INTERNATIONAL STANDARD

ISO 19109

First edition
2005-06-15

Geographic information — Rules for application schema

Information géographique — Règles de schéma d'application

Reference number
ISO 19109:2005(E)

Contents

Page

Foreword

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

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

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

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 19109 was prepared by Technical Committee ISO/TC 211, *Geographic information/Geomatics*.

Introduction

Any description of reality is always an abstraction, always partial, and always just one of many possible "views", depending on the application field.

The widespread application of computers and geographic information systems (GIS) has led to an increased use of geographic data within multiple disciplines. With current technology as an enabler, society's reliance on such data is growing. Geographic datasets are increasingly being shared and exchanged. They are also used for purposes other than those for which they were produced.

To ensure that data will be understood by both computer systems and users, the data structures for data access and exchange must be fully documented. The interfaces between systems, therefore, need to be defined with respect to data and operations, using the methods standardized in this International Standard. For the construction of internal software and data storage within proprietary systems, any method may be used that enables the standardized interfaces to be supported.

An application schema provides the formal description of the data structure and content required by one or more applications. An application schema contains the descriptions of both geographic data and other related data. A fundamental concept of geographic data is the feature.

Geographic information — Rules for application schema

1 Scope

This International Standard defines rules for creating and documenting application schemas, including principles for the definition of features.

The scope of this International Standard includes the following:

— conceptual modelling of features and their properties from a universe of discourse;

— definition of application schemas;

— use of the conceptual schema language for application schemas;

— transition from the concepts in the conceptual model to the data types in the application schema;

— integration of standardized schemas from other ISO geographic information standards with the application schema.

The following are outside the scope:

— choice of one particular conceptual schema language for application schemas;

— definition of any particular application schema;

— representation of feature types and their properties in a feature catalogue;

— representation of metadata;

— rules for mapping one application schema to another;

— implementation of the application schema in a computer environment;

— computer system and application software design;

— programming.

2 Conformance

Any application schema claiming conformance to this International Standard shall pass all of the requirements described in the abstract test suites in Annex A.

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 19107:2003, *Geographic information — Spatial schema*

ISO 19108:2002, *Geographic information — Temporal schema*

ISO 19112:2003, *Geographic information — Spatial referencing by geographic identifiers*

ISO 19113:2002, *Geographic information — Quality principles*

ISO 19115:2003, *Geographic information — Metadata*

ISO/IEC 19501, *Information technology — Open Distributed Processing — Unified Modeling Language (UML) Version 1.4.2*

4 Terms and definitions

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

4.1
application
manipulation and processing of data in support of user requirements

[ISO 19101]

4.2
application schema
conceptual schema for data required by one or more applications

[ISO 19101]

4.3
complex feature
feature composed of other features

4.4
conceptual model
model that defines concepts of a universe of discourse

[ISO 19101]

4.5
conceptual schema
formal description of a conceptual model

[ISO 19101]

1) To be published.

4.6
dataset
identifiable collection of data

[ISO 19115]

4.7
domain
well-defined set

[ISO 19107]

NOTE Well-defined means that the definition is both necessary and sufficient, as everything that satisfies the definition is in the set and everything that does not satisfy the definition is necessarily outside the set.

4.8
feature
abstraction of real-world phenomena

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

[ISO 19101]

4.9
feature association
relationship that links instances of one feature type with instances of the same or a different feature type

[ISO 19110]

4.10
feature attribute
characteristic of a feature

NOTE 1 A feature attribute may occur as a type or an instance. Feature attribute type or feature attribute instance is used when only one is meant.

NOTE 2 A feature attribute type has a name, a data type and a domain associated to it. A feature attribute instance has an attribute value taken from the domain of the feature attribute type.

[adapted from ISO 19101]

4.11
feature operation
operation that every instance of a feature type may perform

EXAMPLE 1 A feature operation upon the feature type “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”.

EXAMPLE 2 A feature operation by the feature type “dam” might be to block vessels from navigating along a watercourse.

[adapted from ISO 19110]

4.12
geographic data
data with implicit or explicit reference to a location relative to the earth

NOTE Geographic information is also used as a term for information concerning phenomena implicitly or explicitly associated with a location relative to the earth.

4.13
metadata
data about data

[ISO 19115]

4.14
model
abstraction of some aspects of reality

4.15
portrayal
presentation of information to humans

[ISO 19117]

4.16
quality
totality of characteristics of a product that bear on its ability to satisfy stated and implied needs

[ISO 19101]

4.17
universe of discourse
view of the real or hypothetical world that includes everything of interest

[ISO 19101]

5 Presentation and abbreviations

5.1 Presentation

This International Standard describes how to create an application schema that integrates conceptual schemas defined in the ISO 19100 series of International Standards for geographic information. In addition to stating the rules for creating application schemas, this International Standard provides guidance through examples. This International Standard adopts the following conventions for presentation purposes:

a) Rules:

All rules are normative, and are described as follows.

Rules:

1) <Rule 1>

2) <Rule 2>

b) Tables:

Tables that are not referenced from the rules are informative.

c) Conceptual schemas:

Conceptual schemas in the normative part of this International Standard are presented in the Unified Modelling Language (UML) in conformance with ISO/TS 19103. UML diagrams are presented in compliance with ISO/IEC 19501.

5.2 Abbreviations

CSL	Conceptual schema language
GFM	General feature model
OCL	Object constraint language
UML	Unified modelling language

6 Context

6.1 Purpose of an application schema

An application schema is a conceptual schema for data required by one or more applications. An application schema defines

— content and structure of data; and

— specifications of operations for manipulating and processing data by an application.

The purpose of an application schema is twofold:

— to provide a computer-readable data description defining the data structure, which makes it possible to apply automated mechanisms for data management; and

— to achieve a common and correct understanding of the data, by documenting the data content of the particular application field, thereby making it possible to unambiguously retrieve information from the data.

6.2 Rules for application schema

This International Standard does not standardize application schemas; it only defines rules for creating application schemas in a consistent manner (including the consistent definition of features) to facilitate the acquiring, processing, analysing, accessing, presenting and transferring of geographic data between different users, systems and locations. The rules in this International Standard are, in the case of data transfer or interchange, used by suppliers and users of geographic data to

— build a transfer application schema for data interchange;

— interpret the semantics of the transferred dataset with respect to user's local data and content and structure of data; and

— determine the necessary transformations between the two datasets.

The rules in this International Standard will assist the users of applications with similar data requirements in creating a common application schema for the interface between their systems and data. This includes an agreement about the elements from the universe of discourse. This is described in more detail in 6.3.

The mapping from one application schema to another application schema may be difficult and even impossible if the two schemas are too divergent. The mapping is facilitated if the application schema used within a system is designed considering also the requirements for the data interchange. The rules can also be used for building an application schema used within a system, although such application schemas are not within the scope of this International Standard.

The creation of an application schema is a process. The content of an application schema has to be settled according to the view of reality in the universe of discourse. This is modelled in terms of types of features and their properties. Clause 7 contains principles for consistently defining features.

The application schema defines the structure and content of data. It is expressed in a conceptual schema language (CSL). Clause 7 also includes a model expressed in UML that defines the concepts required to describe types of features. Feature type definition may be documented in feature catalogues. Such definitions may be used in an applications schema. Other standards in the ISO 19100 series define reusable modules of conceptual schemas that may be integrated in an application schema. Clause 8 gives the main rules for integrating these predefined modules into a conceptual schema in UML.

NOTE ISO 19118 defines how a dataset defined by an application schema in UML is encoded.

6.3 Application schema supporting data interchange

6.3.1 Introduction

Data interchange between information systems may take place in two ways.

— In the traditional data transfer model, the data supplier creates a dataset that is transferred to the user. The structure and the content of data are described in the application schema for the dataset. The dataset is sent in a transfer format.

— In the interoperability model, the user application communicates with the supplier application through a common communication protocol. In this scenario, the user invokes services that result in data being passed from the service provider to the user application. The application schema describes not only the structure and content of the exchanged data, but also the structure of the interfaces involved in the transaction.

There is a fundamental distinction between a data transfer and a data transaction. In data transfer, a data set is predefined in an application schema. The spatial extent and the rules for inclusion of feature instances are also predefined. The user requests and receives a copy of the dataset (or may receive it automatically through a long-term agreement for distribution of datasets). In a data transaction, a requester first specifies selection criteria, such as spatial extent and feature instance inclusion rules for the data from the producer's data store. Data meeting the selection criteria are then retrieved from the data store and provided to the user.

NOTE Conformance to the rules in this International Standard does not guarantee that data conforming to any given application schema can be transformed in a meaningful way to conform to any other application schema. At best, it allows the user to determine which elements are common to the two schemas and which could be transformed from one schema to another, as well as those that cannot be transformed. Complete interoperability can only occur when user and supplier have identical application schemas.

6.3.2 Data interchange by transfer

Figure 1 shows the traditional data transfer model for data suppliers and data users. The structure and content of the data provided by the supplier and received by the user are described in application schemas. To be able to transfer data, three conditions must be fulfilled.

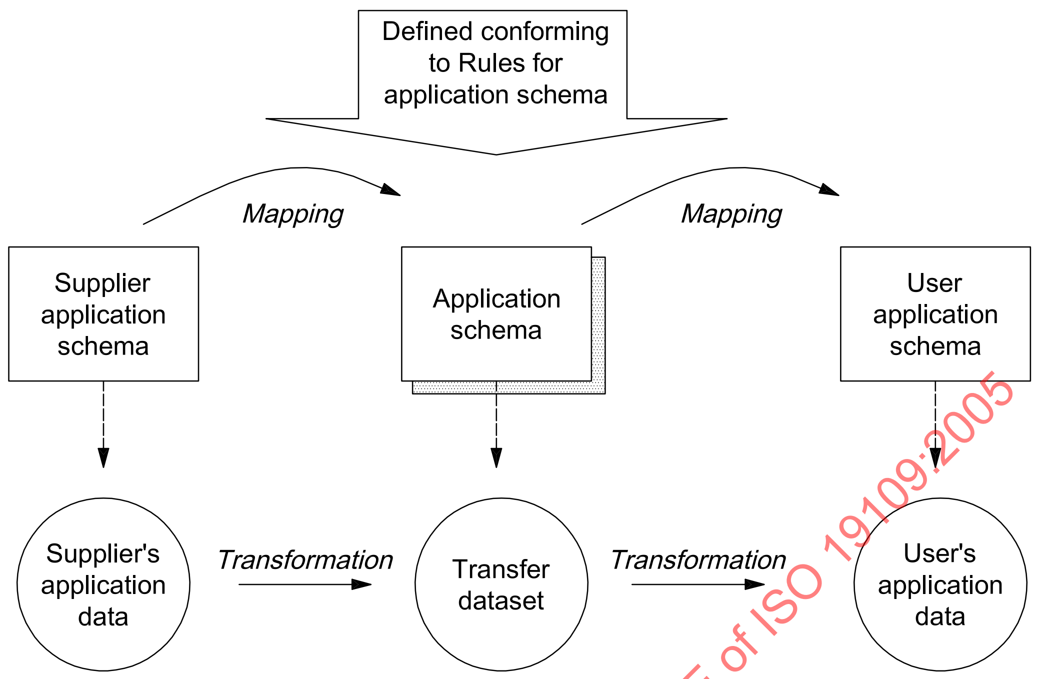


Figure 1 — Data interchange by transfer

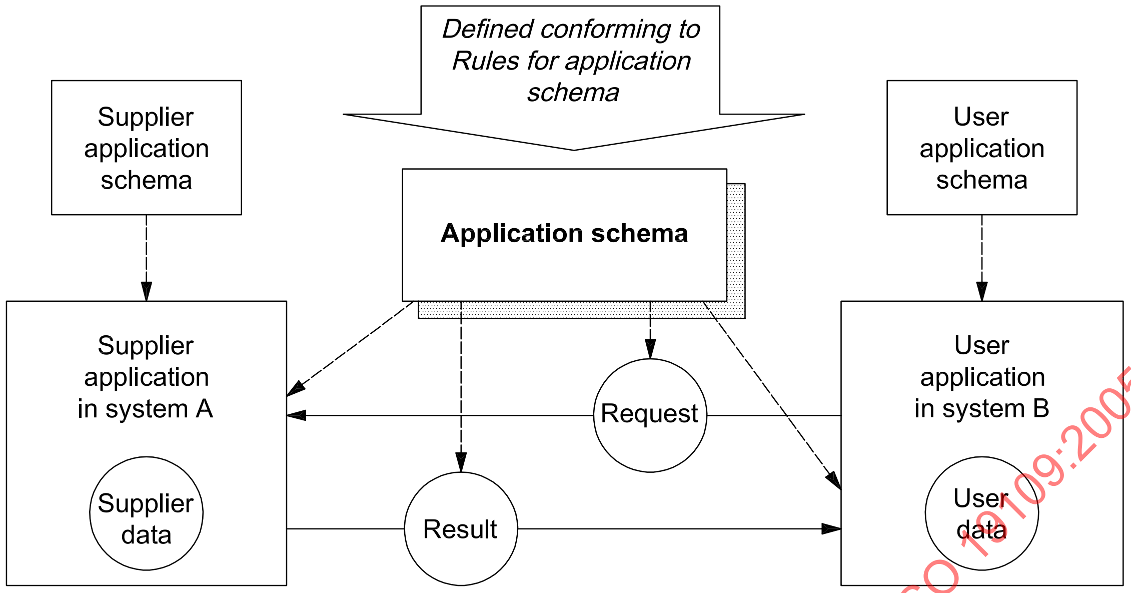
First, the user and the supplier must agree on creating an application schema for the data being exchanged in accordance with this International Standard. In order to facilitate the transfer of data, this application schema shall be developed using the application schemas from the user and the supplier. One mapping will be made from the supplier's application schema to this application schema for the exchanged data, and a second mapping will be made from this application schema to the application schema of the user.

Second, the supplier must be able to transform the application data defined according to the supplier application schema into a transfer dataset defined according to the application schema for the exchanged data.

Third, the user must be able to transform the transfer dataset defined according to its application schema to the application data defined according to the user's application schema.

6.3.3 Data interchange by transactions

Figure 2 shows data interchange through transactions described in the interoperability model. The user application makes a request for data that is received by the supplier application. In response, the supplier application delivers a resulting dataset. Both the request and the resulting dataset are defined according to a common application schema. The supplier application is responsible for transforming the data in system A into the data in the exchanged dataset. After receipt, the user application is responsible for transforming the exchanged data into data in system B. Data interchange by transaction is provided by geospatial services as defined in ISO 19119. In particular, feature access services are defined in Geographic model/information management services.


Defined conforming to Rules for application schema
Supplier application schema
Application schema
User application schema
Supplier application in system A
Supplier data
Request
Result
User application in system B
User data

NOTE The unbroken lines show the flow of data. Broken lines denote the role of the application schema on the data interchange.

Figure 2 — Data interchange by transactions

7 Principles for defining features

7.1 Features

A fundamental unit of geographic information is called a feature. ISO 19110 provides a standard framework for organizing and reporting the classification of features. It also gives a broader discussion on different aspects of geographic features.

This International Standard gives rules for creating application schemas, including the principles for the definition of features. The term feature is used in different contexts defined according to the four-layer architecture. Annex B describes the use of the term feature in the four-layer architecture.

This International Standard distinguishes four aspects of defining features: the definitions or description used to group them into types, the attributes associated with each type, the relationships among the types and the behaviour of the features.

EXAMPLE "Tower Bridge" is the abstraction of a certain real-world bridge in London. The term "bridge" is the abstraction of the collection of all real-world phenomena that is classified into the concept behind the term "bridge". Later in the document, the terms "feature type" and "feature instance" are used to separate the concept of "feature" describing the whole collection from the concept of describing a certain instance occurrence.

Figure 3 describes the most abstract level of defining and structuring geographic data. The classification of real-world phenomena as features depends on their significance to a particular universe of discourse.

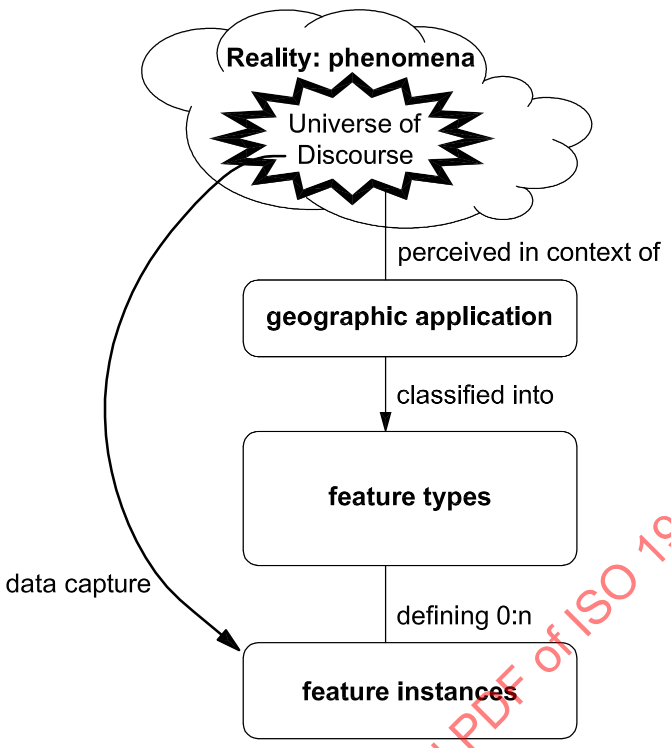


Figure 3 — The process from universe of discourse to data

7.2 Features and the application schema

This International Standard supports the definition of features with respect to their representation in data structures defined by application schemas.

Figure 4 shows the process of structuring data from the universe of discourse to the geographic dataset. The definitions of the feature types and their properties, as perceived in context of an application field, will be derived from the universe of discourse. A feature catalogue documents the feature types.

An application schema defines the logical structure of data and may define operations that can be performed on or with the data. An application schema addresses the logical organization, rather than the physical.

The developer of an application schema may use feature definitions from feature catalogues that already exist. This will reduce the costs of data acquisition, allowing the developer to use existing data, and simplify the process of developing the application schema.

The application schema shall be expressed in a conceptual schema language. Each conceptual schema language has its own terms and concepts. When creating an application schema, the concepts of the General Feature Model (GFM; see 7.3) are mapped to the concepts of the chosen conceptual schema language. For UML, these rules are described in 8.3.

NOTE Annex C gives as an example the rules for mapping the concepts of the GFM to the concepts of ISO 10303-11.

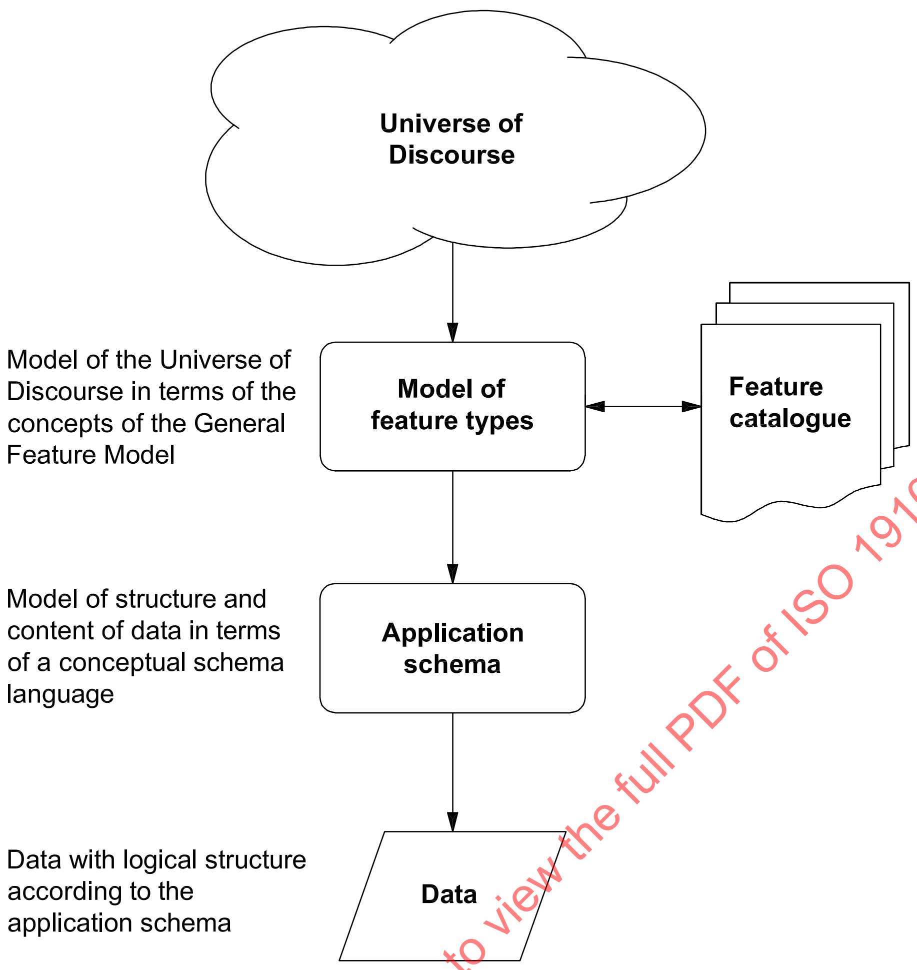

Universe of Discourse
Model of the Universe of Discourse in terms of the concepts of the General Feature Model
Model of feature types
Feature catalogue
Model of structure and content of data in terms of a conceptual schema language
Application schema
Data with logical structure according to the application schema
Data

Figure 4 — From reality to geographic data

7.3 The General Feature Model

7.3.1 Introduction

This subclause identifies and describes the concepts used to define features and how these concepts are related. The description is expressed in a conceptual model, also called the General Feature Model (GFM).

Annex B provides discussions regarding the purpose and design of the GFM.

NOTE The complete kernel of GFM is found in Annex B.3.

The concepts of the GFM are used in the feature catalogue structure described in ISO 19110. ISO 19117 also uses these concepts for specifying the portrayal of geographic information. The concepts of the GFM are used to establish categories of geographic processing services taxonomy in ISO 19119.

Subclauses 7.4 to 7.7 depict how different aspects of the properties of a feature are managed. Subclause 7.4 describes different aspects of attributes and 7.5 describes different aspects of relationships. Subclause 7.6 includes a more detailed description of behaviour of features, and 7.7 describes the concept of constraints.

7.3.2 The purpose of the GFM

The GFM is a model of the concepts required to classify a view of the real world. It is expressed in a CSL, that is in UML class diagrams, but it could be in any CSL. UML has its own model of concepts (metamodel). As both the GFM and the UML metamodel deal with classification, the concepts are very similar. There is one big difference: the concepts in the GFM establish a basis for the classification of features, whereas the UML metamodel provides a basis for classification of any kind.

The things we want to classify we call features; the relations between feature types are feature association types and inheritance. Feature types have properties that are feature attributes, feature operations and feature association roles. All these concepts are expressed as UML metaclasses in the GFM. The GFM is a metamodel of feature types.

Feature types may be documented in feature catalogues. The GFM may serve as the conceptual model of the feature-catalogue structure, but the feature catalogue has additional concepts for documenting feature types. There is a Feature Catalogue Model (FCM) that realizes the GFM concepts and also adds some more concepts (see ISO 19110). Some new concepts are, for example, a list of the feature attributes of each feature type, aliases of the feature type name, and codes for the feature type name. These additional concepts are not in conflict with the concepts from the GFM.

An application schema shall be expressed in a CSL. It shall describe the structure and content of the dataset that represents a universe of discourse. The GFM specifies the requirements for the classification of features, but is not a CSL. This means that we have to use an existing CSL to define the application schema. Within the ISO 19100 series of International Standards, UML is used. As we want to integrate standard schemas from the ISO 19100 series into our application schemas, it is convenient to express the application schema in UML. In this International Standard, we have defined the main rules for mapping the GFM concepts into the UML language. This may be done for other CSLs as well; in Annex C we have exemplified the mapping from GFM to ISO 10303-11 EXPRESS.

The GFM defines the structure for classifying features that we need to keep in mind when we make our application schema in UML. However, the mapping from GFM to UML is a one-way mapping; it is not possible to map backwards. For example, the application schema has UML classes. Some of these classes are GFM feature types and some are the datatypes for feature attributes; it is not possible to keep these things apart. The GFM does not define feature-attribute values to the depth that is needed. That is not necessary to do as the GFM only specifies the structure and content of definition of features.

The conclusion is that the GFM is a metamodel for definition of features that also is used to define the structure of feature catalogues. The chosen CSL metamodel (e.g. the UML metamodel as restricted by ISO 19103) is the metamodel for an application schema. As the application schema deals with data representing features, the structure of the GFM has to be kept in mind while creating the application schema.

7.3.3 The main structure of the GFM

Figure 5 shows the concepts used to define types of features. Figure 5 is an extract from the whole model. Subclause B.3 provides a figure (Figure B.2) showing all concepts of the GFM and the relationships between them.

Besides a name and a description, a feature type is defined by its properties such as

— feature attributes;

— feature association roles characterizing the feature type; and

— defined behaviour of the feature type.

Additional concepts are

— feature associations between the feature type and itself or other feature types;

— generalization and specialization relationships to other feature types; and

— constraints on the feature type.

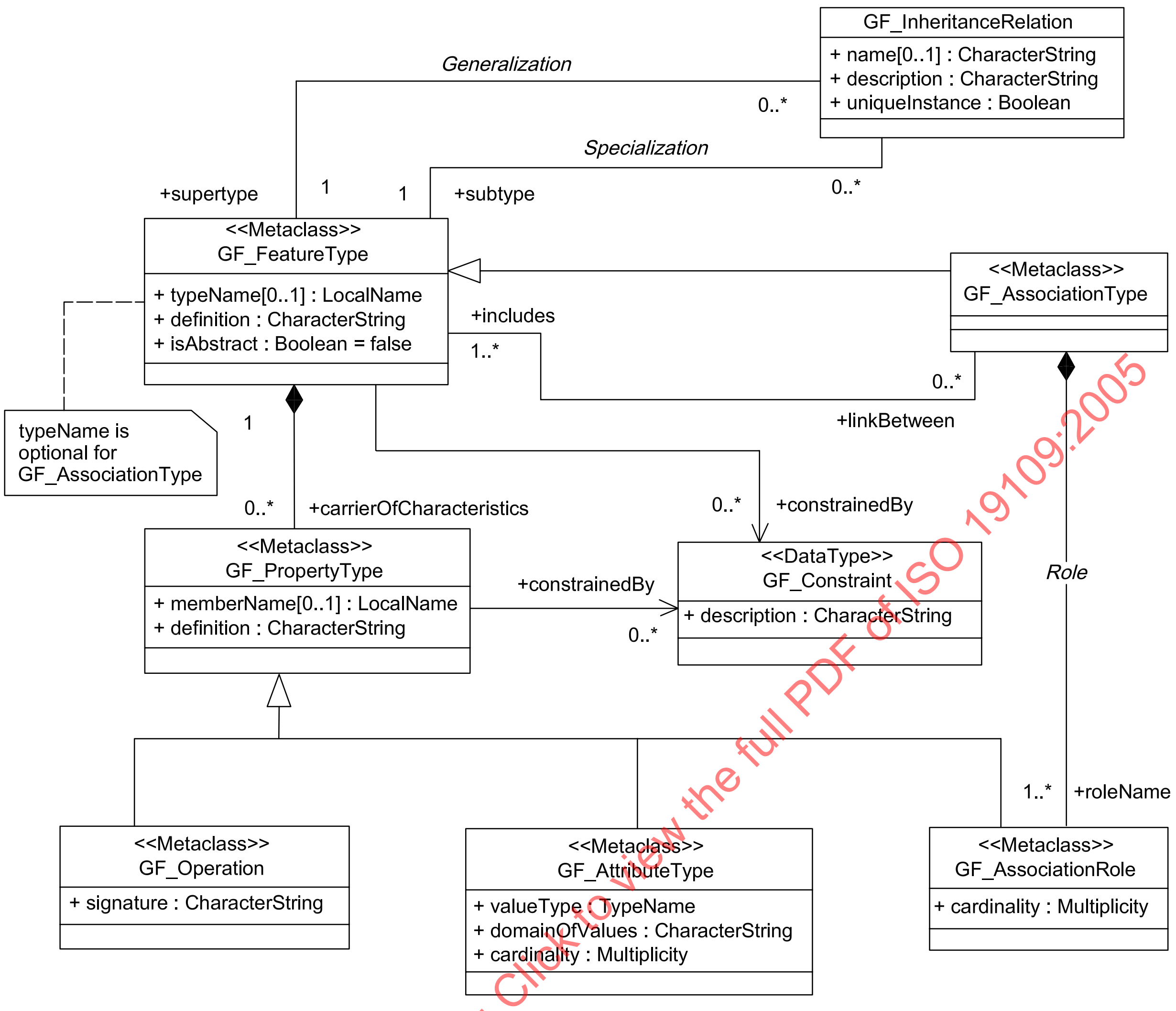

GF_InheritanceRelation
+ name[0..1] : CharacterString
+ description : CharacterString
+ uniqueInstance : Boolean
Generalization
0..*
Specialization
0..*
+supertype
1
1
+subtype
<<Metaclass>>
GF_FeatureType
+ typeName[0..1] : LocalName
+ definition : CharacterString
+ isAbstract : Boolean = false
<<Metaclass>>
GF_AssociationType
+includes
1..*
0..*
+linkBetween
typeName is optional for GF_AssociationType
1
0..*
+carrierOfCharacteristics
0..*
+constrainedBy
<<Metaclass>>
GF_PropertyType
+ memberName[0..1] : LocalName
+ definition : CharacterString
+constrainedBy
0..*
<<DataType>>
GF_Constraint
+ description : CharacterString
Role
1..*
+roleName
<<Metaclass>>
GF_Operation
+ signature : CharacterString
<<Metaclass>>
GF_AttributeType
+ valueType : TypeName
+ domainOfValues : CharacterString
+ cardinality : Multiplicity
<<Metaclass>>
GF_AssociationRole
+ cardinality : Multiplicity

Figure 5 — Extract from the General Feature Model

NOTE Figure 8 shows additional relationships from GF_Operation to GF_AttributeType and GF_AssociationType.

7.3.4 GF_FeatureType

A feature is an abstraction of real-world phenomena. GF_FeatureType is a metaclass that is instantiated as classes that represent individual feature types. A certain feature type is the class for all instances of that feature type. The instances of a class that represents an individual feature type are feature instances.

NOTE 1 Feature types are equivalent to classes and feature instances are equivalent to objects, in object-oriented modelling.

NOTE 2 Annex B provides a table on the use of the term “feature”.

— **typeName**

name of the feature type. The name shall be unique within an application schema. TypeName is optional only for GF_AssociationType.

— **LocalName**

identifier within a name space for a local object. Types defined in ISO/TS 19103, LocalName is a subtype of GenericName which is a component of the NameSpace definition. This could be the target object of the GenericName, or a pointer to another NameSpace (with a new GenericName) one step closer to the target of the identifier.

— **definition**

definition that describes the feature type.

— **isAbstract**

Boolean attribute. If true, the feature type acts as an abstract supertype.

— **includes**

The association role specifies that an instance of a feature association may include any number of instances of feature types.

7.3.5 GF_PropertyType

GF_PropertyType is the metaclass for any class of property of a feature type that describes characteristics of the feature, the behaviour of a feature, or the association roles that the feature is in.

GF_Property Type is the supertype of GF_Operation, GF_AttributeType and GF_AssociationRole.

— **memberName**

name of the behaviour, attribute or role. Only name of role is optional.

— **LocalName**

identifier within a name space for a local object. This could be the target object of the GenericName, or a pointer to another NameSpace (with a new GenericName) one step closer to the target of the identifier. In the Basic Types defined in ISO/TS 19103, LocalName is a subtype of GenericName which is a component of the NameSpace definition.

— **definition**

— description of the behaviour, attribute or role of a feature type.

— **carrierOfCharacteristics**

The association role carrierOfCharacteristics specifies that any feature operation, any feature attribute type and any feature association role carries characteristics of a feature type.

7.3.6 GF_AttributeType

— GF_AttributeType is the metaclass for attribute definitions of a feature type (see also 7.4).

— **valueType**

data type of the attribute value.

NOTE ISO/TS 19103 defines data types that may be used for the valueType of a feature attribute.

EXAMPLE 1 Integer, character String or GM_Object.

— **TypeName**

identifier within a type space for a local object. In the Basic Types defined in ISO/TS 19103, TypeName is subtype of LocalName (see 7.3.4).

— **domainOfValues**

description of a set of values.

EXAMPLE 2 Positive, from 3 to 7, GM_Object and all its subtypes as defined in ISO 19107.

— **cardinality**

number of instances of the attribute that may be associated with a single instance of a feature type.

— **Multiplicity**

specification of the range of allowable cardinalities that a set may assume. This data type is defined in ISO/TS 19103.

7.3.7 GF_AssociationRole

GF_AssociationRole is the metaclass for the classes of roles that are part of a GF_AssociationType (see 7.3.9).

NOTE GF_AssociationRole indicates the role played by the feature type through the association. The instance of GF_AssociationRole that gives the role for one feature type can also be seen as part of this feature type.

— **cardinality**

number of instances of the feature type that can act in this role relative to a single instance of the feature type at the other end of the association.

— **Multiplicity**

specification of the range of allowable cardinalities that a set may assume.

— **Role**

The association role specifies the roles associated to a GF_AssociationType.

— **roleName**

role of being a specific role assigned to a GF_AssociationType.

7.3.8 GF_Operation

GF_Operation is the metaclass for describing behaviour of feature types in terms of operations (see also 7.6).

NOTE 1 GF_operations only apply to the interoperability model and do not apply to the data transfer model as described in 6.3.1

NOTE 2 Instances of GF_Operation are of three kinds: observer operations, mutator operations and constructor operations. Observer operations return the current values of attributes. Mutator operations include actions that change those values. A mutator operation creates an instance of a class for which it is defined. For example, an observer operation may be used to find the height of a dam. Raising the dam is a mutator operation that changes the height of the dam and also affects the attributes of the watercourse and the reservoir associated with the dam. Values may be observed or affected for another feature instance if there is an association between the feature types involved.

— **signature**

description that indicates the name, the arguments and the return values of an operation.

NOTE 3 In UML, signature is expressed in the form *operation_name(input_parameter1, input_parameter2,...) : output_value_type*, for example has_height() : real.

7.3.9 GF_AssociationType

GF_AssociationType is the metaclass for describing associations between feature types (see also 7.5). A feature association may have attributes. This is allowed because GF_AssociationType is a subtype of GF_FeatureType.

NOTE GF_AssociationType is subclassed under GF_Feature type for a variety of reasons. First, from a philosophical point of view, an association instance between feature instances meets the requirements in the definition of feature as an "abstraction of real-world phenomenon". In the case of an instance of an association type, the "phenomenon" is the interaction of the two features. Second, from a pragmatic point of view, because an association meets the philosophical test for feature, it also meets some of the more pragmatic requirements. Associations often have spatial attributes, such as the location of the feature interaction. Associations may need to carry other attributes to describe the interaction, such as in 2D, road-rail intersections need to be classified as "road-overpass", "road-underpass", "rail-overpass", "rail underpass" or "at-grade", and may need to carry other attributes such as "clearance". In many cases, the infrastructure at the point of interaction is naturally treated as a feature in its own right (such as a "rail-bridge"). In addition to the classical feature attributes, association instances can also carry metadata information. Pragmatically, association instances carry attributes just like feature instances, and inheriting the semantics and the code to accomplish that simplifies geographic information software.

— **linkBetween**

The association role linkBetween specifies that a GF_AssociationType will be a link from one instance of a feature type to the same or another instance of a feature type.

7.3.10 GF_InheritanceRelation

GF_InheritanceRelation is the class for a generic relationship between a more general feature type (supertype) and one specialized feature type (subtype).

Any instance of a specialized feature type is also an instance of the general feature type.

EXAMPLE The feature type "bridge" may belong to both the general class of "transportation feature" for road features and to the general class of "hazards" for navigation features. A specific instance of "bridge" is then also an instance of "transportation feature" and "hazards".

Each specialization expresses a purpose. A feature type can act as supertype in a number of generic relationships, each having a different purpose.

— **name**

name of generalization/specialization; optional.

— **description**

explanation of the generalization/specialization.

— **uniqueInstance**

UniqueInstance is a Boolean variable, where .TRUE. means that an instance of the supertype shall not be an instance of more than one of the subtypes, whereas .FALSE. means that an instance of the supertype may be an instance of more than one subtype.

— **Generalization**

The association Generalization specifies that a feature type has the role of being a supertype in an inheritance relationship with another feature type.

— **Specialization**

The association Specialization specifies that a feature type has the role of being a subtype in an inheritance relationship with another feature type.

— **supertype**

role of being the more generic feature type of one other or other feature types.

— **subtype**

role of being the more specific feature type of one other or other feature types.

7.3.11 GF_Constraint

GF_Constraint is the class for constraints that may be associated with feature types and the properties of feature types (see also 7.7).

— **description**

The constraint described in natural language and/or in formal notation.

— **constrainedBy**

The role specifies that constraint is made on the GF_FeatureType or specified on GF_Properties within a feature type.

7.4 Attributes of feature types

7.4.1 Introduction

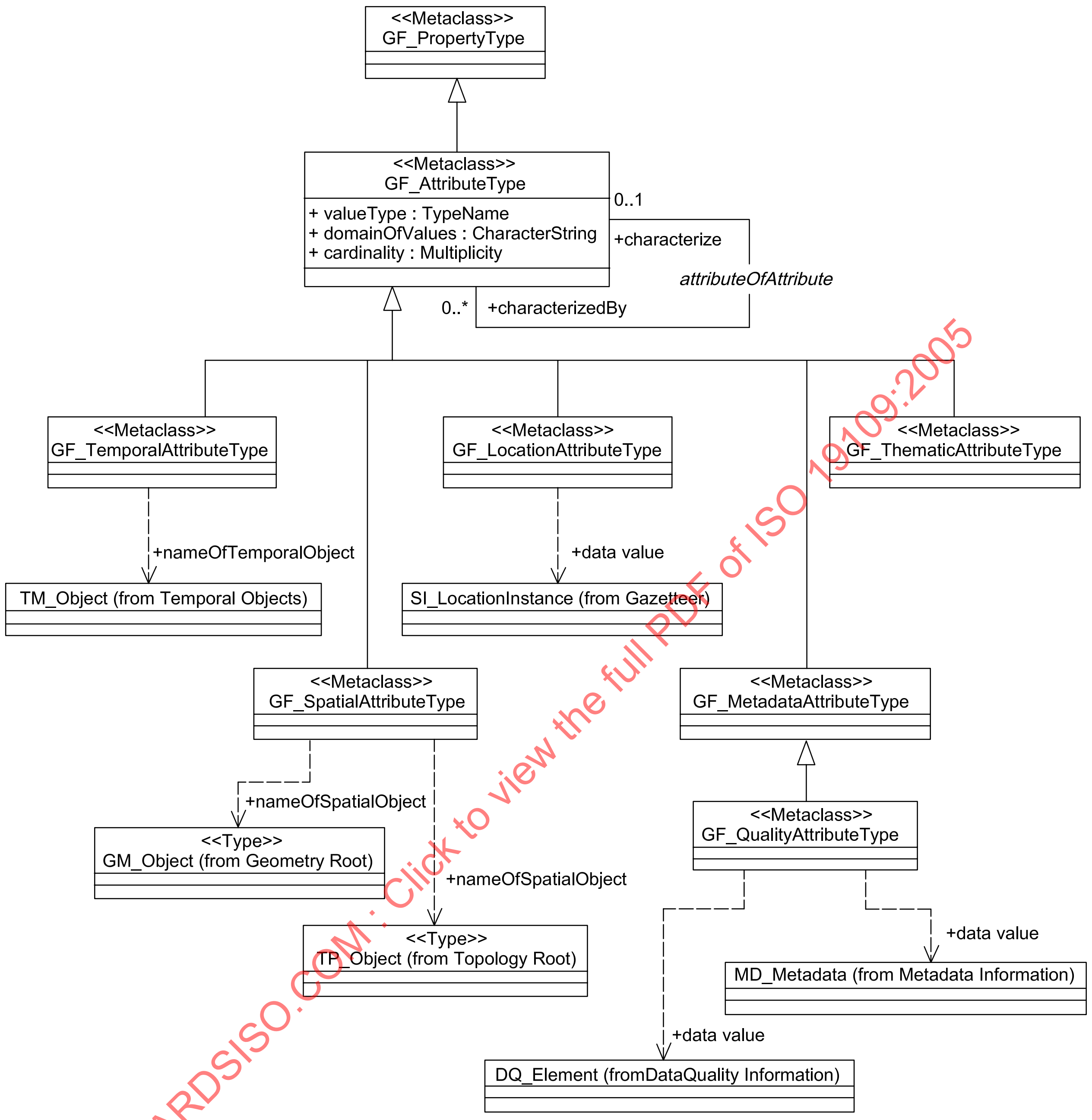
This subclause describes in more detail the role of attributes of features. An attribute type (GF_AttributeType) has a name (memberName), a description (definition), a type, a domain and cardinality associated with it.

The attributes carry all static information of a feature. This covers both spatial and non-spatial properties. In the ISO 19100 series of International Standards, some attribute types are of specific interest. These types are shown in Figure 6 as subtypes of GF_AttributeType. The attribute provides the interface to these other ISO 19100 International Standards because it uses their schemas. The attribute type will get the value type definition from those schemas and the value domain according to those schemas. For example, a spatial attribute type (GF_SpatialAttributeType) will have its value type and value domain according to the definition of GM_Object or TP_Object described in ISO 19107.

7.4.2 attributeOfAttribute

The association attributeOfAttribute links an attribute to another attribute that describes some characteristics of the first attribute.

EXAMPLE An attribute that carries the position of a feature may have another attribute that holds the positional accuracy (data value of GF_QualityAttributeType) of this position.



<<Metaclass>> GF_PropertyType
<<Metaclass>> GF_AttributeType
+ valueType : TypeName
+ domainOfValues : CharacterString
+ cardinality : Multiplicity
0..1
+characterize
attributeOfAttribute
0..*
+characterizedBy
<<Metaclass>> GF_TemporalAttributeType
<<Metaclass>> GF_LocationAttributeType
<<Metaclass>> GF_ThematicAttributeType
+nameOfTemporalObject
+data value
TM_Object (from Temporal Objects)
SI_LocationInstance (from Gazetteer)
<<Metaclass>> GF_SpatialAttributeType
<<Metaclass>> GF_MetadataAttributeType
+nameOfSpatialObject
<<Type>> GM_Object (from Geometry Root)
<<Metaclass>> GF_QualityAttributeType
+nameOfSpatialObject
<<Type>> TP_Object (from Topology Root)
+data value
MD_Metadata (from Metadata Information)
+data value
DQ_Element (fromDataQuality Information)

Figure 6 — Attributes of feature types

7.4.3 GF_SpatialAttributeType

GF_SpatialAttributeType represents a spatial attribute, which shall be used to express spatial characteristics of a feature type. A spatial attribute type shall have a GM_Object or a TP_Object as value type. The structures of GM_Object and TP_Object are defined in the Spatial Schema described in ISO 19107.

7.4.4 GF_TemporalAttributeType

GF_TemporalAttributeType represents a temporal attribute, which shall be used as the time reference characteristic of a feature. A temporal attribute type shall have a TM_Object as value type. The structure of TM_Object is defined in the Temporal Schema described in ISO 19108.

7.4.5 GF_QualityAttributeType

GF_QualityAttributeType represents attributes that carry quality information. These attributes shall be used when a quality characteristic of a feature or its properties is included as data in the dataset. Quality attribute shall have its value type according to the definition of DQ_Element defined in ISO 19115.

7.4.6 GF_LocationAttributeType

GF_LocationAttributeType represents attributes which carry a spatial reference of a feature by a geographic identifier.

EXAMPLE A postal area may be identified by a code, but its location can be found in a gazetteer.

These attribute types shall have their value type defined by SI_LocationInstance, which is defined in ISO 19112.

7.4.7 GF_MetadataAttributeType

GF_MetadataAttributeType represents attributes that carry metadata information when such information is included as data in a dataset. These attribute types shall use the metadata element classes defined in the Metadata Schema (ISO 19115) as their value types.

7.4.8 GF_ThematicAttributeType

GF_ThematicAttributeType represents attributes which carry any other descriptive characteristic of a feature except those specified in 7.4.3 to 7.4.7. Their value types and value domains will normally be defined by the user or by the application area. Both basic types (see ISO/TS 19103) and user defined data types can be used.

NOTE Information about thematic attributes may be found in feature catalogues.

7.5 Relationships between feature types

7.5.1 Introduction

This subclause describes relationships between feature types in some more detail. Figure 7 shows that relationships are classified as follows:

— generalization/specialization of feature types; and

— associations between features.

NOTE Association is a kind of relationship that may exist between both feature types and instances of feature types. The generalization/specialization relationship only occurs between feature types.

7.5.2 GF_InheritanceRelation

The GF_InheritanceRelation represents the specialization and generalization that specifies subtypes and supertypes of feature types. These relationships exist only between feature types, not between feature instances. A typical and powerful property of this kind of relationship is that a subtype will inherit all properties of its supertype.

EXAMPLE A generalization/specialization relationship between the feature types “lake” and “water body” says that an instance of the type “lake” is also an instance of the type “water body”; there are two types but only one instance involved in the relationship.

7.5.3 GF_AssociationType

GF_AssociationType represents all other association types between feature types. These relationships will appear both as association types when they are defined and as instances in the dataset. GF_AssociationType is modelled as a subtype of GF_FeatureType. Accordingly, by definition of feature types in GFM, an association type may be characterized by its own properties, for example having its own attributes.

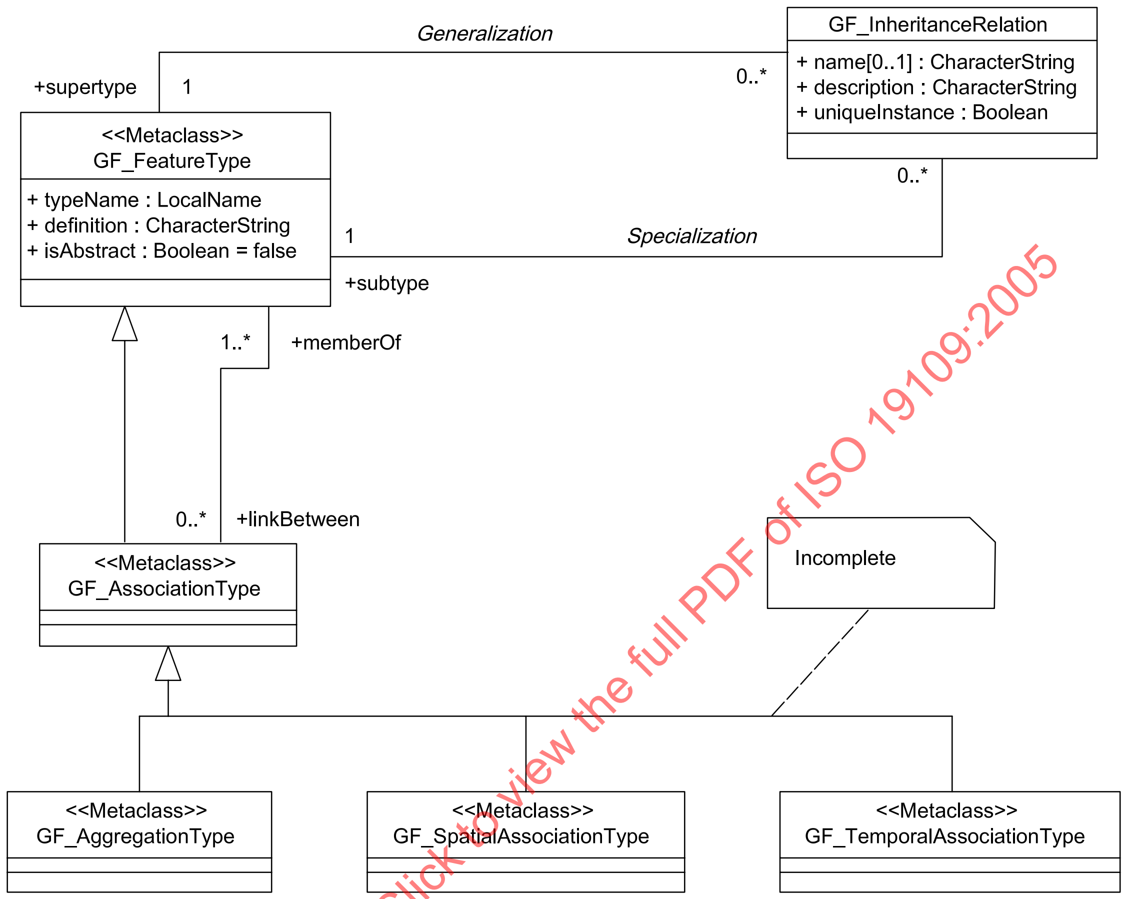


Figure 7 — Relationships between feature types

In the area of geographic information, there exist multiple kinds of associations, which shall be handled by rules defined in this International Standard. Three of these are shown in Figure 7 as subtypes of GF_AssociationType.

— **GF_AggregationType**

GF_AggregationType represents associations between the feature types that are some kind of “whole-part” relationship. The concept of aggregation includes both aggregation where the parts can exist even if the aggregate is destroyed and aggregation where the parts live inside the aggregate and will be destroyed together with the aggregate. Aggregation shall be used to specify feature types which form a complex feature type.

— **GF_SpatialAssociationType**

GF_SpatialAssociationType represents spatial relationships or topological relationships that may exist between features. Those relationships depend upon the relative spatial position of features. See further discussion in 8.7.4.

EXAMPLE 1 Sewerage network consisting of sewers and manholes.

EXAMPLE 2 An intersection is a relationship between two roads, and a road is a relationship between two intersections.

— **GF_TemporalAssociationType**

GF_TemporalAssociationType represents temporal associations that may exist between features. See further discussion in 8.6.3.

EXAMPLE 3 The phrase “built before” is the temporal association in the statement “Amsterdam central railway station was built before Tokyo railway station”.

7.6 Behaviour of feature types

7.6.1 Introduction

The behaviour of features is described by operations that may be performed upon or by all instances of a feature type. A GF_Operation represents the behaviour of a feature type as a function or a method. ISO 19110 provides a broader discussion on behaviour of feature types.

Figure 8 shows the dependencies which may have an impact on the behaviour of a feature.

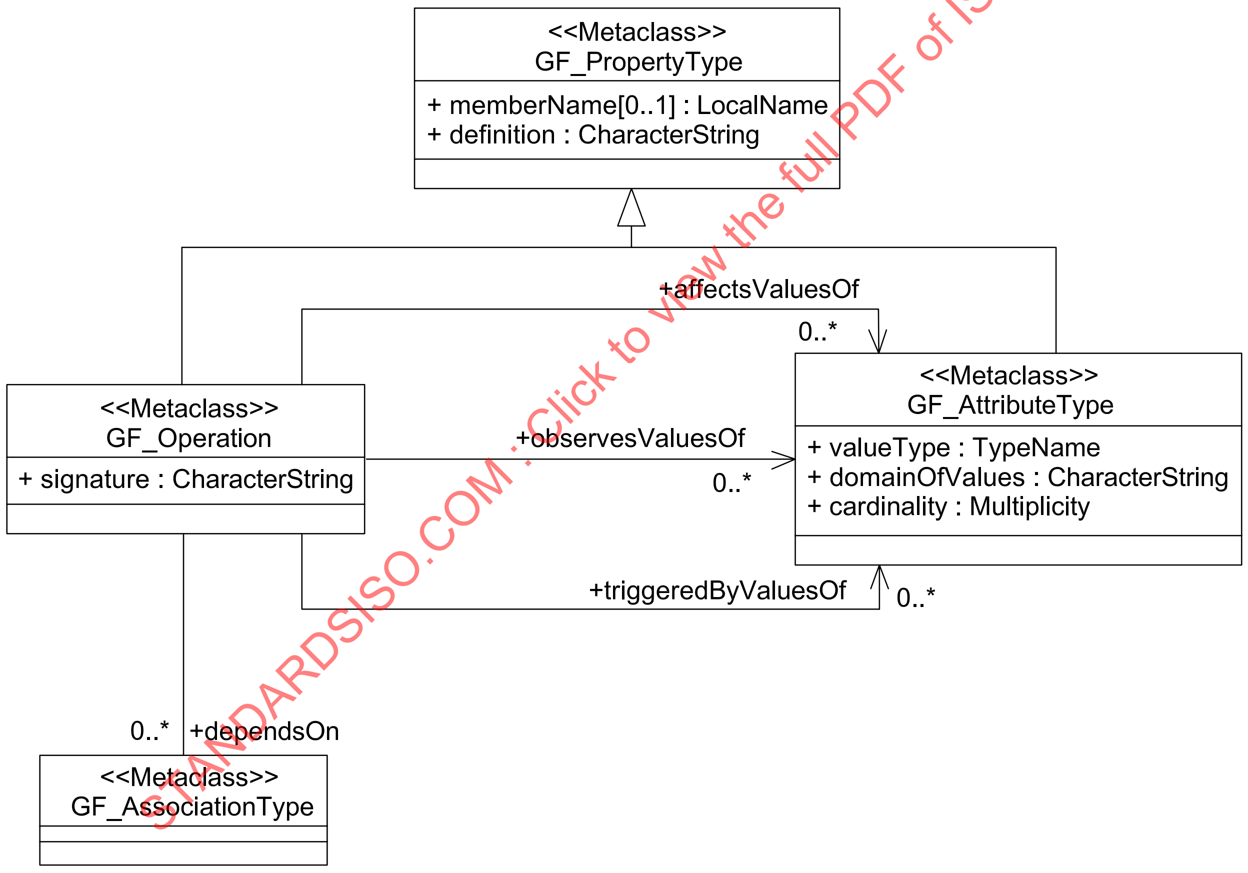


Figure 8 — Behaviour of feature types

7.6.2 observesValuesOf

The association observesValuesOf specifies attributes which may be used as input to perform an operation.

7.6.3 affectsValuesOf

The association affectsValuesOf specifies attributes which will be affected by an operation.

7.6.4 triggeredByValuesOf

The association triggeredByValuesOf specifies attributes which may trigger an operation.

7.6.5 dependsOn

The association dependsOn specifies associations which may be used to perform an operation.

7.7 Constraints

An application schema may introduce constraints to ensure the integrity of the data. Constraints restrict the freedom in an application to prevent creation of erroneous data by specifying combinations of data that are either allowable or not allowable. An application schema shall identify constraints in an unambiguous manner. Both a feature and its properties may have constraints.

EXAMPLE Examples of constraints are as follows:

— A constraint may specify an acceptable combination of attribute values in one or more feature instances (that may belong to different types).

— A constraint may restrict the cardinality of an association between feature instances.

— A constraint may require that if the real-world phenomenon is of a certain size, the feature instance be represented by a certain subtype of GM_Object.

— The behaviour of a feature, as defined in a feature operation, may be restricted by a constraint.

Application schema developers may express constraints in constraint languages specific to their implementation environments.

8 Rules for application schema

8.1 The application modelling process

The application schema serves two purposes. Firstly, it achieves a common and correct understanding of the content and structure of data within a particular application field. Secondly, it may provide a computer-readable schema for applying automated mechanisms for data management.

The two roles imply a stepwise process for creating an application schema. The steps are depicted in Figure 4. The steps can be briefly described as follows:

a) surveying the requirements from the intended field of application (Universe of Discourse);

b) making a conceptual model of the application with concepts defined in the General Feature Model. This task consists of identifying feature types, their properties and constraints;

c) describing the application schema in a formal modelling language (for example UML and OCL) according to rules defined in this International Standard;

d) integrating the formal application schema with other standardized schemas (spatial schema, quality schema, etc.) into a complete application schema.

This process requires two sets of rules:

— how to map the feature types expressed in the concepts of the General Feature Model to the formalism used in the application schema; and

— how to use schemas defined in the other ISO 19100 series of International Standards.

8.2 The application schema

8.2.1 Conceptual schema language for application schemas

The use of a formal language provides unambiguous and consistent representation of models, which facilitates implementations of applications. The normative part of this International Standard uses UML as the formal language for the description of application schema. The rules defined in Clause 8 are dependent on the UML formalism. The examples in Clause 8 are shown in UML. ISO/IEC 19501 explains how to use the UML language. ISO/TS 19103 explains how stereotypes and basic datatypes shall be used within the ISO 19100 series of International Standards and application schemas developed according to these standards.

NOTE An example of equivalent rules is shown in Annex C, where EXPRESS (as defined in ISO 10303-11:1994) is the conceptual schema language.

8.2.2 Main rules

Rules:

1) The data structures of the application shall be modelled in the application schema.

2) All classes used within an application schema for data transfer shall be instantiable. This implies that the integrated class must not be stereotyped <<interface>>.

8.2.3 Identification of an application schema

Rules:

1) The identification of each application schema shall include a name and a version. The inclusion of a version ensures that a supplier and a user agree on which version of the application schema describes the contents of a particular dataset.

2) In UML, an application schema shall be described within a PACKAGE, which shall carry the name of the application schema and the version stated in the documentation of the PACKAGE.

8.2.4 Documentation of an application schema

Rules:

1) An application schema shall be documented.

2) The documentation of an application schema in UML may utilize the documentation facilities in the software tool that is used to create the application schema, if this information can be exported.

3) If a CLASS or other UML component corresponds to information in a feature catalogue, the reference to the catalogue shall be documented.

4) Documentation of feature types in an application schema shall be in a catalogue with a structure derived from GFM, for instance in a catalogue in accordance with ISO 19110.

8.2.5 Integration of an application schema and standard schemas

When developing a large information model, the work is often broken down into independent parts that can be integrated by a defined interface. The application schema is one part; the other standardized schemas in the ISO 19100 series of International Standards are other parts.

The complete definition of the data structure of a certain application consists of the application schema integrated with the other standard schemas to which it directly and indirectly refers.

EXAMPLE Figure 9 shows an application schema that uses elements from schemas defined in the other ISO 19100 series of International Standards. The dependencies in Figure 9 mean that the application schema package “uses” structures and definitions from the other packages.

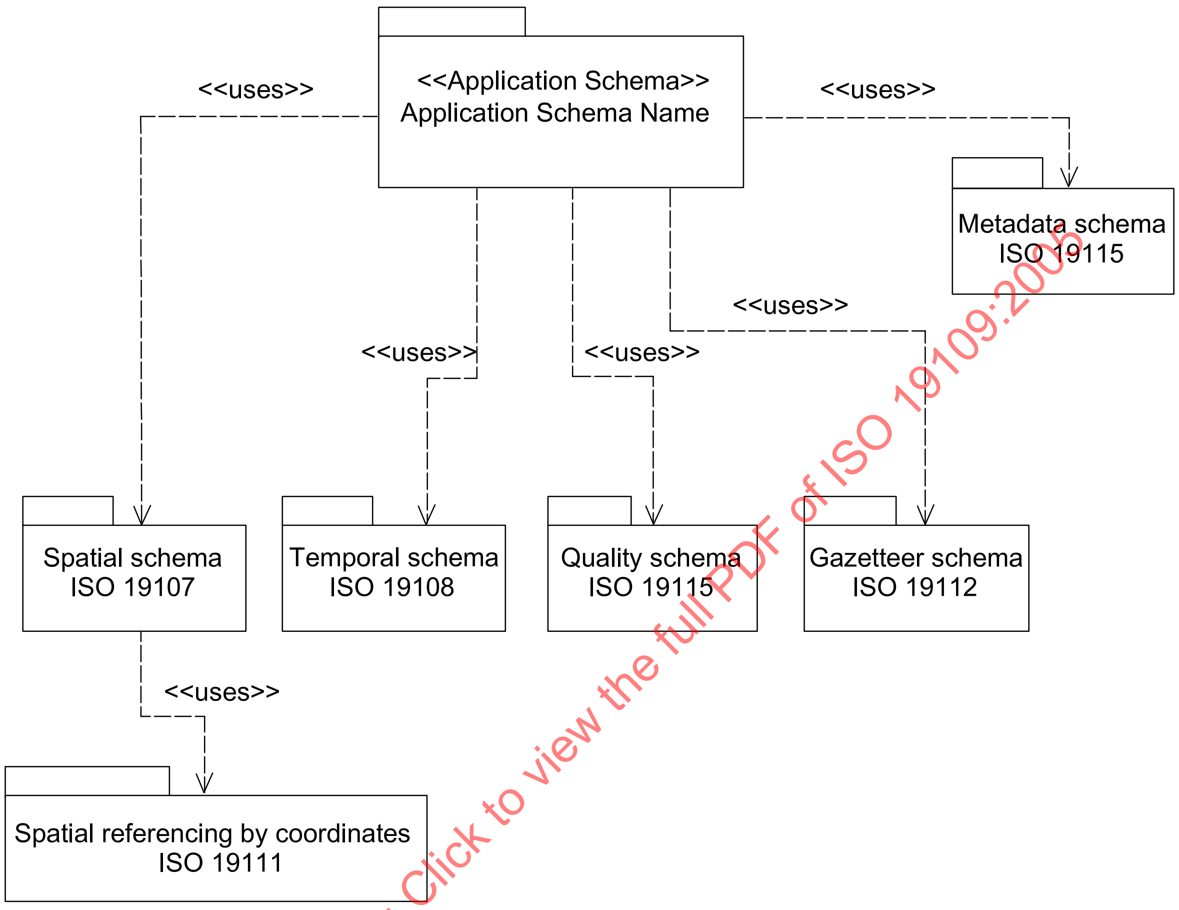


Figure 9 — Example of application schema integration

Rule:

1) The dependency mechanism in UML shall be used to describe the integration of the application schema with the other standard schemas that are required to form the complete definition of the data structure.

8.2.6 Using application schemas to build a new application schema

An application schema can be built up of several other application schemas. Each of these schemas can refer to standardized schemas. This organization can be used to avoid the creation of large and complex schemas.

EXAMPLE Figure 10 illustrates that an application containing roads, rivers and bridges can be described by the three following UML packages:

— a main schema describing the bridges;

— a schema defining data type roads; and

— a schema defining data type rivers.

All of the application schemas use spatial primitives from the spatial schema, ISO 19107.

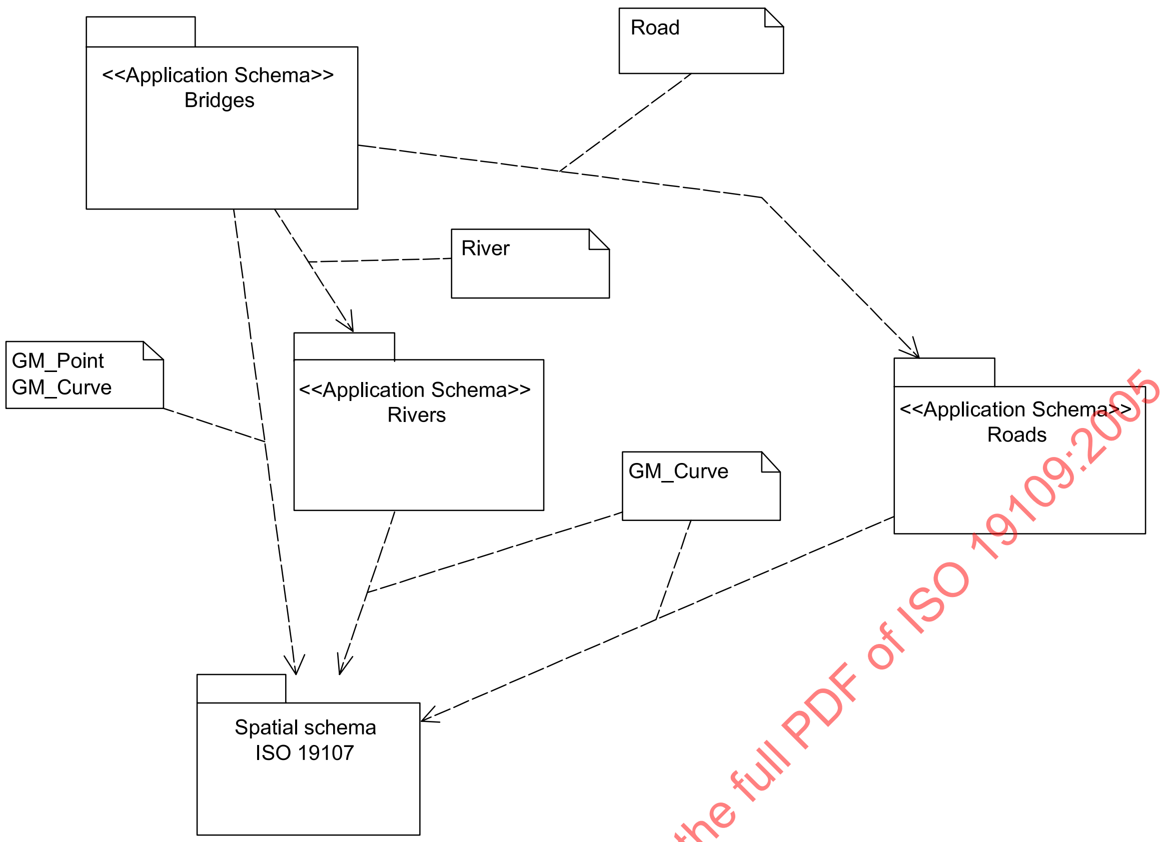


Figure 10 — Example of an application schema based on other application schemas

8.3 Rules for application schema in UML

8.3.1 Main rules

The main rules for creating application schemas in UML are as follows:

Rules:

1) GF_FeatureType: An instance of GF_FeatureType shall be implemented as a CLASS except for Rule 2, case 1 (see GF_AssociationType below).

2) GF_AssociationType: An instance of GF_AssociationType shall be implemented as one of the following cases:

— case 1: an instance of GF_AssociationType that is not associated with any instances of GF_PropertyType. In this case, it has the role of linkBetween in association to instances of GF_FeatureType being implemented as CLASSes. It shall be implemented as an ASSOCIATION between these CLASSes.

— case 2: an instance of GF_AssociationType that is associated with one or more instances of GF_PropertyType. It shall be implemented as an ASSOCIATION CLASS; the associated instances of GF_PropertyType shall be implemented as ATTRIBUTES of the ASSOCIATION CLASS.

3) GF_AggregationType: An instance of GF_AggregationType shall either be implemented as an AGGREGATION (empty diamond) or it shall be implemented as a COMPOSITION (filled diamond). Members of an aggregation can exist independently of the aggregate, and may belong to other aggregates. Members of a composite may not exist independently and may belong to only one composite.

4) GF_AttributeType: An instance of GF_AttributeType shall be implemented as an ATTRIBUTE, unless it is an attribute of an attribute [see Rule 8.3.1 e)].

5) attributeOfAttribute: An instance of GF_AttributeType that acts in the role *characterizedBy* in an attributeOfAttribute association shall be instantiated as a class. That class shall be used either as the data type of the GF_AttributeType, or in an association with the class that contains the GF_AttributeType. Attributes that act in the role *characterizes* shall be instantiated as attributes of the class that represents the attribute that acts in the role *characterizedBy*.

   — step 1: Introduce a new CLASS to represent the attribute that is characterized by other attributes. Use the attribute name as the CLASS name.

   — step 2: If appropriate, insert one ATTRIBUTE in this CLASS to represent the value of the attribute represented by the CLASS. Use the same name as the original attribute name.

   — step 3: Insert additional ATTRIBUTE(s) into this CLASS to represent the attributes that characterize the original ATTRIBUTE.

   — step 4: Use this CLASS as the datatype for the original attribute in the CLASS that contains it, or delete the original attribute from the CLASS that contained it and add an ASSOCIATION from that CLASS to the new CLASS.

6) GF_Operation: An instance of GF_Operation shall be implemented as an OPERATION of the class representing the feature type that it characterizes, which shall have ASSOCIATIONS to other CLASSES from which the operation needs ATTRIBUTE VALUES.

7) GF_AssociationRole: An instance of GF_AssociationRole shall be implemented as a role name at the appropriate end of the ASSOCIATION representing the GF_AssociationType.

8) GF_InheritanceRelation: An instance of GF_InheritanceRelation shall be represented by a UML GENERALIZATION relationship, with additional characteristics depending on the state of the following 2 conditions:

   — case 1: If *uniqueInstance* is .TRUE., the {disjoint} constraint shall be attached to the generalization relationship.

   — case 2: If *uniqueInstance* is .FALSE., the {overlapping} constraint may be attached to the generalization relationship.

9) GF_Constraint: Constraints may be stated in OCL or in plain language and attached to the CLASS, OPERATION or RELATIONSHIP that is constrained.

8.3.2 Example of application schema

Figure 11 shows an example of an application schema expressed in UML, based on real-world concepts in Table 1.

Table 1 — Example of real-world concepts in terms of the General Feature Model

Feature types	Attributes	Kind of subtype of GF_Attribute
Property parcel	Identification Name Area	Thematic attribute type Thematic attribute type Spatial attribute type
Building	Code Centre point Shape Address Type Horizontal accuracy Vertical accuracy Owner	Thematic attribute type Spatial attribute type Spatial attribute type Location attribute type Thematic attribute type Quality attribute type Quality attribute type Thematic attribute type
Loan	Amount Period Classification	Thematic attribute type Temporal attribute type Metadata attribute type

The attributes *Horizontal accuracy* and *Vertical accuracy* describe the quality of the attribute *Centre point*. There are two associations: A *Property parcel* contains zero, one or many *Buildings*. A *Building* is financed by zero, one or many *Loans*.

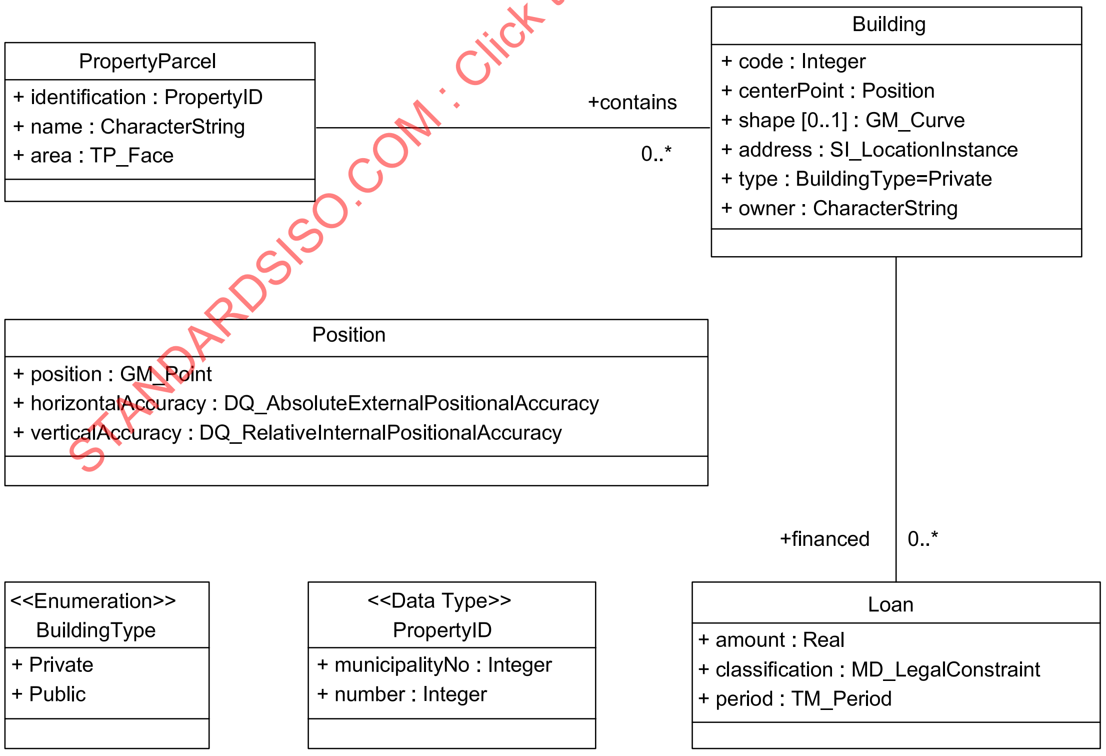

PropertyParcel
+ identification : PropertyID
+ name : CharacterString
+ area : TP_Face
+contains
0..*
Building
+ code : Integer
+ centerPoint : Position
+ shape [0..1] : GM_Curve
+ address : SI_LocationInstance
+ type : BuildingType=Private
+ owner : CharacterString
Position
+ position : GM_Point
+ horizontalAccuracy : DQ_AbsoluteExternalPositionalAccuracy
+ verticalAccuracy : DQ_RelativeInternalPositionalAccuracy
+financed
0..*
<<Enumeration>>
BuildingType
+ Private
+ Public
<<Data Type>>
PropertyID
+ municipalityNo : Integer
+ number : Integer
Loan
+ amount : Real
+ classification : MD_LegalConstraint
+ period : TM_Period

Figure 11 — Example of UML implementation of feature types

8.4 Domain profiles of standard schemas in UML

8.4.1 Introduction

Instead of using the classes as they are defined in the standard schemas directly, it is possible to make adjustments to the standard schemas to fit the actual domain application. The adjustments can be done in either of two ways:

— to add attributes to the classes defined in the standard schemas; or

— to restrict elements of a standard schema as permitted by the conformance clause of the International Standard that specifies that schema.

8.4.2 Rules for adding information to a standard schema

For many purposes, it is convenient or necessary to extend the definitions in a standard schema with additional information.

Rule:

1) If it is necessary to extend or restrict a CLASS specified in a standard schema, a new CLASS shall be defined as a SUBTYPE of the CLASS in the standard schema, and ATTRIBUTEs shall be added to this CLASS to carry the additional information.

NOTE For practical reasons the new classes may be collected in a separate PACKAGE.

EXAMPLE Figure 12 shows an application schema which is using a definition (GM_Point) from the standard spatial schema directly, and a definition (SurfaceWithQuality) in a package called domain spatial package. The domain spatial package contains adjustments to the spatial definition (GM_Surface), and at the same time, it is using a definition from quality schema (DQ_AbsoluteExternalPositionalAccuracy). The class SurfaceWithQuality contains two additional attributes which are appropriate to be used in this domain of application as additional information about the geometry describing the footprint of the building.

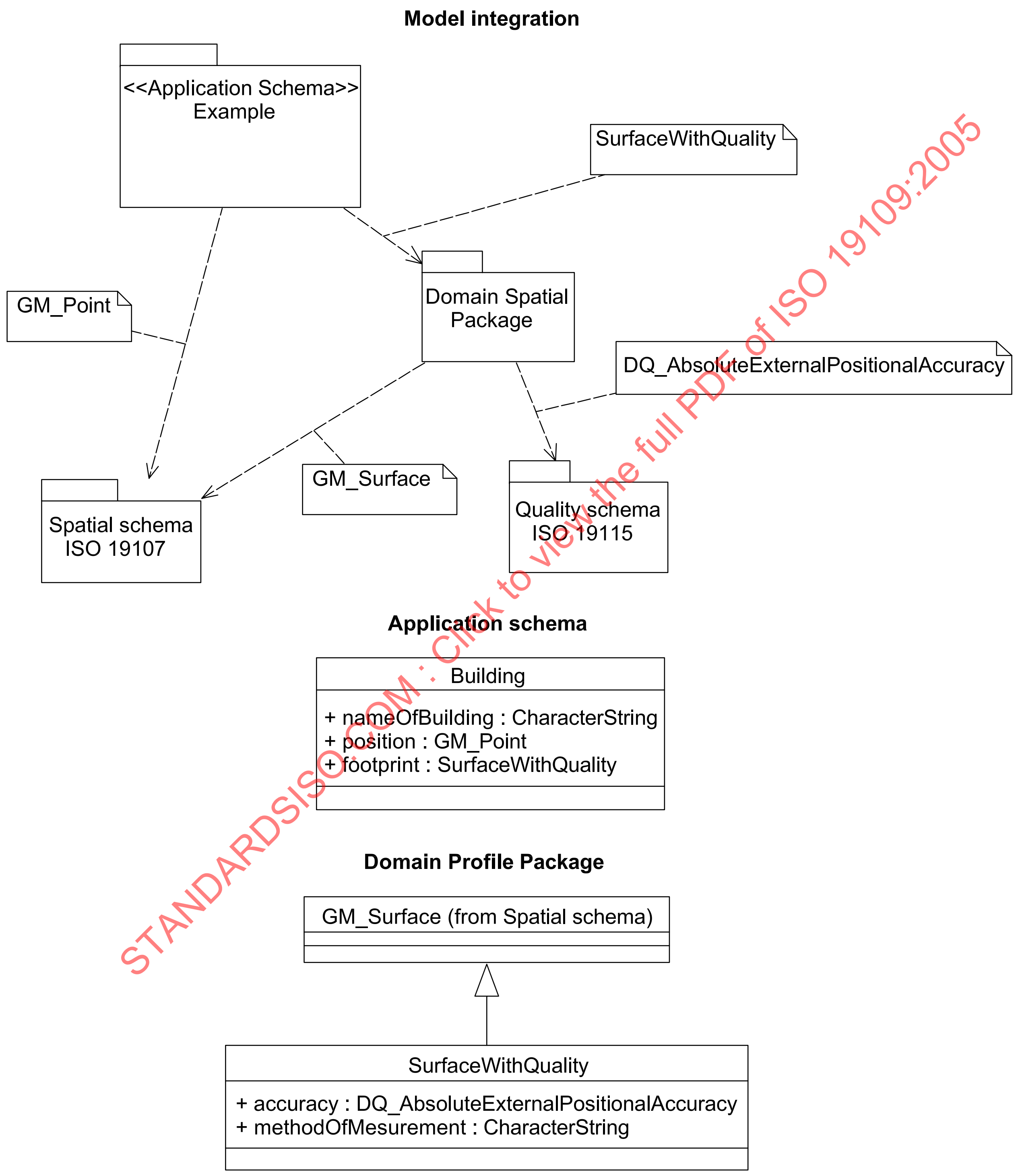

Model integration
<<Application Schema>> Example
SurfaceWithQuality
GM_Point
Domain Spatial Package
DQ_AbsoluteExternalPositionalAccuracy
GM_Surface
Spatial schema ISO 19107
Quality schema ISO 19115
Application schema
Building
+ nameOfBuilding : CharacterString
+ position : GM_Point
+ footprint : SurfaceWithQuality
Domain Profile Package
GM_Surface (from Spatial schema)
SurfaceWithQuality
+ accuracy : DQ_AbsoluteExternalPositionalAccuracy
+ methodOfMesurement : CharacterString

Figure 12 — Example of adding information to a standard schema

8.4.3 Restricted use of standard schemas

For some standard schemas, e.g. ISO 19107, it is possible to redefine the schema in such a way that only selected parts of the schema will be used, and only some of the definitions of classes and relationships will be used.

Rules:

1) Specification of a restricted profile of a standard schema shall be described in a new UML package by copying the actual definitions (classes and relationships) from the standard schema. Attributes and operations within classes may be omitted.

2) Reduction of a standard schema shall be in accordance of the conformance clause given for the actual International Standard.

EXAMPLE A restricted profile of the spatial schema (ISO 19107) could be specified as only using definitions of GM_Object and its subtypes, but not using operations associated to those classes.

8.5 Rules for use of metadata schema

8.5.1 Introduction

The metadata schema (see ISO 19115) is an application schema for metadata data sets. Metadata are data describing and documenting data. Metadata for geographic data typically provide information about their identification, extent, quality, spatial and temporal aspects, spatial reference and distribution.

8.5.2 Providing metadata for features, feature attributes, and feature associations

In some application schemas, definitions from ISO 19115 may be used. This International Standard does not place restrictions on such use.

Rules:

1) A metadata element may be used as a GF_Metadata_AttributeType (subtype of GF_AttributeType) to carry metadata about instances of feature types, feature attributes or associations between features [see 8.3.1, Rules 2), 4) and 5)].

2) The data type of any feature attribute carrying metadata shall be one of the metadata elements specified as classes in the metadata schema.

3) A feature attribute may reuse definitions from a package in ISO 19115 without carrying metadata information.

4) A metadata attribute shall be represented in an application schema as an ATTRIBUTE of a CLASS that represents the data instance it reports [see 8.3.1, Rule 4)].

5) A metadata attribute may be used as an attribute of an attribute, in which case the main rule for *attributeOfAttribute* (see 8.3.1) shall be applied.

EXAMPLE Figure 13 shows an application schema expressed in UML, where two definitions from the metadata schema are used. MD_LegalConstraint is used to identify restrictions on the use of the data [8.5.2, Rule 2)]. EX_GeographicBoundingBox is used to specify a geographical area [8.5.2, Rule 3)].

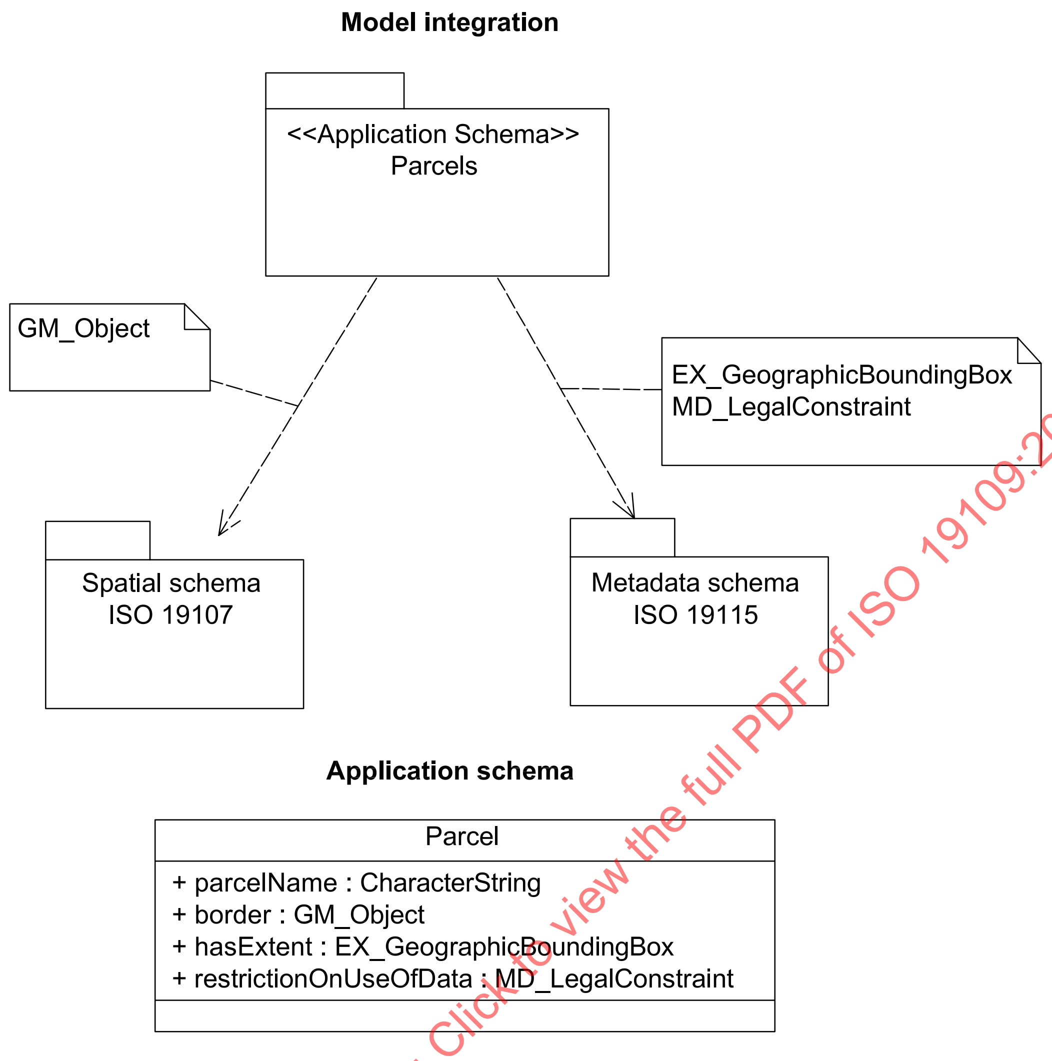


Figure 13 — Example of metadata included as data in an application

8.5.3 Data quality rules

8.5.3.1 Rules for reporting quality information for instances of data

This subclause focuses on the use of elements from the standard quality schema within an application schema. Some data quality sub-elements, and the descriptors of a data quality sub-element, may apply to individual instances of data.

NOTE 1 Both the data quality elements and the data quality subelements identified in ISO 19113 are modelled as subtypes of DQ_Element in ISO 19115.

Quality information for datasets or parts of datasets does not affect the application schema, and should be reported in the metadata for the dataset in accordance with the specifications given in ISO 19115.

<u>**Rules:**</u>

1) Information on the quality of individual instances of features or attributes shall be reported by attribution whenever the quality of an instance is expected to differ from the implied quality for the dataset or parts of the dataset.

2) A quality attribute (an instance of GF_QualityAttributeType) shall be defined in the application schema and shall be used to carry data quality information.

3) A quality attribute shall be represented in an application schema as an ATTRIBUTE of a CLASS that represents the data instance it reports [see 8.3.1, Rule 4)].

4) A quality attribute shall take one of the subtypes of the CLASS DQ_Element (see Table 2) defined in the Data Quality Information Package in ISO 19115 as the data type for its value.

Table 2 — Subtypes of DQ_Element defined in ISO 19115

Quality element	Quality sub-element	Subtype of DQ_Element
Positional Accuracy	Absolute External Accuracy Relative Internal Accuracy Gridded Data Accuracy	DQ_AbsoluteExternalPositionalAccuracy DQ_RelativeInternalPositionalAccuracy DQ_GriddedDataPositionalAcuracy
Temporal Accuracy	Accuracy of a Time Measurement Temporal Consistency Temporal Validity	DQ_AccuracyOfATimeMeasurement DQ_TemporalConsistencyAccuracy DQ_TemporalValidityAccuracy
Thematic Accuracy	Classification Correctness Nonquantitative Attribute Accuracy Quantitative Attribute Accuracy	DQ_ThematicClassificationCorrectness DQ_ThematicNonQuantitativeAttributeAccuracy DQ_ThematicQuantitativeAttributeAccuracy
Logical Consistency	Format Consistency Topological Consistency	DQ_FormatConsistency DQ_TopologicalConsistency

5) A quality attribute may be used as an attribute of an attribute, in which case the main rule for *attributeOfAttribute* (see 8.3.1) shall be applied.

NOTE 2 Wherever an ATTRIBUTE is using DQ_Element or one of its subtypes as data type, this ATTRIBUTE is carrying quality information about the CLASS within which it is defined.

EXAMPLE 1 Figure 14 shows the ATTRIBUTE *featureClassificationQuality* carrying quality information about the classification of the feature *Land Area* into subareas.

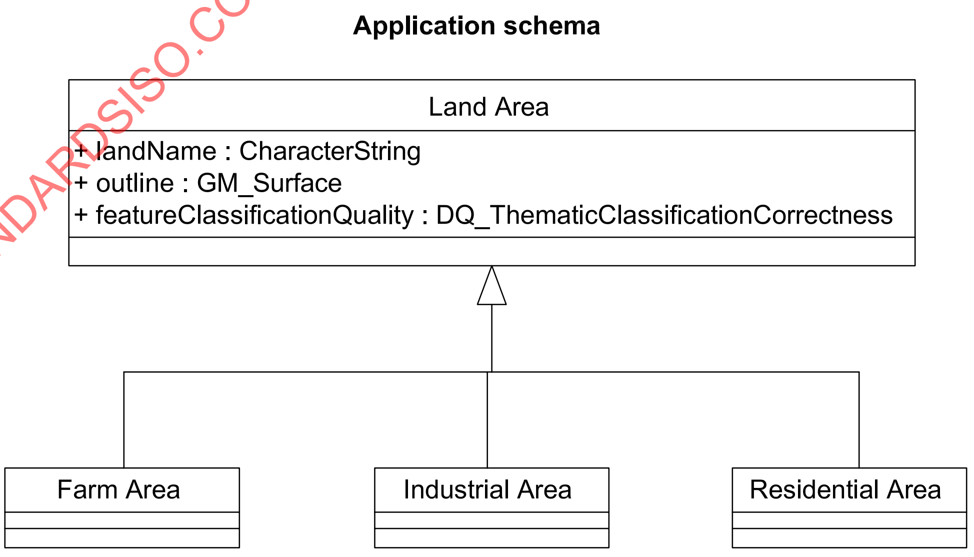

Application schema
Land Area
+ landName : CharacterString
+ outline : GM_Surface
+ featureClassificationQuality : DQ_ThematicClassificationCorrectness
Farm Area
Industrial Area
Residential Area

Figure 14 — Example of quality of features instances

EXAMPLE 2 Figure 15 shows two alternatives for UML implementation of the feature *Land Area* whose attribute *Outline* has a quality attribute *Accuracy*.

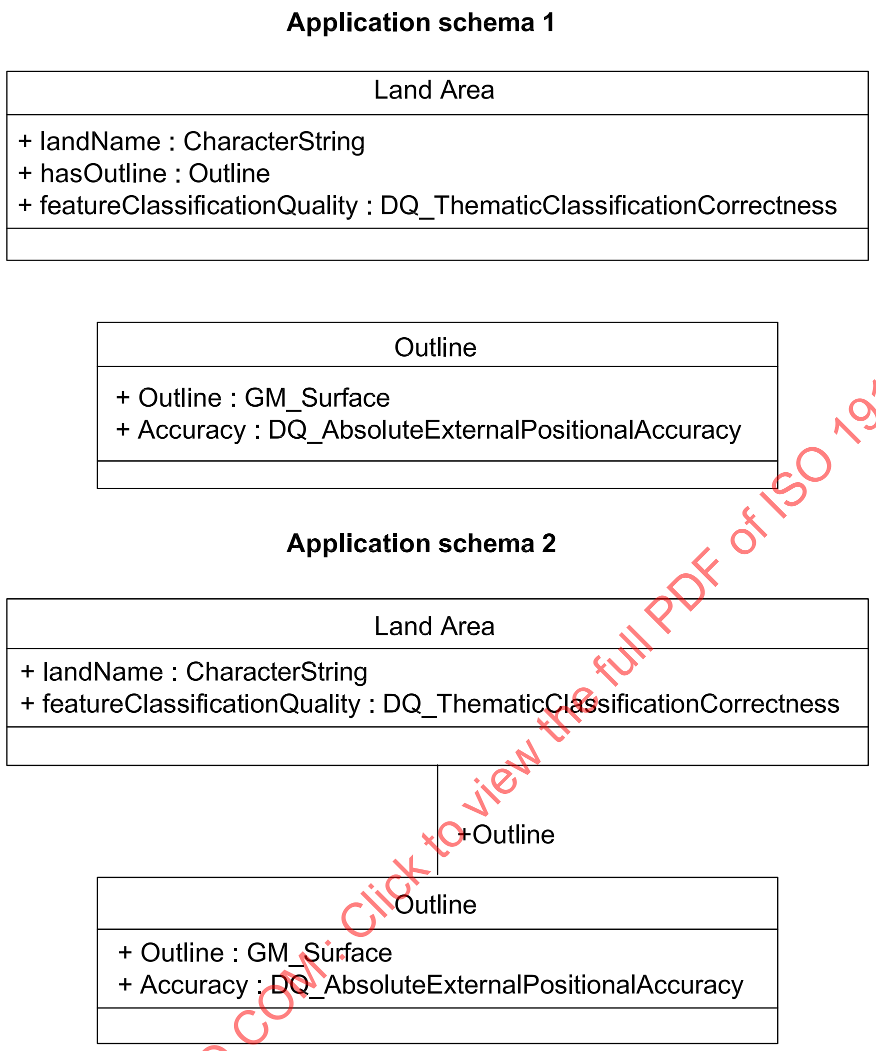

Application schema 1
Land Area
+ landName : CharacterString
+ hasOutline : Outline
+ featureClassificationQuality : DQ_ThematicClassificationCorrectness
Outline
+ Outline : GM_Surface
+ Accuracy : DQ_AbsoluteExternalPositionalAccuracy
Application schema 2
Land Area
+ landName : CharacterString
+ featureClassificationQuality : DQ_ThematicClassificationCorrectness
+Outline
Outline
+ Outline : GM_Surface
+ Accuracy : DQ_AbsoluteExternalPositionalAccuracy

Figure 15 — Example of quality of attributes of features

8.5.3.2 Rule for reporting additional quality information

For many purposes, it is convenient or necessary to extend the data quality information as it is defined in the Data Quality Information Package in ISO 19115 with an additional description called an additional quality sub-element.

<u>Rule:</u>

1) User-defined quality sub-elements shall be defined as a CLASS as a subtype of DQ_Element or one of its subtypes (see Table 2) according to rules for domain profiles of standard schemas (see 8.4.2).

EXAMPLE Figure 16 shows a user-defined quality sub-element.

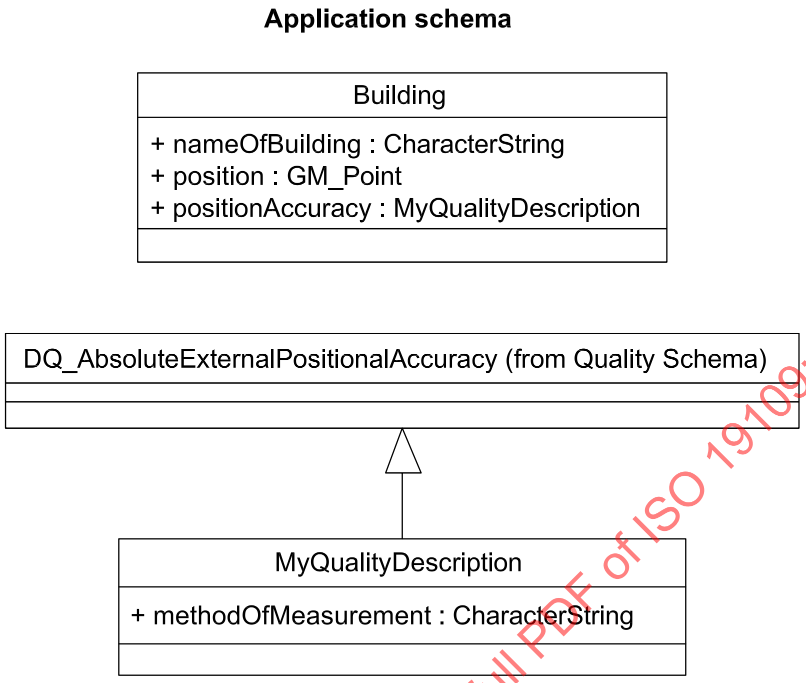

Application schema
Building
+ nameOfBuilding : CharacterString
+ position : GM_Point
+ positionAccuracy : MyQualityDescription
DQ_AbsoluteExternalPositionalAccuracy (from Quality Schema)
MyQualityDescription
+ methodOfMeasurement : CharacterString

Figure 16 — Example of user-defined quality

8.5.3.3 Reporting quality information for attributes of feature instances

Rule:

1) Quality characteristics of feature attributes shall be implemented according to 8.3.1, Rule 5): *attributeOfAttribute*.

8.6 Temporal rules

8.6.1 General temporal rules

Rule:

1) Any description of temporal aspects applied to geographic data shall be in accordance with the specifications given by ISO 19108.

NOTE It is possible to use Date, DateTime and Time, but this makes the attribute an instance of Thematic attribute type, not a Temporal attribute type, as there is no reference system connected to them.

8.6.2 Temporal attributes

Rules:

1) A temporal characteristic of a feature type shall be defined as a temporal attribute, which is a subtype of feature attribute.

2) The implementation of temporal attributes in UML shall follow the rules (see 8.2.5) for referencing standardized schemas.

3) A temporal attribute may be represented in an application schema as an attribute of a UML CLASS that represents a feature (see 8.3.1), in which case the attribute shall take one of the temporal objects defined in the Temporal Schema (see ISO 19108) as the data type for its value.

4) A temporal attribute shall be implemented as a UML class (i.e. subclass of a temporal object) when any of the following cases apply [see 8.3.1, Rule 5)]:

— case 1: attributes have multiple components;

— case 2: data types for attributes of temporal objects need to be restricted;

— case 3: operations from the interfaces defined in ISO 19108 need to be referenced explicitly.

5) A temporal attribute may be used as an attribute of an attribute (see 8.3.1), in which case the attribute shall be a subtype of one of the temporal objects defined in ISO 19108.

6) Valid temporal objects, which shall be applied, are given in Table 3.

Table 3 — List of valid temporal objects for temporal attributes in an application schema

Temporal geometric primitives	Temporal topological primitives	Temporal complexes
TM_Instant	TM_Node	TM_TopologicalComplex
TM_Period	TM_Edge	

EXAMPLE 1 Figure 17 shows an application schema for a measurement activity where equipment positioned in a field registers measured values with a defined frequency. Temporal geometric primitives are identified as data types of the temporal attribute types used in the feature types *Station* and *Measurement*.

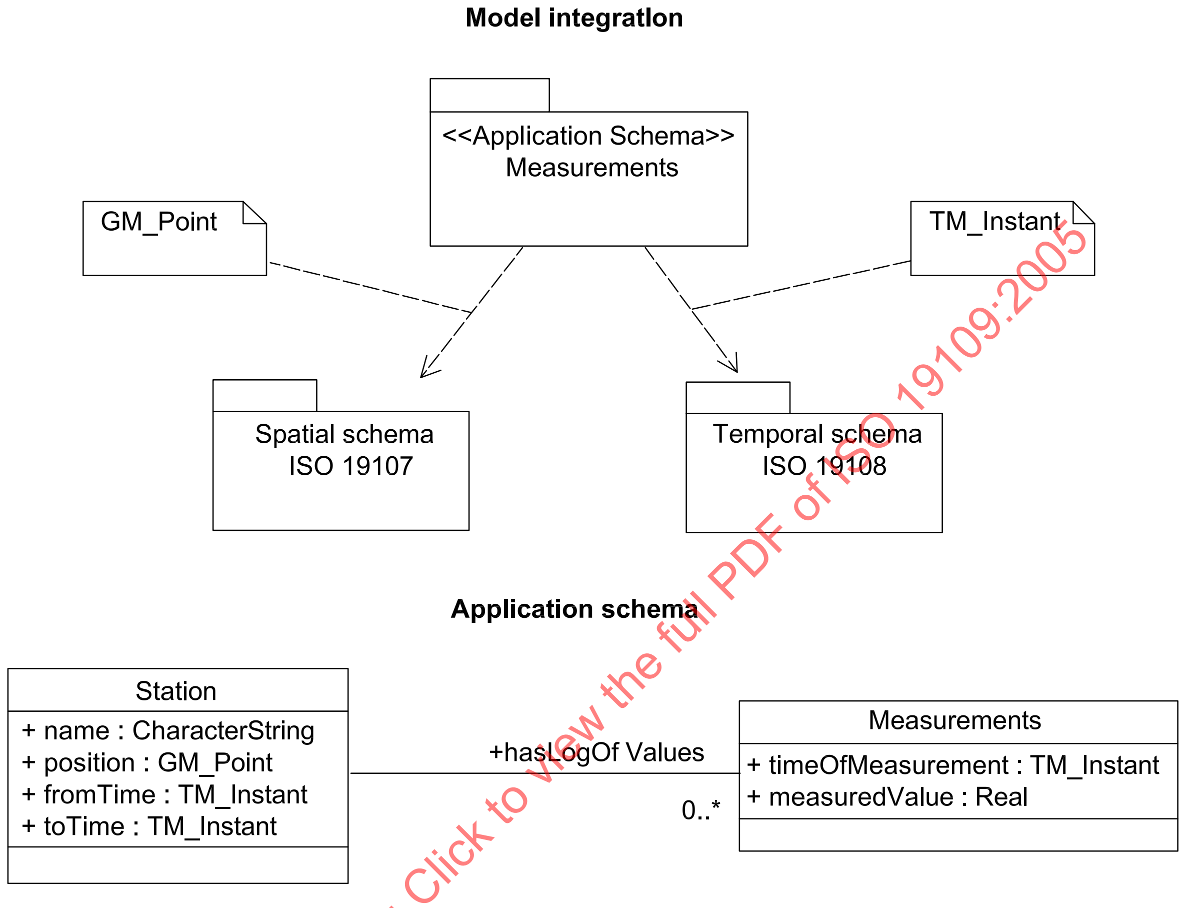


Figure 17 — Example of temporal attribute

EXAMPLE 2 Figure 18 illustrates a use of TM_TopologicalComplex as a temporal feature attribute. BuildingHistory is an attribute of the feature type Building that is represented in the schema as a UML class. As a subtype of TM_TopologicalComplex, it is an aggregation of TM_TopologicalPrimitives that describe events and states in the history of a building. BuildingHistory is an inheritance of TM_TopologicalComplex with a sequence of episodes. Each episode corresponds to a node or edge in the linear graph. BuildingHistory forms a linear graph, thus {TM_Node.previousEdge ->size $\leqslant$1; size of previous edge is 0 at the start node and {TM_Node.nextEdge->size $\leqslant$ 1}; size of next edge is 0 only at the end node. Each episode corresponds to a node or edge in the linear graph.

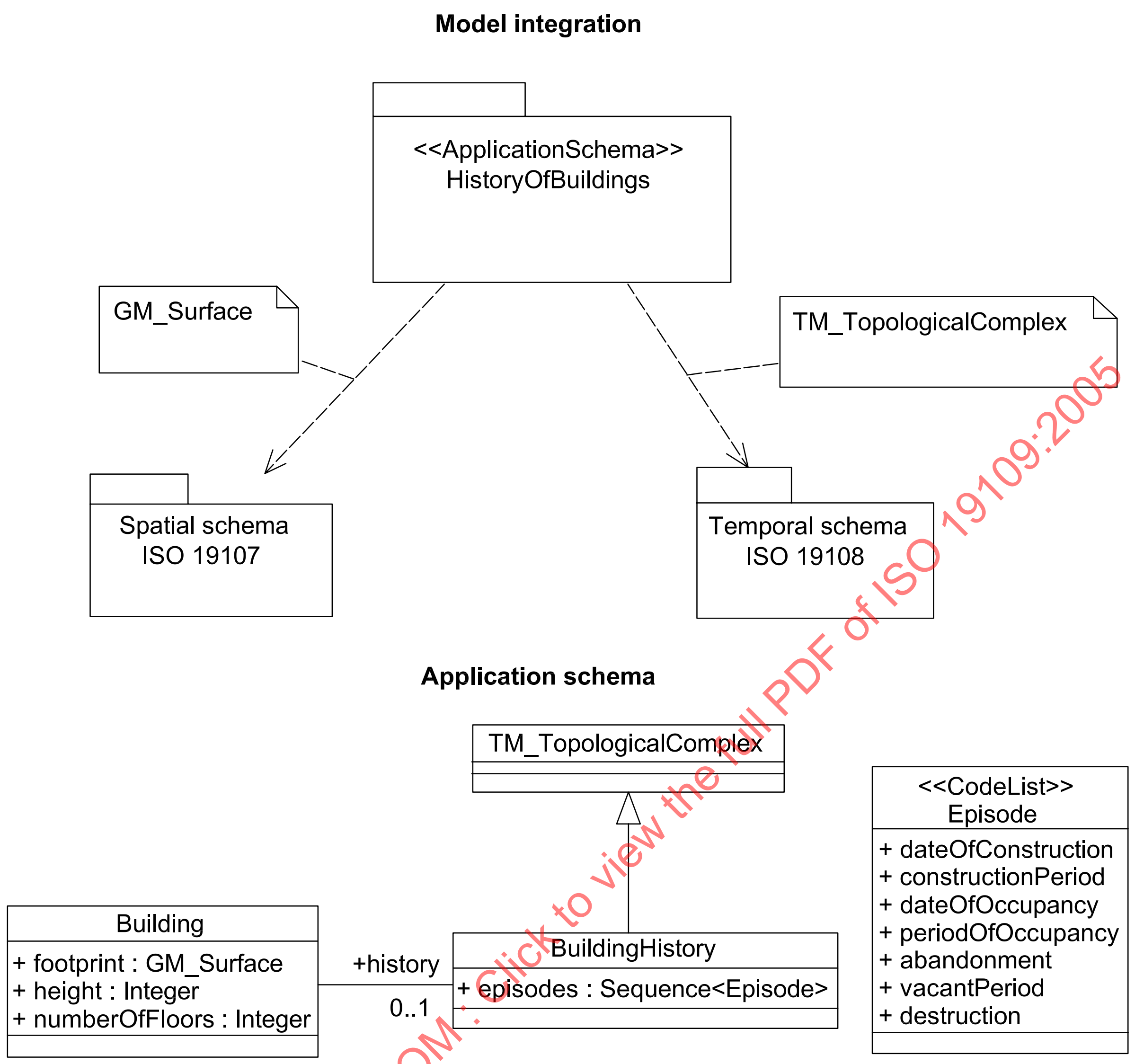


Figure 18 — Example of TM_TopologicalComplex used as a temporal feature attribute

8.6.3 Temporal associations between features

8.6.3.1 Types of relationships

According to the General Feature Model, a temporal association (GF_TemporalAssociation) is a subtype of association between features (GF_AssociationType) (see Figure 7). There are two types of temporal associations: simple temporal associations and feature succession.

8.6.3.2 Simple temporal associations

A simple temporal association can be derived by applying the operations defined for interfaces of TM_Primitives in ISO 19108 to the temporal geometric primitives used as datatypes for temporal attributes.

EXAMPLE 1 An application schema might define a UML class to represent the feature type Building, which has an attribute dateOfConstruction that takes a TM_Instant as its data type. Building A was constructed in 1950 and building B was constructed in 1970. If the operation TM_Order.relativePosition is applied to the TM_Instant that is the value for dateOfConstruction of building A with the TM_Instant that is the value for dateOfConstruction of building B as an input parameter, it will return the value “before”. If the operation TM_Separation.distance is applied in the same way, it will return the value “20 years”.

Rule:

1) Simple temporal associations shall be implemented in an application schema in one of two ways:

— case 1: Temporal primitives used as data types of attributes shall implement the operationTM_RelativePosition from the interface TM_Order defined in ISO 19108, so that simple temporal associations can be derived from the data.

— case 2: A simple temporal association between features shall be implemented in an application schema using an ASSOCIATION in UML [see 8.3.1, Rule 2)].

EXAMPLE 2 Figure 19 shows a simple temporal association implemented as an ASSOCIATION. The schema indicates that roads exist before service stations exist.

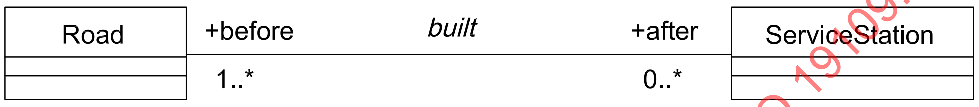


Figure 19 — Example of explicit representation of a simple temporal association

8.6.3.3 Feature succession

Feature succession is a sequence of changes over time involving the replacement of one or more feature instances by other feature instances. There are three types of feature succession: feature substitution, feature division, and feature fusion. Feature substitution is the replacement of one feature instance by another feature instance. It establishes a one-to-one relationship between two feature instances. Feature division occurs when a single feature instance separates into two or more feature instances. It establishes a one-to-many relationship between feature instances. Feature fusion occurs when two or more feature instances merge into a single feature instance. It establishes a many-to-one relationship between feature instances. Combinations of these types are possible.

There are both spatial and temporal aspects to feature succession, in that the features in the relationship occupy the same spatial location, at different times and in a particular order.

Feature succession relationships may be derived from the spatial and temporal attributes of the features. However, this requires every feature type to have a temporal attribute that identifies the period for which the feature exists. The feature succession can be derived by identifying those features that share spatial primitives, and then determining the TM_RelativePositions of their periods of existence.

Feature succession is not always type-dependent. That is, the type of a feature instance is not always a predictor of the type of the feature instance that replaces it. Feature succession can be modelled at the generic feature level, but not always at the feature type level.

Rules:

1) Feature associations of the feature succession type may be instantiated in an application schema as UML associations between feature type classes.

2) Feature associations of the feature succession type may be instantiated in an application schema as self-referent associations of a generic feature class.

3) The names, roles, and multiplicities shall be appropriate for the type of feature succession.

EXAMPLE 1 Figure 20 shows a feature succession modelled as an ASSOCIATION between feature type classes. It is an example of a type of ecological succession, known as old-field succession, common in the Eastern United States. The types occur on a single site in the sequence shown, if the site is left undisturbed.

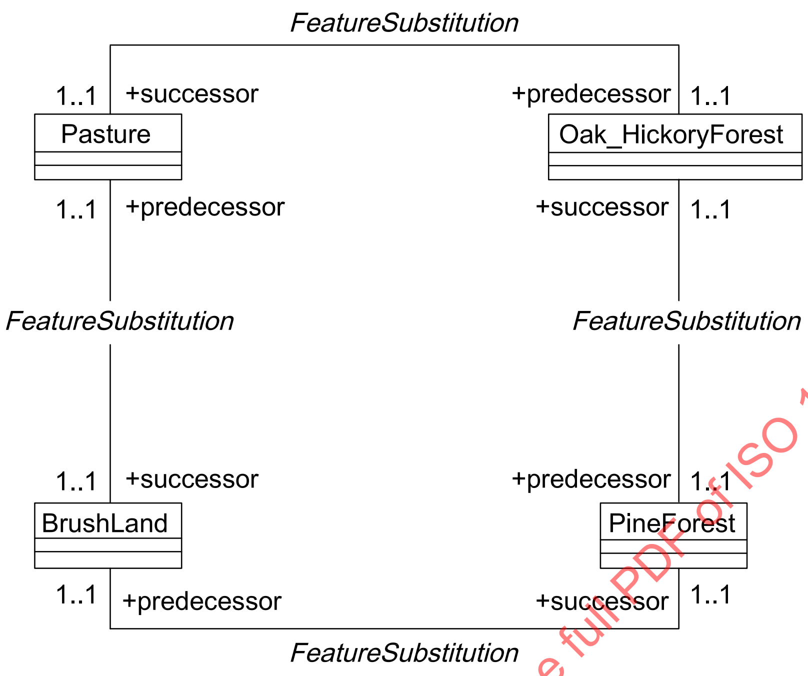


Figure 20 — Example of feature succession between feature types

EXAMPLE 2 Figure 21 shows a feature succession modelled as a self-referent ASSOCIATION of a UML class that is a supertype for the various feature types that may be involved in succession relationships. Modelling in this way is necessary because there is no way to predict the order in which instances of these feature types might succeed each other.

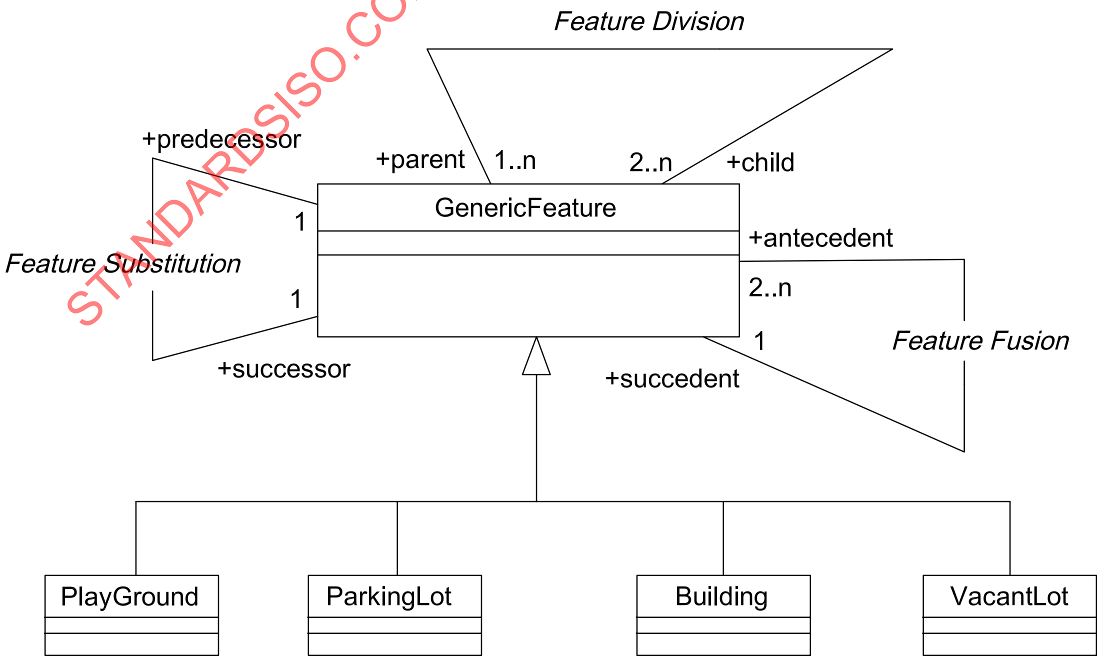


Figure 21 — Example of feature succession at the generic feature level

8.7 Spatial rules

8.7.1 General spatial rules

Rule:

1) The value domain of spatial attribute types shall be in accordance with the specifications given by ISO 19107, which provides conceptual schemas for describing the spatial characteristics of features and a set of spatial operators consistent with these schemas.

ISO 19107 covers vector data, which consists of geometric and topological primitives (spatial objects) used to construct objects that express the spatial characteristics of features.

Geometry provides the means for the quantitative description, by means of coordinates and mathematical functions, of the spatial characteristics of features, including dimension, position, size, shape, and orientation. The mathematical functions used for describing the geometry of an object depend on the coordinate system used to define the spatial position. Geometry is the only aspect of geographic information that changes when the information is transformed from one coordinate system to another.

Topology deals with the characteristics of geometric figures that remain invariant if the space is deformed elastically and continuously, for example, when geographic data are transformed from one coordinate system to another. Within the context of geographic information, topology is commonly used to describe the connectivity of a n-dimensional graph, a property that is invariant under continuous transformation of the graph.

Computational topology provides information about the connectivity of geometric primitives that may be derived from the underlying geometry. The most productive use of topology is to accelerate computational geometry. Geometric calculations such as containment (point-in-polygon), adjacency, boundary, and network tracking are computationally intensive. For this reason, combinatorial structures known as topological complexes, networks or graphs are often constructed for the purpose of optimizing computational geometry algorithms. These data and processing structures convert computational geometry algorithms into combinatorial ones.

8.7.2 Spatial attributes

Rules:

1) Spatial characteristics of a feature shall be described by one or more spatial attributes. In an application schema, a spatial attribute is a subtype of a feature attribute (see 7.4), and the taxonomy of its values is defined in the spatial schema, ISO 19107.

2) A spatial attribute shall be represented in an application schema in either of two ways:

— case 1: as an ATTRIBUTE of a UML CLASS that represents a feature, in which case the ATTRIBUTE shall take one of the spatial objects defined in the spatial schema, ISO 19107, as the data type for its value; or

— case 2: as a UML ASSOCIATION between the class that represents a feature and one of the spatial objects defined in the spatial schema, ISO 19107.

3) A spatial attribute shall take a spatial object as its value. Spatial objects are classified as geometric objects or topological objects, both of which are subclassed as primitives, complexes or aggregates (for geometric objects). Table 4 lists spatial objects that shall be used in an application schema as values for spatial attributes.

Table 4 — List of valid spatial objects for spatial attributes in an application schema

Geometric objects			Topological objects	
Geometric primitives	**Geometric complexes**	**Geometric aggregates**	**Topological primitives**	**Topological complexes**
GM_Point GM_Curve GM_Surface GM_Solid	GM_CompositePoint GM_CompositeCurve GM_CompositeSurface GM_CompositeSolid GM_Complex	GM_Aggregate GM_MultiPoint GM_MultiCurve GM_MultiSurface GM_MultiSolid	TP_Node TP_Edge TP_Face TP_Solid TP_DirectedNode TP_DirectedEdge TP_DirectedFace TP_DirectedSolid	TP_Complex
NOTE The table lists only the highest level classes of spatial objects. Subtypes of these may also be used.				

EXAMPLE Figure 22 shows an application schema named Parcels that is dependent on the spatial schema because it uses the UML-class GM_Surface defined in this schema. The feature Parcel has a spatial attribute (named area) which takes an instance of data type GM_Surface as its value. The two alternatives for representing a spatial attribute are shown.

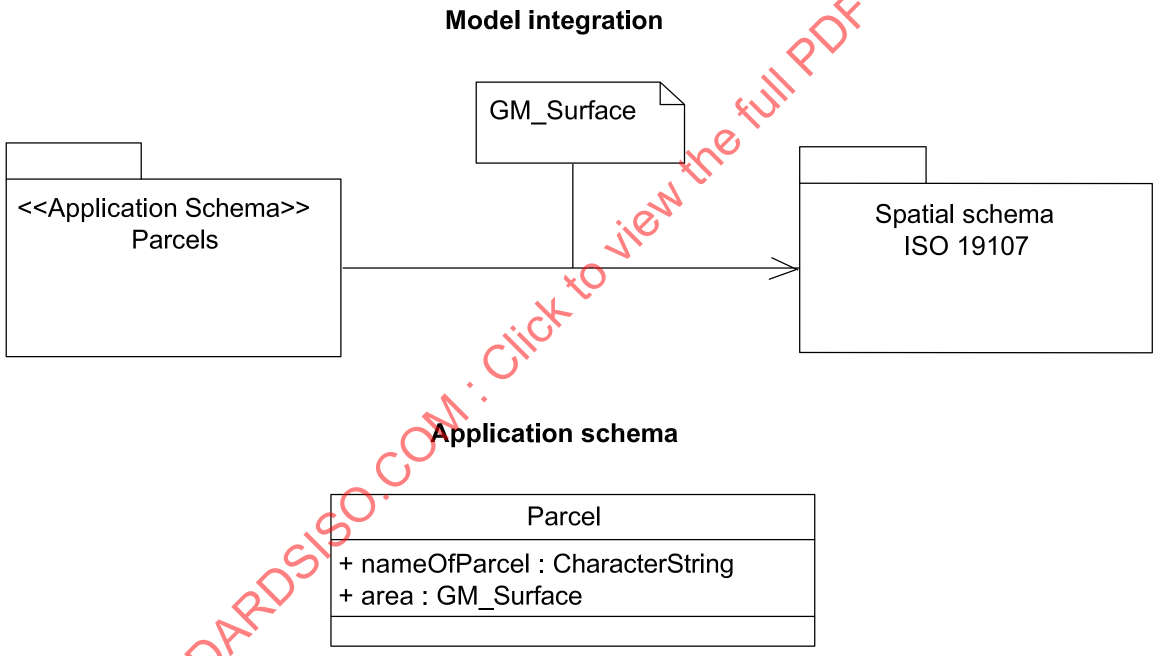


Figure 22 — Example of spatial attributes in UML

8.7.3 Use of geometric aggregates and spatial complexes to represent the values of spatial attributes of features

8.7.3.1 Introduction

The spatial configuration of many features can not be represented by a single geometric primitive. The types GM_Aggregate and GM_Complex support the representation of such features as collections of geometric objects.

8.7.3.2 Geometric aggregates

GM_Aggregates can be arbitrary collections of GM_Objects that have no required additional geometric structure. GM_Aggregates can contain other aggregates. GM_MultiPrimitives can be arbitrary collections of GM_Primitives but cannot contain other aggregates. The particular types GM_MultiPoint, GM_MultiCurve, GM_MultiSurface, and GM_MultiSolid are “type-safe” aggregates that only contain instances of subclasses of GM_Point, GM_Curve, GM_Surface, and GM_Solid, respectively. A direct instance of GM_Aggregate does not have this restriction.

Rules:

1) GM_Aggregate may be used as the value for a spatial attribute that represents a feature as an unstructured collection of GM_Objects.

— case 1: GM_Aggregate shall be used as the value for a spatial attribute that represents a feature as an unstructured collection of different types of GM_Object where there is no restriction on the types of contained objects.

EXAMPLE 1 A power line may be viewed as consisting of two types of geometric objects; the individual poles that support the wire, and the line of the wire itself. In this schema, a GM_Aggregate could be used.

— case 2: GM_MultiPoint, GM_MultiCurve, GM_MultiSurface, or GM_MultiSolid shall be used as the value of a spatial attribute that represents a feature as a set of geometric primitives of the same type.

EXAMPLE 2 An orchard may be viewed as a collection of trees. The spatial attribute of the feature “Orchard” should be a GM_MultiPoint, in which each GM_Point represents an individual tree. Although the orchard could be treated as a complex feature, and the trees as simple features, this is not necessary; the orchard could also be viewed as a simple feature represented by a set of GM_Points.

2) A particular composition or constraint on definition of the spatial attribute shall be implemented in the application schema by introducing a new CLASS that carries the user defined spatial configuration with all constraints as a subtype of GM_MultiPrimitive.

EXAMPLE 3 Figure 23 shows a user defined spatial aggregate named 3points1curve, which is data type of a spatial attribute in the application schema.

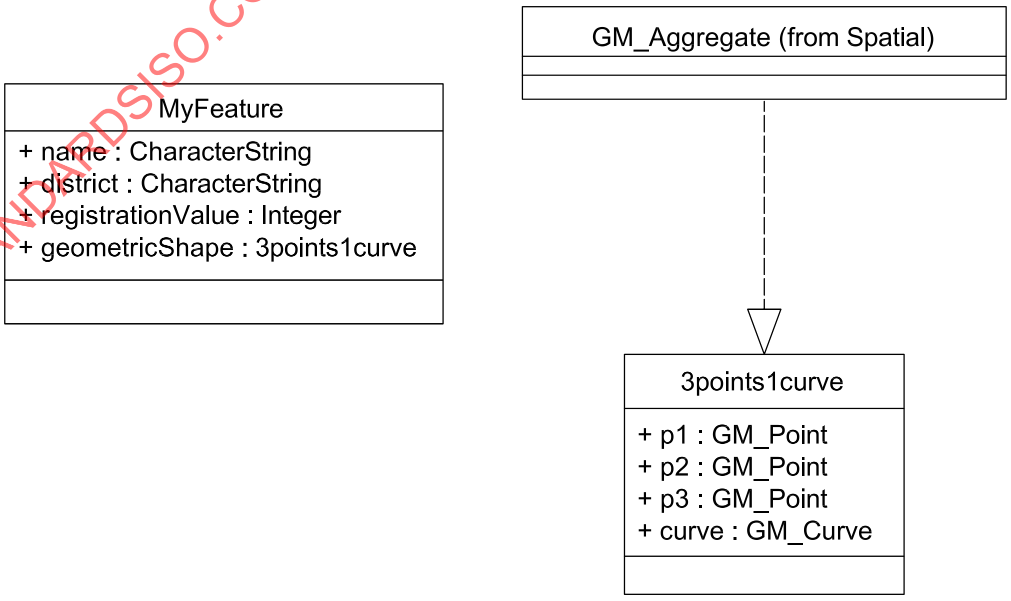

GM_Aggregate (from Spatial)
MyFeature
+ name : CharacterString
+ district : CharacterString
+ registrationValue : Integer
+ geometricShape : 3points1curve
3points1curve
+ p1 : GM_Point
+ p2 : GM_Point
+ p3 : GM_Point
+ curve : GM_Curve

Figure 23 — Example of spatial aggregate defined in application schema

8.7.3.3 Geometric complexes

Geometric complexes are used to represent the spatial characteristics of a feature as a set of connected geometric primitives. In addition, instances of GM_Complex allow geometric primitives to be shared by the spatial attributes of different features. There are no explicit links between the GM_Primitives in a GM_Complex; the connectivity between the GM_Primitives can be derived from the coordinate data.

Rules:

1) A GM_Complex shall be used as the value for a spatial attribute that represents a feature as a collection of connected GM_Objects, which are disjoint except at their boundaries. Subclasses of GM_Complex may be specified to constrain the structure of the GM_Complex used to represent a particular spatial configuration.

The primitives of a single dimension contained in a single complex must be disjoint except where they share a common boundary. A primitive within a complex may intersect another primitive of 1 lower dimension in that complex only where the first primitive contains the second within its boundary. A primitive within a complex may intersect another primitive of even lower dimension in that complex only where the first primitive completely contains the second within its interior and there is an explicit "Interior To" association instance for these primitives.

EXAMPLE 1 A drainage network could be represented as a GM_Complex composed of GM_Curves constrained so that the GM_Curves form a connected graph.

2) Features that share elements of their geometry shall be represented as GM_Complexes that are subcomplexes within a larger GM_Complex.

EXAMPLE 2 Each parcel in a cadastral data set has a boundary composed of GM_Curves. Each GM_Curve is shared by two parcel boundaries. The boundary of each parcel is a GM_Complex, and the set of all parcel boundaries is a larger GM_Complex (see Figure 24).

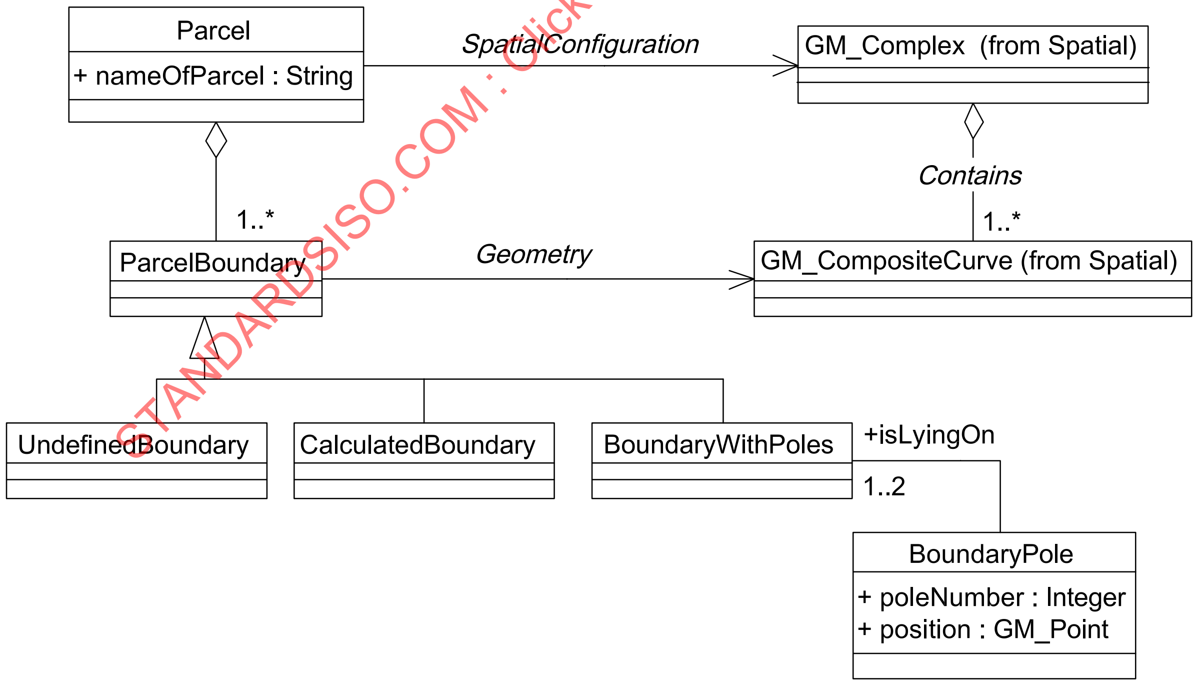


Figure 24 — Example of spatial complex defined in application schema

8.7.3.4 Geometric composites

A geometric composite is a geometric complex that has all the properties of a geometric primitive except that it is composed of smaller geometric primitives of the same kind. Geometric composites are used to represent complex features that are composed of smaller geometric objects that have the same kind of geometry.

Rule:

1) A GM_Composite shall be used to represent a complex feature that has the geometric properties of a geometric primitive.

NOTE A Road Network is composed of Roads, each of which may be composed of simple features representing different kinds of Road Elements (see Figure 25). The spatial attributes of the Road Elements could be GM_Curves. Each Road could be represented by a GM_CompositeCurve that contains the GM_Curves that represent the Road Elements of that Road. The Road Network could be represented by a GM_Complex that contains the set of GM_CompositeCurves that represent the Roads.

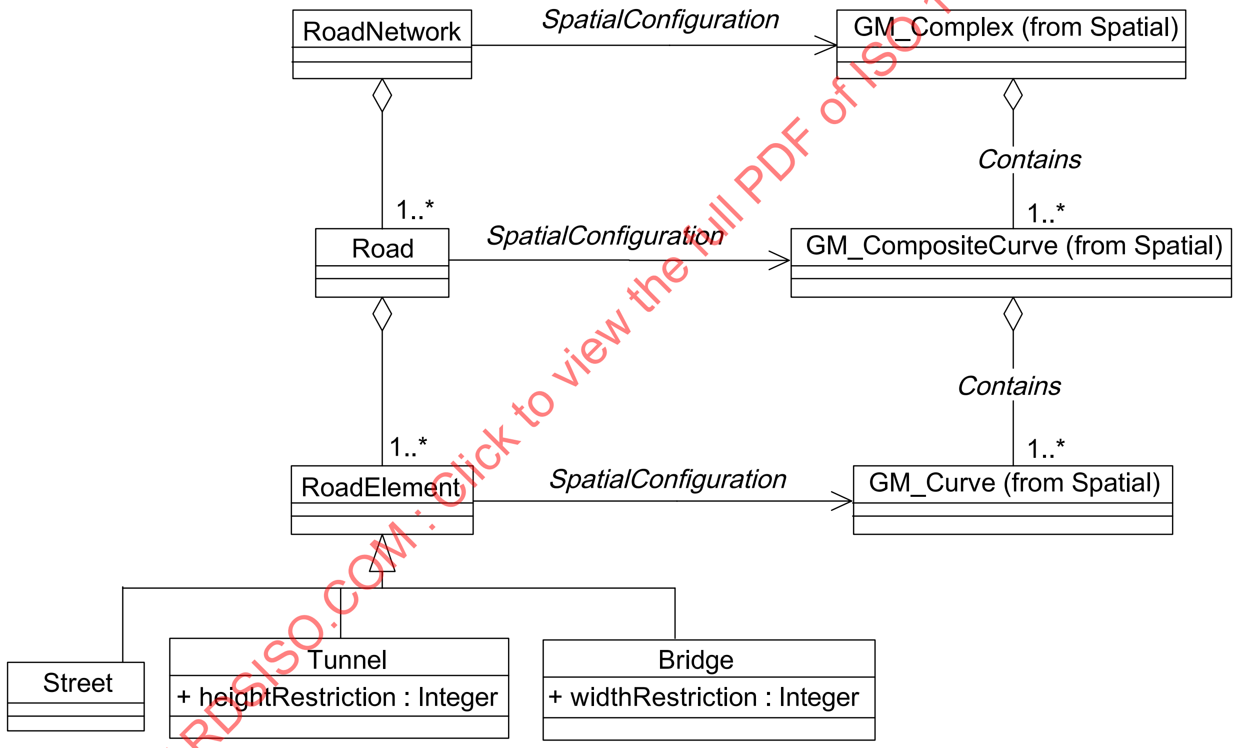


Figure 25 — Example of geometric composites defined in application schema

8.7.3.5 Global geometric complexes

Primitives may, and usually are, used simultaneously in multiple complexes. This means that additional structure may be placed on features whose geometry is structured as composites. This structure may be used to explicitly store the topological relations between features. The usual manner in which this happens is to create large (sometimes implicit) features, which are often called “themes”, “layers” or “maps”.

Rule:

1) Global features consisting of geometric objects that are used to represent the value of one or more attributes of one or more feature classes shall be represented as instances of GM_Complex in such a manner that the attributes so represented are subcomplexes of this object. This inclusion shall be specified by the application schema. Such global features shall be unique by name within each dataset.

Feature attributes that are required to be subcomplexes of such a global feature shall be specified by name as to which global feature they belong.

Like any other feature, one of these global themes may be specified as belonging to a larger theme. In this case, the base features would logically also belong to the larger theme. For example, a "Stream Theme" and a "Canal theme" may be involved in a "Hydrographic Theme". Nothing prevents a feature from belonging to many global features or "themes" through multiple attributes, but each attribute should, in most cases other than the one mentioned here, be in at most one theme.

NOTE If all roads in a dataset are to have their centrelines represented as a network (with or without additional topological structure), then a "Road Theme" feature may be defined by the application schema. It will have restrictions that 1) only one such feature exists, and 2) all road centrelines must be defined as composites of primitives also contained in the "Road Theme".

8.7.3.6 Topological complexes

Topological complexes carry explicit descriptions of the connections between the topological primitives of which they are composed. One use of topological complexes is to enable the application of methods from computational topology to geometric complexes. This requires implementation of the *Realization* relationship between a TP_Complex and a GM_Complex. Another use of topological complexes is to enable the description of the connectivity between features independently of their geometric configuration.

<u>**Rules**</u>:

1) A TP_Complex shall be used as a spatial attribute of a feature whenever the application requires the explicit representation of connectivity between the geometric primitives that represent the feature. The TP_Complex shall be linked in a *Realization* association to a GM_Complex that represents the geometric configuration of the feature. The TP_Complex may be subcomplex within a larger TP_Complex that represents multiple features.

2) TP_Complexes may be used to represent attributes of features in order to represent connectivity that is independent of the geometric configuration of the features. In this case, the TP_Complex is not a member of a *Realization* association.

EXAMPLE An electrical power distribution network could be represented as a TP_Complex in which TP_Nodes represent power plants, transformers, switches or other connection points, and TP_Edges represent the power lines.

NOTE In cases where a single class is defined for each valid combination of TP and GM classes, under the Realization association (through a multiple inheritance of the types), the relationship between TP_Objects and GM_Objects may be "to-self".

8.7.4 Spatial associations between features

According to the General Feature Model (see 7.3), a spatial association between features is a subtype of feature association. Some spatial associations — e.g. within, intersecting, touching — involve the topology of the features, and may be described implicitly by that topology. Others — e.g. east of, above — cannot be described by topology, and can only be described by explicit associations between the features.

NOTE ISO 19107 defines geometric and topological complexes that carry relationships between geometric and topological primitives, respectively. These may be relationships between the components of a spatial attribute of a single feature or relationships between the spatial attributes of different features. They describe how the geometry of one feature relates to the geometry of another feature, but provide no information about how the features relate to each other. On the instance level, the relationships between features are derivable by computation from the geometry.

<u>**Rules:**</u>

1) A spatial association between features may be described implicitly by representing the features as geometric or topological objects that are associated with each other in a geometric or topological complex.

2) A spatial association that is not described by an association between the underlying geometric or topological primitives shall be defined in an application schema as an ASSOCIATION between the features.

EXAMPLE 1 Figure 26 shows the spatial association implemented by mechanisms in the Spatial schema. The feature to feature topology is described by the topological primitives TP_Node and TP_Edge, which belong to a TP_Complex that represents the road network. These spatial objects implicitly carry the spatial association (see also the example in D.1). TP_Edge for Road represents the centreline of the road, and is associated in the application schema with a GM_Curve that is its geometric realization. In an application schema, some features may introduce another attribute carrying a second geometric description of the feature, for example, geometryDescribedBy, which in this example represents the surface of the road. The geometric object and the topological object of a feature may have different dimensions, but the geometric object, in this case, is not the geometric realization of the topological object. In this example, road and crossing features are described geometrically once as surfaces, and a second time as a curve and a point, respectively. They are also described topologically by an edge and a node, where the curve (but not the surface) is the geometric realization of the edge, and the point (but not the surface) is the geometric realization of the node.

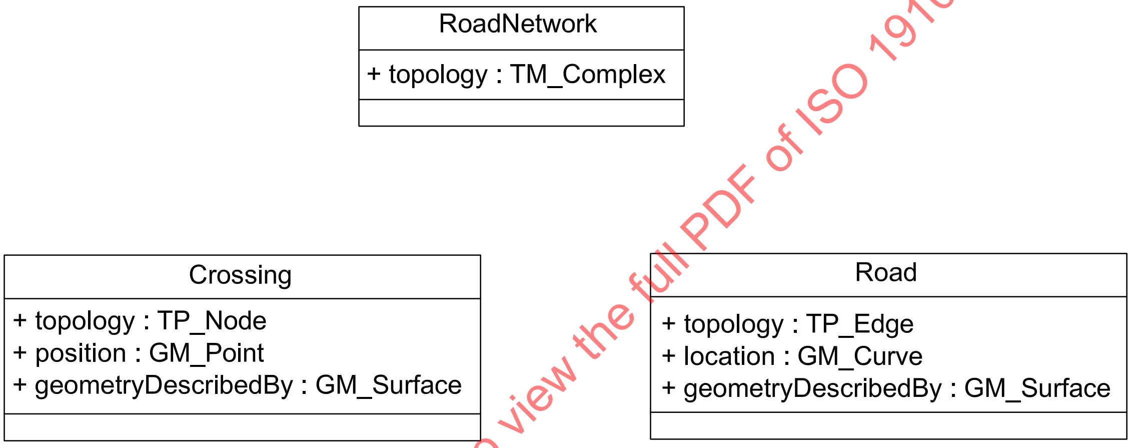


Figure 26 — Example of spatial associations implemented by use of spatial objects

EXAMPLE 2 Figure 27 shows a spatial association between road and crossing described explicitly by the association between the two classes.

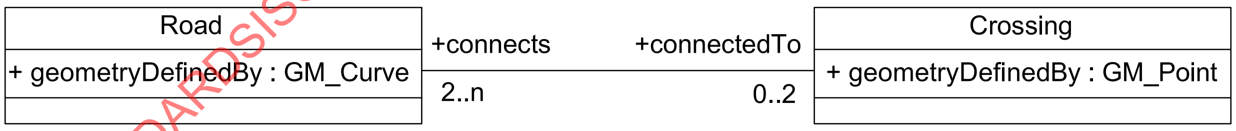

Road
+ geometryDefinedBy : GM_Curve
+connects
2..n
+connectedTo
0..2
Crossing
+ geometryDefinedBy : GM_Point

Figure 27 — Example of spatial associations implemented as associations in the application schema

8.7.5 Features sharing geometry

Different features can share, partly or completely, the same geometry when they appear to occupy the same position. To share a common geometry, spatial feature attributes must share one or more GM_Objects.

There are two ways to share geometry. Complete sharing occurs when two feature instances both take the same instance of a GM_Object as the value of a spatial attribute. This can be required, or precluded, by stating a constraint in the application schema. In the absence of such constraints, it may be done whenever necessary.

Rules:

1) An application schema may require instances of two or more feature types to share their geometry completely by including a constraint that the GM_Objects representing the features be equal.

2) An application schema may preclude instances of two or more feature types from sharing their geometry completely by including a constraint that the GM_Objects representing the features not be equal.

EXAMPLE 1 Figure 28 shows a transformer mounted on a pole represented by the GM_Point that represents the pole. Note that this application schema requires every transformer to share the geometry of a pole, but a pole does not have to share the geometry of a transformer.

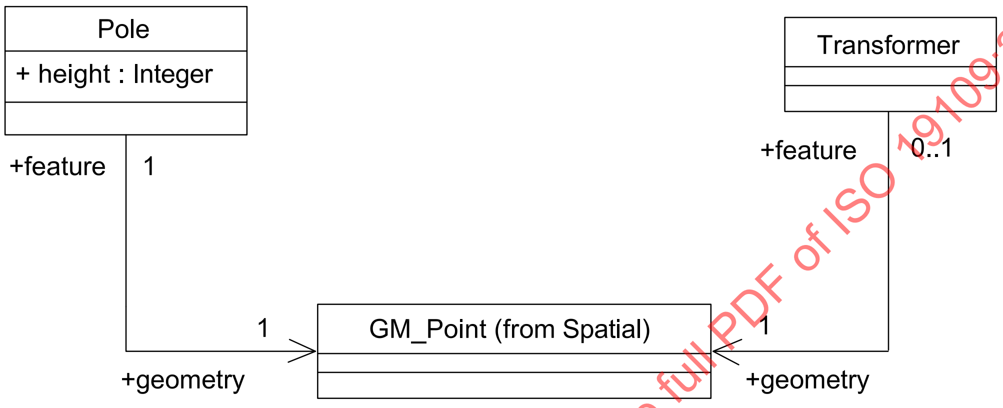


Figure 28 — Example of features sharing geometry

Partial sharing of geometry between feature instances requires the geometric objects that represent the spatial characteristics of the features to be modelled as elements in a geometric complex. See 8.7.3.3 and 8.7.3.4 for rules and examples.

EXAMPLE 2 Figure 29 shows partial sharing of geometry between features where one GM_Curve may depict a bridge and a road simultaneously.

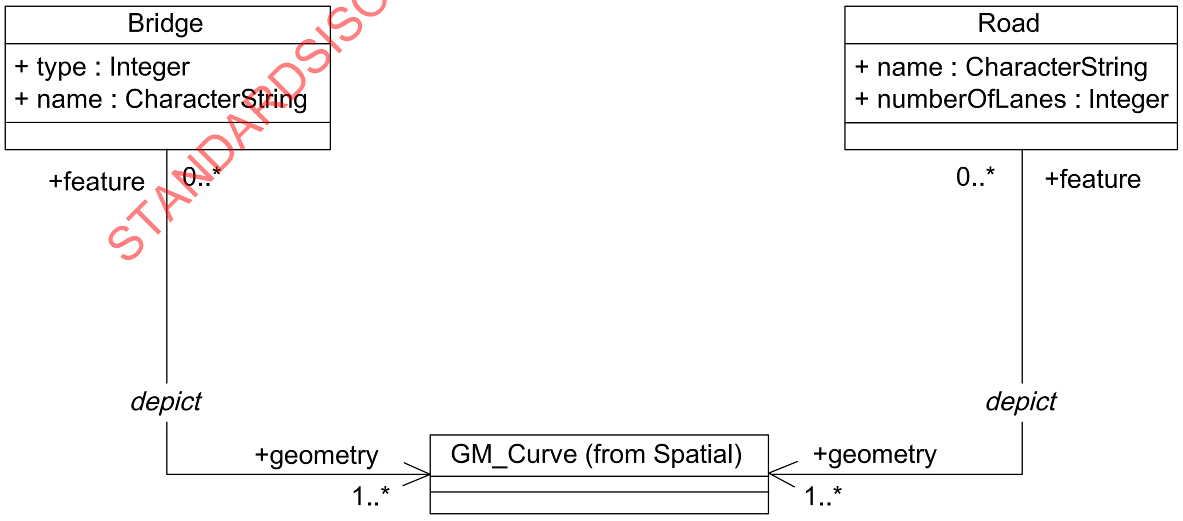


Figure 29 — Example of partial sharing of geometry between features

8.7.6 Point features, line features and area features

The traditional way of structuring geographic data does not distinguish between features and geometric primitives, but includes geometric information in the definition of a feature type. Thus, features are classified as point features, line features and area features because of the nature of the geometry. Large amounts of existing geographic data and functional standards are based on this way of structuring the geographic data.

This International Standard uses the geographic feature as the fundamental unit of geographic information. The geometry is one of several ways of describing the feature. Since a feature type is not defined on the basis of its geometry, several geometric descriptions may be associated to the same feature. It is recommended that point features, line features and area features are redefined in a generalized form as geographic features.

Rules:

1) A point feature shall take a GM_Point as the value of its spatial attribute.

2) A line feature shall take a GM_Curve or a GM_CompositeCurve as the value of its spatial attribute.

3) An area feature shall take a GM_Surface or a GM_CompositeSurface as the value of its spatial attribute.

8.7.7 Defining interpolation methods

In ISO 19107, a GM_Curve can contain any number of segments where every segment is one of the subtypes of GM_CurveSegment. Observe that different segments in one curve do not have to be of the same type. A GM_Surface can contain any number of patches where every patch is one of the subtypes of GM_SurfacePatch. Observe that different patches in one surface do not have to be of the same type. If, and only if, the existing set of subtypes of GM_CurveSegment and GM_SurfacePatch defined in Spatial Schema do not fulfil the requirements of an application schema, a subtype will have to be introduced. This is done by subtyping GM_CurveSegment or GM_SurfacePatch or one of the subtypes of these classes.

Rules:

1) If any existing subtype of GM_CurveSegment or GM_SurfacePatch fulfils the requirements of interpolation method, this subtype shall be used in the application schema.

2) If, and only if, none of the existing classes fulfils the requirements of the application schema, the application schema shall subtype GM_CurveSegment or GM_SurfacePatch (or one of the subtypes) with a CLASS that encapsulates the appropriate data and behaviour for the application-specific interpolation method.

8.7.8 Independent spatial complexes

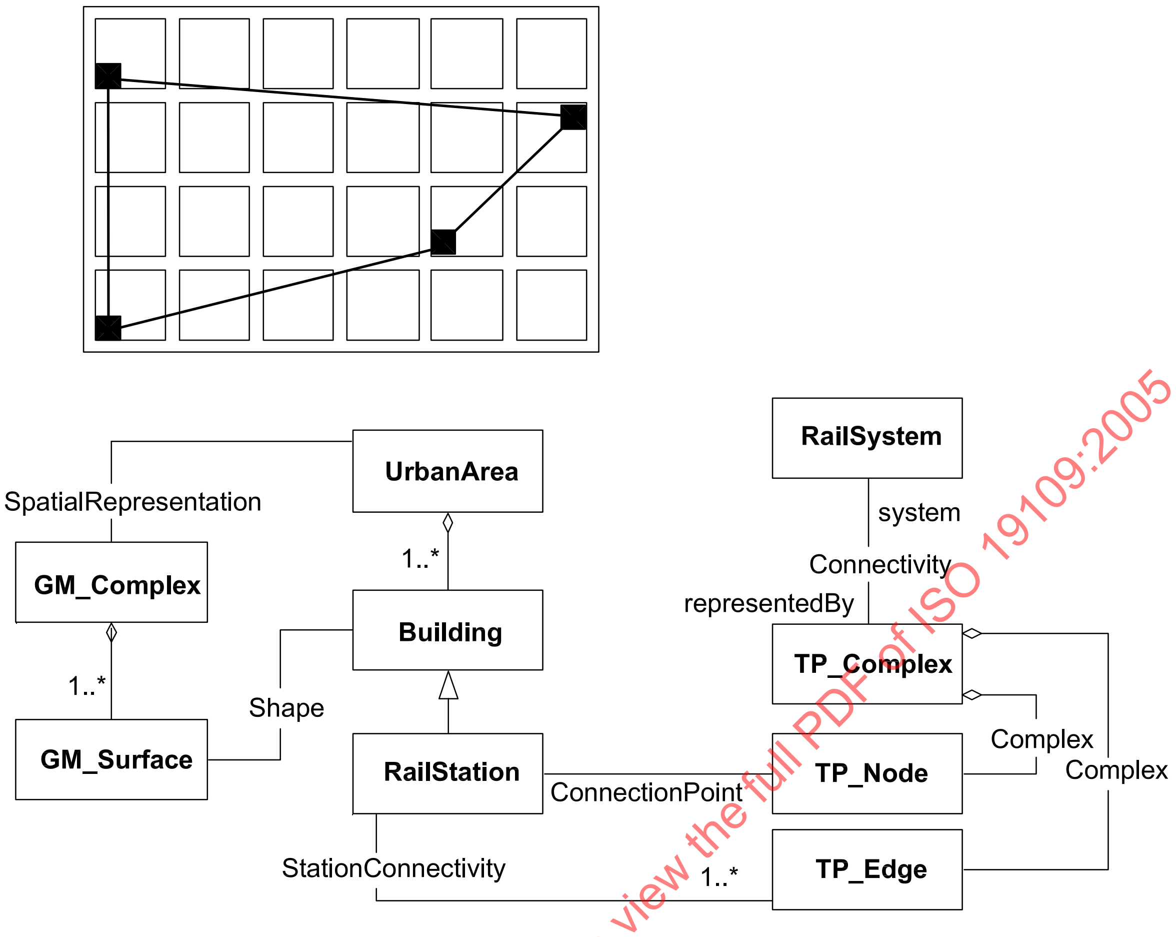
It is possible to include two or more topologically independent sets of features and spatial objects in the same application schema.

Rules:

1) Different spatial representations of the same feature are allowed, but they must belong to different complexes.

2) Topologically independent sets of spatial objects must belong to different TP_Complexes.

EXAMPLE 1 Consider an urban area with an underground rail system, such as that represented by Figure 30. The surface features could be represented as GM_Objects, such as the shaded squares in the figure, which represent stations for the underground rail system. The GM_Objects contain the actual shape and position information for the features. They might belong to a GM_Complex representing the urban area. At the same time, the connectivity of the rail system could be represented by a TP_Complex in which the stations are represented as TP_Nodes and the connections (not the actual routes) are represented as TP_Edges. The TP_Complex does not have a geometric realization. Although the example does not show it, the actual routes of the rail lines could be represented as GM_Curves within the GM_Complex, but these GM_Curves would not be geometric realizations of the TP_Edges that represent station connectivity.



Key

■	rail station as GM_Surface
•	rail station as TP_Node
▬	station connectivity as TP_Edge

Figure 30 — Example of independent spatial complexes in an application schema

EXAMPLE 2 Figure 31 illustrates an application schema that uses independent geometric complexes to represent a single road network. One GM_Complex describes the routing; it is composed of GM_Curves that represent the centrelines of the road segments. The other GM_Complex describes the paved area as an aggregation of GM_Surfaces.

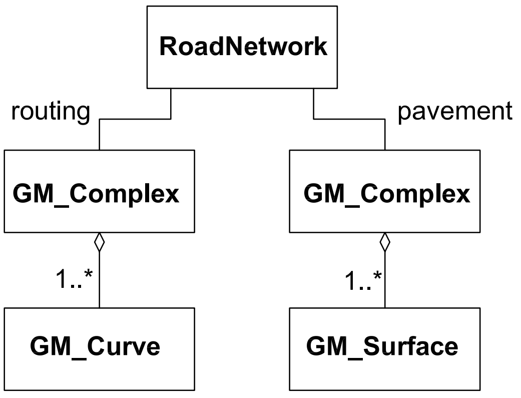


Figure 31 — Example of independent spatial complexes

8.8 Cataloguing rules

8.8.1 Introduction

A feature catalogue is a repository that describes real-world phenomena of significance to a particular universe of discourse. A feature cataloguing methodology provides the details about the organization of the data that represents these phenomena in categories, so that the resulting information is as unambiguous, comprehensible and useful as possible.

NOTE The feature catalogue provides the definition of geographic features at the type level, not the recording and representation of individual instances of each type. Therefore, it excludes spatial referencing, temporal referencing, and portrayal parameters (see also ISO 19107, ISO 19108 and ISO 19117). It also excludes data-capture criteria for feature instances.

8.8.2 Application schema based on a feature catalogue

An application schema may partly or completely be constructed by the definitions provided by a feature catalogue, for instance according to ISO 19110.

Since the conceptual model of a feature catalogue, according to this International Standard (see ISO 19110), is an implementation of GFM, the information in a feature catalogue can be used to construct an application schema by using the rules in subclause 8.3.1.

Rule:

1) Information in a feature catalogue compliant to ISO 19110 may be used to construct an application schema by using the rules in 8.3.1.

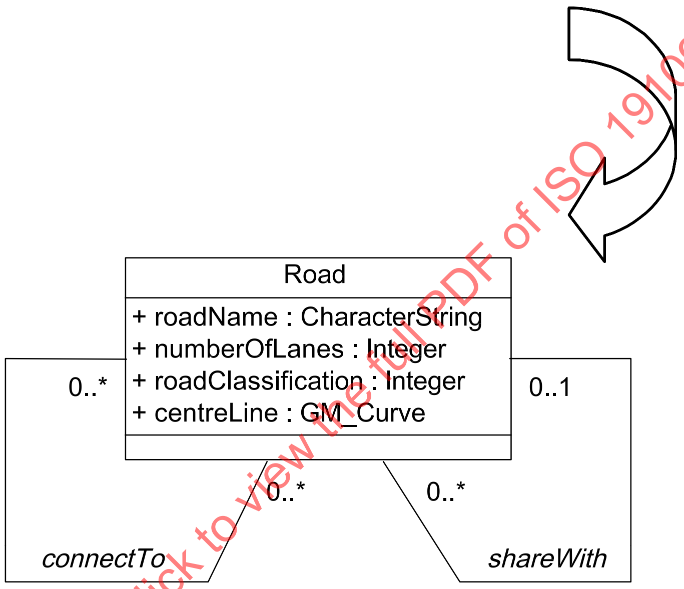
EXAMPLE Figure 32 shows the process of building an application schema based on a feature catalogue, where the schema is extended with a spatial attribute centreLine.

Feature catalogue

Feature Type

Feature Name	Road
Feature Definition	Open way for the movement of motor vehicles on land
Feature Code	1594
Attributes	Name, Number of Lanes, Road Classification
Associations	Road Connect To, Road Share With
etc….	

Application schema



Road
+ roadName : CharacterString
+ numberOfLanes : Integer
+ roadClassification : Integer
+ centreLine : GM_Curve
0..*
0..1
0..*
0..*
connectTo
shareWith

Figure 32 — Example of application schema based on feature catalogue

NOTE The feature catalogue is not part of the application schema, but it is part of the documentation of the application schema.

8.9 Spatial referencing using geographic identifiers

With spatial referencing using geographic identifiers, the position is found by a reference to a location. A geographic identifier is a label or code that identifies a location. A dataset that depends upon spatial referencing by geographic identifiers does not explicitly contain coordinates. A gazetteer may contain the geographic identifier and provide the corresponding coordinates and thus enables the data to be displayed or manipulated geographically. Figure 33 shows the concept of spatial referencing using geographic identifiers where coordinate information is provided.

Multiple gazetteers may exist for a spatial referencing system, containing different coordinate representations of the locations. The choice of gazetteer to be used will be application dependent. For example, for a census application, in one case a gazetteer of census area centroids may be sufficient, while for another case, a full coordinate description of the boundary area may be required.

An attribute of a feature (GF_LocationAttributeType) provides the link to the gazetteer where the location and its position described by coordinates, will be found.

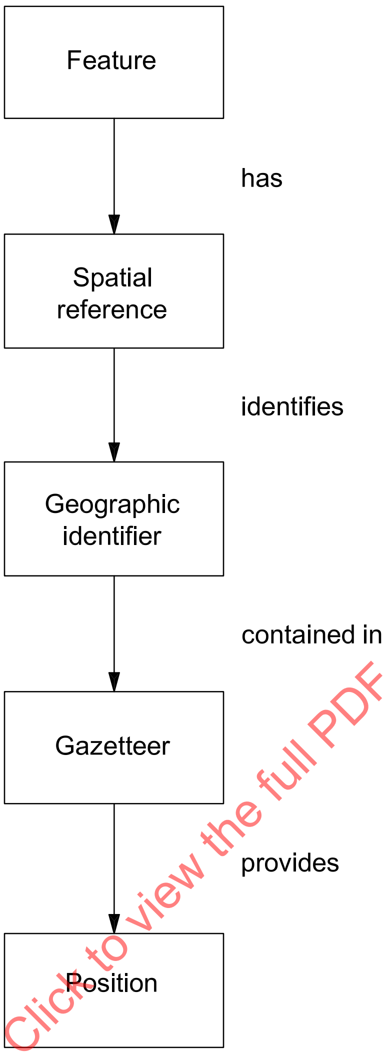


Figure 33 — Spatial referencing by geographic identifiers

Rules:

1) The value domain of attributes using spatial referencing by geographic identifiers shall be in accordance with the specifications given in ISO 19112.

2) The geographic identifier shall be referenced from the application schema by an attribute (an instance of GF_LocationAttributeType) which shall carry the value of the spatial reference.

3) An instance of GF_LocationAttributeType shall be represented in an application schema as an attribute of a UML CLASS that represents the feature, in which case the attribute shall take SI_LocationInstance defined in the Gazetteer Schema (see ISO 19112) as the data type for its value.

EXAMPLE Figure 34 shows the feature type Customer which is located by the attribute postDistrict.

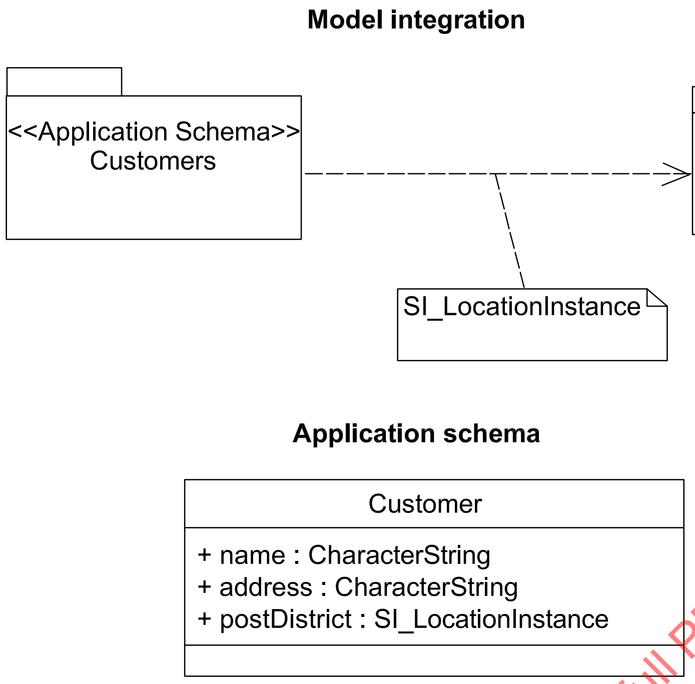


Figure 34 — Example of spatial referencing by geographic identifier

Annex A
(normative)

Abstract test suite

A.1 Feature types in an application schema

The test for feature types is as follows:

a)	test purpose	Verify that all feature types used in an application schema are both defined and implemented according to the rules in this International Standard;
b)	test method	Inspect all feature types in the schema to verify that they are defined as specified in A.2, and A.3;
c)	reference	ISO 19109:2005, Clauses 7 and 8;
d)	test type	capability.

A.2 Defining features

A.2.1 General

The general test for defining feature is as follows:

a)	test purpose	Verify that the elements of features in an application schema are defined according to the General Feature Model;
b)	test method	Inspect the application schema to verify that the rules for defining feature types (A.2.2), feature attribute types (A.2.3), feature association types (A.2.4) and feature operations (A.2.5) have been obeyed;
c)	reference	ISO 19109:2005, 7.3;
d)	test type	capability.

A.2.2 Defining feature types

The test for feature types is as follows:

a)	test purpose	Verify that feature types have been defined according to the General Feature Model;
b)	test method	Inspect the application schema to verify that the feature types have been defined according to the elements and rules specified in A.2.2 c);
c)	reference	ISO 19109:2005, 7.3.4;
d)	test type	capability.

A.2.3 Defining feature attribute types

The test for feature attributes is as follows:

a)	test purpose	Verify that feature attribute types have been defined according to the General Feature Model;
b)	test method	Inspect the application schema to verify that the feature attribute types have been defined according to the elements and rules specified in A.2.3 c);
c)	reference	ISO 19109:2005, 7.3.6 and 7.4;
d)	test type:	capability.

A.2.4 Defining feature association types

The test for feature associations is as follows:

a)	test purpose	Verify that feature association types have been defined according to the General Feature Model;
b)	test method	Inspect the application schema to verify that the feature association types have been defined according to the elements and rules specified in A.2.4 c);
c)	reference	ISO 19109:2005, 7.3.9 and 7.5;
d)	test type:	capability.

A.2.5 Defining feature operations

The test for feature operations is as follows:

a)	test purpose	Verify that feature operations have been defined according to the General Feature Model;
b)	test method	Inspect the application schema to verify that the feature operation types have been defined according to the elements and rules specified in A.2.5 c);
c)	reference	ISO 19109:2005, 7.3.8 and 7.6;
d)	test type:	capability.

A.3 Creating application schemas in UML

A.3.1 General

The general test for application schemas in UML is as follows:

a)	test purpose	Verify that an application schema is created according to rules specified in this International Standard;
b)	test method	Inspect the application schema to verify that the rules have been obeyed for identifying and integrating the application schema (A.3.2), implementing features in the application schema (A.3.3) and the use of the following conceptual schemas in the application schema: metadata (A.3.4), quality (A.3.5), temporal (A.3.6), spatial (A.3.7), cataloguing (A.3.8), geographic identifiers (A.3.9), if they are used;
c)	reference	ISO 19109:2005, Clause 8;
d)	test type	capability.

A.3.2 Identifying and integrating an application schema

The test for identifying and integrating an application schema is as follows:

a)	test purpose	Verify that the application schema has been identified and integrated with the relevant conceptual schemas in the other ISO 19100 series International Standards, according to rules specified in this International Standard;
b)	test method	Inspect the application schema to verify that the rules in A.3.2 c) have been obeyed for identifying and integrating the application schema;
c)	reference	ISO 19109:2005, 8.2;
d)	test type	capability.

A.3.3 Implementation of features in an application schema

The test for implementation of features in an application schema is as follows:

a)	test purpose	Verify that features have been implemented in the application schema according to rules specified in this International Standard;
b)	test method	Inspect the application schema to verify that the general rules in A.3.3 c) have been obeyed for implementing features, as well as the specialization, generalization and aggregation rules in A.3.3 c) where applicable;
c)	reference	ISO 19109:2005, 8.3.1;
d)	test type	capability.

A.3.4 Use of the metadata conceptual schema in an application schema

The test for use of the metadata conceptual schema in an application schema is as follows:

a)	test purpose	Verify that, if a metadata conceptual schema is implemented in the application, it has been implemented according to rules specified in this International Standard;
b)	test method	Inspect the application schema to verify that the rules in A.3.4 c) have been obeyed for implementing the metadata schemas;
c)	reference	ISO 19109:2005, 8.5;
d)	test type	capability.

A.3.5 Use of the quality conceptual schema in an application schema

The test for use of the quality conceptual schema in an application schema is as follows:

a)	test purpose	Verify that, if a quality conceptual schema is implemented in the application, it has been implemented according to rules specified in this International Standard.
b)	test method	Inspect the application schema to verify that the rules in A.3.5 c) have been obeyed for implementing the quality schemas;
c)	reference	ISO 19109:2005, 8.5.3;
d)	test type	capability.

A.3.6 Use of the temporal conceptual schema in an application schema

The test for use of the temporal conceptual schema in an application schema is as follows:

a)	test purpose	Verify that, if a temporal conceptual schema is implemented in the application, it has been implemented according to rules specified in this International Standard;
b)	test method	Inspect the application schema to verify that the rules in A.3.6 c) have been obeyed for implementing the temporal schemas;
c)	reference	ISO 19109:2005, 8.6;
d)	test type	capability.

A.3.7 Use of the spatial conceptual schema in an application schema

The test for use of the spatial conceptual schema in an application schema is as follows:

a)	test purpose	Verify that, if a spatial conceptual schema is implemented in the application, it has been implemented according to rules specified in this International Standard;
b)	test method	Inspect the application schema to verify that the rules in A.3.7 c) have been obeyed for implementing the spatial schemas;
c)	reference	ISO 19109:2005, 8.7;
d)	test type	capability.

A.3.8 Use of the feature catalogue conceptual schema in an application schema

The test for use of the feature catalogue conceptual schema in an application schema is as follows:

a)	test purpose	Verify that, if information from a feature catalogue has been implemented in the application schema, it has been implemented according to rules specified in this International Standard;
b)	test method	Inspect the application schema to verify that the rules in A.3.8 c) have been obeyed for implementing the feature types defined in the feature catalogue;
c)	reference	ISO 19109:2005, 8.8;
d)	test type	capability.

A.3.9 Use of geographic identifiers as in the gazetteer conceptual schema in an application schema

The test for use of geographic identifiers as in the gazetteer conceptual schema in an application schema is as follows:

a) test purpose Verify that, if referencing by geographic identifiers has been used in the application schema, it has been implemented according to rules specified in this International Standard;

b) test method Inspect the application schema to verify that the rules in A.3.9 c) have been obeyed for implementing the gazetteer schemas;

c) reference ISO 19109:2005, 8.9;

d) test type capability.