INTERNATIONAL STANDARD

ISO 19109

Second edition
2015-12-15

Geographic information — Rules for application schema

Information géographique — Règles de schéma d'application

Reference number
ISO 19109:2015(E)

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

Contents

Page

Foreword

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

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

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

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

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT) see the following URL: Foreword - Supplementary information

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

This second edition cancels and replaces the first edition (ISO 19109:2005).

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

2.1 General

This International Standard defines 12 conformance classes shown in Tables 1 to 12, matching the 12 requirements classes described in Clauses 7 and 8. Any application schema claiming conformance to any requirements class in this International Standard shall pass all of the tests listed in the corresponding conformance class, which are described in detail in the abstract test suites in Annex A. Each test relates to one or more specific requirements, which are explicitly indicated in the description of the test.

2.2 Meta-model

Table 1 — Meta-model conformance class

Conformance class	/conf/general
Requirements	/req/general (Clause 7, Table 15)
Tests	All tests in A.2

2.3 UML application schema

Table 2 — UML application schema conformance class

Conformance class	/conf/uml
Dependency	/conf/general (2.2)
Requirements	/req/uml (8.2, Table 16)
Tests	All tests in A.3

2.4 Profiling standard schema

Table 3 — Profiling standard schema conformance class

Conformance class	/conf/profile
Dependency	/conf/uml (2.3)
Requirements	/req/profile (8.3, Table 19)
Tests	All tests in A.4

2.5 Metadata

Table 4 — Metadata conformance class

Conformance class	/conf/metadata
Dependency	/conf/uml (2.3)
Requirements	/req/metadata (8.4, Table 20)
Tests	All tests in A.5

2.6 Quality

Table 5 — Quality conformance class

Conformance class	/conf/quality
Dependency	/conf/uml (2.3)
Requirements	/req/quality (8.5, Table 21)
Tests	All tests in A.6

2.7 Temporal

Table 6 — Temporal conformance class

Conformance class	/conf/temporal
Dependency	/conf/uml (2.3)
Requirements	/req/temporal (8.6, Table 23)
Tests	All tests in A.7

2.8 Spatial

Table 7 — Spatial conformance class

Conformance class	/conf/spatial
Dependency	/conf/uml (2.3)
Requirements	/req/spatial (8.7, Table 25)
Tests	All tests in A.8

2.9 Coverages

Table 8 — Coverages conformance class

Conformance class	/conf/coverage
Dependency	/conf/uml (2.3)
Requirements	/req/coverage (8.8, Table 27)
Tests	All tests in A.9

2.10 Observations

Table 9 — Observations conformance class

Conformance class	/conf/observation
Dependency	/conf/uml (2.3)
Requirements	/req/observation (8.9, Table 29)
Tests	All tests in A.10

2.11 Spatial referencing by identifiers

Table 10 — Spatial referencing by identifiers conformance class

Conformance class	/conf/identifier
Dependency	/conf/uml (2.3)
Requirements	/req/identifier (8.10, Table 30)
Tests	All tests in A.11

2.12 Code list

Table 11 — Code list conformance class

Conformance class	/conf/codeList
Dependency	/conf/uml (2.3)
Requirements	/req/codeList (8.11, Table 31)
Tests	All tests in A.12

2.13 Multi-lingual support

Table 12 — Multi-lingual support conformance class

Conformance class	/conf/multi-lingual
Dependency	/conf/uml (2.3)
Requirements	/req/multi-lingual (8.12, Table 32)
Tests	All tests in A.13

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.

IETF RFC 5646 (2009), *Tags for Identifying Languages, available at <https://www.rfc-editor.org/info/rfc5646>*

ISO 19103:2015, *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 19115-1:2014, *Geographic information — Metadata — Part 1: Fundamentals*

ISO 19115-2:2009, *Geographic information — Metadata — Part 2: Extensions for imagery and gridded data*

ISO 19123:2005, *Geographic information — Schema for coverage geometry and functions*

ISO 19156:2011, *Geographic information — Observations and measurements*

ISO 19157:2013, *Geographic information — Data quality*

ISO/IEC 19505-2:2012, *Information technology — Object Management Group Unified Modeling Language (OMG UML) — Part 2: Superstructure*

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

[SOURCE: ISO 19101-1:2014, 4.1.1]

4.2
application schema
conceptual schema (4.5) for data required by one or more *applications* (4.1)

[SOURCE: ISO 19101-1:2014, 4.1.2]

4.3
complex feature
feature (4.9) composed of other features

4.4
conceptual model
model (4.15) that defines concepts of a *universe of discourse* (4.19)

[SOURCE: ISO 19101-1:2014, 4.1.5]

4.5
conceptual schema
formal description of a *conceptual model* (4.4)

[SOURCE: ISO 19101-1:2014, 4.1.6]

4.6
coverage
feature (4.9) that acts as a function to return *values* (4.20) from its range for any direct position within its spatial, temporal or spatiotemporal *domain* (4.8)

[SOURCE: ISO 19123:2005, 4.1.7]

4.7
dataset
identifiable collection of data

[SOURCE: ISO 19115-1:2014, 4.3]

4.8
domain
well-defined set

Note 1 to entry: 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.

4.9
feature
abstraction of real-world phenomena

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

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

4.10
feature association
relationship that links instances of one *feature* (4.9) type with instances of the same or a different feature type

[SOURCE: ISO 19110:2005, 4.2]

4.11
feature attribute
characteristic of a *feature* (4.9)

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

Note 2 to entry: A feature attribute type has a name, a data type and a *domain* (4.8) associated with it. A feature attribute instance has an attribute *value* (4.20) taken from the domain of the feature attribute type.

[SOURCE: ISO 19101-1:2014, 4.1.12, modified – Notes and examples in 19101-1 have been omitted and other notes have been added.]

4.12
feature operation
operation that every instance of a *feature* (4.9) 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.

[SOURCE: ISO 19110:2005, 4.5, modified - The note given in ISO 19110:2005 for this entry has been omitted. A second example has been added.]

4.13
geographic data
data with implicit or explicit reference to a location relative to the Earth

Note 1 to entry: Geographic information is also used as a term for information concerning phenomena implicitly or explicitly associated with a location relative to the Earth.

4.14
metadata
information about a resource

[SOURCE: ISO 19115-1:2014, 4.10]

4.15
model
abstraction of some aspects of reality

4.16
observation
act of measuring or otherwise determining the *value* (4.20) of a *property* (4.17)

[SOURCE: ISO 19156:2011, 4.11]

4.17
property
facet or attribute of an object referenced by a name

[SOURCE: ISO 19143:2010, 4.21]

4.18
quality
degree to which a set of inherent characteristics fulfils requirements

[SOURCE: ISO 9000:2005, 3.1.1]

4.19
universe of discourse
view of the real or hypothetical world that includes everything of interest

[SOURCE: ISO 19101-1:2014, 4.1.38]

4.20
value
element of a type *domain* (4.8)

[SOURCE: ISO/IEC 19501:2005, 0000_5]

5 Presentation and abbreviations

5.1 Presentation

5.1.1 General

This International Standard describes how to create an application schema that integrates conceptual schemas defined in the ISO geographic information standards. In addition to stating the rules for creating application schemas, this International Standard provides guidance through examples.

5.1.2 Conformance class

Conformance to this International Standard is possible at a number of levels, specified by conformance classes (Clause 2). Each conformance class is summarized using the template shown as Table 13.

Table 13 — Conformance class template

Conformance class	/conf/{classM}
Dependency	[identifier for another conformance class]
Requirements	/req/{classA}
Tests	[reference to clause(s) containing tests]

All tests in a class must be passed, so dependencies are recorded with respect to other conformance classes rather than individual tests . Each conformance class tests conformance to a set of requirements packaged in a requirements class (Clauses 7 and 8).

5.1.3 Requirements class

Each normative statement (requirement or recommendation) in this International Standard is a member of a requirements class. In this International Standard each requirements class is described in a discrete clause or subclause, and summarized using the template shown as Table 14.

Table 14 — Requirements class template

Requirements class	/req/{classM}
Target type	[artefact or technology type]
Dependency	[identifier for another requirements class]
Requirement	/req/{classM}/{reqN}
Recommendation	/rec/{classM}/{recO}
Requirement	/req/{classM}/{reqP}
Requirement/Recommendation	[repeat as necessary]

All requirements in a class must be satisfied, so the requirements class is the unit of re-use and dependency, rather than individual requirements. Hence, the value of a Dependency requirement is another requirements class.

5.1.4 Rules

All rules are normative, and each rule is presented using the following template:

/re(c|q)/[classM]/[reqN]	[Normative statement]

where /re(c|q)/[classM]/[reqN] identifies the requirement or recommendation. The use of this layout convention allows the normative provisions of this International Standard to be easily located by implementers.

5.1.5 Identifiers

Each requirements class, requirement and recommendation has an identifier in the form of a path or partial URI. The identifier supports cross-referencing of class membership, dependencies, and links from each conformance test to requirements tested. The identifier can be appended to a URI that identifies the standard as a whole in order to construct an absolute URI which identifies the requirements class, requirement or recommendation. For example, a URI scheme following the approach described in 2.8 of [IETF RFC 5141] would result in

— http://standards.iso.org/iso/19109/ed-2

Hence, the absolute URI for each requirements class would have the form

— http://standards.iso.org/iso/19109/ed-2/req/[classM]

and the absolute URI for each requirement or recommendation would have the form

— http://standards.iso.org/iso/19109/ed-2/req/[classM]/[reqN]

5.1.6 Conceptual schemas

Conceptual schemas in the normative part of this International Standard are presented in the Unified Modeling Language (UML) in conformance with ISO 19103. UML diagrams are presented in compliance with ISO/IEC 19505-2.

5.1.7 Descriptions of concepts

Concepts from UML are presented in all capitals – e.g. CLASS, PACKAGE, ROLE, ATTRIBUTE, ASSOCIATION.

5.2 Abbreviations

CSL Conceptual Schema Language

GFM General Feature Model

OCL Object Constraint Language

OWL Web Ontology Language

UML Unified Modeling Language

5.3 Package abbreviations

Concepts from schemas defined in some other International Standards are designated with names that start with two-letter codes as follow:

CV ISO 19123:2005 Schema for Coverage Geometry and Functions

DQ ISO 19157:2013 Data Quality

GM ISO 19107:2003 Spatial Schema

LI ISO 19115-2:2009 Metadata extensions for imagery

MD ISO 19115-1:2014 Metadata fundamentals

MI ISO 19115-2:2009 Metadata extensions for imagery

OM ISO 19156:2011 Observations and Measurements

SF ISO 19156:2011 Observations and Measurements

SI ISO 19112:2003 Spatial referencing by geographic identifiers

TM ISO 19108:2002 Temporal Schema, Temporal Objects

TP ISO 19108:2002 Temporal Schema, Temporal Topology

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 shall define

— the content and structure of data;

and may define

— operations for manipulating and processing data;

— constraints to ensure the integrity of the data.

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;

— 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 any 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;

— 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 a UML profile that defines the concepts required to describe types of features. Feature type definitions may be documented in feature catalogues (ISO 19110). Such definitions may be used in an application schema. Other ISO geographic information standards 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, ISO 19136 and ISO/TS 19139 define 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.

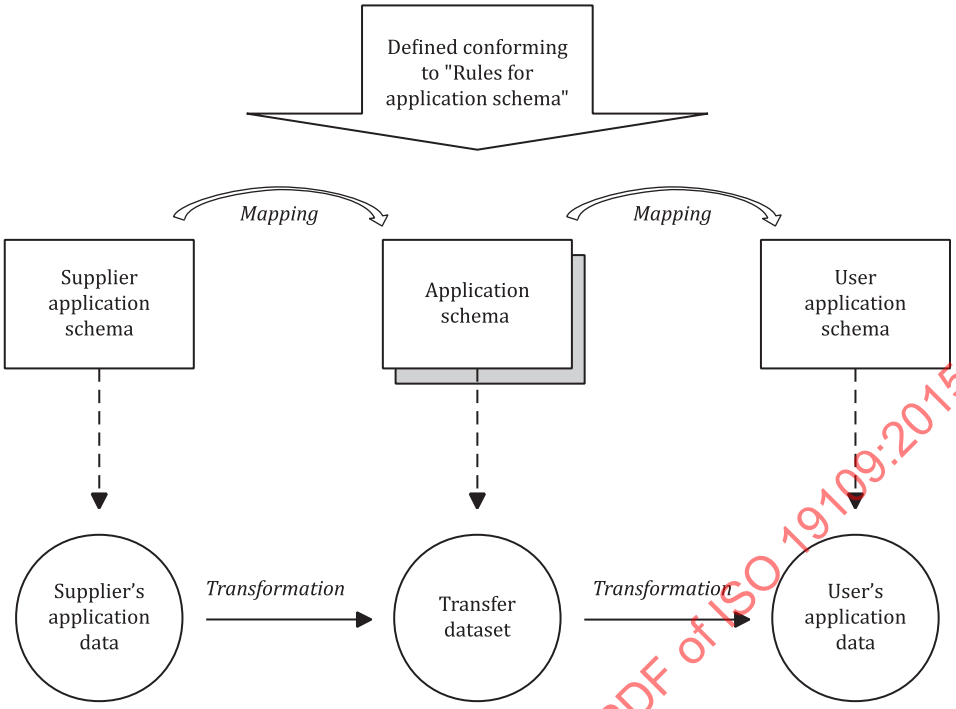


Figure 1 — Data interchange by transfer

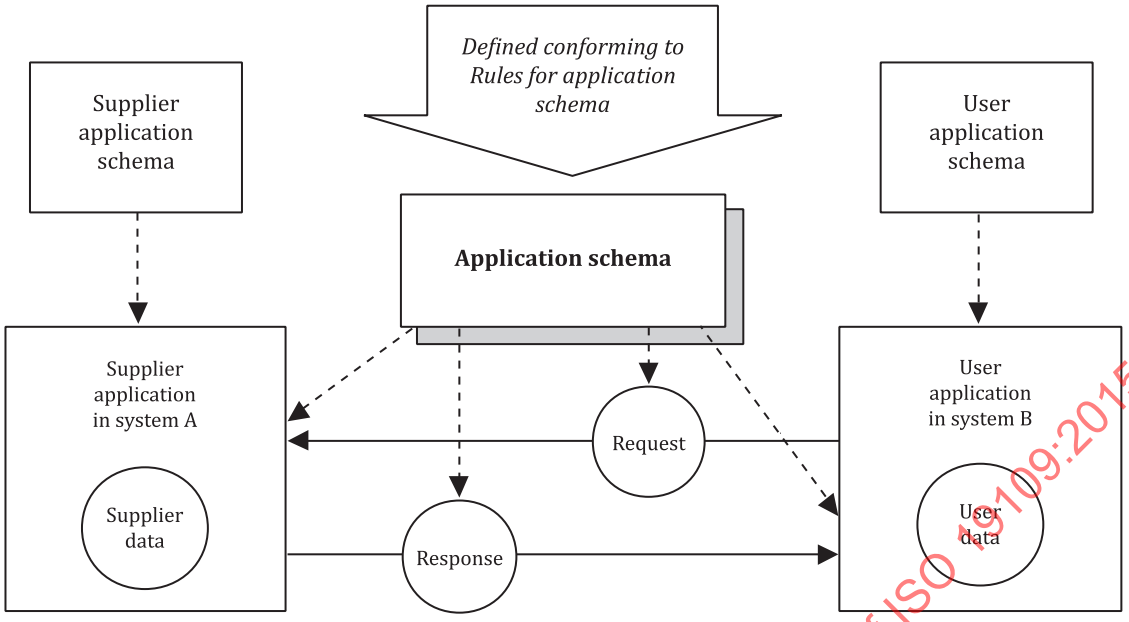
First, the user and the supplier must agree on creating an application schema for the data being exchanged in accordance with this International Standard. In order to facilitate the transfer of data, the application schema is developed using the application schemas from the user and the supplier. One 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.



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 General

This clause describes general rules for formalizing an application schema, independent of any specific conceptual schema language. The General Feature Model is the meta-model for application schemas, and the definition of the General Feature Model is part of the general rules. The rules comprise a requirements class summarized in Table 15.

Table 15 — Requirements class covering general requirements and the meta-model for application schemas

Requirements class	/req/general
Target type	Application schema
Requirement	/req/general/csl
Requirement	/req/general/integration
Requirement	/req/general/feature
Recommendation	/rec/general/property-name
Requirement	/req/general/attribute
Requirement	/req/general/operation
Requirement	/req/general/association-role
Requirement	/req/general/value-assignment
Requirement	/req/general/feature-association
Requirement	/req/general/inheritance
Recommendation	/rec/general/constraint

7.2 Features, Coverages and Properties

7.2.1 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.

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, and 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.

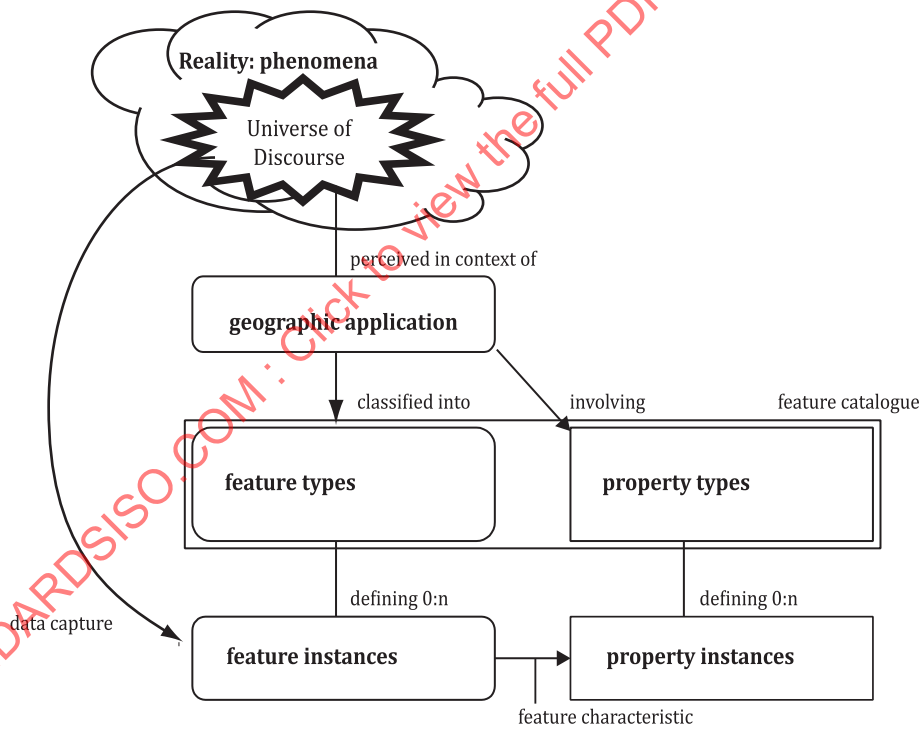


Figure 3 — The process from universe of discourse to data

7.2.2 Coverages

Many aspects of the real-world may be represented as features whose properties are single-valued and static. These conventional features provide a model of the world in terms of discrete objects located in it. However, in some applications it is more useful to use a model focussing on the variation of property values in space and time, formalized as coverages. Users of geographic information may utilize both viewpoints. While coverages are themselves strictly features as well, it is common to contrast coverages and non-coverage features when discussing the functionality provided by each viewpoint. In the following discussion the name 'feature' refers to non-coverage features.

The feature and coverage representations may be related in several ways:

— signal processing to find and characterize features: signals in coverages may provide evidence for the existence, location and type of features, detected through modelling and interpretation;

EXAMPLE 1 Patterns of colour or other radiance bands within a remotely-sensed image may be used to infer the existence of specific objects or features on the ground.

EXAMPLE 2 Signals in a geophysical borehole log may be used to infer the presence of particular rock-units at underground locations.

— coverage-typed feature properties: feature properties whose value vary within the scope of a feature may be described as coverages whose domain extent is the geometry of the feature;

EXAMPLE 3 The variation of concentration of a particular ore-mineral within a mine may be described as a spatial function or coverage within the spatial limits of the mine.

— features sample a coverage: the values of a common property of a set of features provide a discrete sampling of a coverage, whose range type is the property, and whose domain is the aggregate geometry of the set of features.

EXAMPLE 4 The temperature at a set of weather stations may be compiled to show the spatial variation of temperature across the region where the stations are located.

A constraint in the latter two cases is that a property-type from a feature catalogue is the range-type of a coverage description in the same universe of discourse.

The case of features having property values that vary within the scope of the feature can be described using the general feature model (7.5.8).

While the coverage model is described in detail in ISO 19123, an application schema may include both feature- and coverage-types.

NOTE The feature and coverage viewpoints are related to (though not identical with) the so-called 'vector' and 'raster' approaches from traditional GIS implementations.

7.2.3 Properties and observations

Property values are associated with features and coverages. In the case of features, a property value is associated with a classified object. In the case of coverages, a property value is associated with a position in the domain.

Property values can be assigned in a number ways, such as through assertion by a competent authority, by inheritance from a parent feature or type, by derivation from more primitive properties, or by observation. In the latter case the value assigned is an estimate, with an error resulting from the observation procedure.

NOTE Following ISO 19156, 'observation' denotes the generalized case of an event using a procedure to generate the estimate. The procedure may involve a sensor, but may also be a computational or classification process.

The result of an observation is a property value that is assigned to its feature of interest. Where the feature property has a constant value, an associated observation result has a static value. Where the feature property varies with position, the result of an associated observation is a spatial function or coverage.

7.3 Features and the application schema

This International Standard supports the definition of feature types and coverages 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. An application schema is formalized using a Conceptual Schema Language.

/req/general/csl	An application schema shall be expressed in a conceptual schema language. It shall describe the structure and content of the dataset that represents a universe of discourse.

When creating an application schema, the concepts of the General Feature Model (GFM; see 7.4) are mapped to the concepts of the chosen conceptual schema language. For UML, these rules are described in Clause 8.

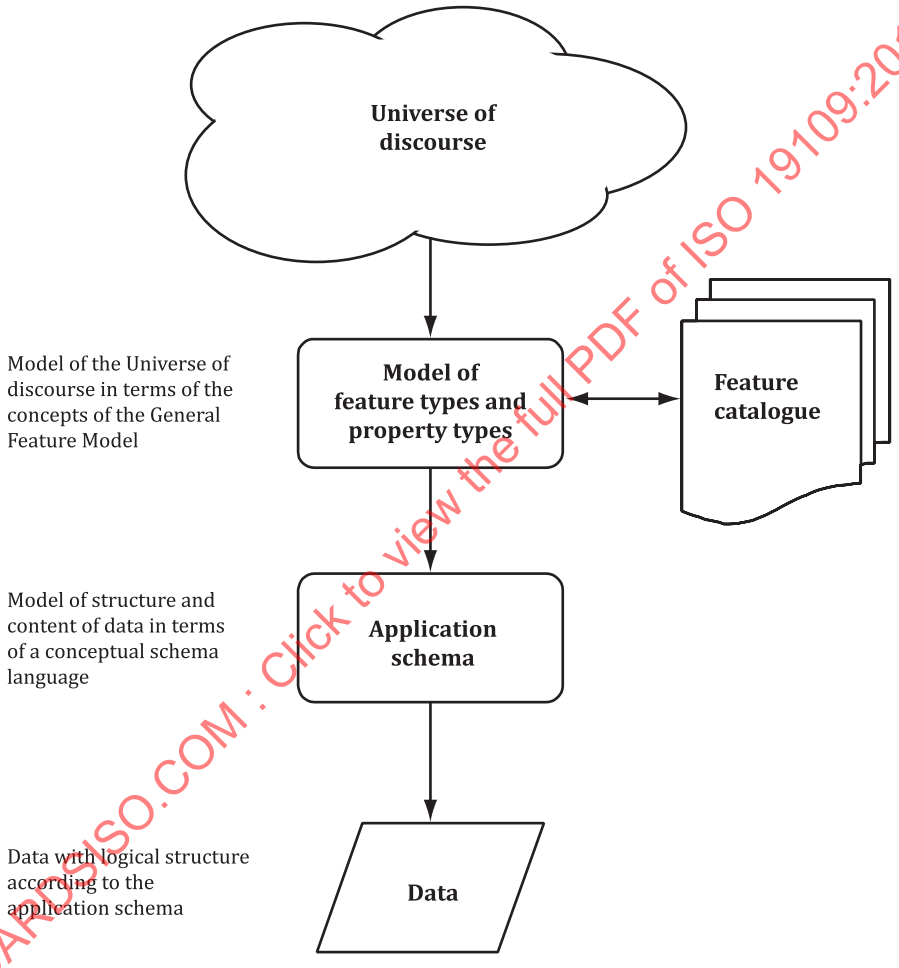


Figure 4 — From reality to geographic data

Use of definitions from existing feature catalogues or feature concept dictionaries can reduce the costs of data acquisition, allowing developers to use existing data, and simplify the process of developing the application schema. Dependencies are clearly recorded.

/req/general/integration	The developer of an application schema may use definitions from feature catalogues that already exist. Where an application schema integrates components from another schema, the dependencies shall be recorded explicitly using mechanisms provided by the conceptual schema language.

7.4 The General Feature Model

7.4.1 Introduction

Concepts used to define features and how these concepts are related are identified and described in 7.4. 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.

7.5 to 7.7 depict how different aspects of the properties of a feature are managed. 7.5 describes different aspects of attributes and 7.6 describes different aspects of relationships. 7.7 describes the concept of constraints.

7.4.2 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, which is UML in accordance with ISO 19103.

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

Feature types can be documented in feature catalogues. The GFM may serve as the conceptual model of the structure of a feature catalogue, but the feature catalogue has additional concepts for documenting feature types. There is a Feature Catalogue Model (FCM) that realizes the GFM concepts and adds 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.

The GFM specifies the requirements for the classification of features, but is not a CSL. We have to use an existing CSL to define the application schema. Within the ISO geographic information standards, UML is used to define standard schemas. As we want to integrate standard schemas into our application schemas, it is convenient to also express the application schema in UML. This International Standard defines the main rules for mapping GFM concepts into UML. This may be done for other CSLs as well. As UML provides a general mechanism for classifications of any kind, it has to be profiled to be used as a basis for the classification of features. This profile is UML as specified in ISO 19103 and augmented in this International Standard and is summarized in 8.2.2.

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.

Hence, the GFM is a metamodel for definition of features that is also used to define the structure of feature catalogues. The chosen CSL (i.e. the UML profile as restricted by ISO 19103 and augmented in this International Standard) is the metamodel for an application schema.

7.4.3 The main structure of the GFM

Figure 5 shows the concepts used to define types of features. 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.

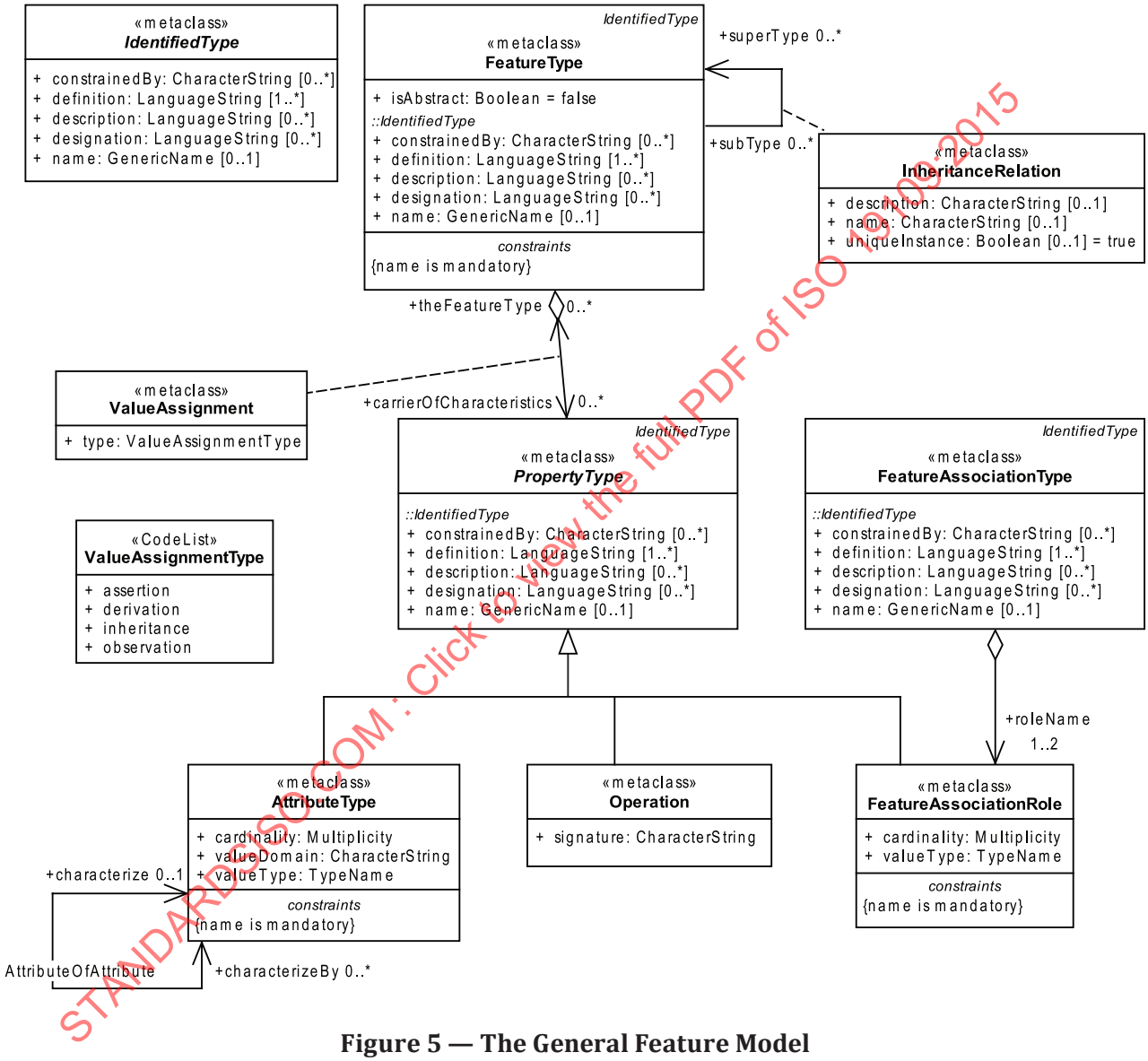


Figure 5 — The General Feature Model

NOTE 1 Earlier versions of this standard prefixed most class names with 'GF_'. That convention is no longer used.

NOTE 2 The types CharacterString, GenericName, LanguageString, LocalName, TypeName, are from ISO 19103.

7.4.4 IdentifiedType

IdentifiedType is a metaclass from which the other key classes in the General Feature Model inherit some common identification and description attributes. IdentifiedType is the supertype of FeatureType, PropertyType and FeatureAssociationType.

— name

  name of the element. Optional in general, but required on some specializations as indicated below. The type GenericName [ISO 19103] allows the namespace to be either explicit or implicit.

The CSL may constrain the scope and namespace of model elements. In UML the name of a class is scoped to the package, and the name of an attribute or role is scoped to a class, so when using UML as the CSL [Clause 8] the Application Schema provides the namespace for feature types, and the feature type provides the namespace for properties. In OWL the namespace for both classes and properties is usually but not necessarily provided by the ontology. When using OWL as the CSL [ISO 19150-2] classes will typically be scoped to the ontology or application schema, but property types may be scoped to a class, to the application schema, or to an external ontology or catalogue.

— definition

  concise definition of the element. One definition is mandatory. Additional definitions can be provided in multiple languages if required.

— designation

  natural language designator for the element to complement the **name**. Optional, with multiple designations allowed to support different languages.

— description

  description of the element, including information beyond that required for concise definition but which may assist in understanding its scope and application. Optional, with multiple descriptions allowed to support different languages.

— constrainedBy

  description of a constraint made on the IdentifiedType or specified on properties within a type. Also see 7.7.

7.4.5 FeatureType

A feature is an abstraction of real-world phenomena. 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.

/req/general/feature	Features shall be modelled as instances of the metaclass FeatureType, as defined in Figure 5 and in 7.4.5. Features shall not be modelled as specializations of GM_Object (ISO 19107:2003) or TM_Object (ISO 19108:2002). The feature shall have a name that is unique in the application schema.

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

NOTE 2 Geometry-, topology- and temporal-objects (GM_Object, TP_Object, TM_Object) are not abstractions of real-world phenomena. These types can provide types for feature properties, but cannot be specialized to features.

NOTE 3 Annex B provides a table on the use of the term “feature”.

FeatureType inherits attributes for identification and description from the abstract metaclass IdentifiedType. The **name** is mandatory on a feature type, and is unique in an application schema.

FeatureType adds the following attributes and association ends:

— isAbstract

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

— carrierOfCharacteristics

association role to specify any feature operation, any feature attribute type and any feature association role that carries characteristics of a feature type. While strictly optional, it is unlikely for a feature type to be defined having no properties. Repeatable.

— superType

association role to specify a supertype (generalized type) of the feature type (optional, repeatable).

7.4.6 PropertyType

PropertyType is the abstract superclass for AttributeType, Operation and FeatureAssociationRole, which provide characteristics that may be associated with a feature type. PropertyType inherits attributes for identification and description from the abstract metaclass IdentifiedType. PropertyType adds no further attributes or associations.

Property types are specified independent of any specific feature type. A property instance and value occurs in the context of a feature, though in some cases the feature is very generic and might not be made explicit in the dataset. For example, the feature type or feature identity for an observed property might not be available because the necessary data interpretation has not yet been completed. Properties will often be described independent of features. It is therefore best if property type names are unique within an application.

/rec/general/property-name	The name of each property type should be unique within an application schema, or should have the same intention with respect to the CLASS owning the property as any other property with the same name in the Application Schema. A property type may be used by one or more feature types if the property type definition is the same in all contexts.

NOTE Decoupling property-types from feature-types is consistent with scientific and observational applications, where properties may be observed prior to feature detection, and where the same property may characterize more than one feature at different stages in a processing chain.

EXAMPLE 1 Physical properties like temperature, magnetic field, gravity vector, are present universally, and may be usefully measured and recorded independently from the feature or medium at any particular location.

EXAMPLE 2 Engineering properties like mass, width, height, distance, velocity, may be used in the same sense in the context of different feature-types.

7.4.7 AttributeType

AttributeType is the metaclass for attribute definitions of a feature type (see also 7.5).

/req/general/attribute	Feature attributes shall be modelled as instances of the metaclass AttributeType, as defined in Figure 5 and in 7.4.7, and further elaborated in 7.5.

AttributeType inherits attributes and associations from the abstract metaclass PropertyType, The **name** is mandatory for an attribute type.

AttributeType adds the following attributes and association ends:

— valueType

data type of the attribute value (mandatory).

NOTE 1 ISO 19103 defines some data types that may be used for the valueType of a feature attribute, and other ISO geographic information standards define other object types that may be used.

EXAMPLE 1 Integer, CharacterString or GM_Object.

— valueDomain

description of a set of values (mandatory). In some cases the domain of values may be a set of concepts used as classifiers, organized in a vocabulary or lexicon.

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

NOTE 2 ISO 19103 provides for a class that defines a vocabulary to be stereotyped «enumeration» or «CodeList», where the members of the vocabulary are modelled as UML attributes. Conventionally a vocabulary was considered as a set of character string values. However, modern knowledge organization theory considers a *term* to be merely a label for a *concept* (ISO/TS 19104:2008; Miles & Bechhofer, 2009).

— cardinality

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

— characterize

the attribute type that is described by this attribute type (in the case of an attribute of attribute) (optional).

— characterizeBy

an attribute type that describes this attribute type (in the case of an attribute of attribute) (optional, repeatable).

EXAMPLE 3 An attribute that carries the position of a feature may have another attribute that holds the positional accuracy (data value of QualityAttributeType) of this position.

A taxonomy of attribute types is described in 7.5.

7.4.8 Operation

The behaviour of features is described by operations that may be performed upon or by all instances of a feature type. An 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. Operation is the metaclass for describing behaviour of feature types in terms of operations.

/req/general/operation	Operations shall be modelled as instances of the metaclass Operation, as defined in Figure 5 and in 7.4.8.

NOTE 1 Operations only apply to the interoperability model and do not apply to the data transfer model as described in 6.3.

NOTE 2 Instances of 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 constructor 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.

In addition to the attributes and associations inherited from the abstract metaclass PropertyType, Operation adds the following attributes:

— signature

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

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

Properties of class operations are determined by application schema designers.

7.4.9 FeatureAssociationRole

FeatureAssociationRole is the metaclass for the classes of roles that are part of a FeatureAssociationType (7.4.11). FeatureAssociationRole indicates the role played by the feature type through the association. The instance of FeatureAssociationRole that gives the role for one feature type can also be seen as part of this feature type.

/req/general/association-role	Feature association roles shall be modelled as instances of the metaclass FeatureAssociationRole, as defined in Figure 5 and in 7.4.9.

FeatureAssociationRole inherits attributes and associations from the abstract metaclass PropertyType. The **name** is mandatory for an association role.

FeatureAssociationRole adds the following attributes:

— valueType

data type of the role value (mandatory).

— 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 (mandatory).

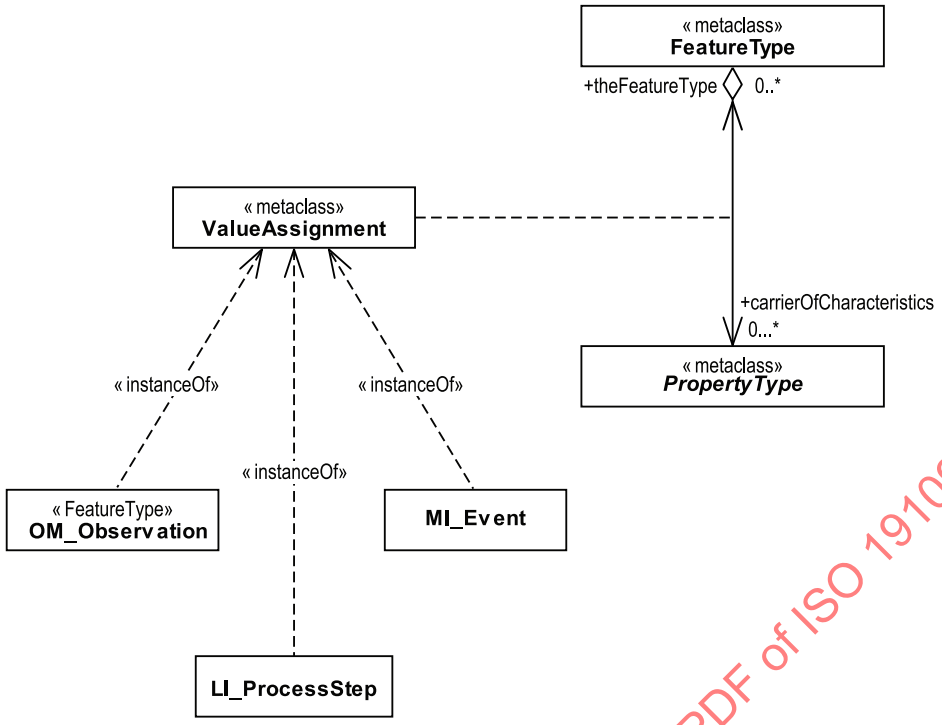
7.4.10 ValueAssignment

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. PropertyType is the supertype of Operation, AttributeType and FeatureAssociationRole.

Values can be assigned to properties through a variety of methods. In many applications and for many properties the method is either not known, or is not of sufficient interest for it to be explicitly modelled. Nevertheless, in principle a value assignment process is associated with every property value. ValueAssignment is the association metaclass for any process by which values are assigned to a property, and when required, instances of this metaclass support auditing the evidence for assignment of a property value.

/req/general/value-assignment	A value-assignment process shall be modelled as an instance of the association class metaclass ValueAssignment, as defined in Figure 6 and in 7.4.10. In a value-assignment instance, the property for which a value is assigned shall be consistent with the feature-type that it is associated with.

The feature type OM_Observation (ISO 19156), and the classes LI_ProcessStep (ISO 19115-1) and MI_Event (ISO 19115-2) may be interpreted as instances of ValueAssignment (Figure 6). The description of a specific observation can provide an evaluation of the likely error in a property value.



NOTE The OM_Observation class in ISO 19156 carries an explicit constraint that the feature-property combination is consistent.

Figure 6 — Example instances of ValueAssignment metaclass

Other examples of value assignment classes are:

— Assertion, in which a property value is assigned by a competent authority.

EXAMPLE 1 The owner of a land parcel or owner of a building is typically asserted by a land registration authority.

— Inheritance from a parent feature or type in which a property value is duplicated from a well defined source.

EXAMPLE 2 All instances of daffodil' have the property colour='yellow' inherited from their class definition.

— Derivation, in which a property value is derived from a number of input parameters based on a defined ruleset.

EXAMPLE 3 If a land area feature has a surface property that equals 'paved', a type property that equals 'road or track' and a make property that equals 'manmade', then the classification value 'road' is assigned.

EXAMPLE 4 Spatial analysis: The municipality a feature falls into given by a point location is automatically determined by spatial analysis against a municipal area dataset.

The ValueAssignment metaclass is shown with a single attribute to indicate the general class involved:

— type

  class of value assignment (mandatory)

The set of value assignment classes described here is not exhaustive, and is therefore shown in Figure 5 as a «CodeList».

7.4.11 FeatureAssociationType

FeatureAssociationType is the metaclass for describing associations between features (see also 7.6.3). A feature association may have attributes.

/req/general/feature-association	Feature associations shall be modelled as instances of the metaclass FeatureAssociationType, as defined in Figure 5 and in 7.4.11, and further elaborated in 7.6.3.

NOTE 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"). Association instances can also carry metadata information.

In addition to the attributes and associations inherited from the abstract metaclass IdentifiedType, FeatureAssociationType adds the following association end:

— roleName

name of a specific role associated with an FeatureAssociationType (mandatory, two may be present).

7.4.12 InheritanceRelation

InheritanceRelation is the class for a 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".

/req/general/inheritance	Inheritance relations shall be modelled as instances of the metaclass InheritanceRelation, as defined in Figure 5 and in 7.4.12, and further elaborated in 7.6.2.

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 (optional).

— 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. Optional with a default value of "true".

— supertype

role of being the more generic feature type of one other or other feature types (optional, repeatable).

— subtype

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

7.5 Attributes of feature types

7.5.1 Introduction

In 7.5 the role of attributes of features is described in more detail. In addition to the attributes inherited from the supertype (PropertyType), an attribute type has a type (valueType), a domain (valueDomain) and cardinality (Multiplicity) associated with it.

The attributes carry all static information of a feature. This covers both spatial and non-spatial properties. In the ISO geographic information standards, some attribute types are of specific interest. These types are shown in Figure 7 as subtypes of AttributeType. The attribute provides the interface to these other ISO geographic information 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 (SpatialAttributeType) will have its value type and value domain according to the definition of GM_Object or TP_Object described in ISO 19107.

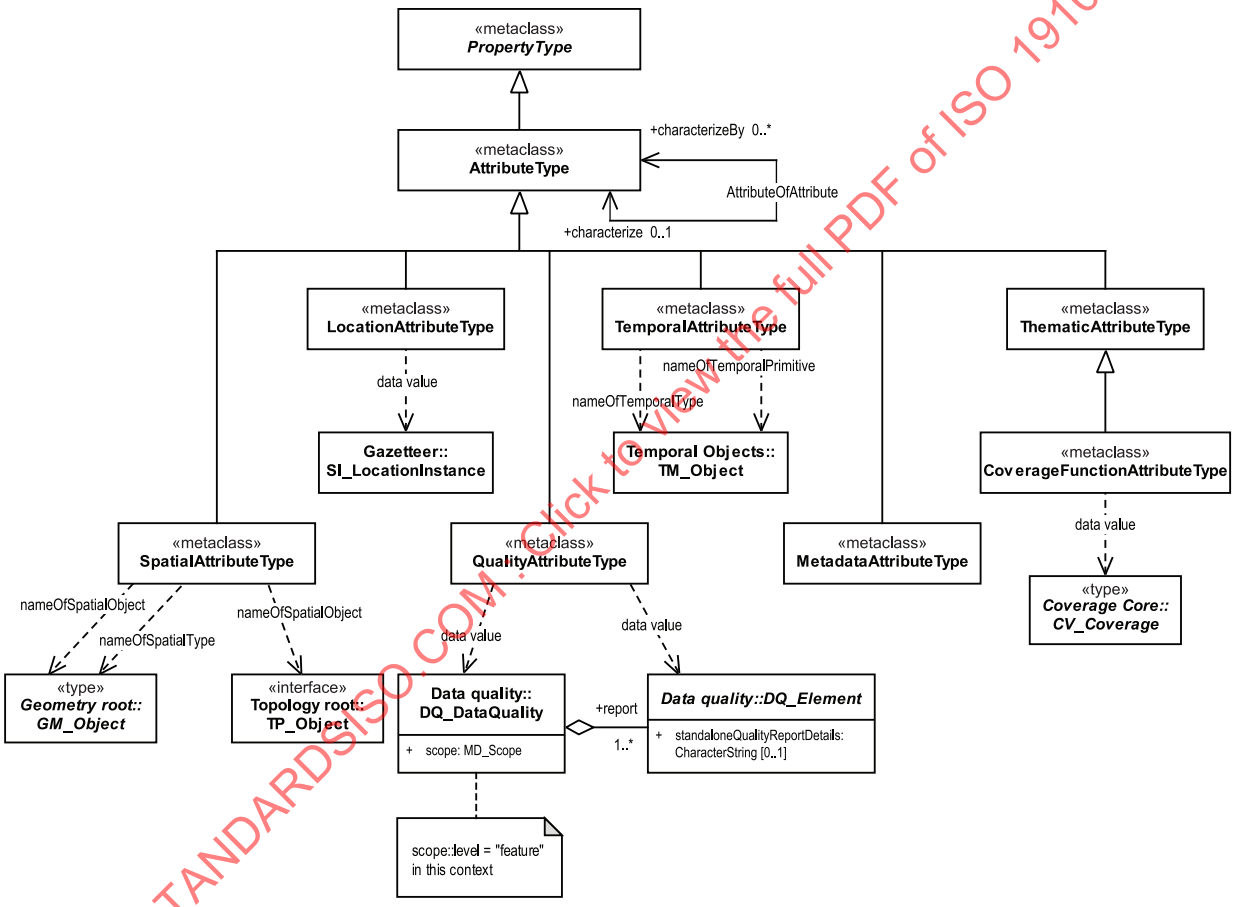


Figure 7 — Attributes of feature types

7.5.2 SpatialAttributeType

SpatialAttributeType represents a spatial attribute, which is used to express spatial characteristics of a feature type. A spatial attribute type has a spatial type or 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.

7.5.3 TemporalAttributeType

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

7.5.4 QualityAttributeType

QualityAttributeType represents attributes that carry quality information. These attributes are used when a quality characteristic of a feature or its properties is included as data in the dataset. Quality attributes can have a value type according to the definition of DQ_Element or DQ_DataQuality defined in ISO 19157.

7.5.5 LocationAttributeType

LocationAttributeType represents attributes which carry a spatial reference to a feature by using a geographic identifier. Ideally that geographicIdentifier is scoped (e.g. is in a namespace) to a gazetteer.

EXAMPLE A postal code references a location by its geographicIdentifier, allowing its location to be found in a relevant gazetteer.

NOTE In practice, the identifier is a character string, matching the lexical form and content rules for a specific gazetteer.

7.5.6 MetadataAttributeType

MetadataAttributeType represents attributes that carry metadata information when such information is included as data in a dataset. These attribute types can use the metadata element classes defined in the Metadata Schema (ISO 19115-1) as their value types.

7.5.7 ThematicAttributeType

ThematicAttributeType represents attributes which carry any descriptive characteristic of a feature except those specified in 7.5.2 to 7.5.6. Their value types and value domains are normally defined by the user or by the application area. Both basic types (e.g. see ISO 19103) and user defined data types can be used.

7.5.8 CoverageFunctionAttributeType

CoverageFunctionAttributeType represents thematic attributes whose value varies as a function of spatio-temporal position within the scope of a feature. Their value types and value domains are defined by the user or by the application area, and provide the rangeType of the CV_Coverage (ISO 19123) that provides the data values for the attribute.

7.6 Relationships between feature types

7.6.1 Introduction

In 7.6 the relationships between feature types are described in some more detail. Figure 8 shows that relationships are classified as follows:

— generalization/specialization of feature types;

— associations between features.

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

7.6.2 InheritanceRelation

The 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.

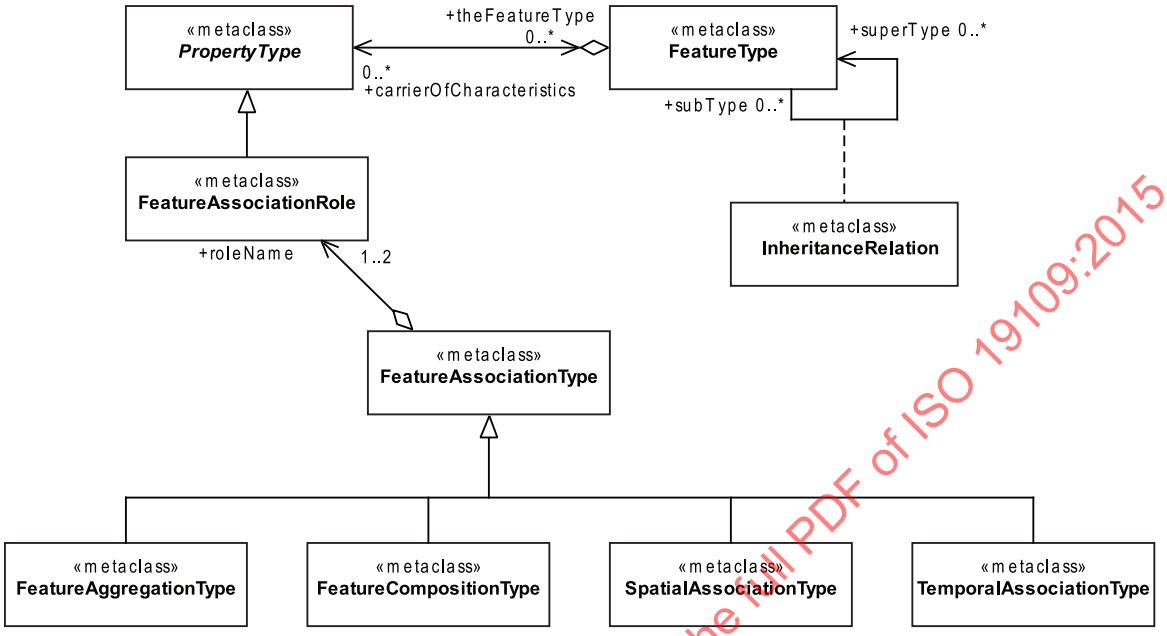


Figure 8 — Relationships between feature types

7.6.3 FeatureAssociationType

FeatureAssociationType 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. An association type can be characterized by its own properties, for example having its own attributes.

In the area of geographic information, there exist multiple kinds of associations, which are handled by rules defined in this International Standard. Four of these are shown in Figure 8 as subtypes of FeatureAssociationType.

— FeatureAggregationType

  FeatureAggregationType represents associations between the types that are some kind of "whole-part" relationship, where the parts can exist even if the aggregate is destroyed. Aggregation is used to specify types which form a complex type.

EXAMPLE 1 International organization and its member states.

— FeatureCompositionType

  FeatureCompositionType represents "whole-part" relationships where features of one type are owned by features of the other type, so the parts will be destroyed together with the composite.

EXAMPLE 2 A bridge structure and the road surface it carries

— SpatialAssociationType

  SpatialAssociationType represents spatial relationships or topological relationships that may exist between features. Those relationships depend upon the relative spatial position and extent of features. See further discussion in 8.7.3.

EXAMPLE 3 An intersection is a relationship between two roads, and a road is a relationship between two intersections.

EXAMPLE 4 Containment relationships such as paddocks within farms, tracts within forests, counties within a state.

— TemporalAssociationType

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

EXAMPLE 5 The phrase "built before" is the temporal association in the statement "Amsterdam central railway station was built before Tokyo railway station".

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. Constraints may be used for various other purposes. 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 the selected CSL.

/rec/general/constraint	Constraints on an application schema that are not otherwise captured in the structural aspects of a model may be expressed formally using a constraint language suitable for the CSL.

NOTE Where the CSL is UML, the constraint language is normally OCL. Nevertheless, a constraint can be expressed in a natural language.

8 Rules for application schema in UML

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 can involve iterations or feedback loops, and 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;

— how to use schemas defined in the other ISO geographic information standards.

8.2 The application schema

8.2.1 General

In 8.2 the requirements for defining an application schema using UML as the CSL are described. These are formalized as a requirements class summarized in Table 16.

Table 16 — Requirements class for modelling an application schema in UML

Requirements class	/req/uml
Target type	UML application schema
Dependency	/req/general (General Feature Model)
Dependency	ISO 19103:2015 (Conceptual schema language)
Requirement	/req/uml/profile
Requirement	/req/uml/identification
Requirement	/req/uml/documentation
Requirement	/req/uml/integration
Requirement	/req/uml/structure
Requirement	/req/uml/feature
Requirement	/req/uml/association
Requirement	/req/uml/aggregation
Requirement	/req/uml/attribute
Requirement	/req/uml/role
Recommendation	/rec/uml/property-name
Requirement	/req/uml/attribute-of-attribute
Requirement	/req/uml/operation
Requirement	/req/uml/inheritance
Requirement	/req/uml/constraint
Requirement	/req/uml/value-assignment

8.2.2 Conceptual schema language for application schemas

The General Feature Model, described in Clause 7, is a metamodel for definition of features. It provides a model of the concepts required to classify a view of the real world. While the GFM itself is formalized in UML, it is not a CSL. A different, though related, profile of UML is used as the CSL for formalizing an application schema.

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, and also provide some constraints on the use of UML for modelling Application Schemas. The examples in Clause 8 are shown in UML. ISO/IEC 19505-2 explains how to use

UML. ISO 19103 explains how stereotypes and basic datatypes shall be used within the ISO geographic information standards and application schemas developed according to these standards, and also describes other requirements on the use of UML. Together with some requirements described in this International Standard, this provides a UML profile which is the CSL for modelling application schemas.

NOTE ISO 19150-2 provides corresponding rules for OWL 2 (as defined by W3C), where OWL 2 is the conceptual schema language.

The UML profile for application schemas is composed of

— primitive types, common implementation types, and derived types, covering text, names, numerics, date and time, truth, collections, records and measures (ISO 19103:2015, Clause 7 and Annex C),

— some requirements on the use of UML associations (ISO 19103:2015, 6.8.2) specifically:

  — All association ends have cardinalities constrained;

  — Any navigable association end has an explicit rolename;

  — An association with role names can be viewed as similar to class attributes for the two classes involved,

— a recommendation that attributes, operations and roles are named uniquely within a package,

— a set of stereotypes and keywords, consistent with the UML profile defined in ISO 19103:2015, Annex D, and summarized in Table 17.

Table 17 — Summary of UML Profile used as the CSL for application schemas

Stereotype or keyword	UML meta-class	Constraints	Tagged values	Source
ApplicationSchema	Package		version	This International Standard, 8.2.3
			description catalogue-entry	This International Standard, 8.2.4
		Not nested in another ApplicationSchema package		This International Standard, 8.2.5
			language designation definition	This International Standard, 8.12
CodeList	Class			ISO 19103:2015, 6.5.3, 6.10
dataType		Use in attributes or strong aggregation (composition)		ISO/IEC 19505-2:2012, 7.3.11 ISO 19103:2015, 6.10
enumeration				ISO/IEC 19505-2:2012, 7.3.16 ISO 19103:2015, 6.5.2, 6.10
Estimated	Property			This International Standard, 8.2.6
FeatureType	Class		description	This International Standard, 8.2.4, 8.2.6
			designation definition	This International Standard, 8.12
Leaf	Package	Cannot contain another package		ISO 19103:2015, 6.10
Union	DataType			ISO 19103:2015, 6.10

Table 17 *(continued)*

Stereotype or keyword	UML meta-class	Constraints	Tagged values	Source
use				ISO/IEC 19505-2:2012, 7.3.54

NOTE Stereotype names are UpperCamelCase, following the UML2 convention [ISO/IEC 19505-2:2012, 18.3.9]. Standard UML keywords are lowerCamelCase [ISO/IEC 19505-2:2012, Annex B]. Nevertheless, since matching is case-insensitive the distinction is only cosmetic.

/req/uml/profile	An application schema shall be modelled using the UML profile defined in ISO 19103:2015 and augmented in this International Standard as summarized in 8.2.2.

8.2.3 Packaging and identification of an application schema

/req/uml/packaging	An application schema shall be described within a PACKAGE. The PACKAGE shall carry the stereotype «ApplicationSchema». The PACKAGE name is the name of the application schema. The application schema version shall be recorded in the tagged value "version".

NOTE 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.

8.2.4 Documentation of an application schema

/req/uml/documentation	An application schema shall be documented. The textual definition of each CLASS, ATTRIBUTE, ASSOCIATION ROLE, OPERATION and CONSTRAINT should be recorded using the primary documentation facility provided by the tool if this information can be exported. Secondary descriptions and informative notes for each PACKAGE, CLASS, ATTRIBUTE, ASSOCIATION ROLE, OPERATION and CONSTRAINT may be recorded using a TAGGED VALUE with the name "description". If a CLASS or other UML component is an implementation of a model element defined in a feature catalogue, the reference to the catalogue shall be documented in a TAGGED VALUE with the name "catalogue-entry".

NOTE 1 Most UML tools support documentation attached to every element of a model, including packages, classes, attributes, associations, association roles, operations, constraints. In the interface this is labelled "Note" or "Scope Note" or similar, and is exported in a tagged value with the tag name "documentation" or similar.

NOTE 2 Feature types and property types from an application schema will also usually be documented in a feature catalogue in accordance with ISO 19110, which may be generated automatically from the UML model.

8.2.5 Integration of application schemas 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 geographic information standards are other parts.

Furthermore, a complete application may be built from more than one application schema package, which may be subject to independent governance arrangements. Each of these schemas can refer to standardized schemas. This organization can be used to avoid the creation of large and complex schemas. It can also assist in maintenance of a complex schema by allowing orderly delegation of design and maintenance of different aspects of the schema to different authorities. This may include an externally maintained vocabulary or lexicon that defines the domain of values for a feature attribute.

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. Integration of

application schemas uses UML dependencies rather than package nesting. The unit of re-use is a UML Package with the stereotype «ApplicationSchema», or a package from one of the standard schemas.

EXAMPLE 1 Figure 9 shows an application schema that uses elements from schemas defined in the other ISO geographic information standards, and from other application schemas. The usage dependencies in Figure 9 mean that the application schema package uses structures and definitions from both standard schemas and other application schemas.

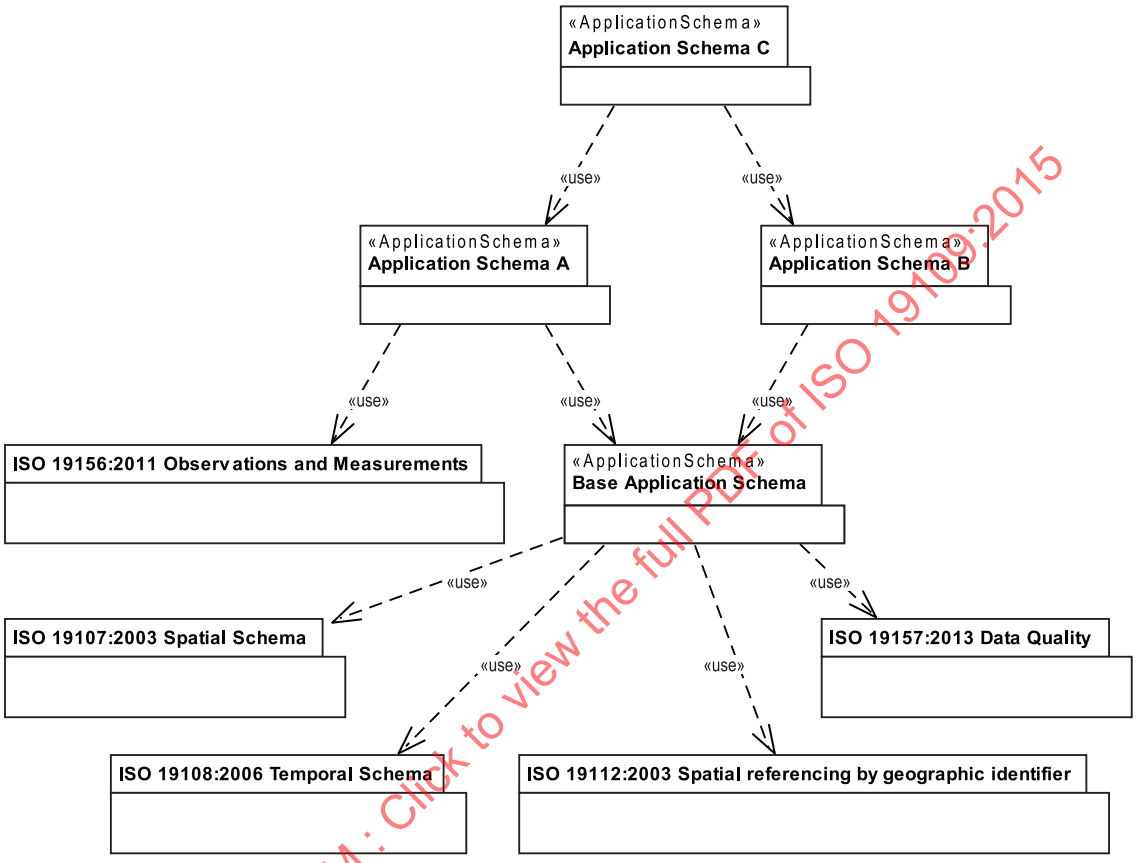


Figure 9 — Example of application schema integration

/req/uml/integration	UML PACKAGE dependency shall be used to describe the integration of the application schema with the other schemas, including standard schemas, as required to form the complete definition of the data structure. A PACKAGE with the STEREOTYPE «ApplicationSchema» shall not contain another PACKAGE with the STEREOTYPE «ApplicationSchema». Dependencies shall be between PACKAGEs with the STEREOTYPE «ApplicationSchema» except for dependencies on standard schemas. A PACKAGE containing, or contained by, a PACKAGE with the STEREOTYPE «ApplicationSchema» shall not have dependencies or dependants.. PACKAGEs shall not have mutual dependencies, and dependency graphs shall not include cycles. Dependencies on standard schemas from the ISO geographic information standards or other application schemas may carry the STEREOTYPE «use».

NOTE Mutual dependencies would require that multiple packages are maintained as a single artefact, which therefore cancels a key advantage of packaging.

EXAMPLE 2 Figure 10 illustrates that an application containing roads, rivers and bridges can be described by the four following UML packages:

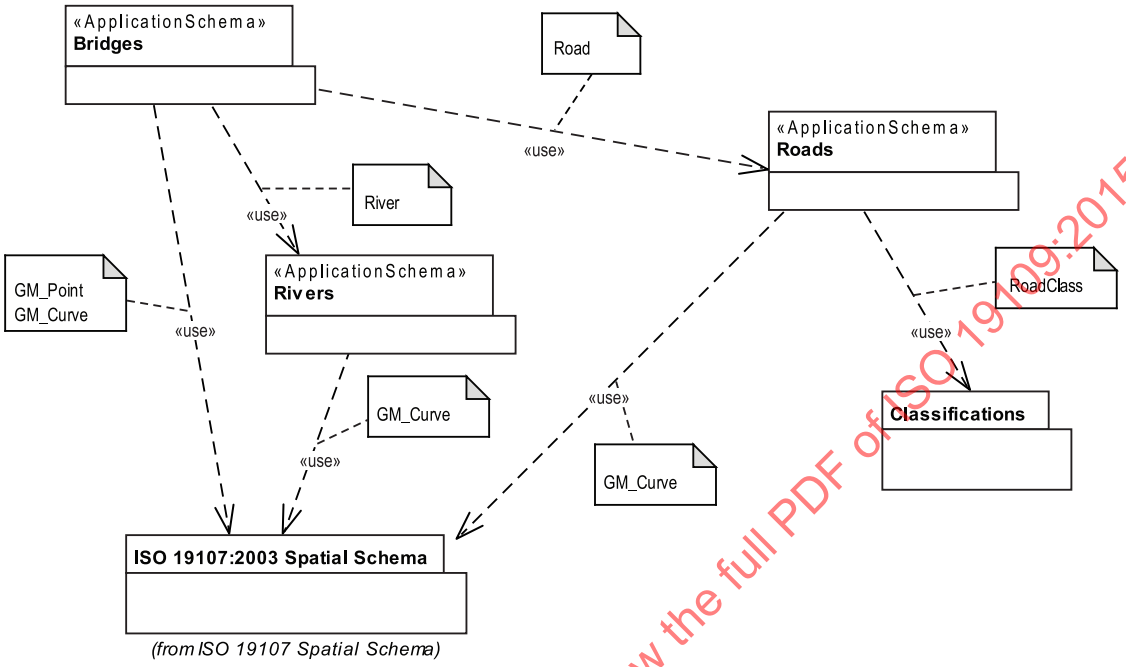
— a main schema describing the bridges;

— a schema defining data type roads;

— a schema defining data type rivers;

— a schema containing a vocabulary that provides the domain of values for road classification.

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



NOTE A diagram note can be used to indicate the classes involved in the dependency.

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

8.2.6 Modelling structures in UML

A basic rule concerns data structures and instantiability.

/req/uml/structure	The data structures of the application shall be modelled in the application schema. All CLASSES used within an application schema for data transfer shall be instantiable, or if abstract, shall have non-abstract specializations. Any CLASS integrated from another PACKAGE must not be STEREOTYPED «interface».

NOTE 1 The stereotype «interface» is defined in ISO 19103. While it is appropriate for use in standards packages, it is not used in application schemas.

The following rules concern the implementation of the structures of the General Feature Model in UML.

AnyFeature is an abstract class that is the generalization of all feature types. AnyFeature is an instance of the metaclass FeatureType (7.4.4) as shown in Figure 11, and thus following /req/uml/feature it has the stereotype «FeatureType». All feature instances are members of the set represented by this class.

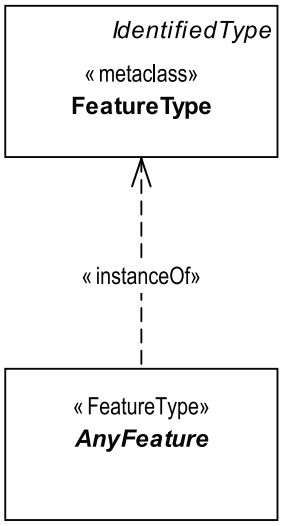


Figure 11 — The AnyFeature feature type

NOTE 2 AnyFeature corresponds to GFI_Feature from ISO 19156.

/req/uml/feature	An instance of FeatureType shall be implemented as a CLASS. The CLASS shall carry the STEREOTYPE «FeatureType». The CLASS shall have a GENERALIZATION with AnyFeature. The CLASS name shall be unique within an Application Schema. The textual definition of the CLASS shall be recorded utilizing the documentation facilities in the software tool, if this information can be exported.

NOTE 3 Most UML tools support documentation attached to a class. In the interface this is labelled “Note” or “Scope Note” or similar, and is exported in a tagged value with the tag name “documentation” or similar.

/req/uml/association	Each instance of FeatureAssociationType shall be associated with one or more instances of PropertyType. It shall be implemented as an ASSOCIATION CLASS; the associated instances of PropertyType shall be implemented as ATTRIBUTES of the ASSOCIATION CLASS.

Complex features are defined as aggregations. Both strong and weak aggregations may be used.

/req/uml/aggregation	An instance of FeatureAggregationType shall be implemented as an AGGREGATION (empty diamond). An instance of FeatureCompositionType 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.

/req/uml/attribute	An instance of AttributeType shall be represented in an application schema in either of two ways: case 1: as an ATTRIBUTE, unless it is an attribute of an attribute [see /req/uml/attributeOfAttribute]; or case 2: as an ASSOCIATION between the CLASS that represents a feature and a class representing the value domain of the AttributeType. The ATTRIBUTE or ASSOCIATION ROLE may carry a stereotype «Estimated» if the value of the attribute is assigned using an observation procedure [ISO 19156:2011].

/req/uml/role	An instance of FeatureAssociationRole shall be implemented as a ROLE name at the appropriate end of the ASSOCIATION representing the FeatureAssociationType. The ASSOCIATION end may carry the STEREOTYPE «Estimated» if the value of the ASSOCIATION target is assigned using an observation procedure [ISO 19156:2011].

It is strongly recommended that the names of attributes, operations and association roles are unique within an Application Schema, or that attributes, operations and association roles with the same name in different classes have the same meaning. This recommendation is consistent with the General Feature Model (Figure 5) where ValueAssignment is an aggregation, so instances of the metaclass PropertyType exist independent of any FeatureType with which they are used. Although in generic UML attributes and association roles are scoped to a class, under this usage each attribute, operation or association role is implicitly scoped to a UML package. As foreseen above (8.2.1) this defines an aspect of the UML profile that is used as the conceptual schema language for Application Schemas.

/rec/uml/property-name	The name of an ATTRIBUTE, OPERATION or ASSOCIATION end should be unique within an Application Schema, or should have the same intention with respect to the CLASS owning the property as any other property with the same name in the Application Schema.

EXAMPLE 1 Road::width and River::width can both appear in an application schema consistent with this recommendation, since the property type "width" has the same meaning in the context of features of both types.

EXAMPLE 2 Forest::classification and Farmland::classification can both appear in an application schema consistent with the recommendation, even if the value types are different (e.g. different code lists), as long as the meaning of the property "classification" is the same in both contexts.

NOTE 4 This approach is consistent with ontologies, where properties are first-class objects alongside classes. If an application schema follows this recommendation then it is possible to have a parallel representation of its structure in UML and OWL using the same names.

NOTE 5 This is also consistent with common practice in observational applications, particularly in natural sciences. Observations of physical phenomenon like temperature, magnetic field, gravity are generally made separately from their association with any specific feature, sometimes in order to support detection and characterization of a feature of unknown type. The same property-type may be characteristic of many different feature types. The property types are typically listed independent of their potential association with one or more feature types.

/req/uml/value-assignment	An instance of ValueAssignment shall be represented by a CLASS that provides metadata for the assignment of the property value.

NOTE 6 OM_Observation and its specializations defined in ISO 19156 provide a suitable model for value assignment using an observation procedure. A requirements class for use of ISO 19156 is provided in 8.9.

/req/uml/attribute-of-attribute	An instance of AttributeType that acts in the role characterizeBy in an attributeOfAttribute association shall be instantiated as a CLASS. That CLASS shall be used either as the data type of the AttributeType, or in an ASSOCIATION with the CLASS that contains the AttributeType. ATTRIBUTEs that act in the role characterize shall be instantiated as ATTRIBUTEs of the CLASS that represents the ATTRIBUTE that acts in the role characterizeBy: 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.

/req/uml/operation	An instance of Operation shall be implemented as an OPERATION of the CLASS representing the feature type that it characterizes.

Operation signatures can reference various attributes in the schema, as required for the application.

EXAMPLE 3 Figure 12 shows some simple feature operations. A feature type "Building" has a feature attribute "temp" which carries the measured temperature. The OPERATION "GetTemperature" measures the current temperature and affects the values of the attribute. The OPERATION "Alarm" will be triggered if the value of the attribute exceeds a maximum value. The UML model is shown below. Additional constraints and rules for the operation must be documented.

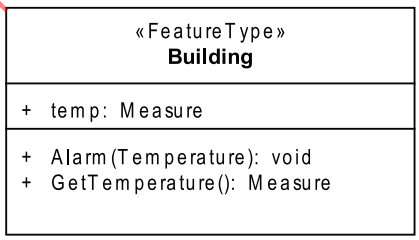


Figure 12 — Example of a feature with operations

/req/uml/inheritance	An instance of InheritanceRelation shall be represented by a GENERALIZATION relationship, with additional characteristics depending on the following condition: case 1: If *uniqueInstance* is *true* (default value), the {disjoint} constraint may be attached to the generalization relationship; case 2: If *uniqueInstance* is *false*, the {overlapping} constraint shall be attached to the generalization relationship.

/req/uml/constraint	CONSTRAINTS may be stated in OCL or in plain language and attached to the CLASS, OPERATION or RELATIONSHIP that is constrained.

EXAMPLE 4 Figure 13 shows an example of an application schema expressed in UML, based on real-world concepts in Table 18. 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.

Table 18 — Example of real-world concepts in terms of the General Feature Model

Feature types	Attributes	Kind of subtype of Attribute
Property parcel	identification name border updates	Thematic attribute type Thematic attribute type Spatial attribute type Metadata attribute type
Building	centre point shape address type owner horizontal accuracy vertical accuracy position	Spatial attribute type Spatial attribute type Location attribute type Thematic attribute type Metadata attribute type Quality attribute type Quality attribute type Spatial attribute type
Loan	amount period classification	Thematic attribute type Temporal attribute type Metadata attribute type

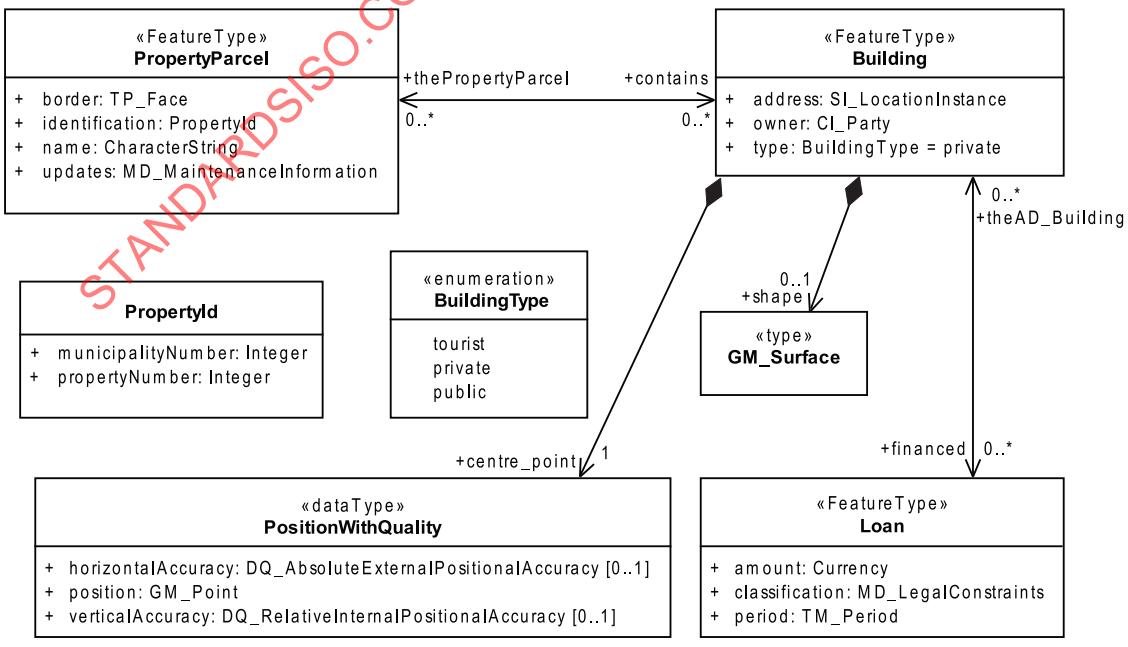


Figure 13 — Example of UML implementation of feature types

8.3 Domain profiles of standard schemas in UML

8.3.1 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;

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

In 8.3 the requirements for defining a profile of a standard schema are described. These are formalized as a single requirements class, summarized in Table 19.

Table 19 — Requirements class for defining a profile of a standard schema

Requirements class	/req/profile
Target type	UML application schema
Dependency	/req/uml (Application schema using UML)
Requirement	/req/profile/extend
Requirement	/req/profile/restrict
Recommendation	/rec/profile/names

8.3.2 Adding information to a standard schema

The definitions in a standard schema can be extended with additional information.

/req/profile/extend	If it is necessary to extend 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 and ASSOCIATIONS shall be added to this CLASS to carry the additional information. The new CLASSES shall be collected in a separate PACKAGE.

EXAMPLE Figure 14 shows an application schema which is using a definition (GM_Surface) from the standard spatial schema directly, and a definition (SurfaceWithQuality) in a package called Domain Spatial. 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.

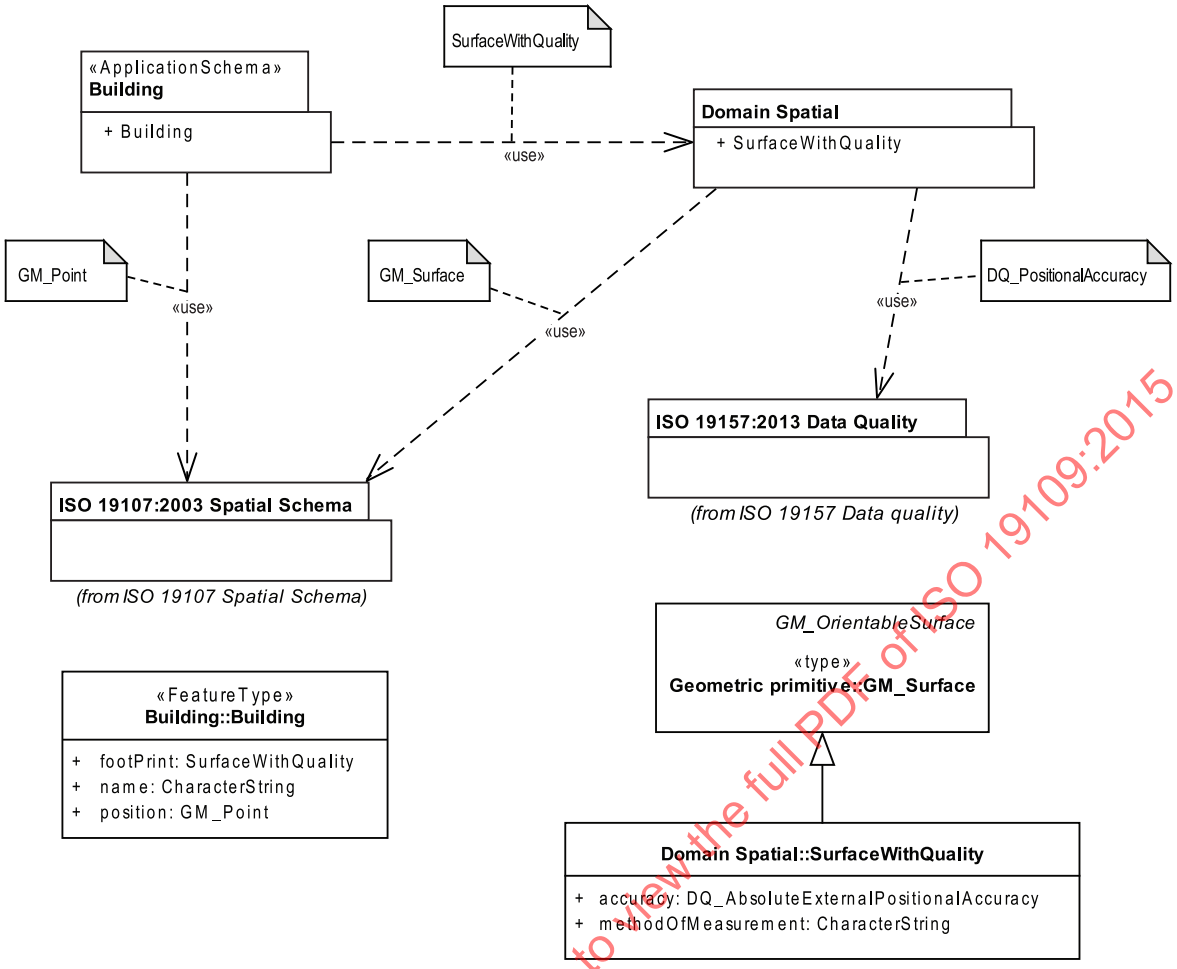


Figure 14 — Example of adding information to a standard schema

8.3.3 Tailored use of standard schemas

A standard schema can be tailored so that only selected parts of that schema, which are relevant for the application schema, are supported. Any tailoring needs to honour the requirements and conformance classes specified in the standard schema.

/req/profile/restrict	A tailored profile of a standard schema may be constructed by defining CLASSES in a new UML PACKAGE, with a dependency on the standard schema, and uses either Case 1: realization relationships between CLASSES in the tailored schema and interfaces in the standard schema Case 2: specialization relationships between interfaces in the tailored schema and interfaces in the standard schema, with CONSTRAINTS on ATTRIBUTES, ASSOCIATION ROLES or OPERATIONS that restrict their cardinality or type to a value allowed by the original definition If the standard schema specifies requirements on the use of the schema, the tailored schema shall conform to all requirements associated with the conformance classes that the tailored schema is intended to conform to.

The two cases have different semantics and characteristics.

In case 1, the tailored schema is on a different abstraction level than the standard schema: the standard schema will typically model interfaces[1)] while the tailored schema will typically model classes and be on the same abstraction level as the application schema.

NOTE 1 As the classes are realizations, they may implement the attributes, association roles, operations, constraints and other characteristics specified in the standard schema in a different way, e.g. use different names for attributes.

In case 2, the specialized classifiers will still be on the same abstraction level as the standard schema.

NOTE 2 See ISO 19103:2015, 6.3 and 6.8.4 as well as ISO 19107:2003, 2.1 provide additional discussion about the use of interfaces from the standard schemas in application schemas.

A naming strategy can be used to express the lineage of the elements in the tailored schema.

/rec/profile/names	Classifiers in such tailored schemas that realize or specialize interfaces from the standard schemas should have the same local name as the interface in the standard schema.

EXAMPLE A tailored 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 with those classes.

8.4 Rules for use of metadata schema

8.4.1 Introduction

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. The metadata schema (ISO 19115-1) is an application schema for metadata data sets. A schema for data quality is defined in ISO 19157.

In application schemas, definitions from ISO 19115-1 and ISO 19157 may be used. This International Standard does not place restrictions on such use. In 8.4 the requirements for an application schema that includes metadata elements from ISO 19115-1 are described, and 8.5 describes the requirements for an application schema that includes quality elements from ISO 19157.

The requirements for metadata elements are formalized as a single requirements class, summarized in Table 20.

Table 20 — Requirements class for an application schema including metadata

Requirements class	/req/metadata
Target type	UML application schema
Dependency	ISO 19115-1.2014 (Metadata)
Dependency	/req/uml (Application schema using UML)
Requirement	/req/metadata/feature

1) ISO 19103 uses the terms “abstract classes” or “types”, too.

8.4.2 Metadata for features, feature attributes, and feature associations

The elements of the ISO 19115-1 metadata schema can be used as feature properties.

/req/metadata/feature	A metadata element may be used as a Metadata_AttributeType (subtype of AttributeType) to carry metadata about instances of feature types, feature attributes or associations between features [see 8.2.6, /req/uml/attribute]. The data type of any feature attribute carrying metadata shall be a metadata element or specialization as specified as CLASSES in the metadata schema ISO 19115-1:2014. If the types are suitable a feature attribute may use definitions from ISO 19115-1:2014 for purposes other than metadata. A metadata attribute shall be represented in an application schema in either of two ways: case 1: as an ATTRIBUTE of a CLASS that represents a feature; or case 2: as an ASSOCIATION between the CLASS that represents a feature and a CLASS from ISO 19115-1:2014. A metadata attribute may be used as an attribute of an attribute, in which case the main rule for *attributeOfAttribute* shall be applied [see 8.2.6, /req/uml/attributeOfAttribute].

EXAMPLE Figure 15 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 [/req/metadata/feature, Rule 2)]. EX_GeographicBoundingBox is used to specify a geographical area [/req/metadata/feature, Rule 3)].

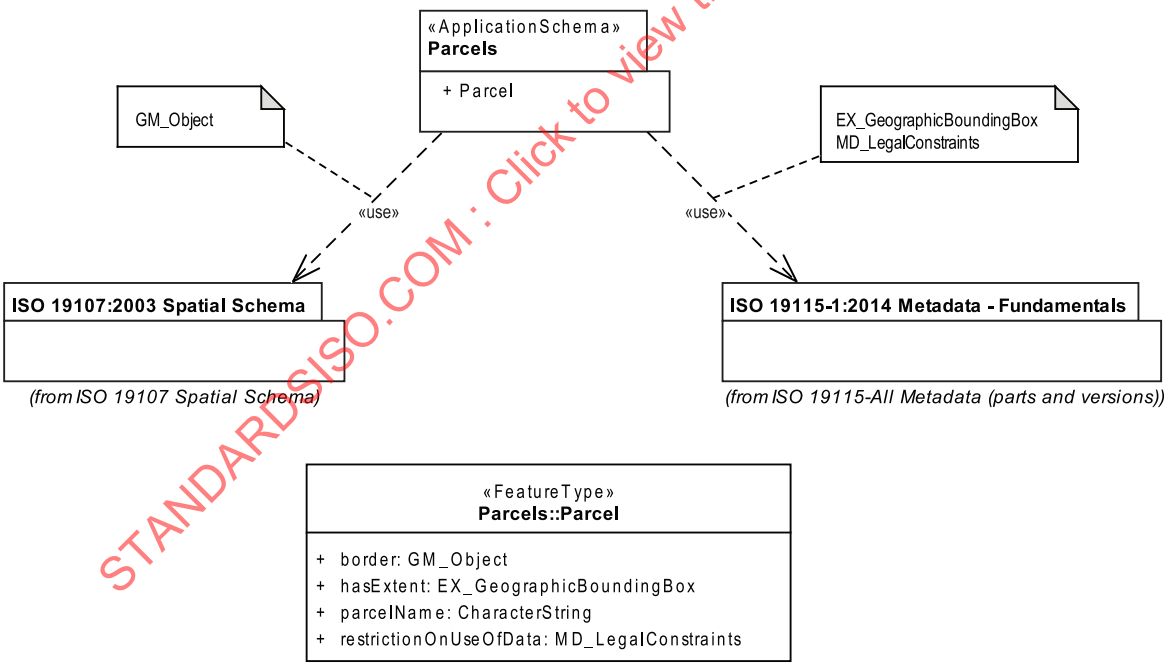


Figure 15 — Example of metadata included as data in an application

8.5 Rules for use of quality schema

8.5.1 Introduction

In 8.5 the requirements for an application schema that uses quality elements from ISO 19157 are described. The requirements for quality elements are formalized as a single requirements class, summarized in Table 21.

Table 21 — Requirements class for an application schema including quality information

Requirements class	/req/quality
Target type	UML application schema
Dependency	ISO 19157:2013 (Data quality)
Dependency	/req/uml (Application schema using UML)
Requirement	/req/quality/attribute
Requirement	/req/quality/additional-quality
Requirement	/req/quality/attribute-quality

8.5.2 Data quality rules

8.5.2.1 Rules for reporting quality information for instances of data

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 19157.

Elements of the ISO 19157 schema provide a means for reporting quality information.

/req/quality/attribute	Information on the quality of individual instances of features or properties 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. A quality attribute (an instance of QualityAttributeType) shall be defined in the application schema and shall be used to carry data quality information. A quality attribute shall be represented in an application schema in either of three ways: case 1: as an ATTRIBUTE of a CLASS that represents a feature, in which case the ATTRIBUTE shall take one of the subtypes of the CLASS DQ_Element or DQ_DataQuality (see Table 22) defined in the Data Quality Information Package in ISO 19157:2013 as the data type for its value; case 2: as an ASSOCIATION between the CLASS that represents a feature and one of the subtypes of the CLASS DQ_Element or DQ_DataQuality (see Table 22) defined in the Data Quality Information Package in ISO 19157:2013 as the data type for its value; or case 3: as an attribute of an attribute, in which case the main rule for *attributeOfAttribute* shall be applied [see /req/uml/attributeOfAttribute].

Table 22 shows the concrete subtypes of DQ_Element defined in ISO 19157.

Table 22 — Concrete subtypes of DQ_Element

Categories of quality element	Quality element	Subtype of DQ_Element
Positional Accuracy	Absolute or External Accuracy Relative or Internal Accuracy Gridded Data Positional Accuracy	DQ_AbsoluteExternalPositionalAccuracy DQ_RelativeInternalPositionalAccuracy DQ_GriddedDataPositionalAcuracy
Temporal Accuracy	Accuracy of a Time Measurement Temporal Consistency Temporal Validity	DQ_AccuracyOfATimeMeasurement DQ_TemporalConsistency DQ_TemporalValidity
Thematic Accuracy	Classification Correctness Non-quantitative Attribute Correctness Quantitative Attribute Accuracy	DQ_ThematicClassificationCorrectness DQ_NonQuantitativeAttributeCorrect-ness DQ_QuantitativeAttributeAccuracy
Logical Consistency	Consistency with conceptual schema Consistency of values Format Consistency Topological Consistency	DQ_ConceptualConsistency DQ_DomainConsistency DQ_FormatConsistency DQ_TopologicalConsistency
Completeness	Excess data present Data absent	DQ_CompletenessCommisssion DQ_CompletenessOmission
Usability	Usability element	DQ_UsabilityElement

NOTE 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 16 shows the ATTRIBUTE *featureClassificationQuality* carrying quality information about the classification of the feature *Land Area* into subareas.

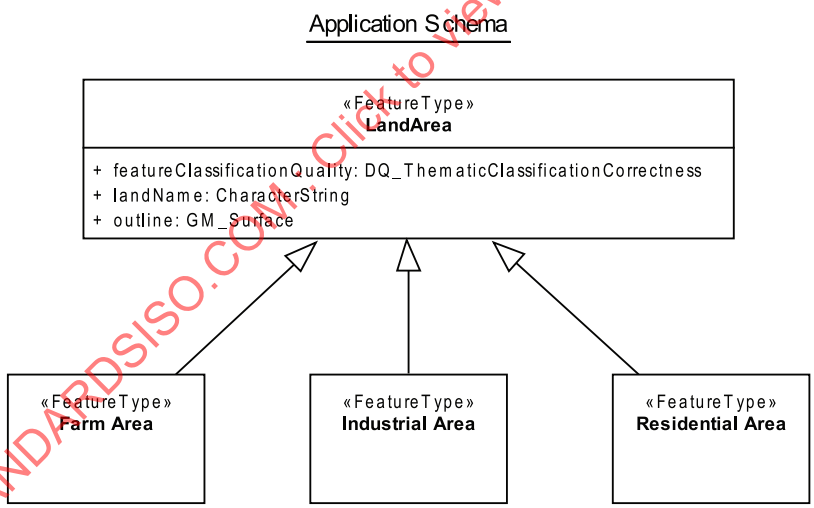


Figure 16 — Example of quality of features instances

EXAMPLE 2 Figure 17 shows two alternatives for UML implementation of the feature Land Area whose attribute Outline has a quality attribute Accuracy.

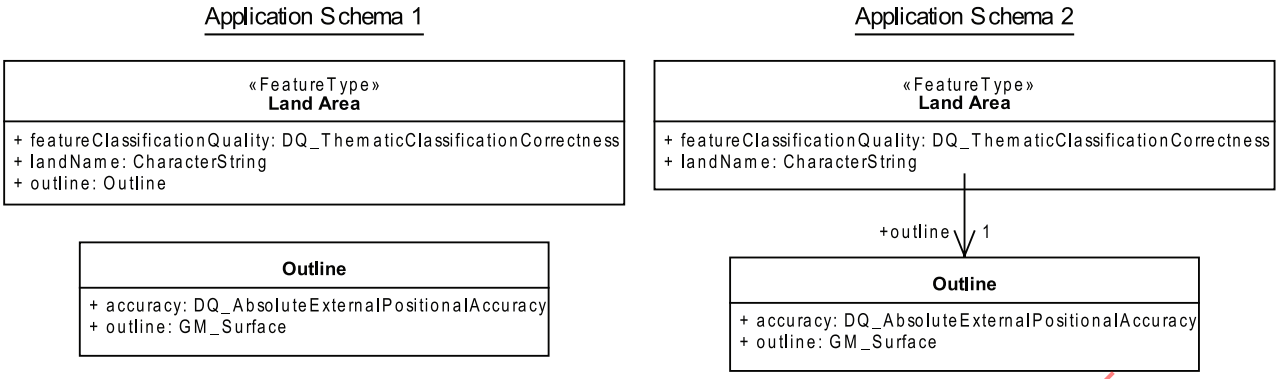


Figure 17 — Examples of quality of attributes of features

8.5.2.2 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 19157 with an additional description called an additional quality sub-element.

Elements of the ISO 19157 schema provide a means to record additional quality information.

/req/quality/additional-quality	User-defined quality sub-elements shall be defined as a specialization of DQ_Element or one of its subtypes (see Table 22) according to rules for domain profiles of standard schemas (see 8.3).

EXAMPLE Figure 18 shows a user-defined quality sub-element.

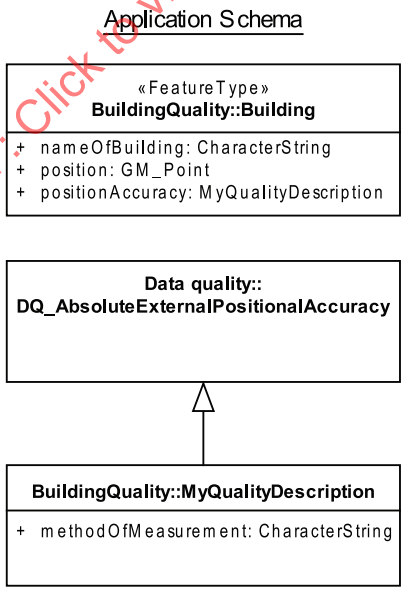


Figure 18 — Example of user-defined quality

8.5.2.3 Reporting quality information for attributes of feature instances

Elements of the ISO 19157 schema provide a means to indicate quality of attributes of feature instances.

/req/quality/attribute-quality	Quality characteristics of feature attributes shall be implemented according to /req/uml/attributeOfAttribute.

8.6 Temporal rules

8.6.1 Rules for modelling applications with temporal properties

In 8.6 the requirements for an application schema with feature types that have temporal properties are described. These are formalized as a single requirements class, summarized in Table 23.

Table 23 — Requirements class for applications with temporal properties

Requirements class	/req/temporal
Target type	UML application schema
Dependency	ISO 19108:2002 (Temporal schema)
Dependency	/req/uml (Application schema using UML)
Requirement	/req/temporal/schema
Requirement	/req/temporal/attribute
Requirement	/req/temporal/association
Requirement	/req/temporal/succession

8.6.2 Use of the temporal conceptual schema

ISO 19108 provides a schema for temporal objects.

/req/temporal/schema	Any description of temporal aspects applied to geographic data shall be in accordance with the specifications given by ISO 19108:2002.

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.

8.6.3 Temporal attributes

/req/temporal/attribute	A temporal characteristic of a feature type shall be defined as a temporal attribute, which is a subtype of feature attribute (7.5.3). The implementation of temporal attributes in UML shall follow the rules for referencing standardized schemas [see /req/uml/integration]. A temporal attribute may be represented in an application schema in either of two ways: case 1: as an ATTRIBUTE of a CLASS that represents a feature, in which case the ATTRIBUTE shall take one of the temporal objects defined in the temporal schema, ISO 19108:2002, as the data type for its value; or case 2: as an ASSOCIATION between the CLASS that represents a feature and one of the temporal objects defined in the temporal schema, ISO 19108:2002. The value of a temporal attribute shall be implemented as a CLASS (i.e. subclass of a temporal object) when any of the following cases apply [see /req/uml/attributeOfAttribute]: 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 19108:2002 need to be referenced explicitly. A temporal attribute may be used as an attribute of an attribute [see /req/uml/attributeOfAttribute], in which case the attribute shall be a subtype of one of the temporal objects defined in ISO 19108:2002. Valid temporal objects, which shall be applied, are given in Table 24.

Table 24 — 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 19 shows part of the application schema for observations and measurements from ISO 19156. Temporal geometric primitives are identified as data types of the temporal attribute types used in the feature types *OM_Observation* and *OM_Measurement*.

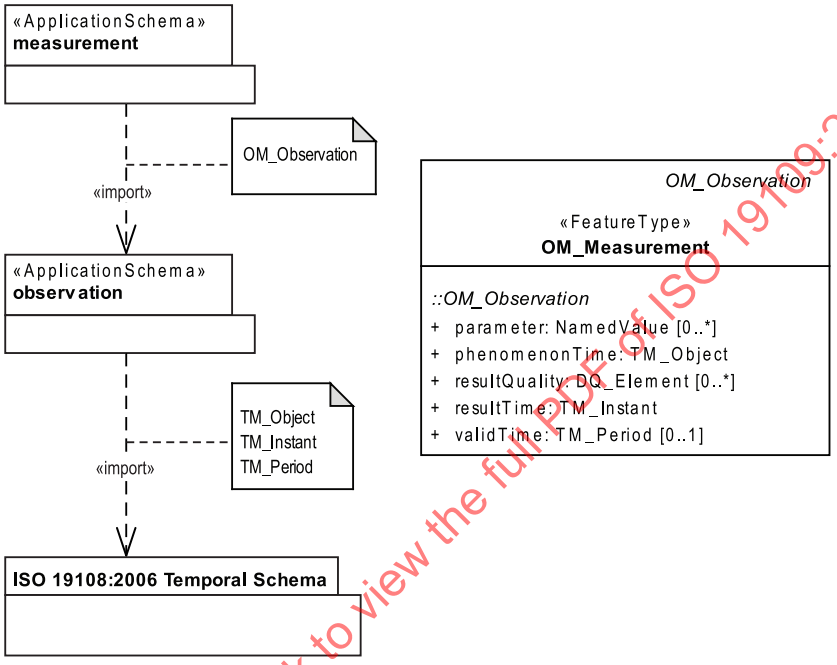


Figure 19 — Example of temporal attribute

EXAMPLE 2 Figure 20 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. BuildingHistory forms a linear graph. Each episode corresponds to a node or edge in the linear graph.

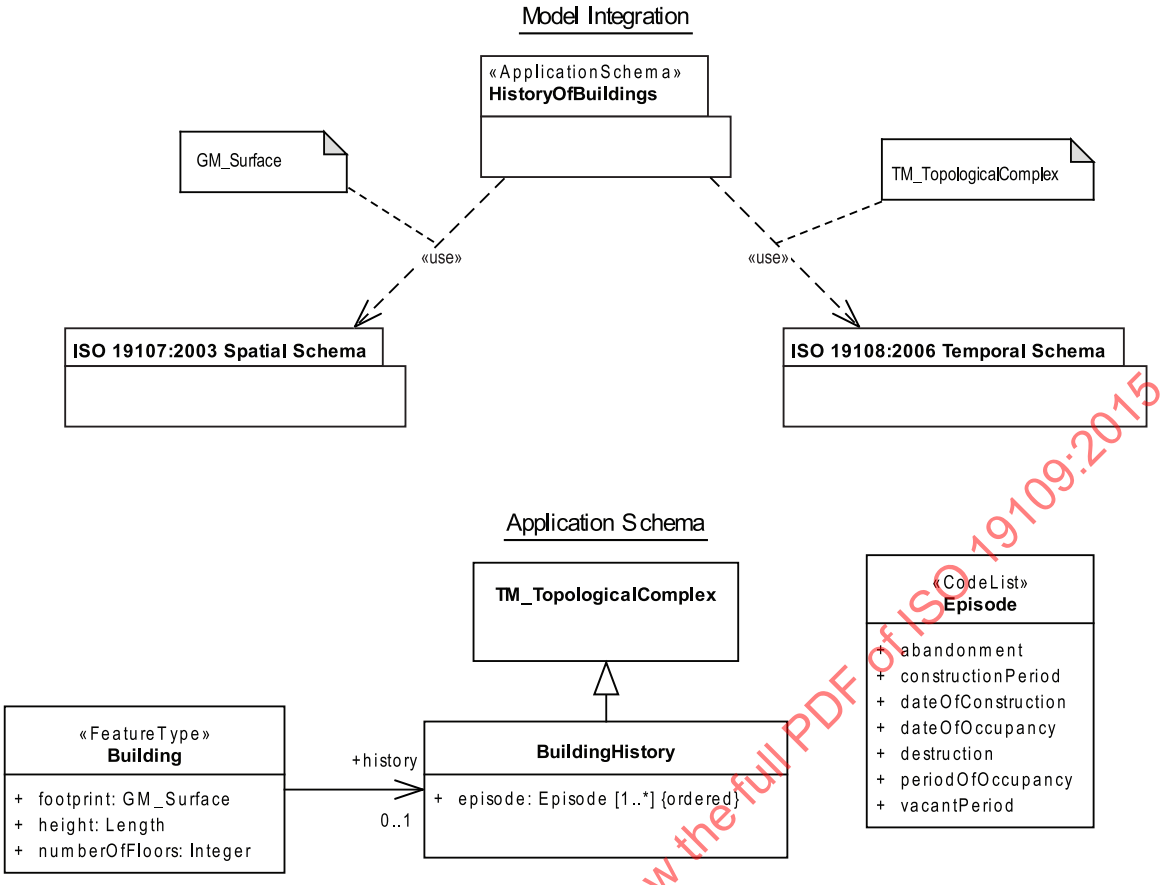


Figure 20 — Example of TM_TopologicalComplex used as a temporal feature attribute

8.6.4 Temporal associations between features

8.6.4.1 Types of relationships

According to the General Feature Model, a temporal association (TemporalAssociation) is a subtype of association between features (FeatureAssociationType) (see Figure 8). There are two types of temporal associations: simple temporal associations and feature succession.

8.6.4.2 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”.

/req/temporal/association	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 operation TM_RelativePosition from the interface TM_Order defined in ISO 19108:2002, 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 [see /req/uml/association].

EXAMPLE 2 Figure 21 shows a simple temporal association implemented as an ASSOCIATION. The schema indicates that roads exist before service stations exist.

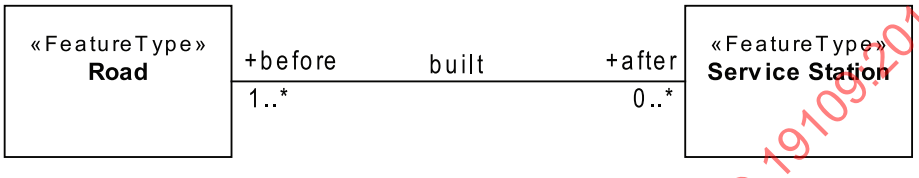


Figure 21 — Example of explicit representation of a simple temporal association

8.6.4.3 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.

/req/temporal/succession	Feature associations of the feature succession type may be instantiated in an application schema as UML associations between feature type classes. Feature associations of the feature succession type may be instantiated in an application schema as self-referent associations of a generic feature class. The names, roles, and multiplicities shall be appropriate for the type of feature succession.

EXAMPLE 1 Figure 22 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.

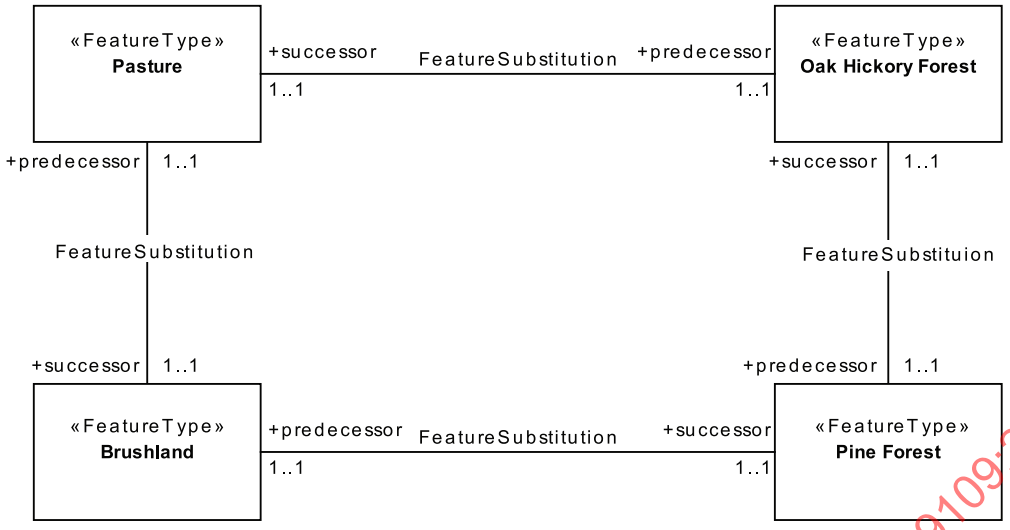


Figure 22 — Example of feature succession between feature types

EXAMPLE 2 Figure 23 shows a feature succession modelled as a self-referent ASSOCIATION of a 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.

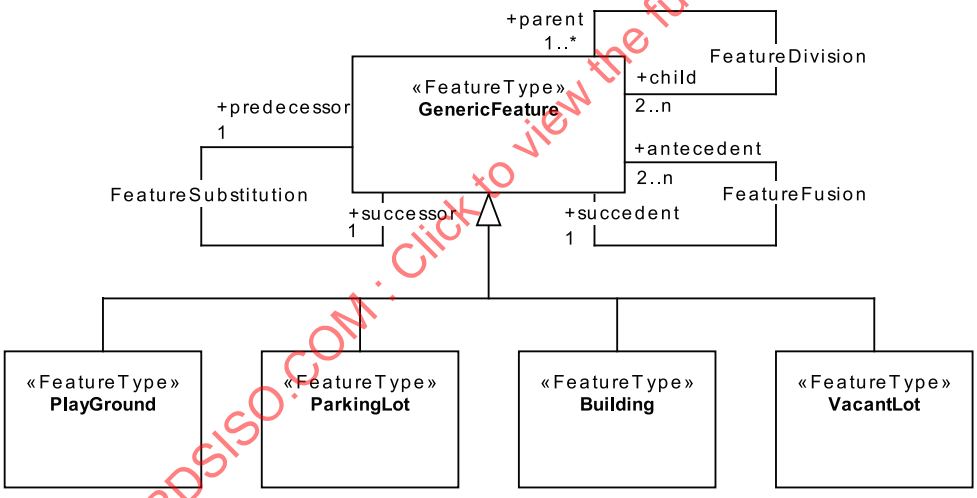


Figure 23 — Example of feature succession at the generic feature level

8.7 Spatial rules

8.7.1 Rules for modelling applications with spatial properties

In 8.7 the requirements for an application schema with feature types that have spatial properties are described. These are formalized as a single requirements class, summarized in Table 25.

Table 25 — Requirements class for applications with spatial properties

Requirements class	/req/spatial
Target type	UML application schema
Dependency	ISO 19107:2003 (Spatial schema)
Dependency	req/uml (Application schema using UML)

Table 25 *(continued)*

Requirement	/req/spatial/object
Requirement	/req/spatial/schema
Requirement	/req/spatial/attribute
Requirement	/req/spatial/aggregate
Requirement	/req/spatial/complex
Requirement	/req/spatial/composite
Requirement	/req/spatial/geom-complex
Requirement	/req/spatial/topo-complex
Requirement	/req/spatial/association
Requirement	/req/spatial/shared
Requirement	/req/spatial/single
Requirement	/req/spatial/interpolation
Requirement	/req/spatial/independent-complex

8.7.2 Use of standard spatial schema

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. However, spatial objects and features are disjoint.

/req/spatial/object	Spatial objects shall not be used as feature types, either directly or by specialization.

Spatial attribute types are required to conform to ISO 19107.

/req/spatial/schema	The value domain of spatial attribute types shall be in accordance with the specifications given by ISO 19107:2003, which provides conceptual schemas for describing the spatial characteristics of features and a set of spatial operators consistent with these schemas.

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.

8.7.3 Spatial attributes

/req/spatial/attribute	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 7.5), and the taxonomy of its values is defined in the spatial schema, ISO 19107:2003. A spatial attribute shall be represented in an application schema in either of two ways: case 1: as an ATTRIBUTE of a CLASS that represents a feature, in which case the ATTRIBUTE shall take one of the spatial objects defined in the spatial schema, ISO 19107:2003, as the data type for its value; or case 2: as an ASSOCIATION between the CLASS that represents a feature and one of the spatial objects defined in the spatial schema, ISO 19107:2003. A spatial attribute shall take a spatial object or type 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 26 lists spatial objects and types that shall be used in an application schema as values for spatial attributes.

NOTE In case 1 the geometry is owned by the feature and in case 2 it is an independent object whose life-cycle is independent of the feature and may be shared between multiple features.

Table 26 — List of valid spatial objects for spatial attributes in an application schema

Geometric objects and types			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
Geometric types				
DirectPosition				
NOTE The table lists only the highest level classes of spatial objects. Subtypes of these may also be used.				

EXAMPLE Figure 26 shows an extract from the application schema for sampling features (ISO 19156). The application schema has a dependency on the spatial schema because it uses the UML-class GM_Point defined in this schema. The feature SF_SamplingPoint has a spatial attribute (shape) which takes an instance of data type GM_Point as its value.

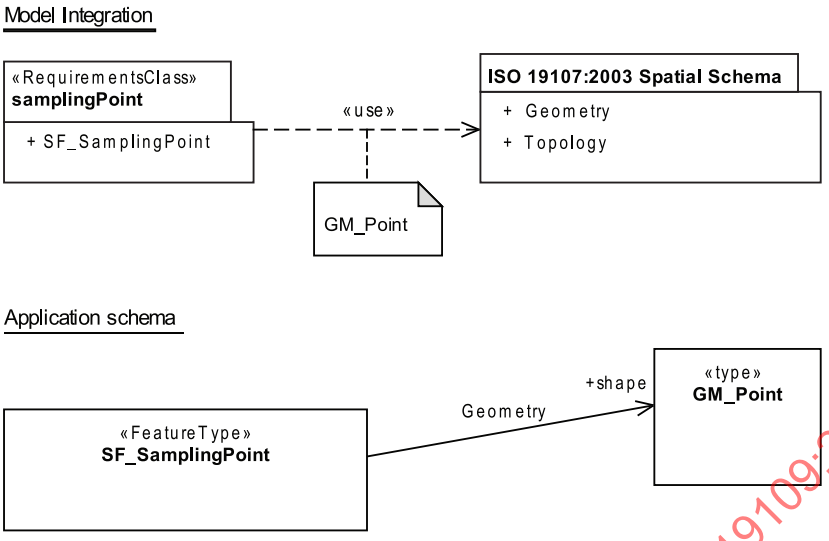


Figure 24 — Example of spatial attributes in UML

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

8.7.4.1 Introduction

The spatial configuration of many features cannot 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.

8.7.4.2 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.

/req/spatial/aggregate	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. 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. 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.

NOTE This rule requires the use of concrete specializations of GM_Aggregate in preference to the use of specializations of GM_Primitive with a multiplicity greater than unity (i.e. GM_MultiPoint *not* GM_Point[0..*], etc.).

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.

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.

EXAMPLE 3 Figure 25 shows a user defined spatial aggregate named 3points1curve, which is data type of a spatial attribute in the application schema.

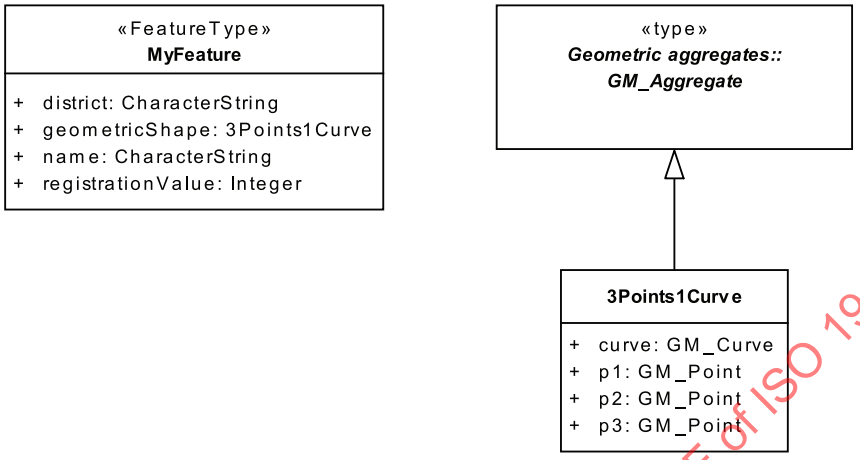


Figure 25 — Example of spatial aggregate defined in application schema

8.7.4.3 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.

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.

/req/spatial/complex	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. Features that share elements of their geometry shall be represented as GM_Complexes that are subcomplexes within a larger GM_Complex.

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.

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 26).

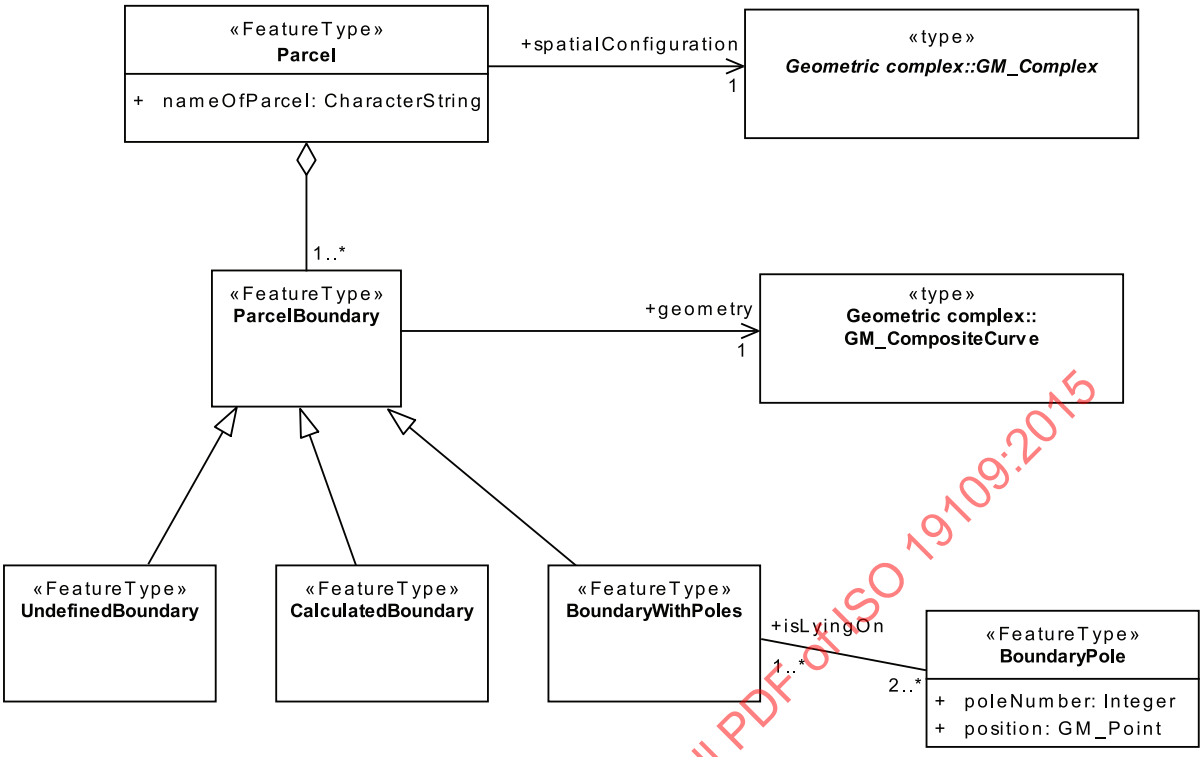


Figure 26 — Example of spatial complex defined in application schema

8.7.4.4 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.

/req/spatial/composite	A GM_Composite shall be used to represent a complex feature that has the geometric properties of multiple geometric primitives.

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 27). 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.

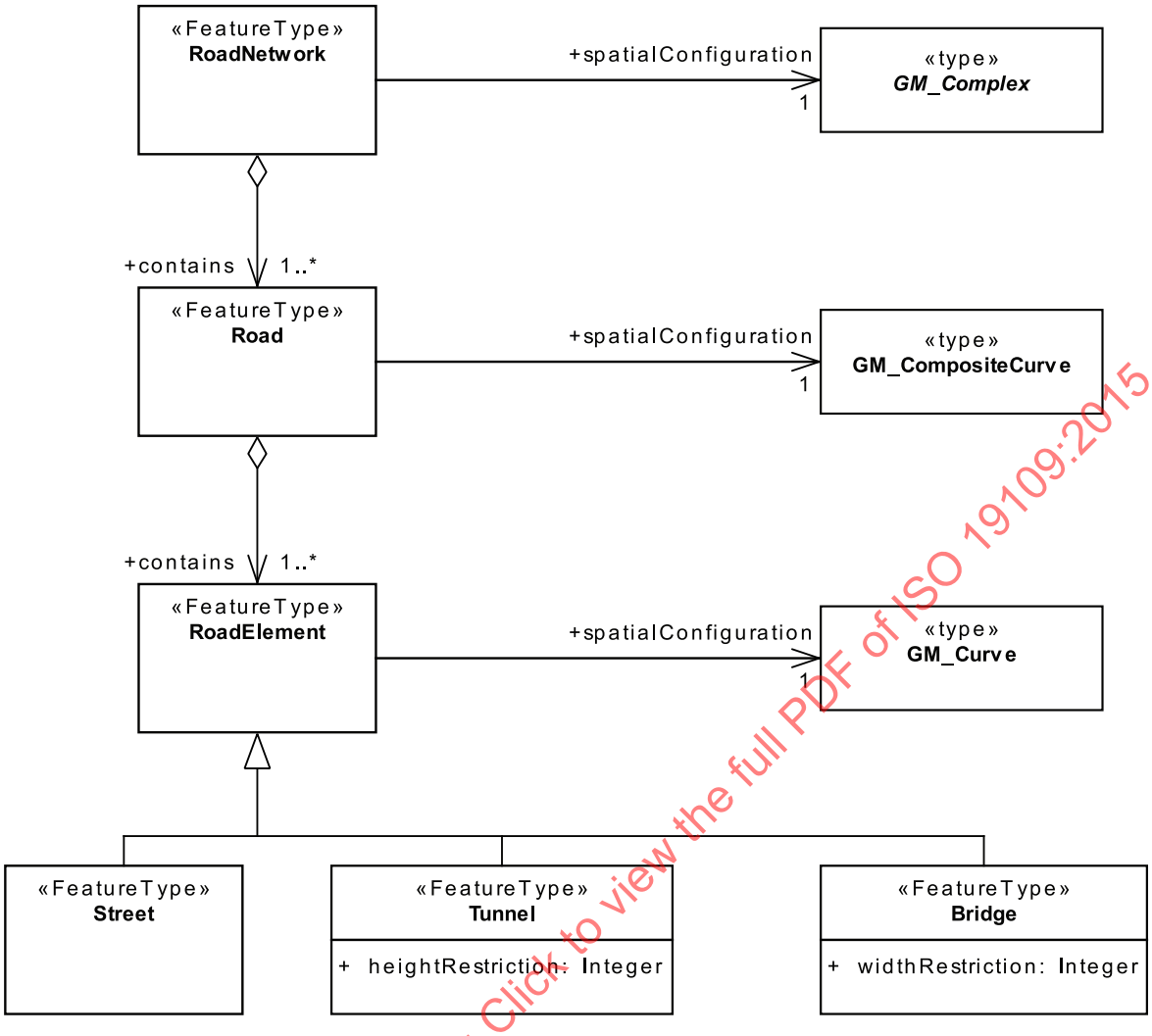


Figure 27 — Example of geometric composites defined in application schema

8.7.4.5 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".

/req/spatial/geom-complex	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 "Hydrologic 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".

8.7.4.6 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.

/req/spatial/topo-complex	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. 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".

8.7.5 Spatial associations between features

According to the General Feature Model (see 7.4), 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 can 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.

/req/spatial/association	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. 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 28 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 C.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.

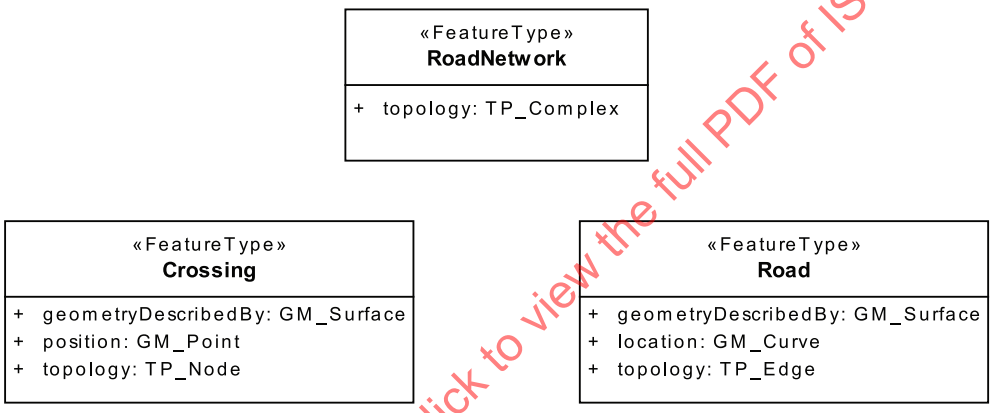


Figure 28 — Example of spatial associations implemented by use of spatial objects

EXAMPLE 2 Figure 29 shows a spatial association between road and crossing described explicitly by the association between the two classes.

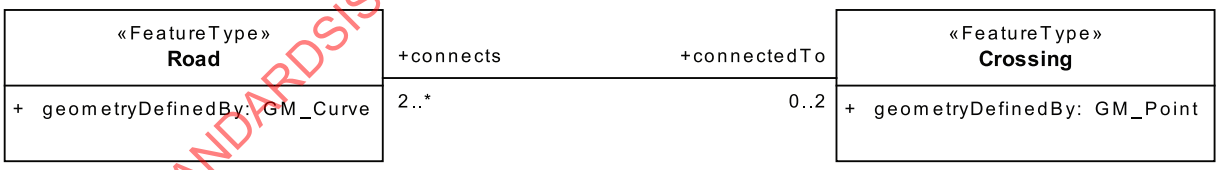


Figure 29 — Example of spatial associations implemented as associations in the application schema

8.7.6 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.

/req/spatial/shared	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. 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.

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.

EXAMPLE 1 Figure 30 A shows a transformer mounted on a pole whose geometry is the same GM_Point that describes the geometry of the pole.

EXAMPLE 2 Figure 30 B shows a similar example with mailboxes and fence-posts. In this case the position is modelled as a class attribute, rather than association.

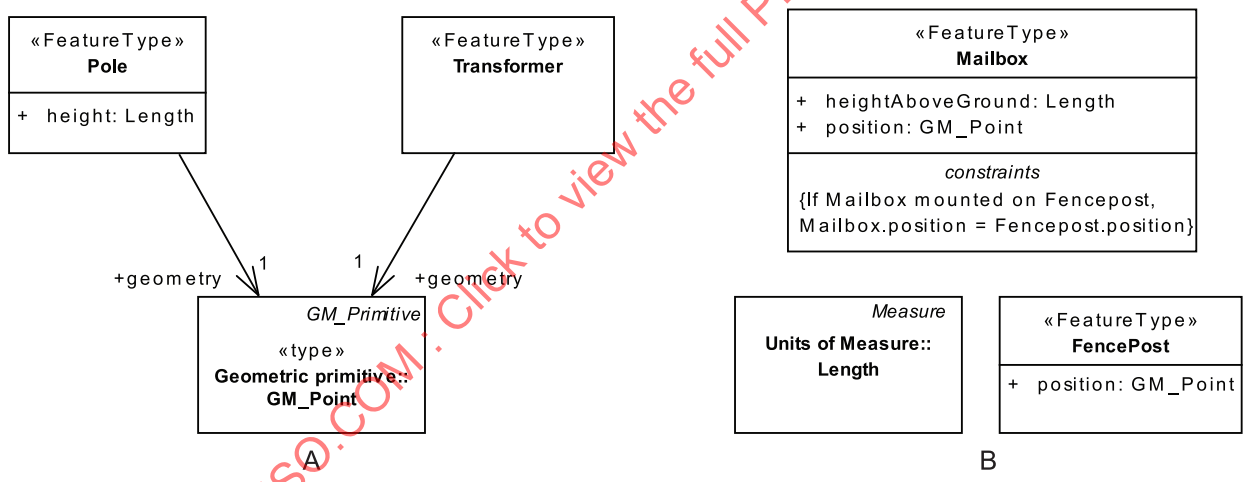


Figure 30 — Examples of features sharing geometry

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 3 Figure 31 shows partial sharing of geometry between features where one GM_Curve may depict (part of) a road and potentially also a bridge simultaneously.

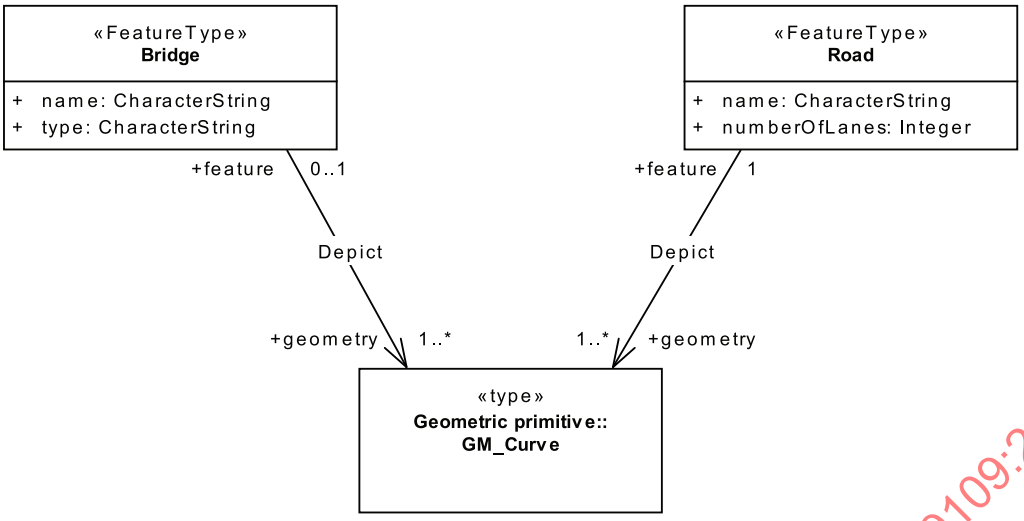


Figure 31 — Example of partial sharing of geometry between features

8.7.7 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 with the same feature. It is recommended that point features, line features and area features are redefined in a generalized form as geographic features.

/req/spatial/single	A point feature shall take a GM_Point as the value of its spatial attribute. A line feature shall take a GM_Curve or a GM_CompositeCurve as the value of its spatial attribute. An area feature shall take a GM_Surface or a GM_CompositeSurface as the value of its spatial attribute. A solid feature shall take a GM_Solid or a GM_CompositeSolid as the value of its spatial attribute.

8.7.8 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. 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. Different patches in one surface do not have to be of the same type. If an application requires, for example, that all the segments in the GM_Curve or GM_Surface have the same interpolation, a subtype of GM_Curve or GM_Surface should be introduced. 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.

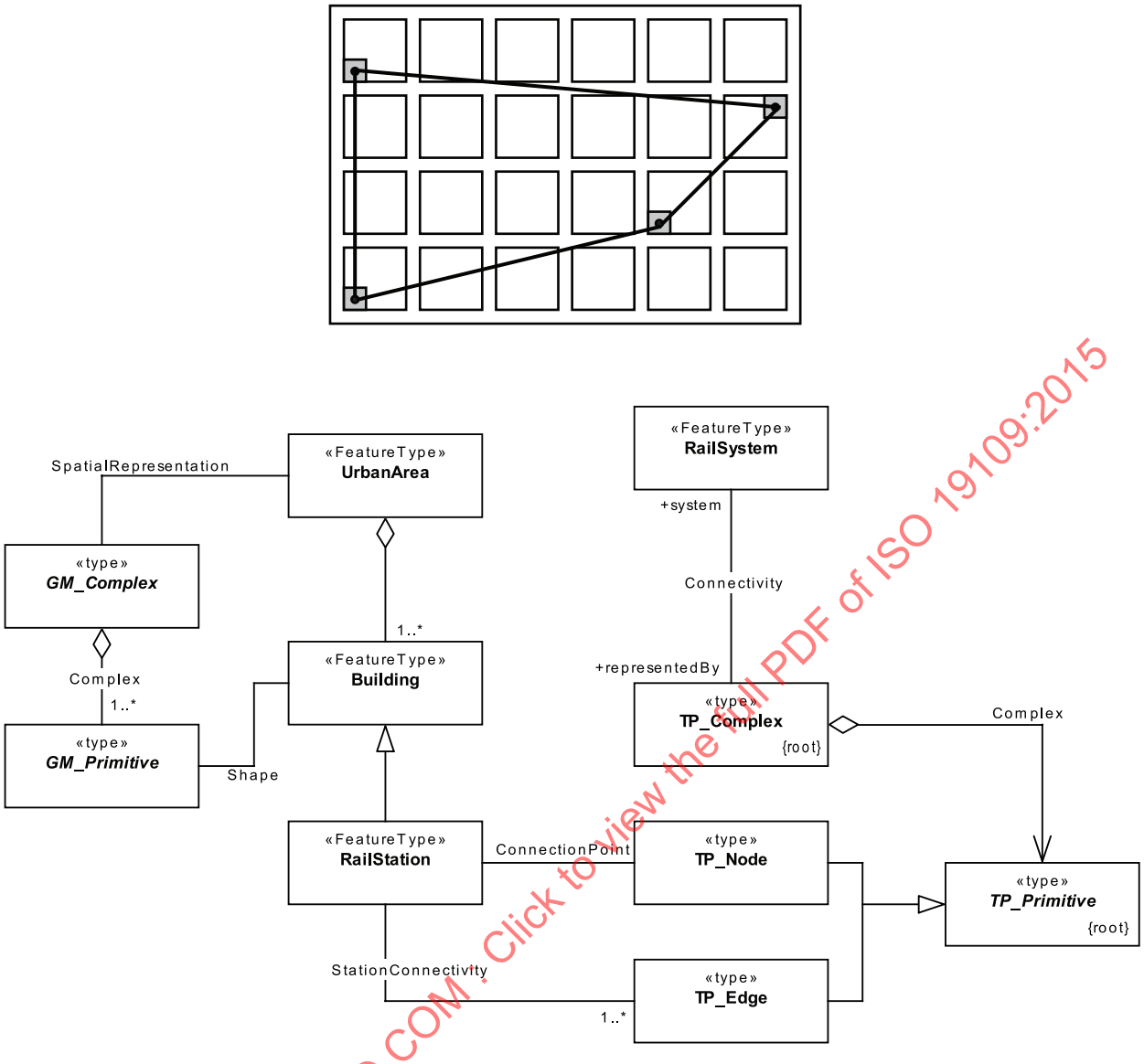
/req/spatial/interpolation	If any existing subtype of GM_CurveSegment or GM_SurfacePatch fulfils the requirements of interpolation method, this subtype shall be used in the application schema. If, and only if, none of the existing classes fulfils the requirements of the application schema, the application 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.

8.7.9 Independent spatial complexes

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

/req/spatial/independent-complex	Different spatial representations of the same feature are allowed, but they must belong to different complexes. 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 32. 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 32 — Example of independent spatial complexes in an application schema

EXAMPLE 2 Figure 33 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.

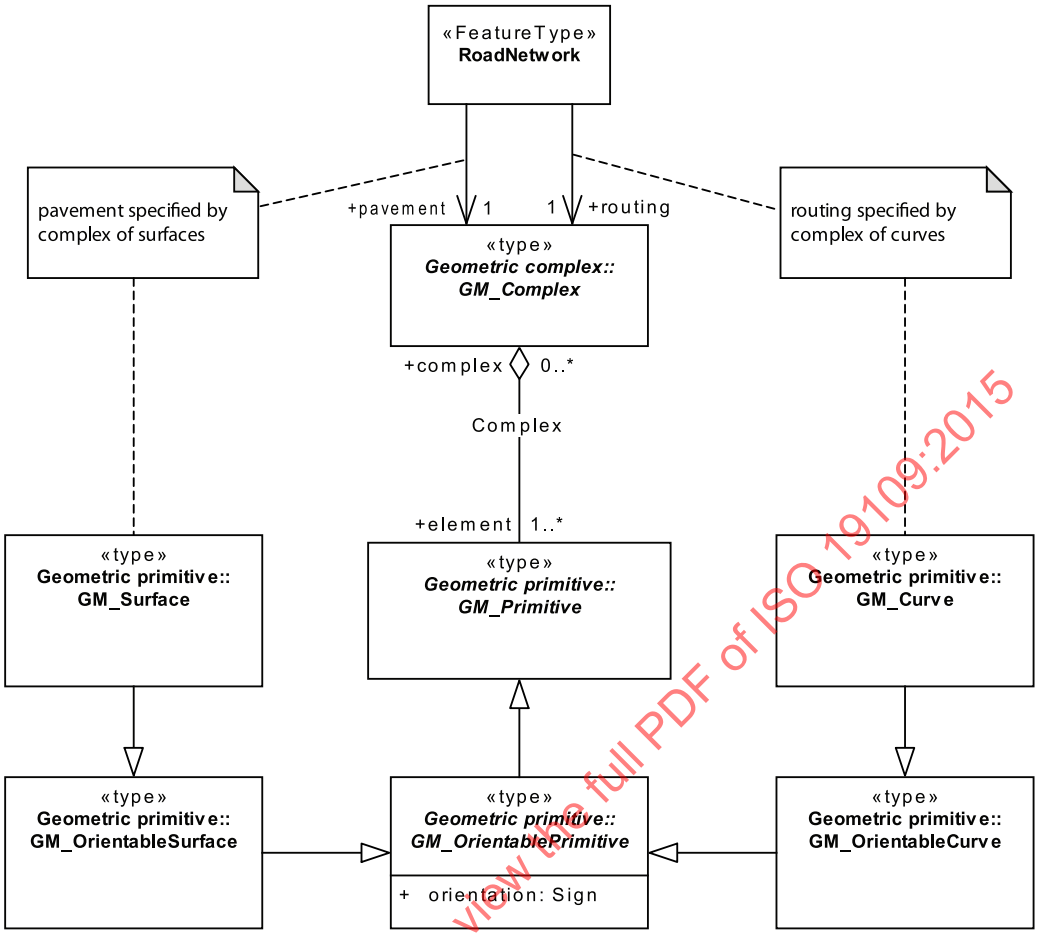


Figure 33 — Example of independent spatial complexes

8.8 Rules for use of coverage functions

Coverage functions are used to describe characteristics of real-world phenomena that vary over space and/or time. Typical examples are temperature, elevation and precipitation. A coverage contains a set of such values, each associated with one of the elements in a spatial, temporal or spatio-temporal domain. Typical spatial domains are point sets (e.g. sensor locations), curve sets (e.g. contour lines), grids (e.g. orthoimages, elevation models), etc. A property whose value varies as a function of time may be represented as a temporal coverage or time-series. A continuous coverage is associated with a method for interpolating values at spatial positions between the elements of a domain, e.g. between two points or contour lines.

In 8.8 the requirements for an application schema with feature types that have coverage-valued properties are described. These are formalized as a single requirements class, summarized in Table 27.

Table 27 — Requirements class for applications with coverage-valued properties

Requirements class	/req/coverage
Target type	UML application schema
Dependency	ISO 19123:2005 (Coverage geometry and functions)
Dependency	/req/uml (Application schema using UML)
Requirement	/coverage/req/schema

/req/coverage/schema	Any specification of a coverage function in an application schema shall be in accordance with ISO 19123:2005 and 7.5.8. An application schema package that uses coverage functions shall import the coverage schema specified by ISO 19123:2005. A coverage function shall be defined as a property of a feature type where the type of the property value is a realization of one of the types given in Table 28.

NOTE This approach creates an implementation profile of the relevant ISO 19123 types in the application schema.

Table 28 — List of valid coverage types in an application schema

Abstract coverage types	Discrete coverages	Continuous coverages
CV_Coverage CV_DiscreteCoverage CV_ContinuousCoverage	CV_DiscretePointCoverage CV_DiscreteGridPointCoverage CV_DiscreteCurveCoverage CV_DiscreteSurfaceCoverage CV_DiscreteSolidCoverage	CV_ThiessenPolygonCoverage CV_ContinousQuadrilateralGridCoverage CV_HexagonalGridCoverage CV_TINCoverage CV_SegmentedCurveCoverage

EXAMPLE 1 Figure 34 illustrates an application schema that uses solid-coverages to represent the grade of ore minerals in a mine.

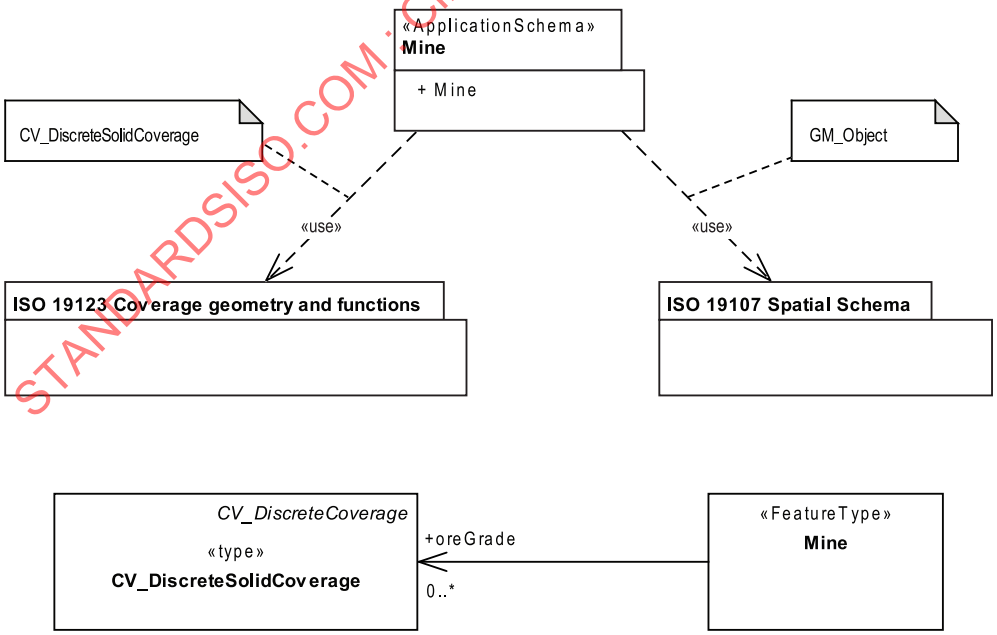


Figure 34 — Example of a coverage-valued property, with package integration of application schema

EXAMPLE 2 Figure 35 illustrates an application schema that uses a discrete point coverage to represent the water level at a gauging station. The points in the coverage domain are given relative to a temporal coordinate system, or a 1-D coordinate reference system in which the single axis measures time.

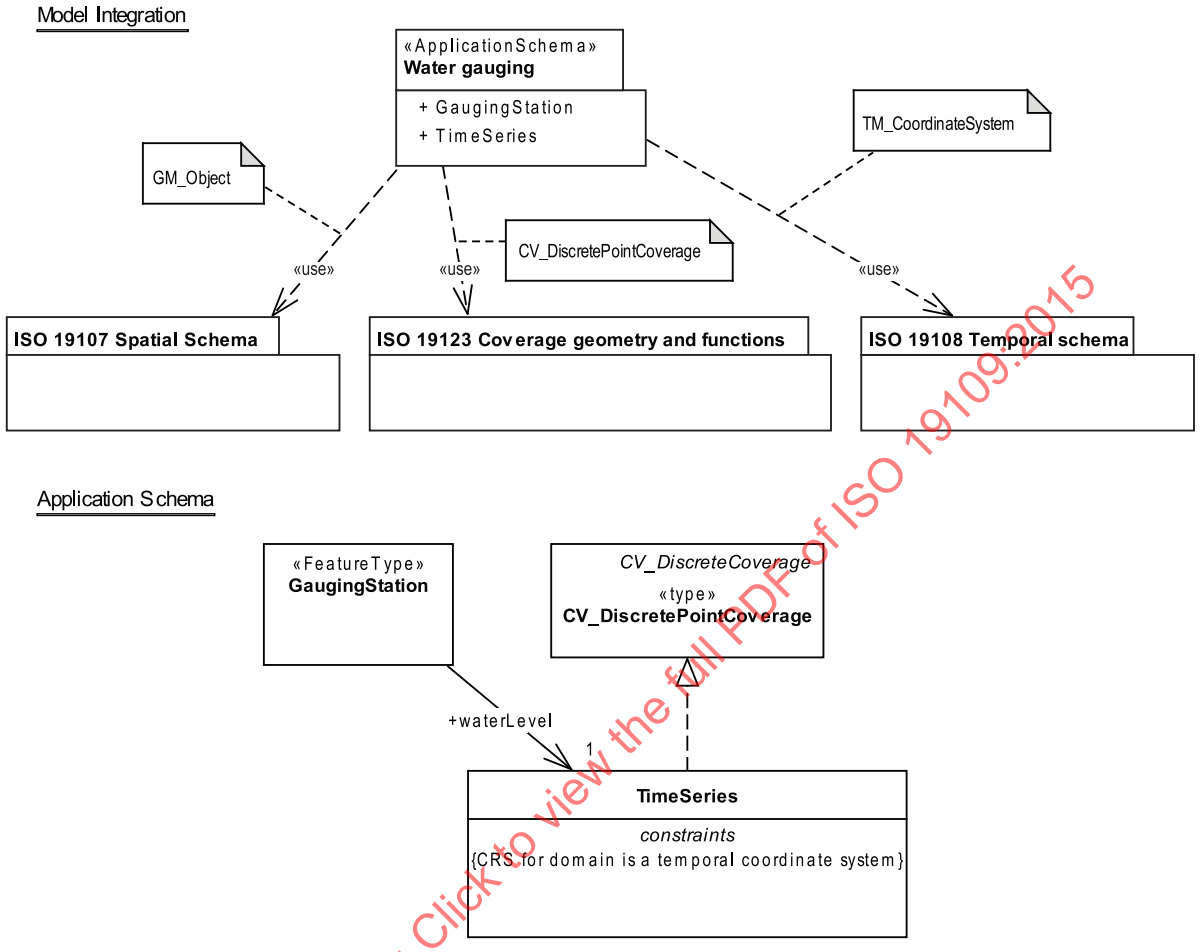


Figure 35 — Example of a coverage-valued property using a specialized coverage

8.9 Rules for the use of observations

An observation is an act in which a procedure is used to estimate the value of a property. The properties of an observation provide metadata for the value assigned to a feature property, for example: the description of the procedure and the time that the observation of the value was made. Observation is an important instantiation of the ValueAssignment metaclass (7.4.10). The relationships between observations, and the features and properties whose value assignment they document, are shown in Figure 36.

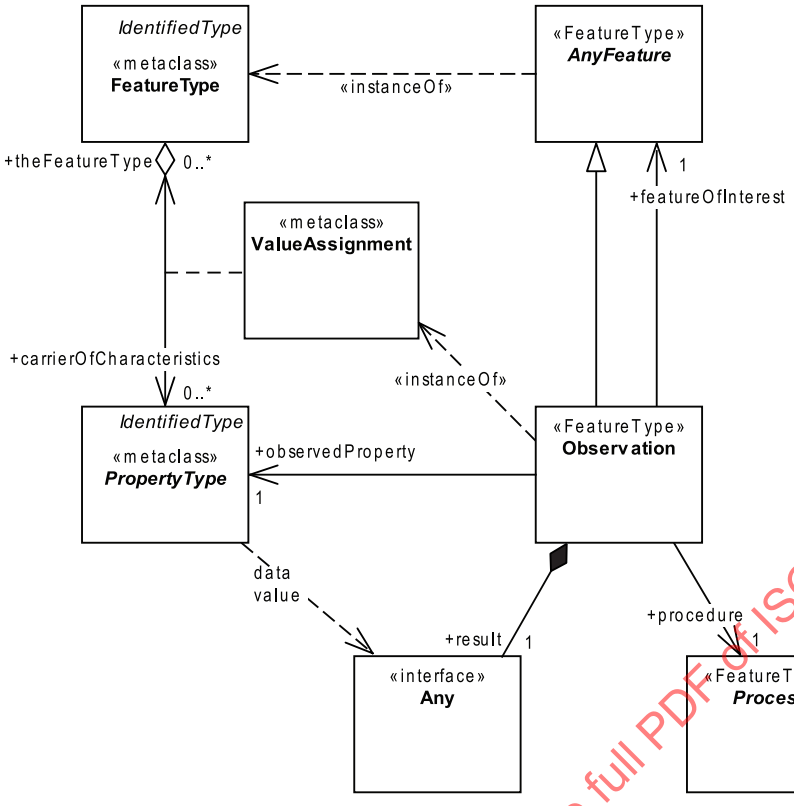


Figure 36 — Key elements of the observation schema (ISO 19156) related to General Feature Model (Clause 7)

It is unusual to have relationships other than dependency or realization that cross metalevels. However, they are included in the model shown in Figure 36 because this expresses requirements of property-value metadata. Without a common mechanism every application schema with this requirement is likely to model it differently. This model was introduced in ISO 19156 to provide a common pattern, and is included here to support a rule for application schemas.

An application schema may include the standard observation feature types through integration of the schema provided in ISO 19156. If necessary, specific observation types may be derived by specialization of one of the types provided in ISO 19156.

In 8.9 the requirements for an application schema with feature types that involve value assignment through observation are described. These are formalized as a single requirements class, summarized in Table 29.

Table 29 — Requirements class for applications involving observations

Requirements class	/req/observation
Target type	UML application schema
Dependency	ISO 19156:2011 (Observations and Measurements)
Dependency	/req/uml (Application schema using UML)
Requirement	/req/observation/model
Requirement	/req/observation/property
Recommendation	/rec/observation/estimated

ISO 19156 provides a model for observations.

/req/observation/model	Observation features in an application schema shall be modelled according to the schema for observations provided in ISO 19156:2011.

The model in Figure 36 implies some consistency requirements relating the observed property, feature types and the application schema.

/req/observation/property	The value of observed-property in data conforming to the application schema shall be constrained to be a property-type from the feature catalogue for the application schema, and shall be a characteristic property-type for the feature-type of the feature-of-interest.

The stereotype «Estimated» can be used in an application schema to indicate feature properties whose values are assigned by observation.

/rec/observation/estimated	If a feature-attribute or association role is stereotyped «Estimated», the metadata for the assigned value may be provided by an instance of the OM_Observation class or one of its specializations.

EXAMPLE Figures 37 and 38 illustrate application schemas in which a property of a feature type has the stereotype «estimated», with suitable observation-types that provide metadata about the value assignment.

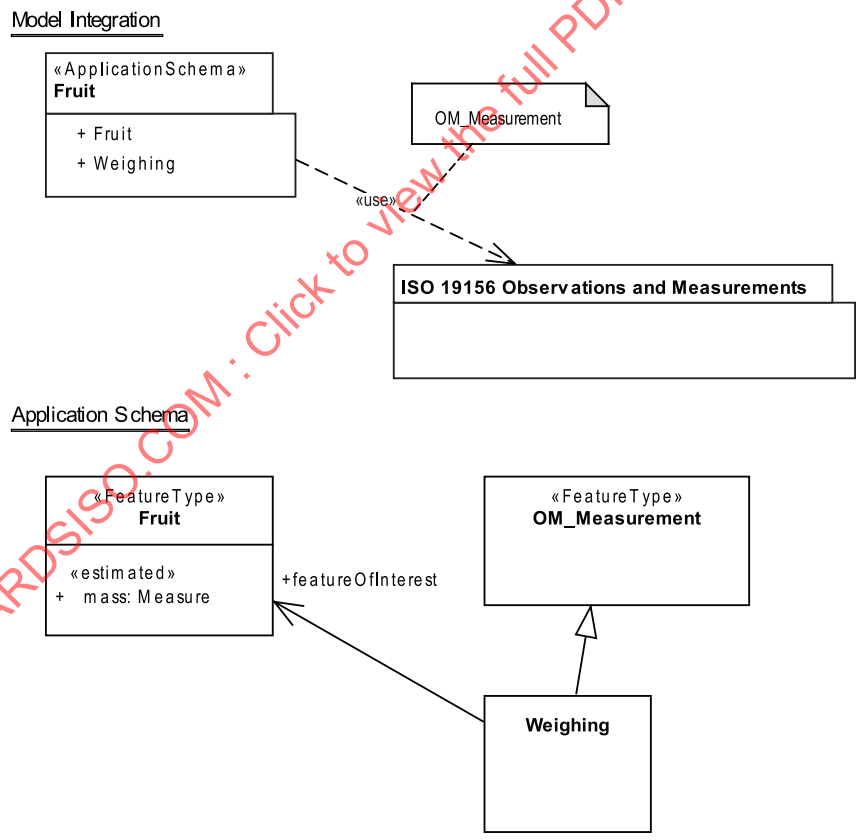


Figure 37 — Example 1 of links between properties and evidence for the assignment of values

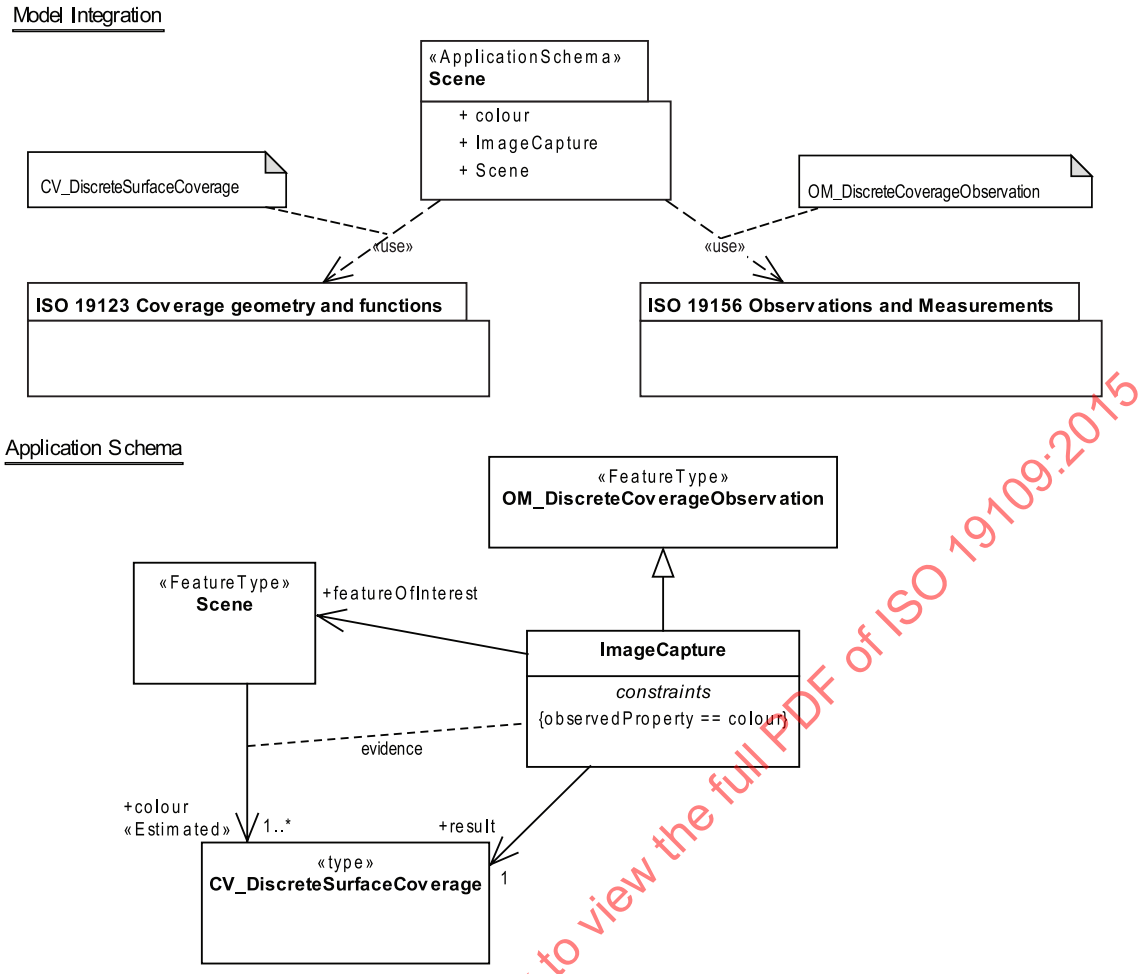


Figure 38 — Example 2 of links between properties and evidence for the assignment of values

8.10 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 39 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.

In 8.10 the requirements for an application schema with feature types that use geographic identifiers are described. These are formalized as a single requirements class, summarized in Table 30.

Table 30 — Requirements class for applications using geographic identifiers

Requirements class	/req/identifier
Target type	UML application schema
Dependency	ISO 19112:2003
Dependency	/req/uml (Application schema using UML)
Requirement	/req/identifier/general

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

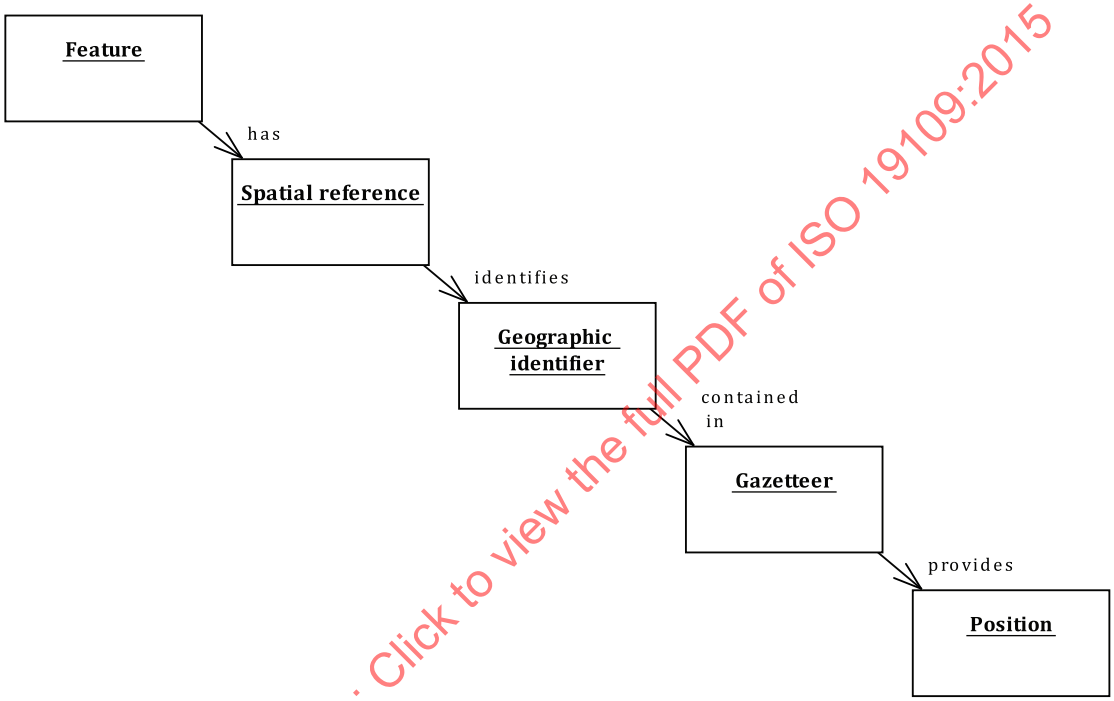


Figure 39 — Spatial referencing by geographic identifiers

/req/identifier/general	The value domain of attributes using spatial referencing by geographic identifiers shall identify objects that are in accordance with ISO 19112:2003. The geographic identifier shall be referenced from the application schema by an attribute (an instance of LocationAttributeType) which shall carry the value of the spatial reference.

EXAMPLE Figure 40 shows the feature type Customer which is located by the attribute postDistrict. Note that while the target of the association is of type SI_LocationInstance, the association is by reference, using the geographic identifier, which is typically a character string.

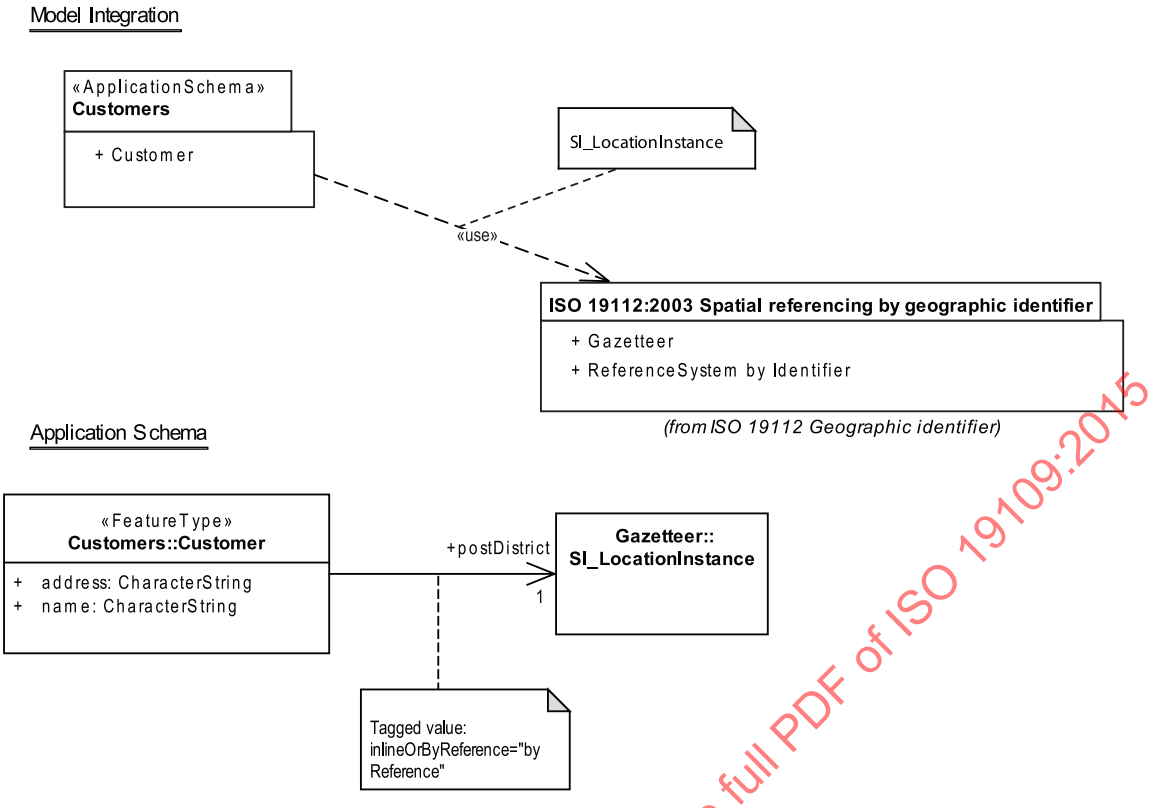


Figure 40 — Example of spatial referencing by geographic identifier

8.11 Code lists, vocabularies, lexicons

In many applications, there is a well-established code list already used in a thematic community. Where possible, such code lists should be re-used in application schemas.

In 8.11 the requirements for the use of code lists in application schemas are described. These are formalized as a single requirements class, summarized in Table 31.

Table 31 — Requirements class for applications using code lists

Requirements class	/req/codeList
Target type	UML application schema
Dependency	ISO 19103:2015
Dependency	/req/uml (Application schema using UML)
Requirement	/req/codeList/management
Requirement	/req/codeList/formalization

A well managed code list will meet a number of requirements.

/req/codeList/management	A code list shall be managed by a competent organisation; The code list and each of its values shall be denoted by a persistent identifier, such as a URI; All values of a code list shall be available, including those that have been deprecated, retired or superseded.

The code list formalization defined in ISO 19103 is used in application schemas.

/req/codeList/formalization	Where a class attribute takes a value from a code list, the code list shall be formalized in the application schema according to ISO 19103:2015.

8.12 Linguistic adaptation

In some contexts it is necessary or desirable for key information in an application schema to be available in more than one language, and it is preferable if the alternative designations and definitions are stored in the model itself. In 8.12 the requirements for annotating a UML application schema for use in multi-lingual environments are described. These are formalized as a single requirements class, summarized in Table 32.

Table 32 — Requirements class for application schemas in multi-lingual environments

Requirements class	/req/multi-lingual
Target type	UML application schema
Dependency	IETF RFC 5646
Dependency	/req/uml (Application schema using UML)
Requirement	/req/multi-lingual/package
Requirement	/req/multi-lingual/feature

/req/multi-lingual/package	A PACKAGE containing an application schema shall use a TAGGED VALUE "language" to indicate the primary language used for names and definitions in the application schema. The TAGGED VALUE shall use a value taken from IETF RFC 5646. A PACKAGE shall use a TAGGED VALUE "designation" to contain the name of the PACKAGE in an alternative language to the primary language. The TAGGED VALUE may be repeated for multiple languages. The value shall be formatted according to the pattern: "{Name}"@{language} Where {Name} is the name in a desired language, and {language} is its language code, taken from IETF RFC 5646.

EXAMPLE "Observations et Mesures"@fr

NOTE This follows the compact syntax for localized strings used in the W3C N-Triples RDF encoding.[13]

/req/multi-lingual/feature	A FeatureType or PropertyType shall use a TAGGED VALUE "designation" to record its name in an alternative language to the primary language. A FeatureType or PropertyType shall use a TAGGED VALUE "definition" to record its textual definition in an alternative language to the primary language. The TAGGED VALUEs "designation", "definition" and "description" may be repeated for multiple languages. The value of these TAGGED VALUEs shall be formatted according to the pattern: "{Name}"@{language} Where {Name} is the name in a desired language, and {language} is its language code, taken from IETF RFC 5646.

NOTE The tagged value "description" is introduced in /req/uml/documentation Rule 3) (8.2.4).

Annex A
(normative)

Abstract test suite

A.1 Introduction

The test suite is structured into 12 conformance classes (Clause 2). Each test relates to one or more specific requirements, which are explicitly indicated in the description of the test.

A.2 Meta-model

A.2.1 Conceptual schema language

The test for use of a CSL is as follows:

a) test identifier: /conf/general/csl;

b) test purpose: Verify that the application schema is formalized using a conceptual schema language;

c) test method: Inspect the application schema to verify that it is expressed consistently using a recognised conceptual schema language;

d) reference: /req/general/csl;

e) test type: basic.

A.2.2 Schema integration

The test for integration of an application schema with other schemas is as follows:

a) test identifier: /conf/general/integration;

b) test purpose: Verify that dependencies on other schemas are recorded;

c) test method: Inspect the application schema to verify that if any components from other application schemas or standard schemas are used, the dependencies on the external schemas are recorded explicitly;

d) reference: /req/general/integration;

e) test type: basic.

A.2.3 Features

The test for modelling features is as follows:

a) test identifier: /conf/general/feature;

b) test purpose: Verify that feature types and their properties are modelled according to the metamodel;

c) test method: Inspect the application schema to verify that feature types, feature attribute types, association roles, and feature operations have been defined, named and documented according to the elements and rules specified in 7.4.5, 7.4.7, 7.4.8 and 7.4.9;

d) reference: /req/general/feature, /req/general/attribute, /req/general/operation, /req/general/association-role;

e) test type: basic.

A.2.4 Value assignment

The test for modelling value assignment metadata is as follows:

a) test identifier: /conf/general/value-assignment;

b) test purpose: Verify that types providing value assignment metadata are modelled according to the metamodel;

c) test method: Inspect the application schema to verify that value assignment type definitions have been defined according to the elements and rules specified in 7.4.10;

d) reference: /req/general/value-assignment;

e) test type: basic.

A.2.5 Feature associations

The test for modelling feature associations is as follows:

a) test identifier: /conf/general/association;

b) test purpose: Verify that feature associations are modelled according to the metamodel;

c) test method: Inspect the application schema to verify that feature associations have been defined according to the elements and rules specified in 7.4.11;

d) reference: /req/general/association;

e) test type: basic.

A.2.6 Feature inheritance

The test for modelling feature inheritance is as follows:

a) test identifier: /conf/general/inheritance;

b) test purpose: Verify that feature inheritance is modelled according to the metamodel;

c) test method: Inspect the application schema to verify that feature inheritance has been defined according to the elements and rules specified in 7.4.12;

d) reference: /req/general/inheritance;

e) test type: basic.

A.2.7 Constraints

The test for modelling constraints is as follows:

a) test identifier: /conf/general/constraint;

b) test purpose: Verify that constraints have been specified according to the metamodel;

c) test method: Inspect the application schema to verify that any constraints have been defined according to the rules specified in 7.7;

d) reference: /rec/general/inheritance;

e) test type: basic.

A.3 Creating application schemas in UML

A.3.1 UML profile

The test for use of the UML profile is as follows:

a) test identifier: /conf/uml/profile;

b) test purpose: Verify that the application schema uses the specified UML profile;

c) test method: Inspect the application schema to verify that the CSL used is UML following the constraints, stereotypes and tagged values described in 8.2.2 and Table 17;

d) reference: /req/uml/profile;

e) test type: capability.

A.3.2 Packaging

The test for packaging the application schema is as follows:

a) test identifier: /conf/uml/packaging;

b) test purpose: Verify that the application schema is contained in a UML package with the correct name and version recorded;

c) test method: Inspect the application schema to verify that all elements owned by the application schema are contained within a package with the stereotype «ApplicationSchema» where the name of the package matches the name of the application schema, and the application schema version is recorded in a tagged value;

d) reference: /req/uml/packaging;

e) test type: capability.

A.3.3 Documentation

The test for documenting the application schema is as follows:

a) test identifier: /conf/uml/documentation;

b) test purpose: Verify that the application schema is properly documented;

c) test method: Inspect the application schema to verify that the definition of each primary element (packages, classifiers, properties) is recorded using the primary exportable documentation facility, that any secondary documentation is recorded in a tagged value “description”, and that any reference to a corresponding catalogue entry is recorded in a tagged value “catalogue-entry”;

d) reference: /req/uml/documentation;

e) test type: capability.

A.3.4 Integrating an application schema

The test for integrating an application schema is as follows:

a) test identifier: /conf/uml/integration;

b) test purpose: Verify that the application schema has been integrated with other schemas according to rules specified in this International Standard;

