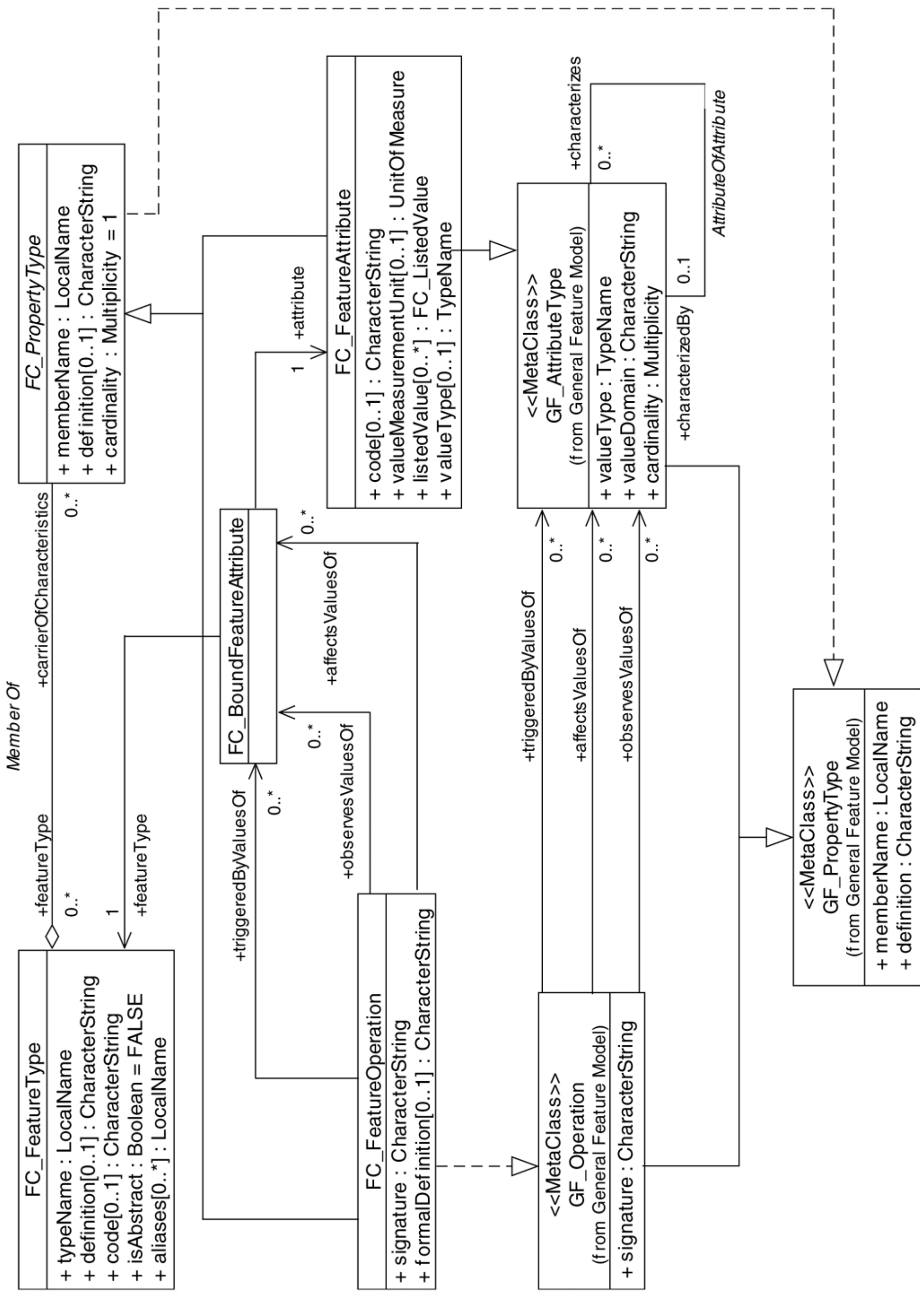

FC_PropertyType
+ memberName : LocalName
+ definition[0..1] : CharacterString
+ cardinality : Multiplicity = 1
FC_FeatureAttribute
+ code[0..1] : CharacterString
+ valueMeasurementUnit[0..1] : UnitOfMeasure
+ listedValue[0..*] : FC_ListedValue
+ valueType[0..1] : TypeName
<<MetaClass>>
GF_AttributeType
(from General Feature Model)
+ valueType : TypeName
+ valueDomain : CharacterString
+ cardinality : Multiplicity
+characterizes
0..*
AttributeOfAttribute
0..1
+characterizedBy
+attribute
1
FC_BoundFeatureAttribute
0..*
+affectsValuesOf
0..*
+observesValuesOf
0..*
+triggeredByValuesOf
Member Of
+carrierOfCharacteristics
0..*
+featureType
0..*
1
+featureType
FC_FeatureType
+ typeName : LocalName
+ definition[0..1] : CharacterString
+ code[0..1] : CharacterString
+ isAbstract : Boolean = FALSE
+ aliases[0..*] : LocalName
FC_FeatureOperation
+ signature : CharacterString
+ formalDefinition[0..1] : CharacterString
<<MetaClass>>
GF_Operation
(from General Feature Model)
+ signature : CharacterString
+triggeredByValuesOf
0..*
+affectsValuesOf
0..*
+observesValuesOf
0..*
<<MetaClass>>
GF_PropertyType
(from General Feature Model)
+ memberName : LocalName
+ definition : CharacterString

Figure B.4 — Derivation of FC_FeatureOperation from the GF_Operation metaclass

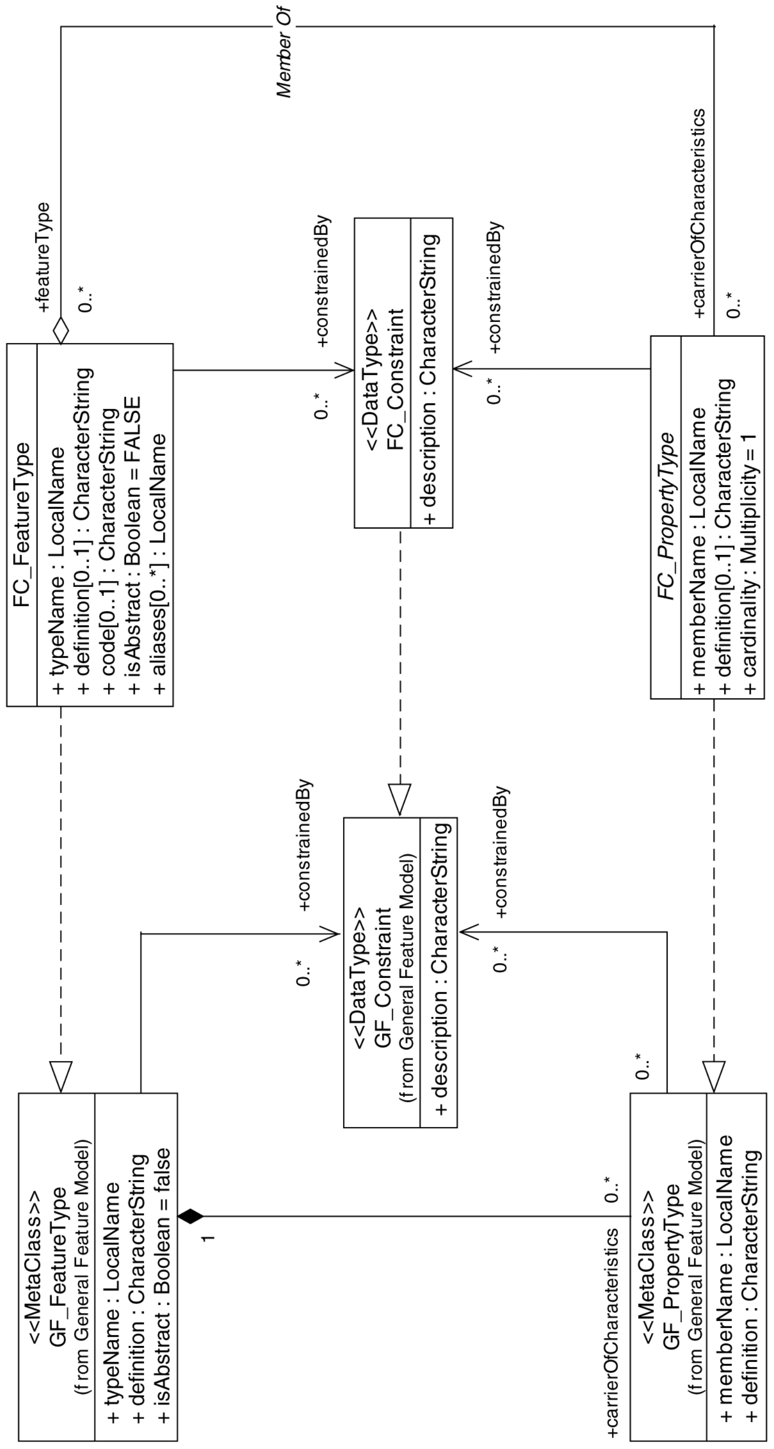

Member Of
+featureType
0..*
FC_FeatureType
+ typeName : LocalName
+ definition[0..1] : CharacterString
+ code[0..1] : CharacterString
+ isAbstract : Boolean = FALSE
+ aliases[0..*] : LocalName
+constrainedBy
0..*
<<DataType>>
FC_Constraint
+ description : CharacterString
+constrainedBy
0..*
+carrierOfCharacteristics
0..*
FC_PropertyType
+ memberName : LocalName
+ definition[0..1] : CharacterString
+ cardinality : Multiplicity = 1
+constrainedBy
0..*
<<DataType>>
GF_Constraint
(from General Feature Model)
+ description : CharacterString
+constrainedBy
0..*
0..*
<<MetaClass>>
GF_FeatureType
(from General Feature Model)
+ typeName : LocalName
+ definition : CharacterString
+ isAbstract : Boolean = false
1
0..*
+carrierOfCharacteristics
<<MetaClass>>
GF_PropertyType
(from General Feature Model)
+ memberName : LocalName
+ definition : CharacterString

Figure B.5 — Derivation of FC_Constraint from the GF_Constraint metaclass

Annex C
(informative)

Feature cataloguing examples

C.1 Introduction

This annex presents examples of the functionality of the feature catalogue template 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.

Many of the examples are based on the Digital Geographic Information Exchange Standard (DIGEST) Edition 2.1[5]. DIGEST Part 4 specifies the Feature and Attribute Coding Catalogue (FACC). The FACC specifies features, attributes, and attribute domain values. It provides the means for denoting real-world entities and concepts, including those which are not necessarily visible or have tangible physical form (e.g. airspace). It does not specify the delineation or geometry of features and is intended to be independent from the level of resolution (scale), representation, or portrayal.

Feature operations have been formally defined using the Gofer functional programming language[7].

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[6]. Unused optional elements are omitted from the examples.

C.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 C.1 illustrates a populated FC_FeatureCatalogue (Table B.1) based on the FACC, edition 2.1. Only one of the contained feature types is illustrated; in addition, there is one feature association (see C.4) in this example feature catalogue.

The FACC does not include definitions for all feature attribute list values; in some cases the IHO Hydrographic Dictionary may be used as a source for these missing definitions. Table C.2 illustrates a populated FC_DefinitionSource (Table B.13) for the example feature catalogue.

Table C.1 — Example feature catalogue

Class FC_FeatureCatalogue (identity = 1)				
Attribute FC_FeatureCatalogue.name	“Digital Geographic Information Exchange Standard (DIGEST) Feature and Attribute Coding Catalogue (FACC)”			
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			
Attribute FC_FeatureCatalogue.producer	**Class ISO 19115 Metadata:: CI_ResponsibleParty**			
	individualName	“John Q. Public”		
	organisationName	“US National Geospatial-Intelligence Agency (NGA)”		
	contactInfo	**Class ISO 19115 Metadata:: CI_Contact**		
		phone	**Class ISO 19115 Metadata:: CI_Telephone**	
			voice	“1 703 xxx xxxx”
			facsimile	"1 703 xxx xxxx”
		address	**Class ISO 19115 Metadata:: CI_Address**	
			deliveryPoint	“12310 Sunrise Valley Drive”
			city	“Reston”
			administrativeArea	“Virginia”
			postalCode	“20191-3449”
			country	“USA”
			electronicMailAddress	“PublicJQ@nga.mil”
	role	007 (pointOfContact)		
Attribute FC_FeatureCatalogue.functionalLanguage	“Gofer”			
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 C.2 — Example definition source

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

C.3 Feature types and feature attributes

C.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 C.3 illustrates a 'mine' feature type; it includes a definition, code, and is not an abstract feature type. It also has an alias.

Table C.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)
	FC_Binding (identity = 6)

The example feature catalogue includes the feature attribute ‘depth’. Table C.4 illustrates its representation using FC_FeatureAttribute (Table B.8); it is real-valued and measured in the unit ‘metre’.

Table C.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)
	FC_Binding (identity = 6)
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 C.5 illustrates how this information is represented using a FC_Constraint (Table B.7).

Table C.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.”

Although independently specified, the ‘depth’ feature attribute has restricted semantics when bound to the ‘mine’ feature type. Table C.6 illustrates the representation of its restricted semantics using FC_Binding (Table B.6).

Table C.6 — Example feature attribute binding

Class FC_Binding (identity = 6)	
Attribute FC_Binding.description	“The starting point for the measurement of distance is the primary mine entrance at surface level.”

C.3.2 The ‘classification’ of a ‘berthing structure’

The example feature catalogue also includes a ‘berthing structure’ feature type (illustrated in Table C.7) and a ‘pier/wharf/quay classification’ feature attribute (illustrated in Table C.8). 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.11) and are illustrated in Tables C.9, C.10, C.12, C.14, C.16, C.17 and C.18.

Table C.7 — 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)
	FC_Binding (identity = 19)

Table C.8 — 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)
	FC_Binding (identity = 19)
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 C.10, C.12 and C.14), the definitions are located outside of the feature catalogue and are found in the source document previously specified in Table C.2. Tables C.11, C.13 and C.15 illustrate the representation, using FC_DefinitionReference (Table B.14) of the additional citation information necessary to locate each listed value definition in the source document.

Table C.9 — 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 C.10 — 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 C.11 — 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 C.12 — 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 C.13 — 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 C.14 — 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 C.15 — 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 C.16 — 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 C.17 — 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 C.18 — 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’.”

The optional FC_Binding.description illustrated in Table C.19 is left empty to emphasize that the ‘pier/wharf/quay classification’ feature attribute does not have restricted semantics when bound to the ‘berthing structure’ feature type.

Table C.19 — Example empty feature attribute binding

Class FC_Binding (identity = 19)	
Attribute FC_Binding.description	“”

C.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 C.20 and C.21, respectively.

Table C.20 — 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_FeatureAssociationRole** (identity = 23)
	FC_Binding *(unspecified in this example)*

Table C.21 — Example feature type ‘bridge’

Class FC_FeatureType (identity = 21)	
Attribute FC_FeatureType.typeName	“Bridge” [a]
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)
	FC_Binding *(unspecified in this example)*
[a] More precisely, FACC edition 2.1 names this feature type as “Bridge/Overpass/Viaduct”.	

Table C.22 illustrates the representation of the ‘stacked on’ feature association using FC_FeatureAssociation (Table B.12).

Table C.22 — 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.”
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.9), illustrated in Tables C.23 and C.24. 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 C.23 — 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.”
Role FC_PropertyType.featureType	**FC_FeatureType** (identity = 20)
Attribute FC_AssociationRole.cardinality	“0..*”
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.10).

Table C.24 — Example association role ‘under’

Class FC_ AssociationRole (identity = 24)	
Attribute FC_PropertyType.memberName	“Under”
Attribute FC_PropertyType.definition	“Roads which cross this bridge.”
Role FC_PropertyType.featureType	**FC_FeatureType** (identity = 21)
Attribute FC_AssociationRole.cardinality	“0..1”
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)

C.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 C.25 and C.26, respectively.

Table C.25 — 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 C.26 — 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, feature associations, and feature operations that apply to the ‘building’ feature type in the example feature catalogue also apply to the ‘lighthouse’ feature type. Table C.27 illustrates the representation of the ‘is a’ inheritance relation using FC_InheritanceRelation (Table B.3).

Table C.27 — 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)

C.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 following tables.

Tables C.28, C.29, and C.30 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 C.28 — 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)
	FC_Binding *(unspecified in this example)*
Role FC_FeatureType.carrierOfCharacteristics	**FC_FeatureAttribute** (identity = 30)
	FC_Binding *(unspecified in this example)*
Role FC_FeatureType.carrierOfCharacteristics	**FC_FeatureOperation** (identity = 36)
	FC_Binding *(unspecified in this example)*

Table C.29 — 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)
	FC_Binding *(unspecified in this example)*
Attribute FC_FeatureAttribute.code	“damHeight”
Attribute FC_FeatureAttribute.valueMeasurementUnit	“Metre”
Attribute FC_FeatureAttribute.valueType	Positive real

Table C.30 — 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)
	FC_Binding *(unspecified in this example)*
Attribute FC_FeatureAttribute.code	"maxHeight"
Attribute FC_FeatureAttribute.valueMeasurementUnit	"Metre"
Attribute FC_FeatureAttribute.valueType	Positive real

Tables C.31 and C.32 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 C.31 — 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"
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)
	FC_Binding *s*

Table C.32 — 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)
	FC_Binding *(unspecified in this example)*
Attribute FC_FeatureAttribute.code	"reservoirDepth"
Attribute FC_FeatureAttribute.valueMeasurementUnit	"Metre"
Attribute FC_FeatureAttribute.valueType	Positive real

Tables C.33, C.34 and C.35 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 C.33 — 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)
	FC_Binding *(unspecified in this example)*
Role FC_FeatureType.carrierOfCharacteristics	**FC_FeatureAttribute** (identity = 35)
	FC_Binding *(unspecified in this example)*

Table C.34 — 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)
	FC_Binding *(unspecified in this example)*
Attribute FC_FeatureAttribute.code	"streamDepth"
Attribute FC_FeatureAttribute.valueMeasurementUnit	"Metre"
Attribute FC_FeatureAttribute.valueType	Positive real

Table C.35 — 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)
	FC_Binding *(unspecified in this example)*
Attribute FC_FeatureAttribute.code	"streamFlow"
Attribute FC_FeatureAttribute.valueMeasurementUnit	"Cubic metres per second"
Attribute FC_FeatureAttribute.valueType	Positive integer

Table C.28, illustrating the specification of the ‘dam’ feature type, identified 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.5), is illustrated in Table C.36.

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 detailed formal definition is illustrated in part; the remainder is given in D.6.2.4.

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 C.36 — 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”
Attribute FC_FeatureOperation.formalDefinition	`"damRaise(ConstructDam(d),h) = error ‘Cannot raise height of a dam under construction’ damRaise(Operate(d,i,j),h) | (h > i) && (h < maxHeight(d)) = Operate(d,h,j) | otherwise = error ‘Illegal new height for dam’"` *(remainder as specified in D.6.2.4)*
Role FC_FeatureOperation.observesValuesOf	**FC_FeatureAttribute** (identity = 30)
Role FC_FeatureOperation.affectsValuesOf	**FC_FeatureAttribute** (identity = 29)
Role FC_FeatureOperation.affectsValuesOf	**FC_BoundFeatureAttribute** (identity = 37)
Role FC_FeatureOperation.affectsValuesOf	**FC_BoundFeatureAttribute** (identity = 38)
Role FC_FeatureOperation.affectsValuesOf	**FC_BoundFeatureAttribute** (identity = 39)

While the maxHeight and damHeight feature attributes are properties of the same Dam feature type as the ‘raise dam’ feature operation, the three other feature attributes must be specified in terms of the feature types of which they are properties. Their specifications, represented using FC_BoundFeatureAttribute (Table B.15), are illustrated in Tables C.37, C.38 and C.39.

Table C.37 — Example bound feature attribute ‘watercourse:streamDepth’

Class FC_BoundFeatureAttribute (identity = 37)	
Role FC_BoundFeatureAttribute.featureType	**FC_FeatureType** (identity = 33)
Role FC_BoundFeatureAttribute.attribute	**FC_FeatureAttribute** (identity = 34)

Table C.38 — Example bound feature attribute ‘watercourse:streamFlow’

Class FC_BoundFeatureAttribute (identity = 38)	
Role FC_BoundFeatureAttribute.featureType	**FC_FeatureType** (identity = 33)
Role FC_BoundFeatureAttribute.attribute	**FC_FeatureAttribute** (identity = 35)

Table C.39 — Example bound feature attribute ‘reservoir:reservoirDepth’

Class FC_BoundFeatureAttribute (identity = 39)	
Role FC_BoundFeatureAttribute.featureType	**FC_FeatureType** (identity = 31)
Role FC_BoundFeatureAttribute.attribute	**FC_FeatureAttribute** (identity = 32)

Annex D
(informative)

Feature cataloguing concepts

D.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 into categories 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 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 following examples 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 International Standard, feature operations are presented as a fourth major aspect of feature classification, 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 classification, this International Standard seeks to support the anticipated transition from current practice to a future, more rigorously functional, approach; see Reference [8].

D.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.

D.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.

D.4 Feature relationships

D.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.

D.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 both to 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’.

D.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.

D.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 C.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.

D.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. Functional specification of the feature types provides an unambiguous method for evaluating equivalence.

D.6 Examples of feature operations

D.6.1 ‘Road’ with operation ‘road passable’

D.6.1.1 Introduction

The following example uses an unspecified formal description language to specify the object type ‘road vehicle’ and the feature type ‘road’ algebraically. By knowing the characteristics of the ‘road vehicle’, it is possible to determine whether or not the ‘road’ is passable.

D.6.1.2 'Road vehicle' specification

The ‘road vehicle’ object supports five operations; two determine the type of traction mechanism of the vehicle and three determine its measured characteristics. They are specified as follows:

```
SYNTAX OF OPERATIONS:
    VEHICLETRACKED: V → b
    VEHICLEWHEELED: V → b
    VEHICLELOADCLASS: V → i
    VEHICLEWIDTH: V → i
    VEHICLEHEIGHT: V → i
SEMANTICS OF OPERATIONS:
    pre- VEHICLETRACKED(v) ::= true
    post- VEHICLETRACKED(v;b) ::=    if vehicle_tracked then b=true
                                        else b=false
    pre- VEHICLEWHEELED(v) ::= true
    post- VEHICLEWHEELED(v;b) ::=    VEHICLETRACKED(v;b);
                                     b = NOT b
    pre- VEHICLELOADCLASS(v) ::= true
    post- VEHICLELOADCLASS(v;i) ::= i=load_class_of_vehicle
    pre- VEHICLEWIDTH(v) ::= true
    post- VEHICLEWIDTH(v;i) ::= i=width_of_vehicle
    pre- VEHICLEHEIGHT(v) ::= true
    post- VEHICLEHEIGTHv;i) ::= i=height_of_vehicle

    WHERE ...
            b     The set consisting of Boolean values true and false.
            i     The set of integers.
            V     The set of RoadVehicles.
```

D.6.1.3 'Road' specification

The feature type ‘road’ is defined as ‘An open way maintained for vehicular use.’ It has three feature attributes, specified as follows.

a) ‘Existence Category’ defined as ‘The state or condition of the feature.’ and whose domain is a set of listed values as follows:

   1) ‘Unknown’ with code 0,

   2) ‘Under Construction’ with code 5, and

   3) ‘Operational’ with code 28.

b) ‘Minimum Travelled Way Width’ defined as ‘Minimum width of the travelled way, excluding hard pavements and shoulders.’ Its value is measured in decimetres and is from the domain of the positive integers.

c) ‘Weather Type Category’ defined as ‘Weather conditions under which a feature is usable.’ and whose domain is a set of listed values as follows:

   1) ‘Unknown’ with code 0,

   2) ‘All weather’ with code 1,

   3) ‘Fair/Dry Weather with code 2, and

   4) ‘Winter Only’ with code 3.

The feature type ‘road’ has a single feature operation, **ROADPASSABLE**, defined as ‘Indicates whether a road is passable by a given road vehicle.’ The operation depends on all three feature attributes for ‘road’, and is specified as follows:

```
SYNTAX OF OPERATION:
     ROADPASSABLE: R, V, SY, WT → b
SEMANTICS OF OPERATION:
     pre- ROADPASSABLE(r,v,sy,wt) ::= true
     post- ROADPASSABLE(r,v,sy,wt;b) ::=
               if ((GETROADEXISTENCECATEGORY(r) = 28) AND
                    ((SY = winter AND GETROADWEATHERTYPECATEGORY(r) = 3) OR
                         (GETROADWEATHERTYPECATEGORY (r) = 1) OR
                         (GETROADWEATHERTYPECATEGORY (r) = 2 AND WT = Fair/Dry)) AND
                    (VEHICLEWIDTH (v) <= GETROADMINIMUMTRAVELEDWAYWIDTH(r))
                    b = true
               else
                    b = false
     WHERE ...
```

b	The set consisting of Boolean values *true* and *false*.
R	The set of road instances.
V	The set of RoadVehicles.
SY	The set of seasons, consisting of *spring, summer, autumn* and *winter*.
WT	The set of weather conditions consisting of *Fair/Dry, Rain*, and *Snow/Ice*.

D.6.2 'Dam' with multiple operations

D.6.2.1 Introduction

The following example uses the Gofer functional programming language[7] Feature types are defined in Gofer as 'abstract data types'. 'Operations' in Gofer correspond to feature operations. 'Axioms' in Gofer specify the results of each operation in terms of the attribute values of the types. 'Observer functions' return the current value of an existing attribute or one derived from others by mathematical manipulation. 'Constructor functions' affect the values of one or more attributes of the subject feature or a related feature. Taken together, the 'abstract data types' and the 'operations' of a feature type constitute the 'signature' of that feature type. Such 'signatures' provide a formal basis for assessing the interoperability of feature definitions between applications and datasets.

The feature type ‘dam’ is specified algebraically in terms of its feature operations[8]. In this example, the variables used are defined as follows:

```
d is a Dam
w is a Watercourse
r is a Reservoir

k is the maxHeight of a Dam
i is the damHeight of a Dam
h is the newHeight of a Dam
j is the dischargeFlow of a Dam

u is the streamDepth of a Watercourse
v is the streamFlow of a Watercourse

m is the maxDepth of a Reservoir
```

D.6.2.2 Before constructing a ‘dam’

The definition for a ‘dam’ is 'a barrier constructed across a watercourse to control the flow or raise the level of water in a reservoir'. Given this natural-language definition, a series of Gofer operations can be specified as follows. Initially, while the ‘dam’ is only proposed, there is a single feature type, the ‘watercourse’, with feature attributes of depth and flow.

NOTE To simplify the metrics of the problem, it is assumed that the gradient of the ‘watercourse’ and its cross-sectional area, taken together, result in a constant value for the attribute ‘stream depth’ over the segments where the ‘dam’ will be built. This attribute can then be used to operationalize the variable ‘water level’ in the definition of ‘dam’.

The situation can be represented algebraically in Gofer (Specification 1), with an abstract data type `Watercourse`, the only operations of which are to observe its depth and flow (operations *`streamDepth`* and *`streamFlow`*).

```
Specification 1:

--abstract data type

data Watercourse = Stream(Int,Int)

--operations (observer functions)

streamDepth :: Watercourse -> Int
streamFlow :: Watercourse -> Int

--axioms

streamDepth(Stream(u,v)) = u
streamFlow(Stream(u,v)) = v
```

D.6.2.3 Constructing a ‘dam’

The first phrase in the definition of ‘dam’ is 'a barrier constructed across…' There are thus two conditions of ‘dam’, the one under construction and the one ready to operate. As a result of constructing the ‘dam’, several changes occur. First, there is a new ‘dam’ where there was none before. The ‘watercourse’ is now split into two parts that will behave differently, an upstream part and a downstream part. Second, a portion of the valley through which the ‘watercourse’ flows is about to be flooded. This area will become a new ‘reservoir’.

The algebraic specification involves the new Gofer abstract data types `Dam` and `Reservoir`, in addition to `Watercourse`. The operations include constructing a ‘dam’ and creating a new ‘reservoir’. It is the case that the new ‘dam’ is (as yet) open, its height is zero, and its discharge is zero. The new ‘reservoir’ is empty and its ‘depth’ is also zero. The maximum ‘height’ of the ‘dam’ is set at the time of construction (Specification 2).

```
Specification 2:

--abstract data types

data Dam = ConstructDam(Int) | Operate(Dam,Int,Int)
data Watercourse = Upstream(Int,Int)| Downstream(Watercourse,Int,Int)
data Reservoir = NewReservoir(Int) | Fill(Reservoir,Int)

--operations (observer functions)

maxHeight :: Dam -> Int
damHeight :: Dam -> Int
damOpen :: Dam -> Bool
streamDepth :: Watercourse -> Int
streamFlow :: Watercourse -> Int
```

```
reservoirDepth :: Reservoir -> Int
reservoirEmpty :: Reservoir -> Bool

--axioms

maxHeight (ConstructDam(k)) = k
damHeight (ConstructDam(k)) = 0
damOpen (ConstructDam(k)) = True
streamDepth (Upstream(u,v)) = u
streamDepth (Downstream(w,u,v)) = streamDepth(Upstream(u,v))
streamFlow (Upstream(u,v)) = v
streamFlow (Downstream(w,u,v)) = streamFlow(Upstream(u,v))
reservoirDepth (NewReservoir(m)) = 0
reservoirEmpty (NewReservoir(m)) = True
```

D.6.2.4 Raising a ‘dam’

The next operation is to raise the height of the ‘dam’ and begin to fill the ‘reservoir’. The operation of raising the ‘dam’ results in the ‘dam’ being closed and stopping the flow of water downstream. A condition is added to the feature operation of raising the ‘dam’ that prevents the ‘dam’ from being raised higher than its maximum possible ‘height’ (or lower than zero). The mutual interdependence of the feature types is reflected in more complex operations and equations (Specification 3).

```
Specification 3:

--abstract data types

data Dam = ConstructDam(Int) | Operate(Dam,Int,Int)
data Watercourse= Upstream(Int,Int)|Downstream(Watercourse,Int,Int)
data Reservoir = NewReservoir(Int) | Fill(Reservoir,Int)

--operations (damRaise is a constructor function)

damRaise :: (Dam,Int) -> Dam
maxHeight :: Dam -> Int
damHeight :: Dam -> Int
discharge :: Dam -> Int
damOpen :: (Dam,Watercourse,Reservoir) -> Bool
damClose :: (Dam,Watercourse,Reservoir) -> Bool
streamDepth :: (Dam,Watercourse,Reservoir) -> Int
streamFlow :: (Dam,Watercourse,Reservoir) -> Int
reservoirDepth :: (Dam,Watercourse,Reservoir) -> Int
reservoirEmpty :: (Dam,Watercourse,Reservoir) -> Bool

--axioms

maxHeight (ConstructDam(k)) = k
maxHeight (Operate(d,i,j)) = maxHeight (d)
damHeight (ConstructDam(k)) = 0
damHeight (Operate(d,i,j)) = i
discharge (ConstructDam(k)) = 0
discharge (Operate(d,i,j)) = j

streamDepth (d,Upstream(u,v),r) = u
streamDepth (d,Downstream(w,u,v),r)
     | damClose(d,w,r) == True = 0
     | damOpen(d,w,r) == True = streamDepth(d,w,r)
                         + reservoirDepth(d,w,r) - damHeight(d)
     | otherwise = streamDepth(d,Upstream(u,v),r)
streamFlow(d,Upstream(u,v),r) = v
streamFlow(d,Downstream(w,u,v),r)
    | damClose(d,w,r) == True = 0
    | damOpen(d,w,r) == True = streamFlow(d,w,r) + discharge(d)
    | otherwise = streamFlow(d,Upstream(u,v),r)

damOpen(ConstructDam(k),w,r) = True
damOpen(d,w,r) = damHeight(d) < reservoirDepth(d,w,r)

reservoirEmpty(ConstructDam(k),w,r) = True
reservoirDepth(ConstructDam(k),w,r) = 0

damRaise(ConstructDam(d),h) = error
    "Cannot raise height of a dam under construction"
damRaise(Operate(d,i,j),h)
    | (h > i) && (h < maxHeight(d)) = Operate(d,h,j)
    | otherwise = error "Illegal new height for dam"
damClose(d,w,r) = damHeight(d) > reservoirDepth(d,w,r)
```

D.6.2.5 'Reservoir' is full

The dam-closed state of affairs continues until the 'reservoir' fills to the level of the 'dam'. An observer function is added to indicate when the 'reservoir' is full (Specification 4).

Specification 4:

```
--observer function reservoirFull

reservoirFull :: (Dam,Watercourse,Reservoir) -> Bool
reservoirFull(d,w,r) = reservoirDepth(d,w,r) == damHeight(d)
```

When this occurs, the 'dam' is neither open (discharging extra water into the downstream segment of the 'watercourse') nor closed (preventing any water from flowing downstream). The downstream segment of the 'watercourse' returns to its normal upstream depth.

D.6.2.6 Lowering a 'dam' (discharging)

A feature operation that has different consequences is lowering the 'dam'. When this happens, there is a period of time when the 'height' of the 'reservoir' exceeds that of the 'dam'. The downstream flow is increased relative to the upstream flow by the additional amount of discharge from the 'dam'. The dam-lowering operation and its effects require additional operations and equations (Specification 5).

Specification 5:

```
discharge :: (Dam,Watercourse,Reservoir) -> Int
discharge (ConstructDam(k),w,r) = 0
discharge (Operate(d,i,j),w,r)
    | damOpen(d,w,r) = ((reservoirDepth(d,w,r) - damHeight(d))
                        / streamDepth(d,w,r)) * streamFlow(d,w,r)
    | otherwise = 0

--constructor function damLower

damLower :: (Dam,Int) -> Dam
damLower (ConstructDam(k),h) = error "Cannot lower a new dam"
damLower (Operate(d,i,j),h)
    | (h < i) && (h >= 0) = Operate(d,h,j)
    | otherwise = error "Illegal new height for dam"
```

Eventually, the level of the ‘reservoir’ falls to the ‘height’ of the ‘dam’ and the system is again in equilibrium (as observed by the operations `damOpen`, `damClose`, and `reservoirFull`). This condition was specified in D.6.2.5; the only difference is that there are new values for the ‘height’ of the ‘dam’ and the ‘depth’ of the ‘reservoir’.

Bibliography

[1] ISO 19101:2002, *Geographic information — Reference model*

[2] ISO 19107:2003, *Geographic information — Spatial schema*

[3] ISO 19108:2002, *Geographic information — Temporal schema*

[4] ISO 19117:—2), *Geographic information — Portrayal*

[5] Digital Geographic 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 13 December 2003]. Available from World Wide Web: <http://www.digest.org/Navigate2.htm>.

[6] Internet Engineering Task Force (IETF), The Internet Society. *Uniform Resource Identifiers (URI): Generic Syntax.* RFC 2396. Reston (Virginia): IETF, 1998. [cited 13 December 2003]. Available from World Wide Web: <http://www.faqs.org/rfcs/rfc2396.html>.

[7] JONES, M. P., *Gofer Functional Programming Environment*, 1993. [cited 13 December 2003]. Available from World Wide Web: <http://www.cse.ogi.edu/~mpj/goferarc/index.html>.

[8] RUGG, R. D., EGENHOFER, M. J. and KUHN, W., *Formalizing Behavior of Geographic Features, Geographical Systems*, Vol. 4, No. 2, pp. 159-179, 1997. [cited 13 December 2003]. Available from World Wide Web: <http://www.spatial.maine.edu/~max/RJ24.html>.

2) Under preparation.

ICS 35.240.70
Price based on 55 pages