



## ISO/TC 211 N 3603

---

Replaces N 3519

2013-07-29

Number of pages: 113

### ISO/TC 211 Geographic information/Geomatics Secretariat: SN (Norway)

**Document type:** Other draft

**ISO reference number:** 19109

**Title:** **ISO 19109 Geographic information — Rules for application schema - Submission of the final CD to ISO for issuing as a Draft International Standard**

**Status:** Enquiry (DIS) will be sent out from the ISO Central Secretariat and the ballot will open on 2013-10-10 and close on 2014-03-10

**Source:** ISO/TC 211/WG 9/PT 19109 Editing Committee

**Expected action:** Info

**Reference:** N 3515, 3604

**Email to secretary:** [bjs@standard.no](mailto:bjs@standard.no)

**Committee URL:** <http://isotc.iso.org/livelink/livelink/open/tc211> and <http://www.isotc211.org>



**ISO TC 211/SC**

Date: 2013-07-22

**ISO/DIS 19109**

ISO TC 211/SC /WG 9

Secretariat: SN

## **Geographic information — Rules for application schema**

*Information géographique — Règles de schéma d'application*

### **Warning**

This document is not an ISO International Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an International Standard.

Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Document type: International Standard

Document subtype:

Document stage: (40) Enquiry

Document language: E

C:\Users\by16\Desktop\ISO 19109\ISO\_DIS\_19109\_(E).doc STD Version 2.1c2

### Copyright notice

This ISO document is a Draft International Standard and is copyright-protected by ISO. Except as permitted under the applicable laws of the user's country, neither this ISO draft nor any extract from it may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, photocopying, recording or otherwise, without prior written permission being secured.

Requests for permission to reproduce should be addressed to either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office  
Case postale 56 • CH-1211 Geneva 20  
Tel. + 41 22 749 01 11  
Fax + 41 22 749 09 47  
E-mail [copyright@iso.org](mailto:copyright@iso.org)  
Web [www.iso.org](http://www.iso.org)

Reproduction may be subject to royalty payments or a licensing agreement.

Violators may be prosecuted.

# Contents

Page

Foreword .....	vii
Introduction.....	viii
<b>1</b> <b>Scope</b> .....	<b>1</b>
<b>2</b> <b>Conformance</b> .....	<b>1</b>
2.1 <b>General</b> .....	<b>1</b>
2.2 <b>Meta-model</b> .....	<b>2</b>
2.3 <b>UML application schema</b> .....	<b>2</b>
2.4 <b>Profiling standard schema</b> .....	<b>2</b>
2.5 <b>Metadata</b> .....	<b>2</b>
2.6 <b>Quality</b> .....	<b>2</b>
2.7 <b>Temporal</b> .....	<b>3</b>
2.8 <b>Spatial</b> .....	<b>3</b>
2.9 <b>Coverages</b> .....	<b>3</b>
2.10 <b>Observations</b> .....	<b>3</b>
2.11 <b>Spatial referencing by identifiers</b> .....	<b>4</b>
2.12 <b>Code-list</b> .....	<b>4</b>
2.13 <b>Multi-lingual support</b> .....	<b>4</b>
<b>3</b> <b>Normative references</b> .....	<b>4</b>
<b>4</b> <b>Terms and definitions</b> .....	<b>5</b>
<b>5</b> <b>Presentation and abbreviations</b> .....	<b>7</b>
5.1 <b>Presentation</b> .....	<b>7</b>
5.1.1 <b>General</b> .....	<b>7</b>
5.1.2 <b>Conformance class</b> .....	<b>7</b>
5.1.3 <b>Requirements class</b> .....	<b>8</b>
5.1.4 <b>Rules</b> .....	<b>8</b>
5.1.5 <b>Identifiers</b> .....	<b>8</b>
5.1.6 <b>Conceptual schemas</b> .....	<b>9</b>
5.1.7 <b>Descriptions of concepts</b> .....	<b>9</b>
5.2 <b>Abbreviations</b> .....	<b>9</b>
5.3 <b>Package abbreviations</b> .....	<b>9</b>
<b>6</b> <b>Context</b> .....	<b>10</b>
6.1 <b>Purpose of an application schema</b> .....	<b>10</b>
6.2 <b>Rules for application schema</b> .....	<b>10</b>
6.3 <b>Application schema supporting data interchange</b> .....	<b>11</b>
6.3.1 <b>Introduction</b> .....	<b>11</b>
6.3.2 <b>Data interchange by transfer</b> .....	<b>11</b>
6.3.3 <b>Data interchange by transactions</b> .....	<b>12</b>
<b>7</b> <b>Principles for defining features</b> .....	<b>13</b>
7.1 <b>General</b> .....	<b>13</b>
7.2 <b>Features, Coverages and Properties</b> .....	<b>14</b>
7.2.1 <b>Features</b> .....	<b>14</b>
7.2.2 <b>Coverages</b> .....	<b>15</b>
7.2.3 <b>Properties and observations</b> .....	<b>16</b>
7.3 <b>Features and the application schema</b> .....	<b>16</b>
7.4 <b>The General Feature Model</b> .....	<b>17</b>
7.4.1 <b>Introduction</b> .....	<b>17</b>
7.4.2 <b>The purpose of the GFM</b> .....	<b>18</b>
7.4.3 <b>The main structure of the GFM</b> .....	<b>18</b>

## ISO/DIS 19109

7.4.4	IdentifiedType.....	19
7.4.5	FeatureType.....	20
7.4.6	PropertyType.....	21
7.4.7	AttributeType.....	21
7.4.8	Operation.....	22
7.4.9	FeatureAssociationRole.....	22
7.4.10	ValueAssignment.....	23
7.4.11	FeatureAssociationType.....	25
7.4.12	InheritanceRelation.....	25
7.5	Attributes of feature types.....	26
7.5.1	Introduction.....	26
7.5.2	SpatialAttributeType.....	27
7.5.3	TemporalAttributeType.....	27
7.5.4	QualityAttributeType.....	27
7.5.5	LocationAttributeType.....	27
7.5.6	MetadataAttributeType.....	28
7.5.7	ThematicAttributeType.....	28
7.5.8	CoverageFunctionAttributeType.....	28
7.6	Relationships between feature types.....	28
7.6.1	Introduction.....	28
7.6.2	InheritanceRelation.....	28
7.6.3	FeatureAssociationType.....	29
7.7	Constraints.....	30
8	Rules for application schema in UML.....	30
8.1	The application modelling process.....	30
8.2	The application schema.....	31
8.2.1	General.....	31
8.2.2	Conceptual schema language for application schemas.....	32
8.2.3	Packaging and identification of an application schema.....	33
8.2.4	Documentation of an application schema.....	33
8.2.5	Integration of application schemas and standard schemas.....	34
8.2.6	Modelling structures in UML.....	36
8.3	Domain profiles of standard schemas in UML.....	41
8.3.1	Introduction.....	41
8.3.2	Adding information to a standard schema.....	41
8.3.3	Tailored use of standard schemas.....	42
8.4	Rules for use of metadata schema.....	43
8.4.1	Introduction.....	43
8.4.2	Metadata for features, feature attributes, and feature associations.....	43
8.5	Rules for use of quality schema.....	44
8.5.1	Introduction.....	44
8.5.2	Data quality rules.....	45
8.6	Temporal rules.....	48
8.6.1	Rules for modelling applications with temporal properties.....	48
8.6.2	Use of the temporal conceptual schema.....	49
8.6.3	Temporal attributes.....	50
8.6.4	Temporal associations between features.....	52
8.7	Spatial rules.....	55
8.7.1	Rules for modelling applications with spatial properties.....	55
8.7.2	Use of standard spatial schema.....	56
8.7.3	Spatial attributes.....	57
8.7.4	Use of geometric aggregates and spatial complexes to represent the values of spatial attributes of features.....	58
8.7.5	Spatial associations between features.....	63
8.7.6	Features sharing geometry.....	64
8.7.7	Point features, line features and area features.....	65
8.7.8	Defining interpolation methods.....	65
8.7.9	Independent spatial complexes.....	66
8.8	Rules for use of coverage functions.....	68

8.9	Rules for the use of observations .....	71
8.10	Spatial referencing using geographic identifiers.....	75
8.11	Code-lists, vocabularies, lexicons.....	77
8.12	Linguistic adaptation .....	78
<b>Annex A</b>	<b>(normative) Abstract test suite .....</b>	<b>80</b>
<b>A.1</b>	<b>Introduction.....</b>	<b>80</b>
<b>A.2</b>	<b>Meta-model.....</b>	<b>80</b>
<b>A.2.1</b>	<b>Conceptual schema language.....</b>	<b>80</b>
<b>A.2.2</b>	<b>Schema integration .....</b>	<b>80</b>
<b>A.2.3</b>	<b>Features.....</b>	<b>80</b>
<b>A.2.4</b>	<b>Value assignment .....</b>	<b>81</b>
<b>A.2.5</b>	<b>Feature associations.....</b>	<b>81</b>
<b>A.2.6</b>	<b>Feature inheritance .....</b>	<b>81</b>
<b>A.2.7</b>	<b>Constraints.....</b>	<b>82</b>
<b>A.3</b>	<b>Creating application schemas in UML .....</b>	<b>82</b>
<b>A.3.1</b>	<b>UML profile.....</b>	<b>82</b>
<b>A.3.2</b>	<b>Packaging.....</b>	<b>82</b>
<b>A.3.3</b>	<b>Documentation .....</b>	<b>82</b>
<b>A.3.4</b>	<b>Integrating an application schema .....</b>	<b>83</b>
<b>A.3.5</b>	<b>Modelling structure .....</b>	<b>83</b>
<b>A.3.6</b>	<b>Modelling features.....</b>	<b>83</b>
<b>A.3.7</b>	<b>Property names .....</b>	<b>84</b>
<b>A.3.8</b>	<b>Value assignment .....</b>	<b>84</b>
<b>A.3.9</b>	<b>Attribute of attribute.....</b>	<b>84</b>
<b>A.3.10</b>	<b>Feature operation .....</b>	<b>84</b>
<b>A.3.11</b>	<b>Feature inheritance .....</b>	<b>85</b>
<b>A.3.12</b>	<b>Constraints.....</b>	<b>85</b>
<b>A.4</b>	<b>Profiling standard schemas .....</b>	<b>85</b>
<b>A.4.1</b>	<b>Extension.....</b>	<b>85</b>
<b>A.4.2</b>	<b>Restriction.....</b>	<b>86</b>
<b>A.5</b>	<b>Use of metadata schema .....</b>	<b>86</b>
<b>A.6</b>	<b>Use of quality schema.....</b>	<b>86</b>
<b>A.6.1</b>	<b>Use of the quality schema in an application schema .....</b>	<b>86</b>
<b>A.6.2</b>	<b>Additional quality metadata in an application schema .....</b>	<b>87</b>
<b>A.6.3</b>	<b>Instance quality metadata in an application schema .....</b>	<b>87</b>
<b>A.7</b>	<b>Temporal elements.....</b>	<b>87</b>
<b>A.7.1</b>	<b>Use of the temporal conceptual schema in an application schema .....</b>	<b>87</b>
<b>A.7.2</b>	<b>Temporal attributes .....</b>	<b>87</b>
<b>A.7.3</b>	<b>Temporal associations.....</b>	<b>88</b>
<b>A.7.4</b>	<b>Feature succession .....</b>	<b>88</b>
<b>A.8</b>	<b>Spatial elements .....</b>	<b>88</b>
<b>A.8.1</b>	<b>Use of the spatial conceptual schema in an application schema .....</b>	<b>88</b>
<b>A.8.2</b>	<b>Spatial attributes .....</b>	<b>89</b>
<b>A.8.3</b>	<b>Spatial complexes .....</b>	<b>89</b>
<b>A.8.4</b>	<b>Spatial associations .....</b>	<b>89</b>
<b>A.8.5</b>	<b>Shared geometry .....</b>	<b>89</b>
<b>A.8.6</b>	<b>Single geometry.....</b>	<b>90</b>
<b>A.8.7</b>	<b>Interpolation method.....</b>	<b>90</b>
<b>A.8.8</b>	<b>Independent spatial complexes .....</b>	<b>90</b>
<b>A.9</b>	<b>Coverages .....</b>	<b>91</b>
<b>A.10</b>	<b>Observations.....</b>	<b>91</b>
<b>A.11</b>	<b>Spatial referencing using geographic identifiers.....</b>	<b>91</b>
<b>A.12</b>	<b>Code-lists and vocabularies.....</b>	<b>92</b>
<b>A.12.1</b>	<b>Management of code-lists .....</b>	<b>92</b>
<b>A.12.2</b>	<b>Formalization of code-lists .....</b>	<b>92</b>
<b>A.13</b>	<b>Multi-lingual support .....</b>	<b>92</b>
<b>A.13.1</b>	<b>Application schema.....</b>	<b>92</b>
<b>A.13.2</b>	<b>Feature and property types .....</b>	<b>92</b>
<b>Annex B</b>	<b>(informative) The modelling approach and the General Feature Model .....</b>	<b>94</b>

## ISO/DIS 19109

<b>B.1</b>	<b>The four-layer architecture .....</b>	<b>94</b>
<b>B.1.1</b>	<b>Introduction .....</b>	<b>94</b>
<b>B.1.2</b>	<b>The Meta Meta level and the Meta level .....</b>	<b>95</b>
<b>B.1.3</b>	<b>The Meta level and the Application level .....</b>	<b>95</b>
<b>B.1.4</b>	<b>The Application level and the Data level .....</b>	<b>95</b>
<b>B.1.5</b>	<b>The General Feature Model and the conceptual schema language of the application schema.....</b>	<b>95</b>
<b>B.2</b>	<b>The term “feature” .....</b>	<b>96</b>
<b>Annex C</b>	<b>(informative) Application schema examples.....</b>	<b>97</b>
<b>C.1</b>	<b>Utility network .....</b>	<b>97</b>
<b>C.1.1</b>	<b>Introduction .....</b>	<b>97</b>
<b>C.1.2</b>	<b>The application schema .....</b>	<b>98</b>
<b>C.1.3</b>	<b>An example of the document on Utility Network.....</b>	<b>99</b>
<b>C.2</b>	<b>Administrative units .....</b>	<b>101</b>
<b>C.2.1</b>	<b>Introduction .....</b>	<b>101</b>
<b>C.2.2</b>	<b>The application schema in UML .....</b>	<b>102</b>
<b>Bibliography</b>	<b>.....</b>	<b>103</b>

## Tables

<b>Table 1</b>	<b>— Meta-model conformance class .....</b>	<b>2</b>
<b>Table 2</b>	<b>— UML application schema conformance class.....</b>	<b>2</b>
<b>Table 3</b>	<b>— Profiling standard schema conformance class.....</b>	<b>2</b>
<b>Table 4</b>	<b>— Metadata conformance class.....</b>	<b>2</b>
<b>Table 5</b>	<b>— Quality conformance class.....</b>	<b>2</b>
<b>Table 6</b>	<b>— Temporal conformance class .....</b>	<b>3</b>
<b>Table 7</b>	<b>— Spatial conformance class .....</b>	<b>3</b>
<b>Table 8</b>	<b>— Coverages conformance class.....</b>	<b>3</b>
<b>Table 9</b>	<b>— Observations conformance class .....</b>	<b>3</b>
<b>Table 10</b>	<b>— Spatial referencing by identifiers conformance class .....</b>	<b>4</b>
<b>Table 11</b>	<b>— Code-list conformance class.....</b>	<b>4</b>
<b>Table 12</b>	<b>— Multi-lingual support conformance class .....</b>	<b>4</b>
<b>Table 13</b>	<b>— Conformance class template.....</b>	<b>8</b>
<b>Table 14</b>	<b>— Requirements class template.....</b>	<b>8</b>
<b>Table 15</b>	<b>— Requirements class covering general requirements and the meta-model for application schemas .....</b>	<b>14</b>
<b>Table 16</b>	<b>— Requirements class for modelling an application schema in UML.....</b>	<b>31</b>
<b>Table 17</b>	<b>— Summary of UML Profile used as the CSL for application schemas .....</b>	<b>33</b>
<b>Table 18</b>	<b>— Example of real-world concepts in terms of the General Feature Model .....</b>	<b>40</b>
<b>Table 19</b>	<b>— Requirements class for defining a profile of a standard schema.....</b>	<b>41</b>
<b>Table 20</b>	<b>— Requirements class for an application schema including metadata .....</b>	<b>43</b>
<b>Table 21</b>	<b>— Requirements class for an application schema including metadata .....</b>	<b>45</b>
<b>Table 22</b>	<b>— Concrete subtypes of DQ_Element .....</b>	<b>46</b>
<b>Table 23</b>	<b>— Requirements class for applications with temporal properties.....</b>	<b>49</b>
<b>Table 24</b>	<b>— List of valid temporal objects for temporal attributes in an application schema .....</b>	<b>50</b>
<b>Table 25</b>	<b>— Requirements class for applications with spatial properties .....</b>	<b>56</b>
<b>Table 26</b>	<b>— List of valid spatial objects for spatial attributes in an application schema.....</b>	<b>57</b>
<b>Table 27</b>	<b>— Requirements class for applications with coverage-valued properties .....</b>	<b>69</b>
<b>Table 28</b>	<b>— List of valid coverage types in an application schema .....</b>	<b>69</b>
<b>Table 29</b>	<b>— Requirements class for applications involving observations .....</b>	<b>73</b>
<b>Table 30</b>	<b>— Requirements class for applications using geographic identifiers .....</b>	<b>76</b>
<b>Table 31</b>	<b>— Requirements class for applications schemas using code-lists.....</b>	<b>78</b>
<b>Table 32</b>	<b>— Requirements class for application schemas in multi-lingual environments.....</b>	<b>78</b>
<b>Table B.1</b>	<b>— Feature used in different levels .....</b>	<b>96</b>

## Foreword

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

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

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

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

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

This second edition cancels and replaces the first edition (ISO 19109:2005), which has been technically revised.

## **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

ISO 19103:—<sup>1)</sup>, *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*

---

1) To be published.

ISO 19115-1:—<sup>2)</sup>, *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:—<sup>3)</sup>, *Geographic information — Data Quality*

ISO 19505-2:2012, *Information technology — Object Management Group Unified Modeling Language (OMG UML) — Part 2: Superstructure (Version 2.4.1)*

## 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:—<sup>4)</sup>, 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:—<sup>4)</sup>, 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:—<sup>4)</sup>, 4.1.5]

### 4.5

#### **conceptual schema**

formal description of a *conceptual model* (4.4)

[SOURCE: ISO 19101-1:—<sup>4)</sup>, 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]

---

2) To be published

3) To be published.

4) To be published.

**ISO/DIS 19109****4.7****dataset**

identifiable collection of data

[SOURCE: ISO 19115-1:—<sup>4</sup>) 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.

[SOURCE: ISO/TS 19103:2005, 4.1.6]

**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:—<sup>4</sup>), 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:—<sup>4</sup>), 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:—<sup>4</sup>), 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:—<sup>4</sup>), 4.1.38]

#### 4.20

##### **value**

element of a type and *domain* (4.8)

[SOURCE: ISO/IEC 19505-2:2012, N.M]

## 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 on other conformance classes<sup>5)</sup>. 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 sub-clause, 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	/req/{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. 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:

/req/{classM}/{reqN}	[Normative statement]
----------------------	-----------------------

where /req/{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 a complete URI which allows the requirements class, requirement or recommendation to be identified in an external context. For example, following the URI scheme for ISO standards [IETF RFC 5141] a URI denoting this standard is

5) See ISO/TC 211 Resolution 570.

<http://standards.iso.org/iso/19109/2>

The URI for each requirements class has the form

[http://standards.iso.org/iso/19109/2/req/\[classM\]](http://standards.iso.org/iso/19109/2/req/[classM])

and the URI for each requirement or recommendation has the form

[http://standards.iso.org/iso/19109/2/req/\[classM\]/\[reqN\]](http://standards.iso.org/iso/19109/2/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 19103:201X. 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 bi-alpha codes as follow:

GM	ISO 19107:2003 Spatial Schema
TM	ISO 19108:2002 Temporal Schema, Temporal Objects
TP	ISO 19108:2002 Temporal Schema, Temporal Topology
MD	ISO 19115 Metadata
CV	ISO 19123:2005 Schema for Coverage Geometry and Functions
OM	ISO 19156:2011 Observations and Measurements
DQ	ISO 19157:201X Data Quality

## 6 Context

### 6.1 Purpose of an application schema

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

- the content and structure of data, and
- operations for manipulating and processing 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, and
- determine the necessary transformations between the two datasets.

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

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

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

The application schema defines the structure and content of data. It is expressed in a conceptual schema language (CSL). Clause 7 also includes a model expressed in a UML profile that defines the concepts required to describe types of features. Feature type definitions may be documented in feature catalogues (ISO 19110:2005). 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 19118:2005, ISO 19136:2007 and ISO/TS 19139:2007 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:

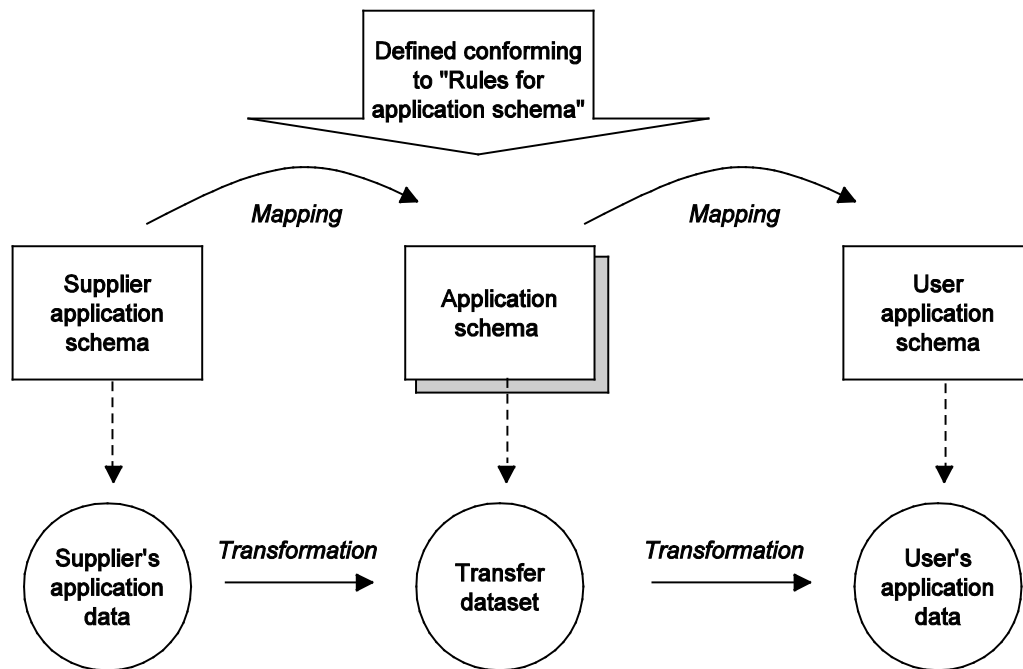
- In the traditional data transfer model, the data supplier creates a dataset that is transferred to the user. The structure and the content of data are described in the application schema for the dataset. The dataset is sent in a transfer format;
- In the interoperability model, the user application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

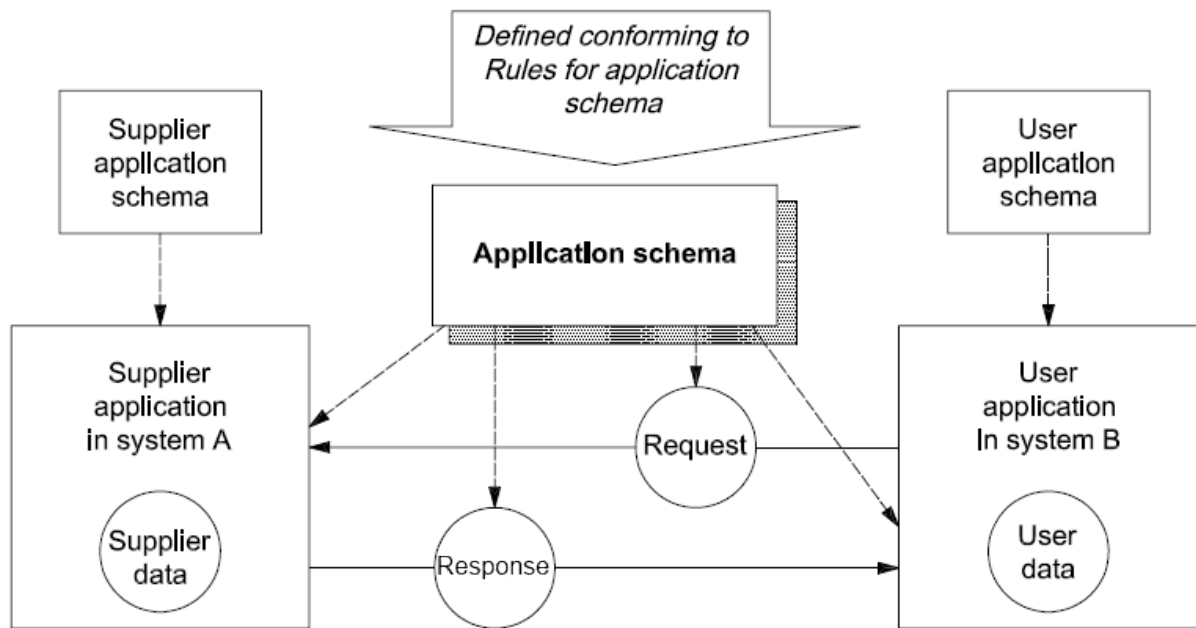
First, the user and the supplier must agree on creating an application schema for the data being exchanged in accordance with this International Standard. In order to facilitate the transfer of data, the application schema is developed using the application schemas from the user and the supplier. One 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 19119:2005. 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
Requirement	/req/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	/req/general/constraint

## 7.2 Features, Coverages and Properties

### 7.2.1 Features

A fundamental unit of geographic information is called a feature. ISO 19110:2005 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.

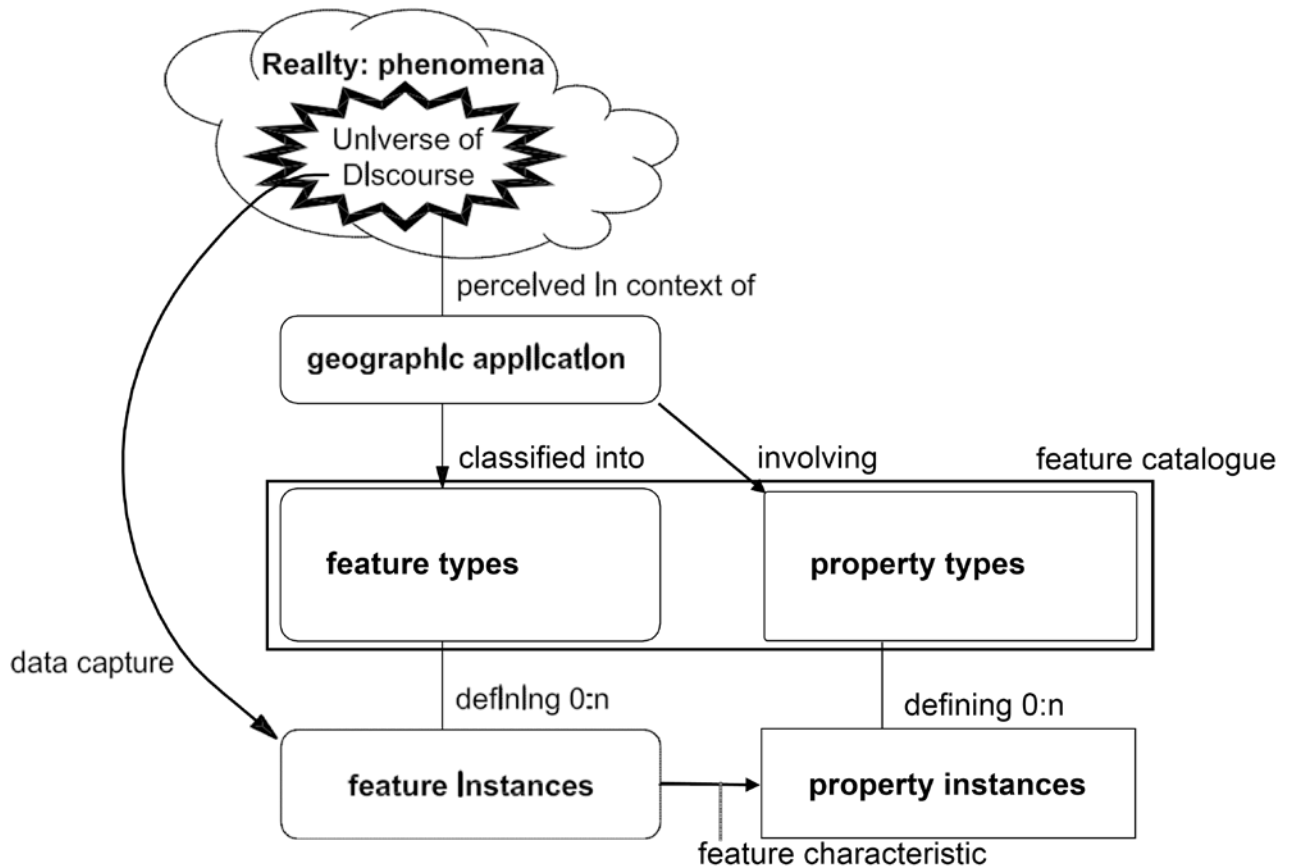


Figure 3 — The process from universe of discourse to data

### 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

## ISO/DIS 19109

- 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 19123:2005, 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 19156:2011, '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.

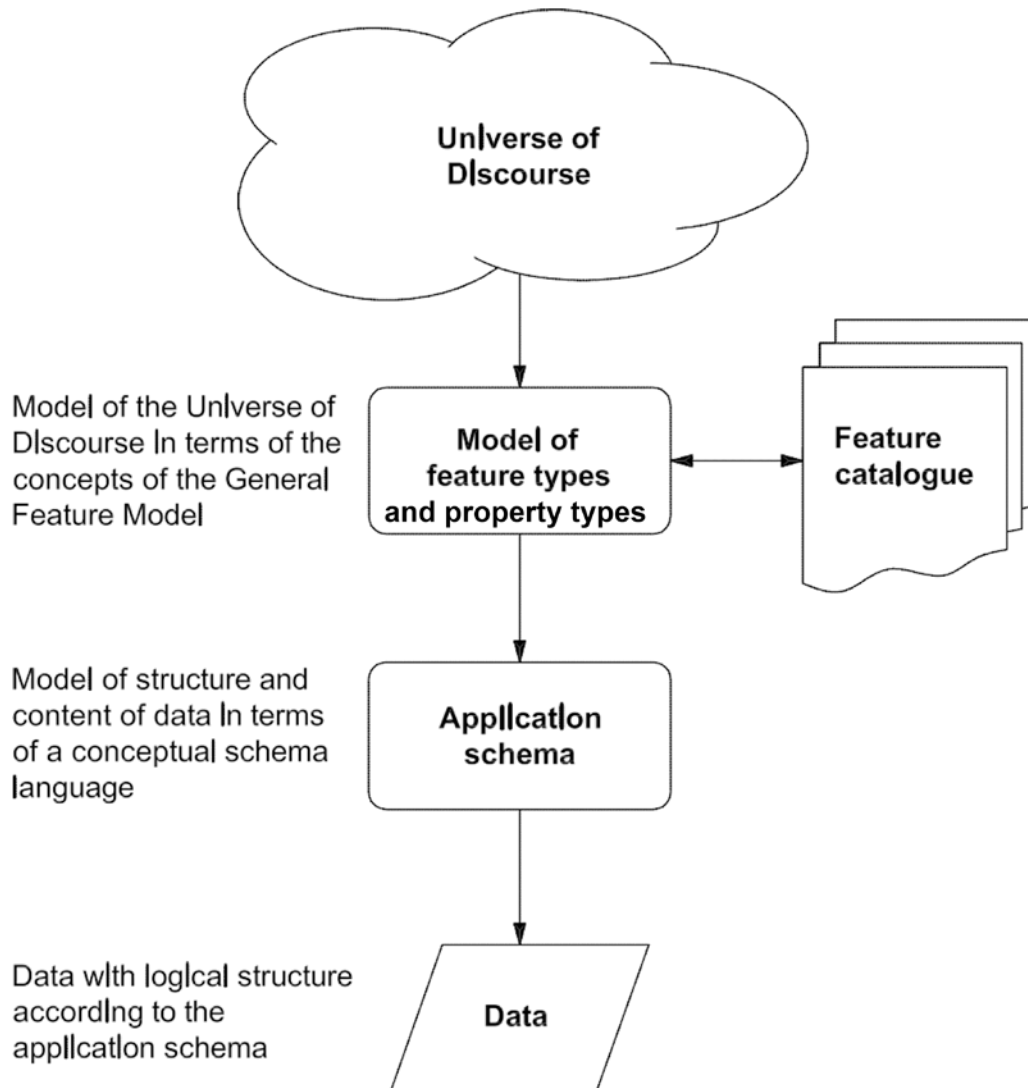
Figure 4 shows the process of structuring data from the universe of discourse to the geographic dataset. The definitions of the feature types and their properties, as perceived in context of an application field, will be derived from the universe of discourse. A feature catalogue documents the feature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

## ISO/DIS 19109

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 19103:201X.

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 19110:2005). 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 19103:201X 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 19103:201X 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.

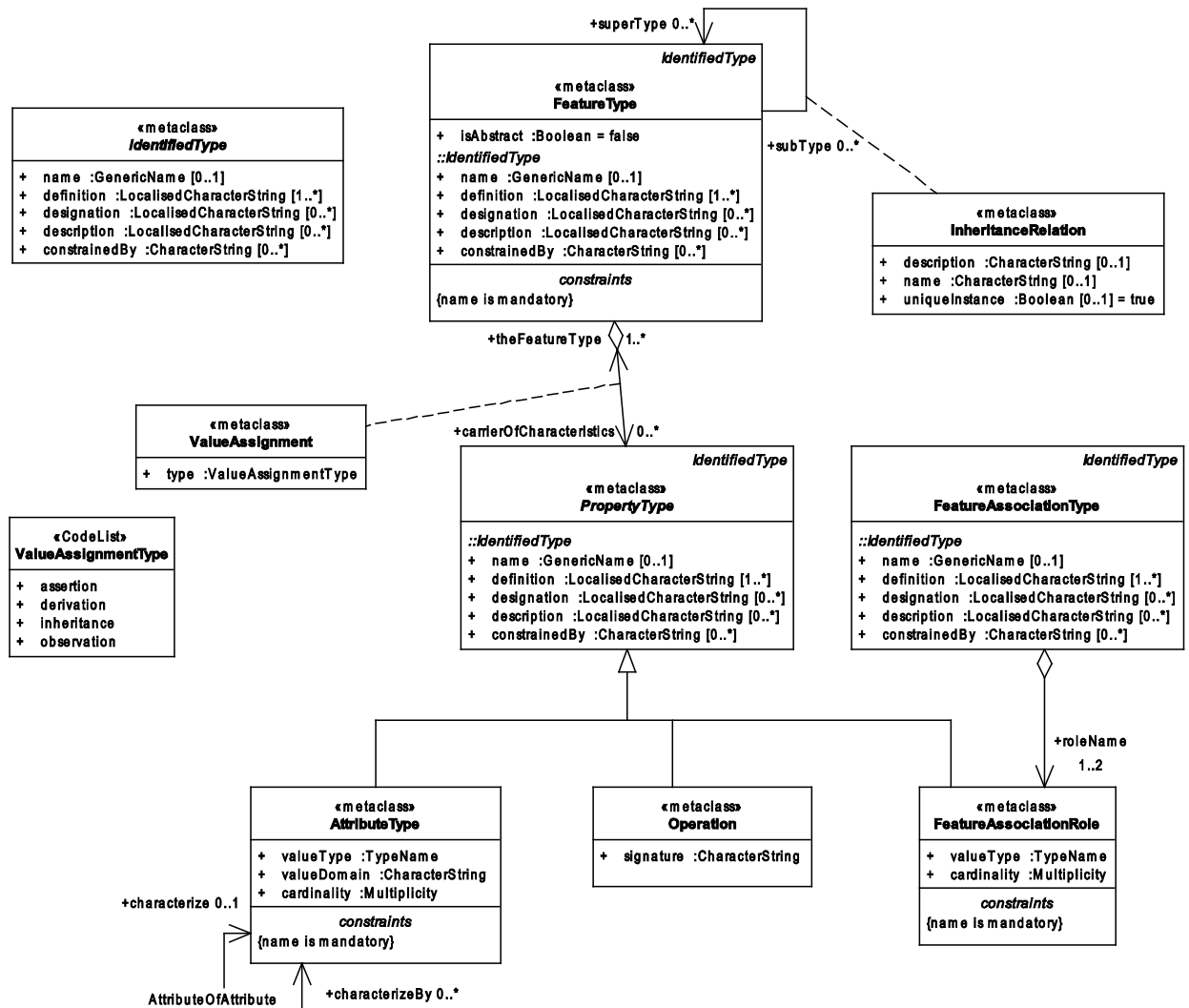


Figure 5 — The General Feature Model

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

NOTE 2 The types `CharacterString`, `LocalisedCharacterString`, `LocalName`, `TypeName`, `Multiplicity` are from ISO 19103:201X.

#### 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 19103:201X] 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

## ISO/DIS 19109

[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 19150-2:201X] 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.

## 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.

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

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, but a property instance and value occurs in the context of a feature. Nevertheless, the feature type or feature identity for an observed property may not be available, for example because the necessary data interpretation has not yet been done. Therefore, properties may be described independent of features. Property type names therefore need to also be unique within an application.

<b>/req/general/property-name</b>	The name of each property type shall be unique within an application schema.
-----------------------------------	--

#### 7.4.7 AttributeType

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

<b>/req/general/attribute</b>	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 19103:201X 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, character String 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 19107:2003; enumerated set.

NOTE 2 ISO 19103:201X 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**

## ISO/DIS 19109

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 19110:2005 provides a broader discussion on behaviour of feature types. Operation is the metaclass for describing behaviour of feature types in terms of operations.

<b>/req/general/operation</b>	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.

<b>/req/general/association-role</b>	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).

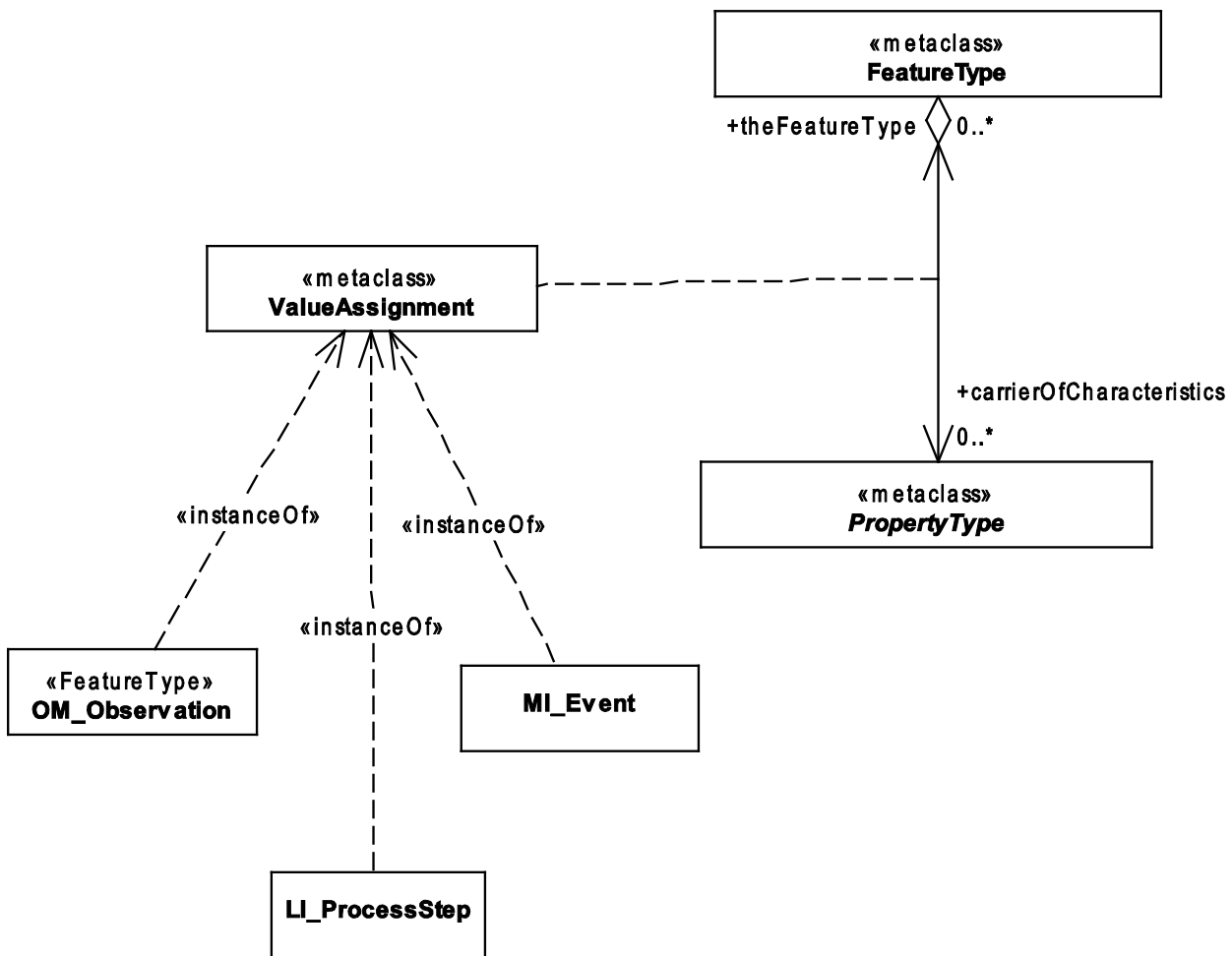
#### 7.4.10 ValueAssignment

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.

<b>/req/general/value-assignment</b>	<p>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.</p> <p>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.</p>
--------------------------------------	--

The feature type OM\_Observation (ISO 19156:2011), and the classes LI\_ProcessStep (ISO 19115-1:201X) and MI\_Event (ISO 19115-2:2009) 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 19156:2011 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).

## ISO/DIS 19109

### — **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**

This sub-clause describes in more detail the role of attributes of features. 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 19107:2003.

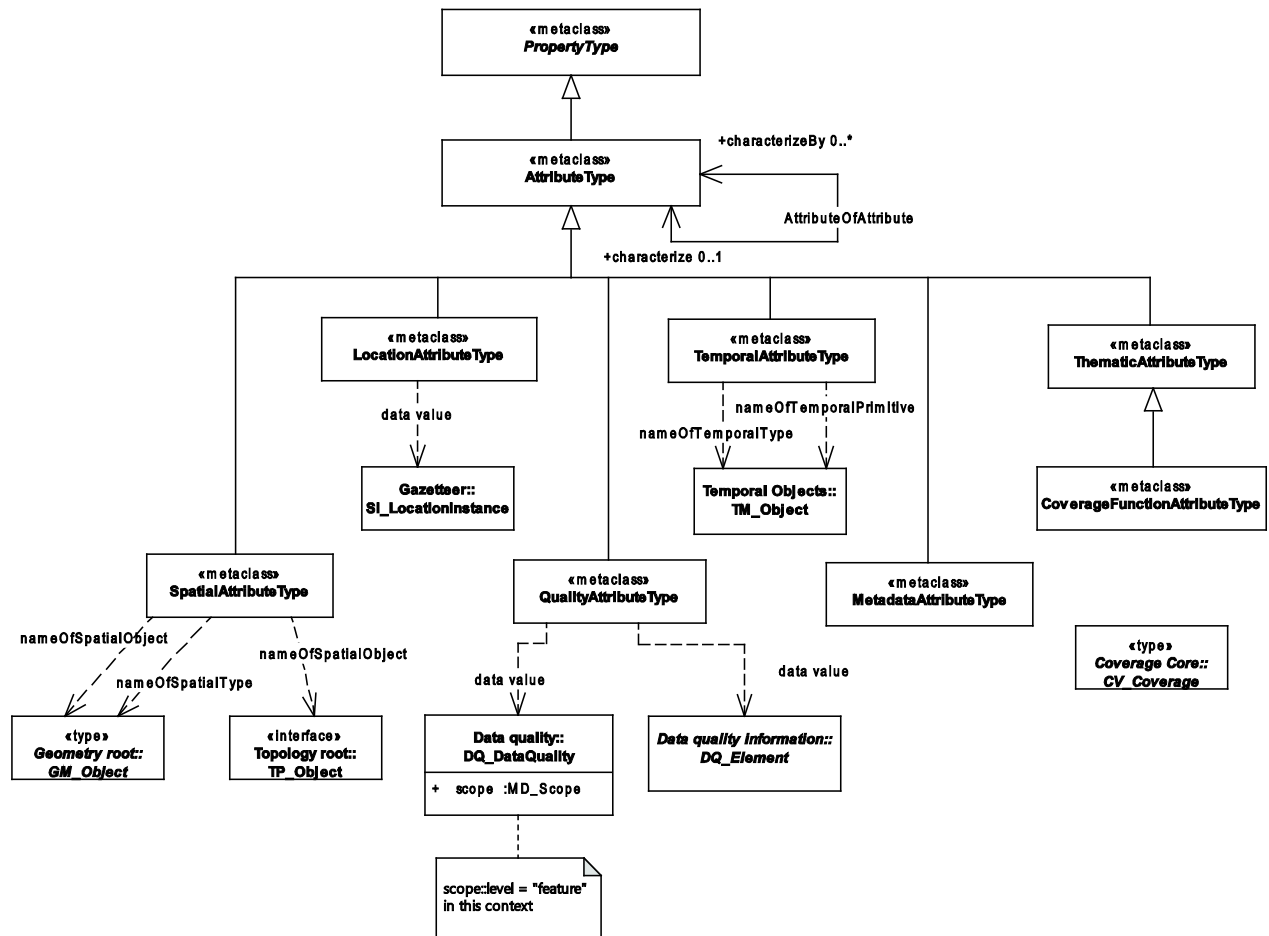


Figure 7 — Attributes of feature types

### 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 19107:2003.

### 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 19108:2002.

### 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 19157:201X.

### 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.

## ISO/DIS 19109

**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 19115-1:201X) 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 19103:201X) 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 19123:2005) that provides the data values for the attribute.

## 7.6 Relationships between feature types

### 7.6.1 Introduction

This sub-clause describes relationships between feature types 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.

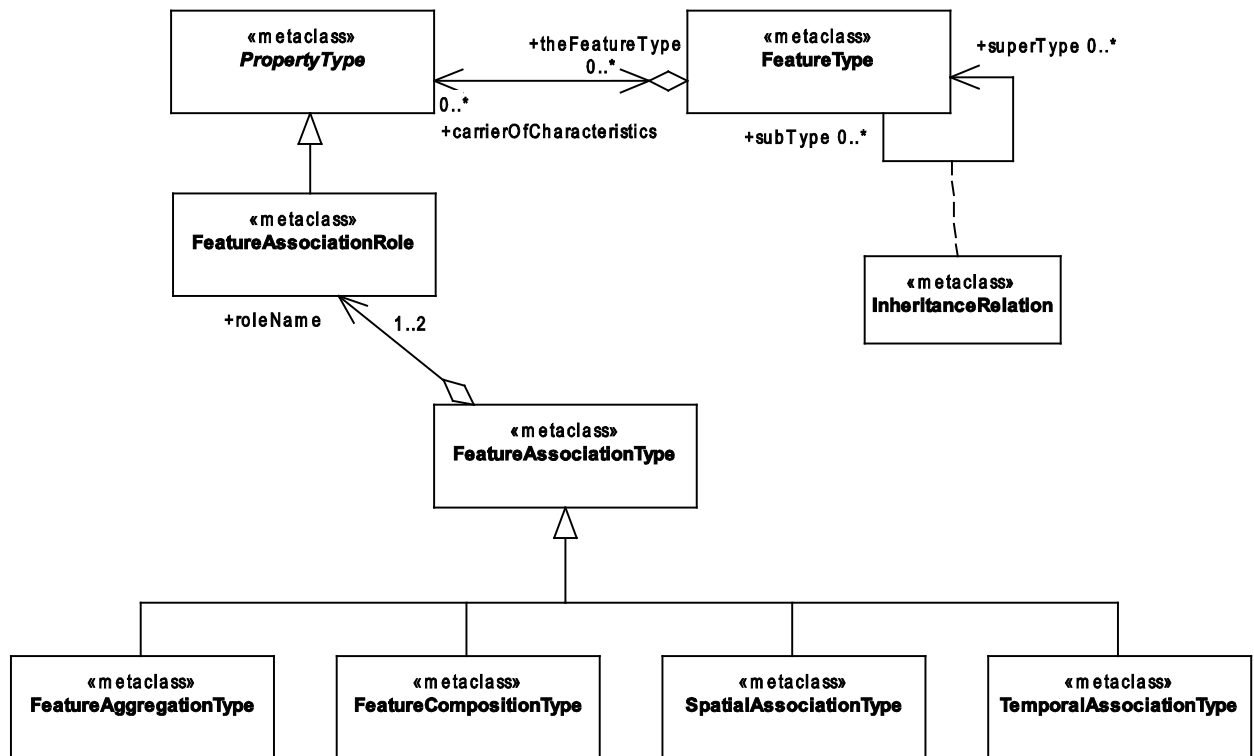


Figure 8 — Relationships between feature types

### 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

## ISO/DIS 19109

`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.

<b>/req/general/constraint</b>	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.

## 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

This sub-clause describes the requirements for defining an application schema using UML as the CSL. 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 19103:201X (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
Requirement	/req/uml/propertyName
Requirement	/req/uml/attributeOfAttribute
Requirement	/req/uml/operation
Requirement	/req/uml/inheritance
Requirement	/req/uml/constraint
Requirement	/req/uml/valueAssignment

## 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 19505-2:2012 explains how to use UML. ISO 19103:201X 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 19150-2:201X 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 19103:201X Clause 7 and Annex C),
- some requirements on the use of UML associations (ISO 19103:201X, 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, summarized in Table 17.

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

Stereotype	Scope	Constraints	Tagged values	Source
ApplicationSchema	package	Not nested in another applicationSchema package		This International Standard, 8.2.3
			language designation	This International Standard, 8.12
Leaf	package			19103:201X , 6.10
Datatype	classifier	Use in attributes or strong aggregation (composition)		UML2 19103:201X, 6.10
Enumeration	classifier			19103:201X, 6.5.2, 6.10
CodeList	classifier			19103:201X, 6.5.3, 6.10
Union	classifier			19103:201X, 6.10
FeatureType	classifier		description	This International Standard, 8.2.6
			designation definition	This International Standard, 8.12
estimated	attribute, role			This International Standard, 8.2.6

**/req/uml/profile**

An application schema shall be modelled using the UML profile defined in ISO 19103:201X 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**

- 1) An application schema shall be documented.
- 2) 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.
- 3) 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”.
- 4) 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”.

## ISO/DIS 19109

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 19110:2005, 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 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 dependencies in Figure 9 mean that the application schema package “imports” structures and definitions from both standard schemas and other application schemas.

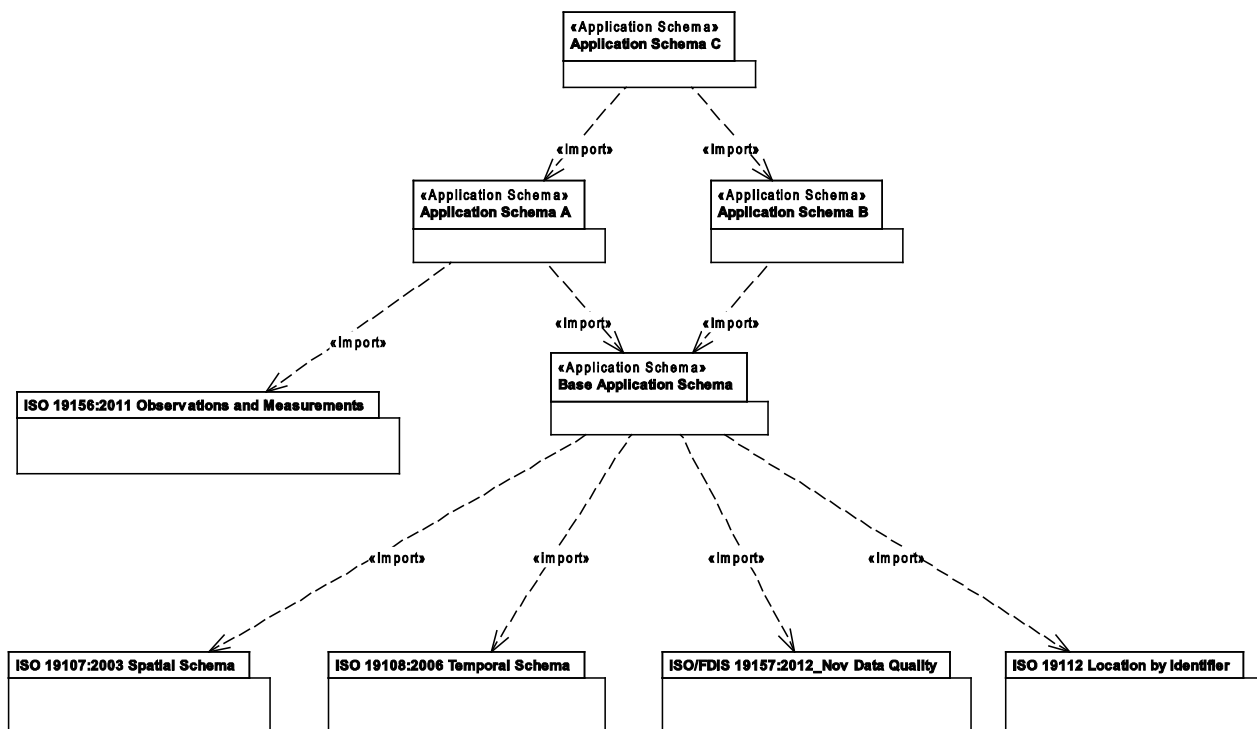


Figure 9 — Example of application schema integration

**/req/uml/integration**

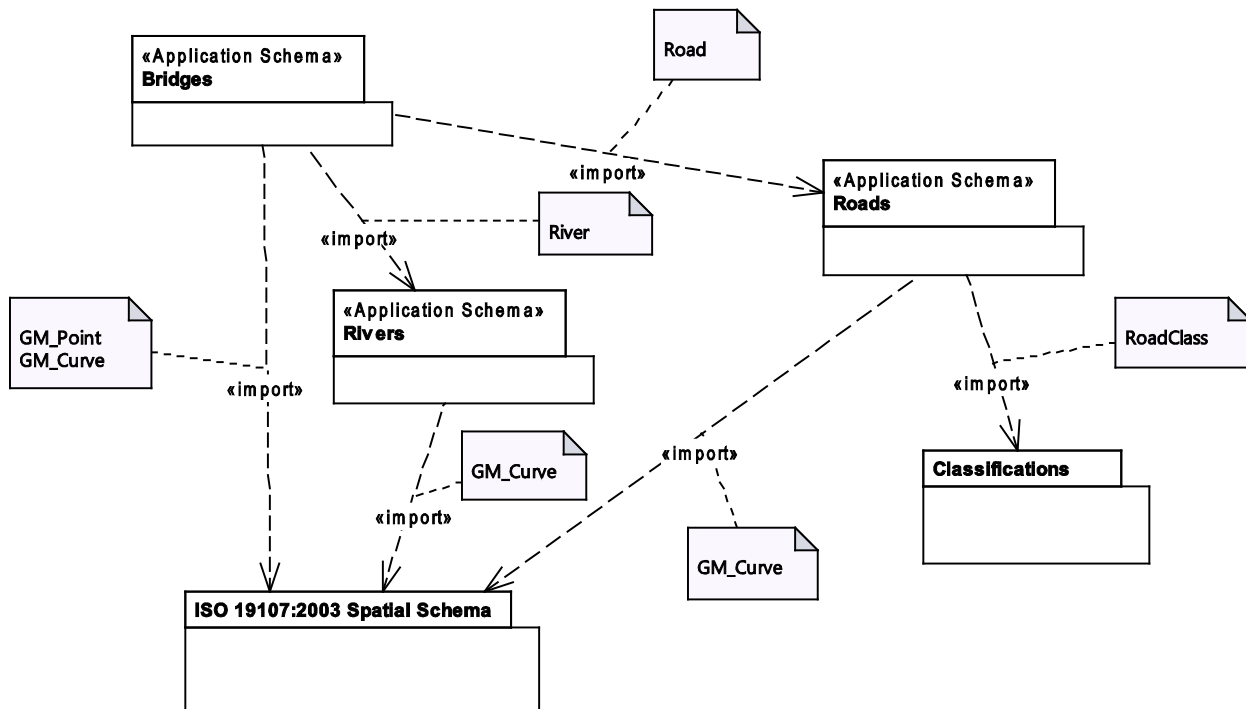
- 1) UML PACKAGE dependency shall be used to describe the integration of the application schema with the other schemas, including standard schemas, as required to form the complete definition of the data structure.
- 2) A PACKAGE with the STEREOType «ApplicationSchema» shall not contain another PACKAGE with the STEREOType «ApplicationSchema».
- 3) A PACKAGE containing, or contained by, a PACKAGE with the STEREOType «ApplicationSchema» shall not have dependencies or dependants.
- 4) PACKAGEs shall not have mutual dependencies, and dependency graphs shall not include cycles.
- 5) Dependencies on standard schemas from the ISO geographic information standards or other application schemas may carry the STEREOType «import».

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

**EXAMPLE** 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 19107:2003.



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

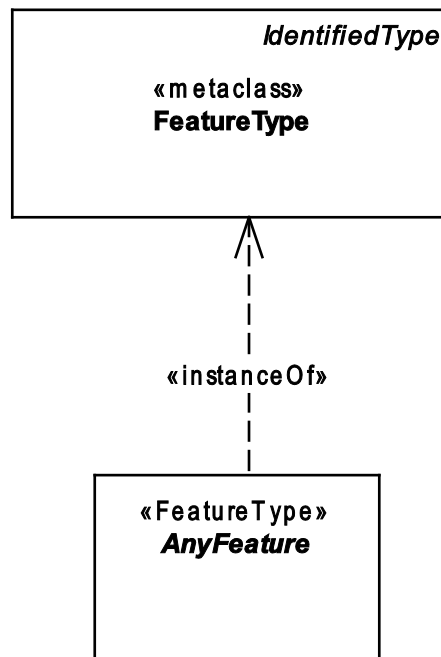
A basic rule concerns data structures and instantiability.

#### **/req/uml/structure**

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

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 1 AnyFeature corresponds to GFI\_Feature from ISO 19156:2011.

<b>/req/uml/feature</b>	<ol style="list-style-type: none"> <li>1) An instance of FeatureType shall be implemented as a CLASS. The CLASS shall carry the STEREOType «FeatureType». The CLASS shall have a GENERALIZATION ASSOCIATION with AnyFeature.</li> <li>2) The CLASS name shall be unique within an Application Schema.</li> <li>3) The textual definition of the CLASS shall be recorded utilizing the documentation facilities in the software tool, if this information can be exported.</li> </ol>
-------------------------	--

NOTE 2 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.

<b>/req/uml/association</b>	An instance of FeatureAssociationType shall be implemented as an instance of FeatureAssociationType that is 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.

<b>/req/uml/aggregation</b>	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.
-----------------------------	--

<b>/req/uml/attribute</b>	<p>An instance of AttributeType shall be represented in an application schema in either of two ways:</p> <ul style="list-style-type: none"> <li>— case 1: as an ATTRIBUTE, unless it is an attribute of an attribute [see /req/uml/attributeOfAttribute]; or</li> <li>— case 2: as an ASSOCIATION between the CLASS that represents a feature and a class representing the value domain of the AttributeType.</li> </ul> <p>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]</p>
---------------------------	--

<b>/req/uml/role</b>	<p>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].</p>
----------------------	--

It is recommended that the names of attributes, operations and association role names are unique within an Application Schema, or that attributes, operations and association roles with the same name have the same meaning within an Application Schema. 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. In this usage each attribute, operation or association role is implicitly scoped to a UML package. In generic UML, attributes and association roles are scoped to a class. As foreseen above (8.2.1) this usage is an aspect of the UML profile used as a conceptual schema language for Application Schemas.

<b>/req/uml/propertyName</b>	<p>The name of an ATTRIBUTE, OPERATION or ASSOCIATION end shall be unique within an Application Schema, or shall have the same intention with respect to the CLASS owning the property as any other property with the same name in the Application Schema.</p>
------------------------------	--

NOTE 3 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. Class names must also be unique within a package, but this requirement is enforced by basic UML conformance.

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

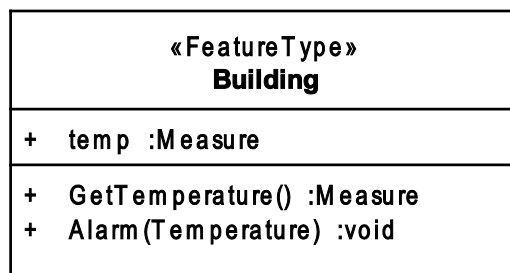
NOTE 4 OM\_Observation and its specializations defined in ISO 19156:2011 provide a suitable model for value assignment using an observation procedure. A requirements class for use of ISO 19156:2011 is provided in 8.9.

<b>/req/uml/attributeOfAttribute</b>	<p>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:</p> <ul style="list-style-type: none"> <li>— step 1: Introduce a new CLASS to represent the ATTRIBUTE that is characterized by other ATTRIBUTES. Use the ATTRIBUTE name as the CLASS name;</li> <li>— 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;</li> <li>— step 3: Insert additional ATTRIBUTE(s) into this CLASS to represent the attributes that characterize the original ATTRIBUTE;</li> <li>— 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.</li> </ul>
--------------------------------------	--

<b>/req/uml/operation</b>	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 1 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**

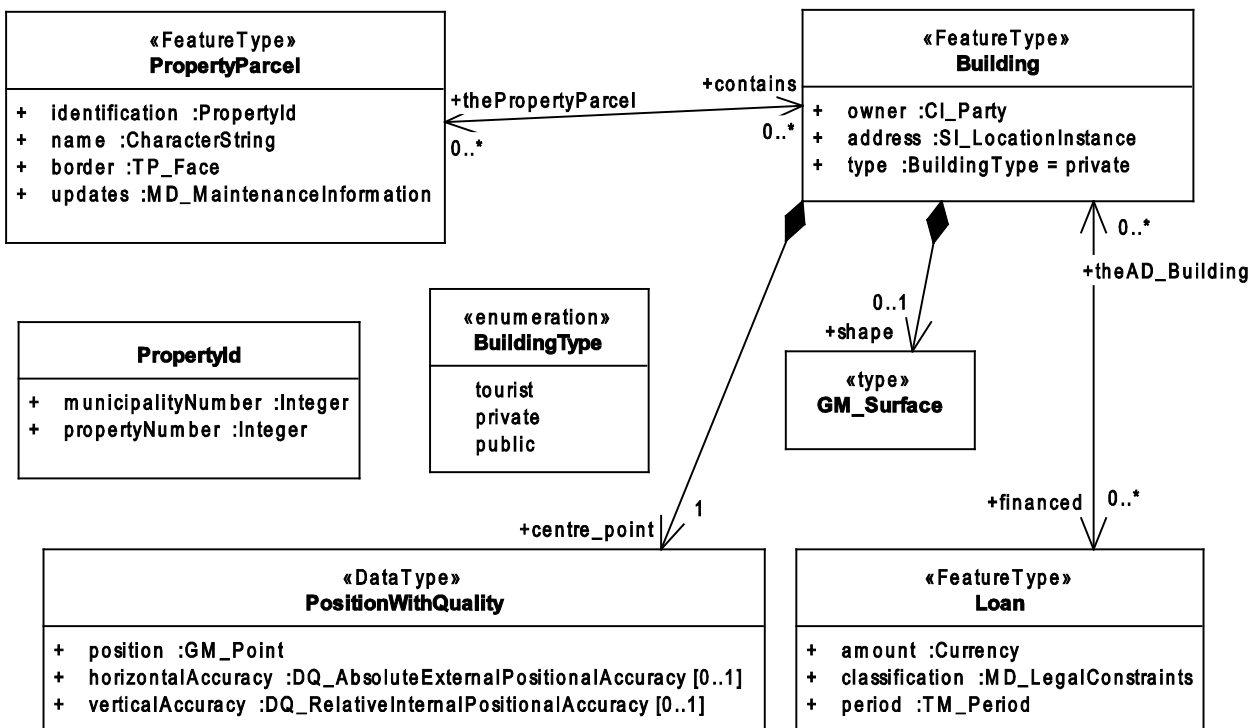
<b>/req/uml/inheritance</b>	<p>An instance of InheritanceRelation shall be represented by a GENERALIZATION relationship, with additional characteristics depending on the following condition:</p> <ul style="list-style-type: none"> <li>— case 1: If <i>uniqueInstance</i> is <i>true</i> (default value), the {disjoint} constraint may be attached to the generalization relationship;</li> <li>— case 2: If <i>uniqueInstance</i> is <i>false</i>, the {overlapping} constraint shall be attached to the generalization relationship.</li> </ul>
-----------------------------	---

**/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 2 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.

This sub-clause describes the requirements for defining a profile of a standard schema. 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	/req/profile/names

### 8.3.2 Adding information to a standard schema

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

<b>/req/profile/extend</b>	<ol style="list-style-type: none"> <li>1) 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.</li> <li>2) The new CLASSES shall be collected in a separate PACKAGE.</li> </ol>
----------------------------	--

**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.

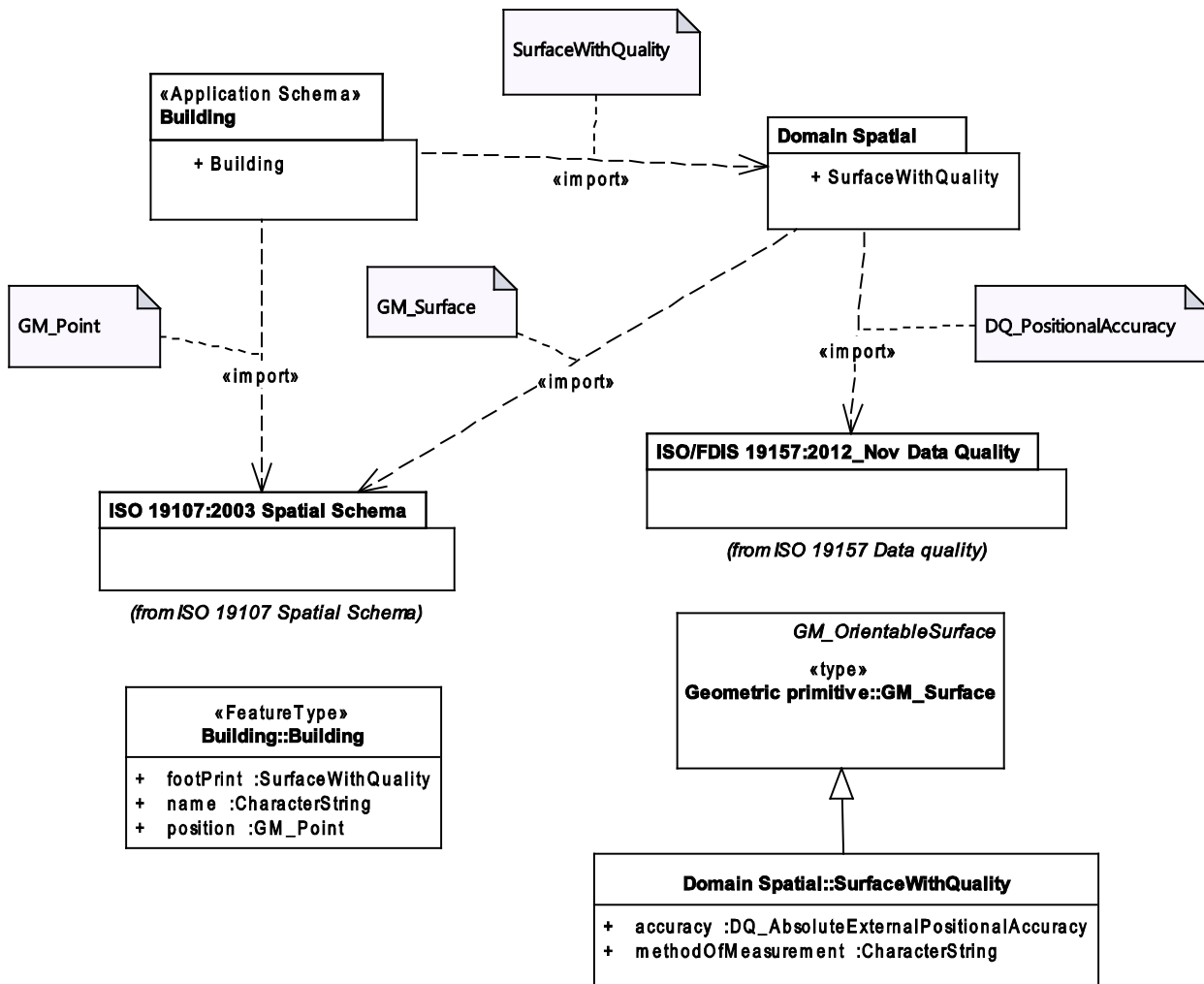


Figure 14 — Example of adding information to a standard schema

### 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<sup>6)</sup> 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 19103:201X, 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.

<b>/req/profile/names</b>	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 19107:2003) 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 19115-1:201X) is an application schema for metadata data sets. A schema for data quality is defined in ISO 19157:201X.

In application schemas, definitions from ISO 19115-1:201X and ISO 19157:201X may be used. This International Standard does not place restrictions on such use. This sub-clause describes the requirements for an application schema that includes metadata elements from ISO 19115-1:201X, and 8.5 describes the requirements for an application schema that includes quality elements from ISO 19157:201X.

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
<b>Target type</b>	UML application schema
<b>Dependency</b>	ISO 19115-1:201X (Metadata)
<b>Dependency</b>	/req/uml (Application schema using UML)
<b>Requirement</b>	/req/metadata/feature

### 8.4.2 Metadata for features, feature attributes, and feature associations

The elements of the ISO 19115-1:201X metadata schema can be used as feature properties.

6) ISO 19103 uses the terms "abstract classes" or "types", too.

<b>/req/metadata/feature</b>	<ol style="list-style-type: none"> <li>1) 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].</li> <li>2) 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 19115-1:201X.</li> <li>3) If the types are suitable a feature attribute may use definitions from ISO 19115-1:201X for purposes other than metadata.</li> <li>4) A metadata attribute shall be represented in an application schema in either of two ways: <ul style="list-style-type: none"> <li>— case 1: as an ATTRIBUTE of a CLASS that represents a feature; or</li> <li>— case 2: as an ASSOCIATION between the CLASS that represents a feature and a CLASS from ISO 19115-1:201X..</li> </ul> </li> <li>5) A metadata attribute may be used as an attribute of an attribute, in which case the main rule for <i>attributeOfAttribute</i> shall be applied [see 8.2.6, /req/uml/attributeOfAttribute].</li> </ol>
------------------------------	---

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)].

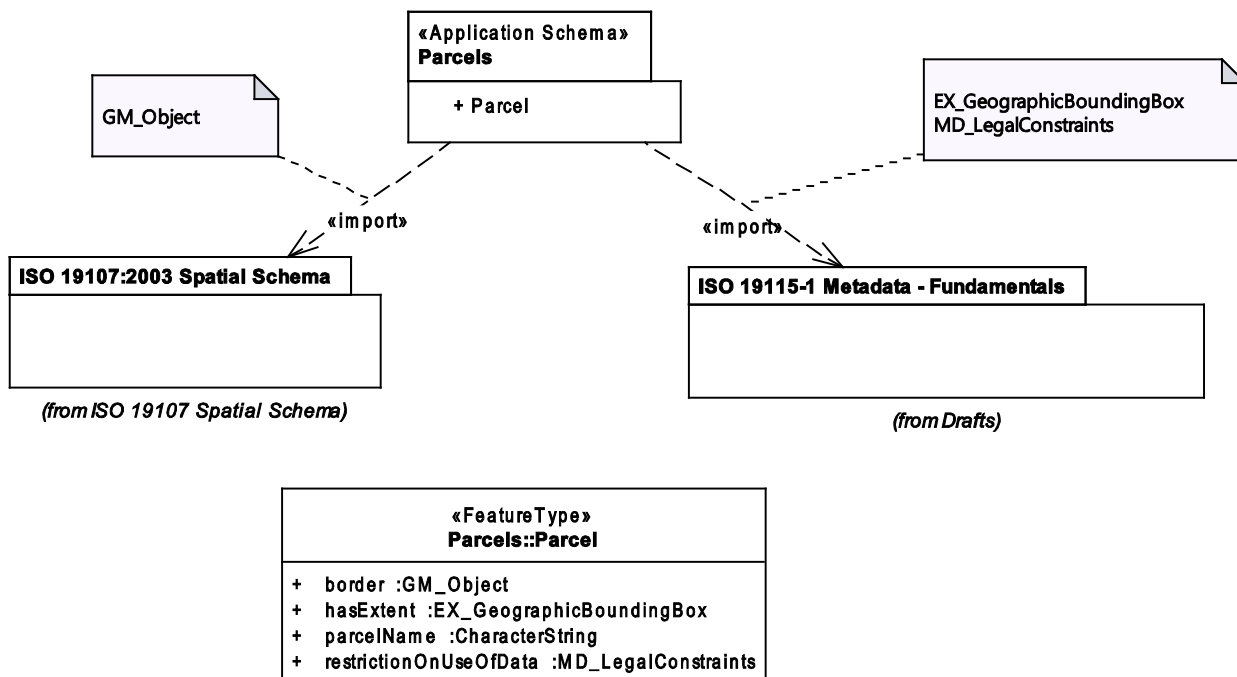


Figure 15 — Example of metadata included as data in an application

## 8.5 Rules for use of quality schema

### 8.5.1 Introduction

This sub-clause describes the requirements for an application schema that uses quality elements from ISO 19157:201X. 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 metadata

Requirements class	/req/quality
Target type	UML application schema
Dependency	ISO 19157:201X (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

This subclause focuses on the use of elements from the standard quality schema within an application schema. Some data quality elements may apply to individual 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 19157:201X.

Elements of the ISO 19157:201X schema provide a means for reporting quality information.

<b>/req/quality/attribute</b>	<ol style="list-style-type: none"> <li>1) 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.</li> <li>2) A quality attribute (an instance of QualityAttributeType) shall be defined in the application schema and shall be used to carry data quality information.</li> <li>3) A quality attribute shall be represented in an application schema in either of three ways: <ul style="list-style-type: none"> <li>— 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 19157:201X as the data type for its value;</li> <li>— 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 19157:201X as the data type for its value; or</li> <li>— case 3: as an attribute of an attribute, in which case the main rule for <i>attributeOfAttribute</i> shall be applied [see /req/uml/attributeOfAttribute].</li> </ul> </li> </ol>
-------------------------------	---

Table 22 shows the concrete sub-types of DQ\_Element defined in ISO 19157:201X.

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_GriddedDataPositionalAccuracy
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_NonQuantitativeAttributeCorrectness 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_CompletenessCommission 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.

### Application Schema

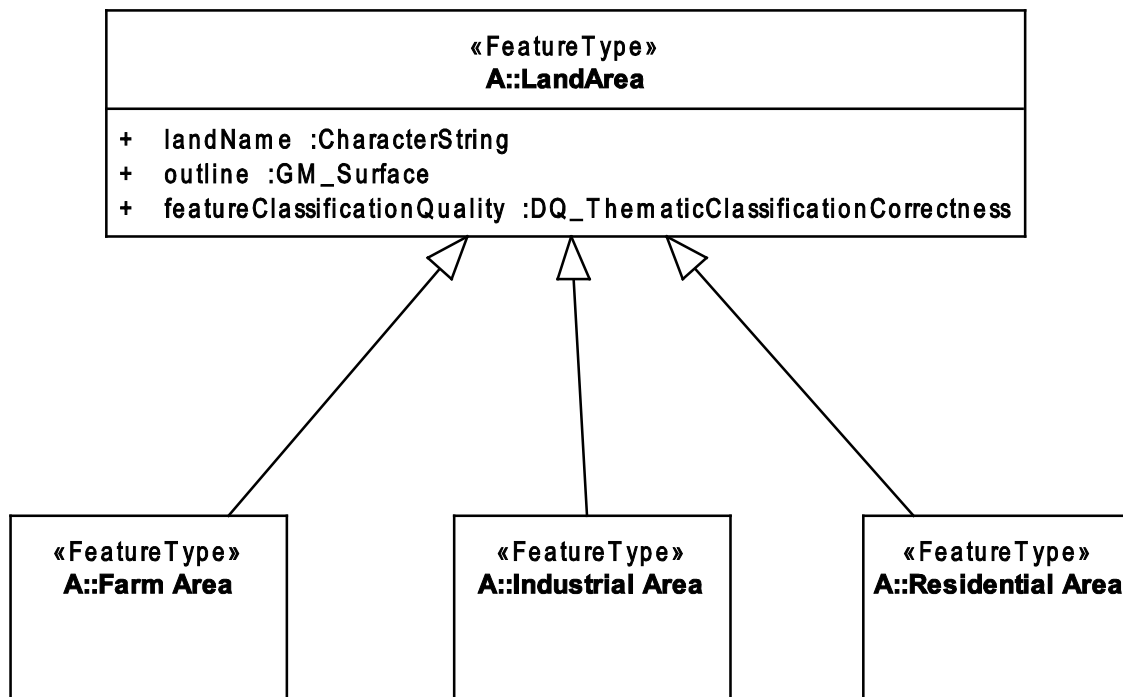


Figure 16 — Example of quality of features instances

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 19157:201X with an additional description called an additional quality sub element.

Elements of the ISO 19157:201X schema provide a means to record additional quality information.

<b>/req/quality/additional-quality</b>	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.

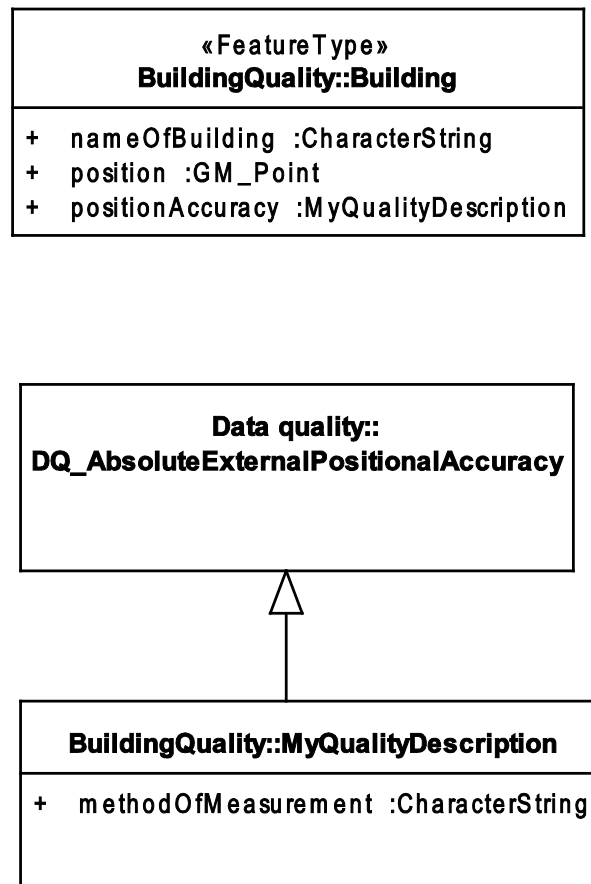
**Application Schema**

Figure 18 — Example of user-defined quality

**8.5.2.3 Reporting quality information for attributes of feature instances**

Elements of the ISO 19157:201X schema provide a means to indicate quality of attributes of feature instances.

<b>/req/quality/attribute-quality</b>	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**

This sub-clause describes the requirements for an application schema with feature types that have temporal properties. 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 19108:2002 provides a schema for temporal objects.

<b>/req/temporal/schema</b>	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

<b>/req/temporal/attribute</b>	<ol style="list-style-type: none"> <li>1) A temporal characteristic of a feature type shall be defined as a temporal attribute, which is a subtype of feature attribute (7.5.3).</li> <li>2) The implementation of temporal attributes in UML shall follow the rules for referencing standardized schemas [see /req/uml/integration].</li> <li>3) A temporal attribute may be represented in an application schema in either of two ways: <ul style="list-style-type: none"> <li>— 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</li> <li>— 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.</li> </ul> </li> <li>4) as an attribute of a CLASS [see /req/uml/feature], in which the attribute shall take one of the temporal types (Date, DateTime, TM_Position) or one of the temporal objects defined in the Temporal Schema (ISO 19108:2002) as the data type for its value.</li> <li>5) 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]: <ul style="list-style-type: none"> <li>— case 1: attributes have multiple components;</li> <li>— case 2: data types for attributes of temporal objects need to be restricted;</li> <li>— case 3: operations from the interfaces defined in ISO 19108:2002 need to be referenced explicitly.</li> </ul> </li> <li>6) 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.</li> <li>7) Valid temporal objects, which shall be applied, are given in Table 24.</li> </ol>
--------------------------------	--

**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*.

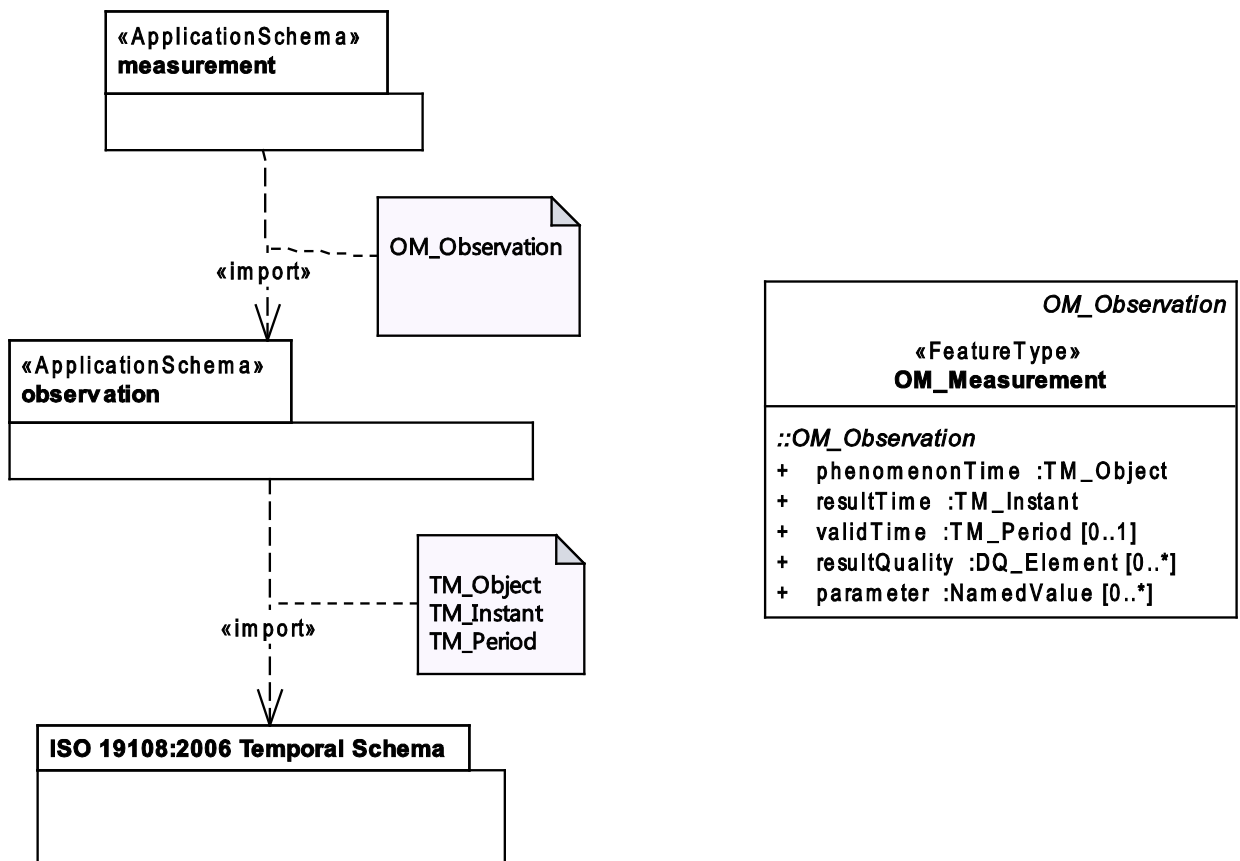


Figure 19 — Example of temporal attribute

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.

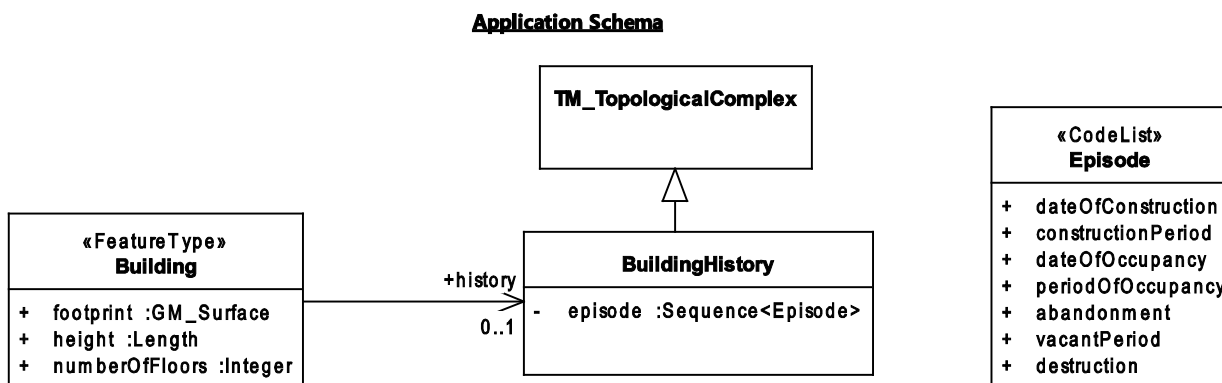
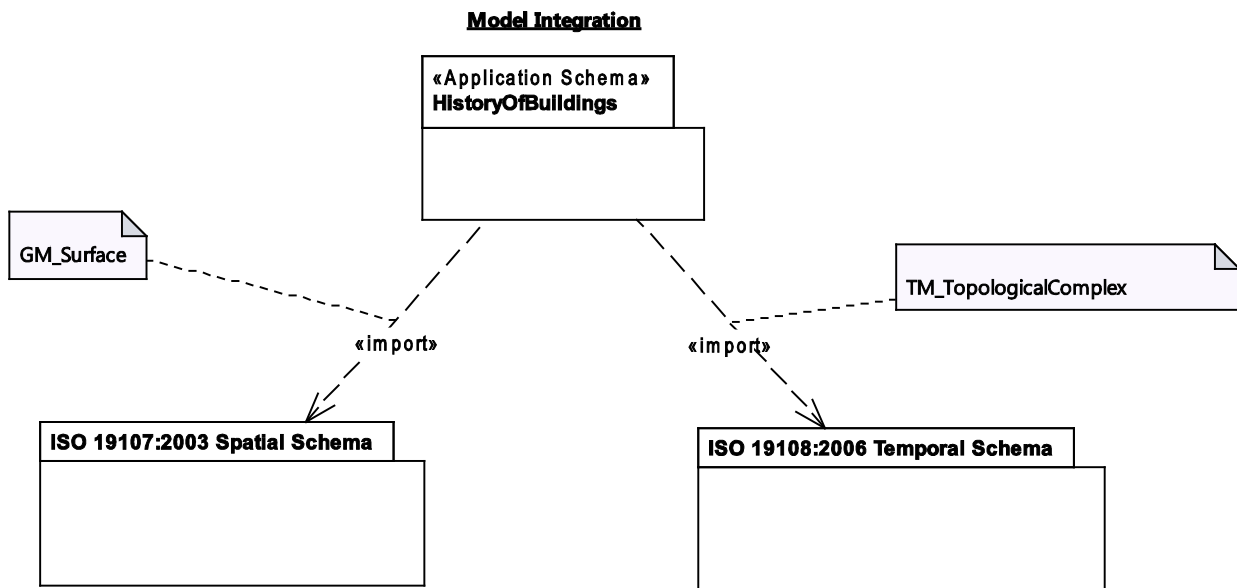


Figure 20 — Example of TM\_TopologicalComplex used as a temporal feature attribute

## 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 19108:2002 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”.

<b>/req/temporal/association</b>	<p>Simple temporal associations shall be implemented in an application schema in one of two ways:</p> <ul style="list-style-type: none"> <li>— case 1: Temporal primitives used as data types of attributes shall implement the operation <code>TM_RelativePosition</code> from the interface <code>TM_Order</code> defined in ISO 19108:2002, so that simple temporal associations can be derived from the data;</li> <li>— case 2: A simple temporal association between features shall be implemented in an application schema using an ASSOCIATION [see <code>/req/uml/association</code>].</li> </ul>
----------------------------------	--

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

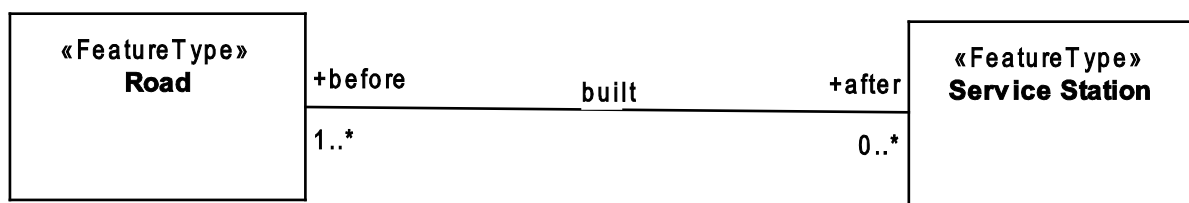


Figure 21 — Example of explicit representation of a simple temporal association

#### 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**
- 1) Feature associations of the feature succession type may be instantiated in an application schema as UML associations between feature type classes.
  - 2) Feature associations of the feature succession type may be instantiated in an application schema as self-referent associations of a generic feature class.
  - 3) The names, roles, and multiplicities shall be appropriate for the type of feature succession.

EXAMPLE 1 Figure 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.

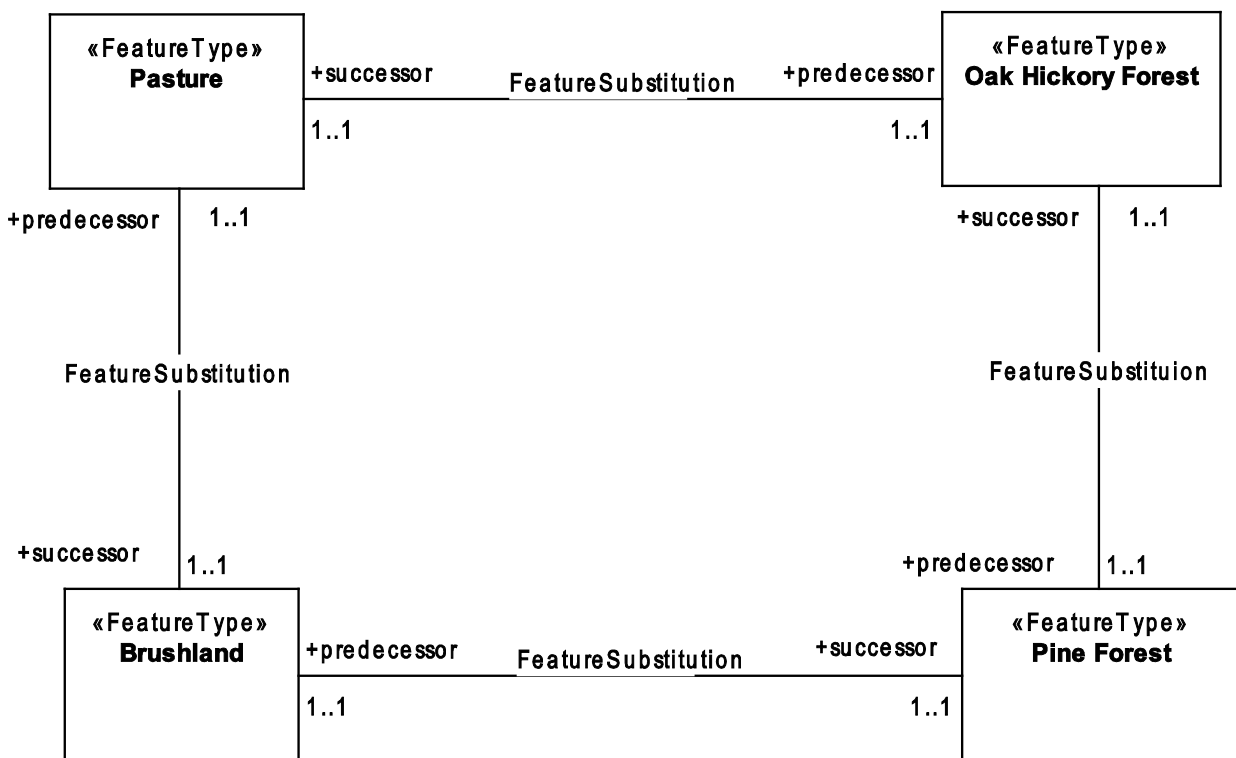


Figure 22 — Example of feature succession between feature types

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.

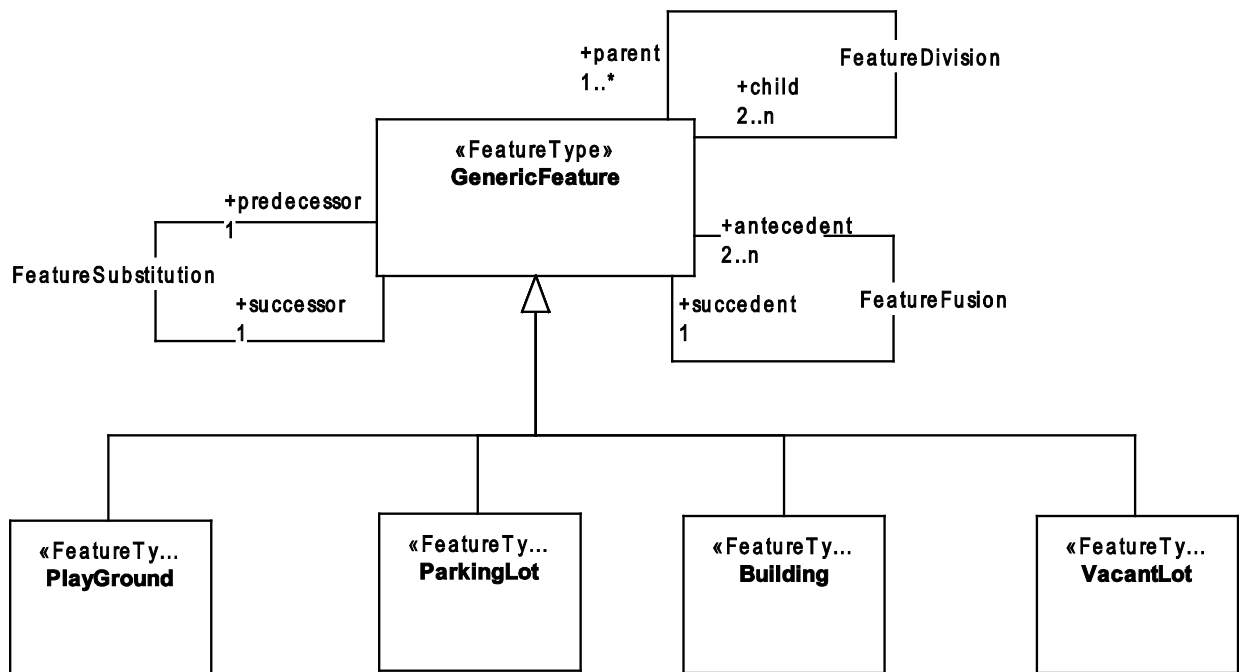


Figure 23 — Example of feature succession at the generic feature level

## 8.7 Spatial rules

### 8.7.1 Rules for modelling applications with spatial properties

This sub-clause describes the requirements for an application schema with feature types that have spatial properties. 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)
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 19107:2003 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.

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

Spatial attribute types are required to conform to ISO 19107:2003.

<b>/req/spatial/schema</b>	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

<b>/req/spatial/attribute</b>	<ol style="list-style-type: none"> <li>1) Spatial characteristics of a feature shall be described by one or more spatial attributes. In an application schema, a spatial attribute is a subtype of a feature attribute (see 7.5), and the taxonomy of its values is defined in the spatial schema, ISO 19107:2003.</li> <li>2) A spatial attribute shall be represented in an application schema in either of two ways: <ul style="list-style-type: none"> <li>— 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</li> <li>— 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.</li> </ul> </li> <li>3) 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.</li> </ol>
-------------------------------	--

NOTE In case 1 the geometry is owned by the feature and in case 2 it is an independent object which 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
<b>Geometric types</b>				
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.

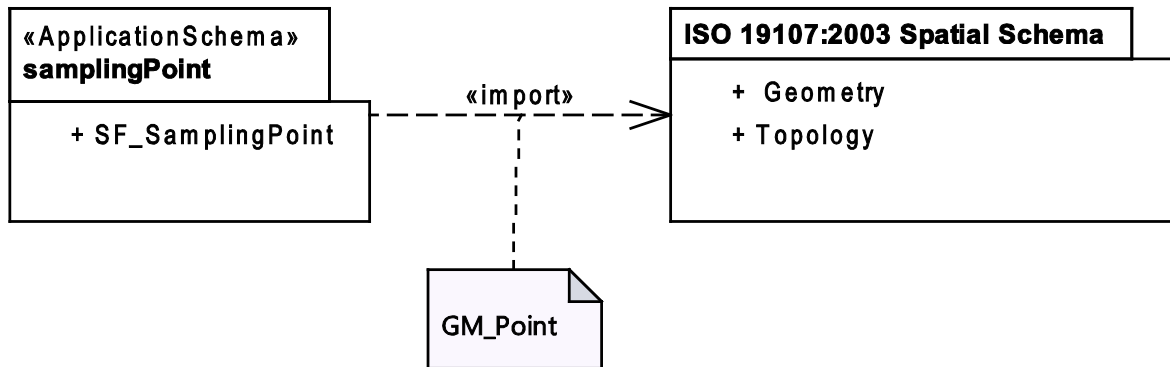
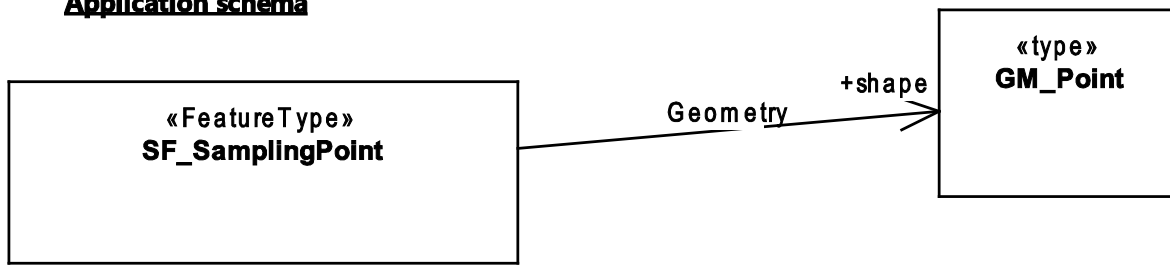
**Model Integration****Application schema**

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.

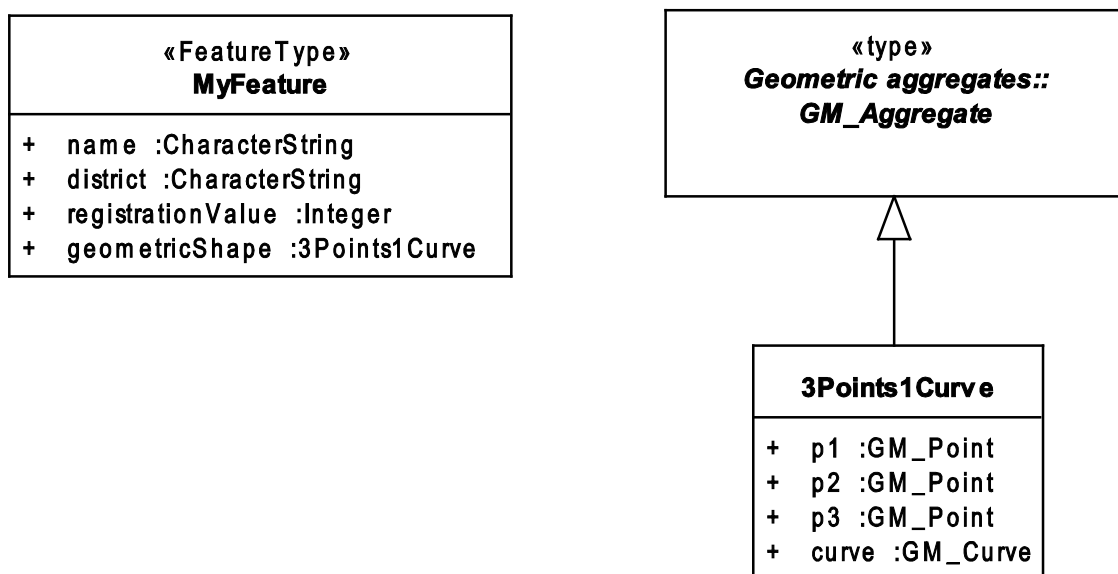
<b>/req/spatial/aggregate</b>	<p>1) GM_Aggregate may be used as the value for a spatial attribute that represents a feature as an unstructured collection of GM_Objects.</p> <ul style="list-style-type: none"> <li>— case 1: GM_Aggregate shall be used as the value for a spatial attribute that represents a feature as an unstructured collection of different types of GM_Object where there is no restriction on the types of contained objects.</li> <li>— 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.</li> </ul> <p>2) A particular composition or constraint on definition of the spatial attribute shall be implemented in the application schema by introducing a new CLASS that carries the user defined spatial configuration with all constraints as a subtype of GM_MultiPrimitive.</p>
-------------------------------	---

**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

## ISO/DIS 19109

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.

- |                             |   |
|-----------------------------|---|
| <b>/req/spatial/complex</b> | <ol style="list-style-type: none"> <li>1) A GM_Complex shall be used as the value for a spatial attribute that represents a feature as a collection of connected GM_Objects, which are disjoint except at their boundaries. Subclasses of GM_Complex may be specified to constrain the structure of the GM_Complex used to represent a particular spatial configuration.</li> <li>2) Features that share elements of their geometry shall be represented as GM_Complexes that are subcomplexes within a larger GM_Complex.</li> </ol> |
|-----------------------------|---|

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).

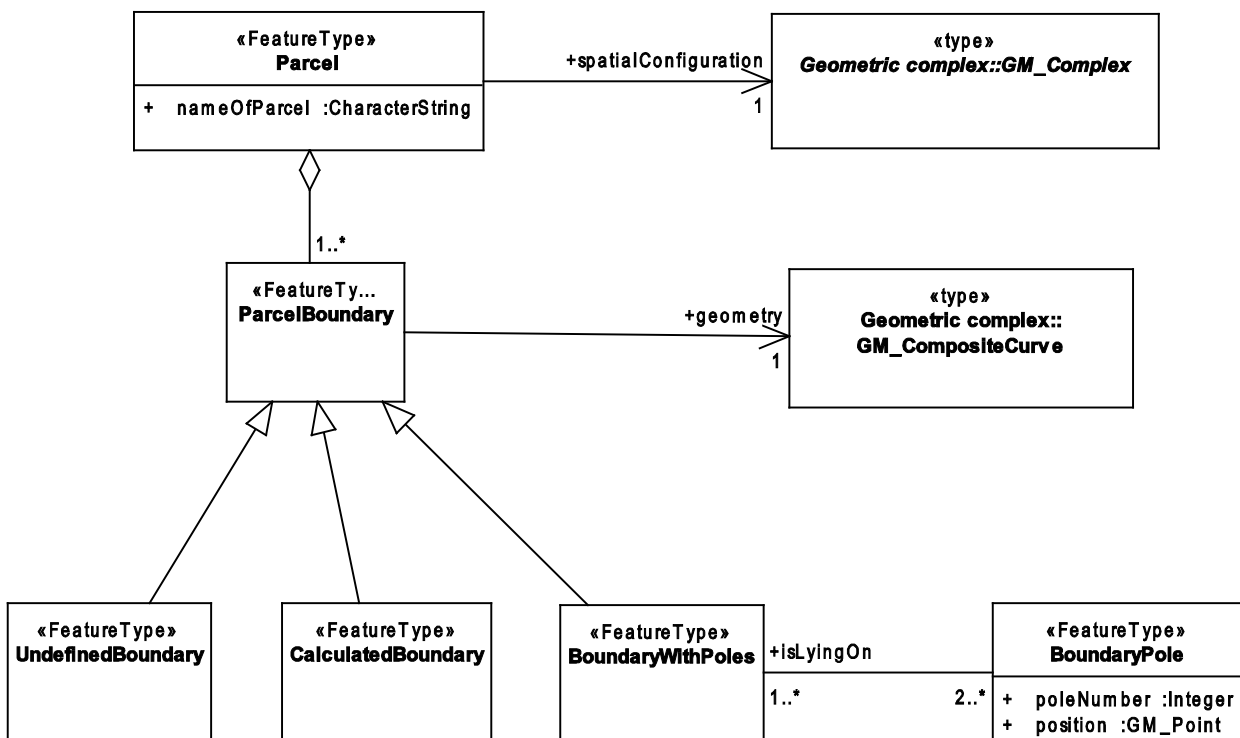


Figure 26 — Example of spatial complex defined in application schema

#### 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.

<b>/req/spatial/composite</b>	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.

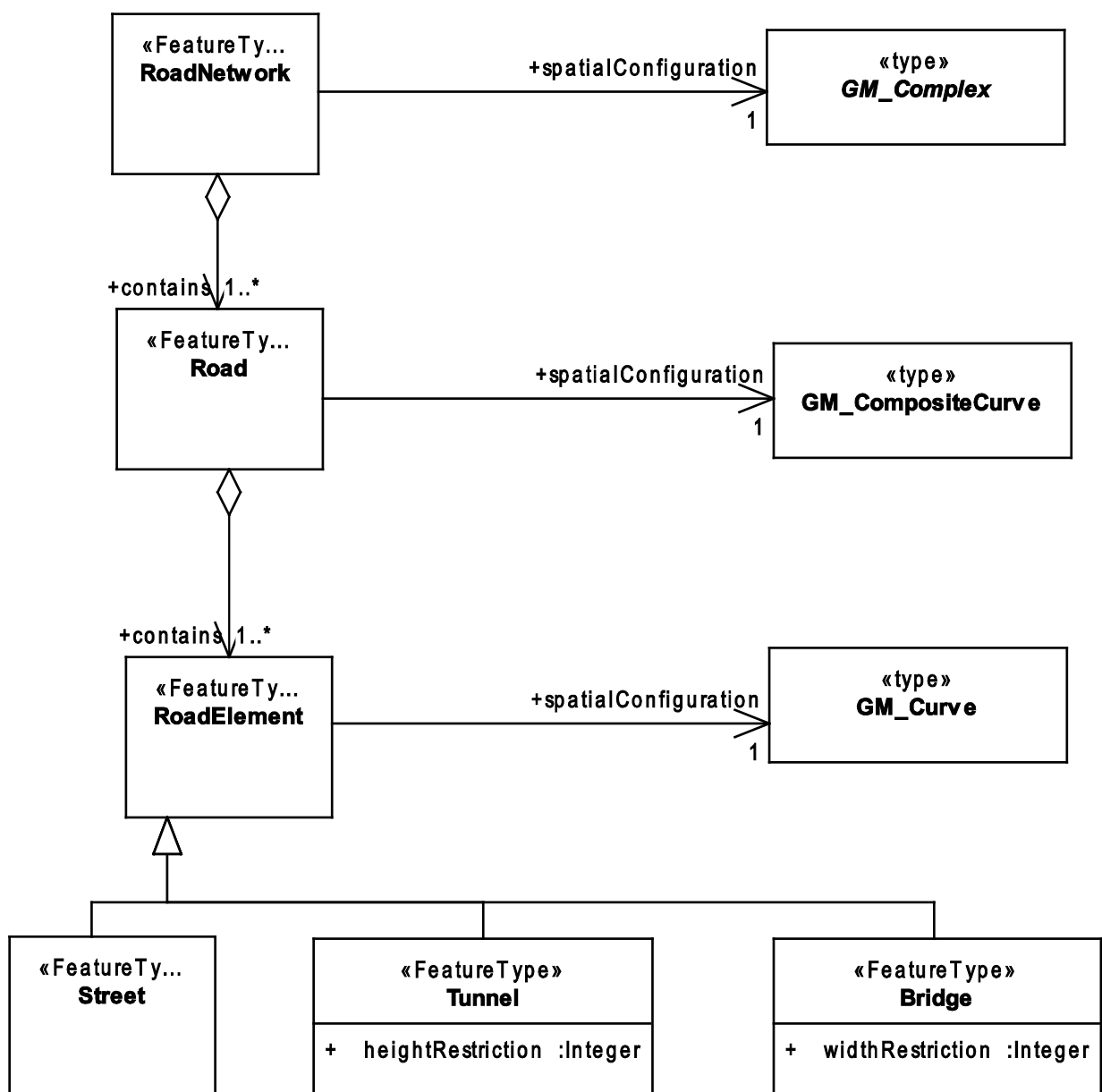


Figure 27 — Example of geometric composites defined in application schema

## ISO/DIS 19109

## 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”.

<b>/req/spatial/geom-complex</b>	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.

<b>/req/spatial/topo-complex</b>	<ol style="list-style-type: none"> <li>1) A TP_Complex shall be used as a spatial attribute of a feature whenever the application requires the explicit representation of connectivity between the geometric primitives that represent the feature. The TP_Complex shall be linked in a REALIZATION ASSOCIATION to a GM_Complex that represents the geometric configuration of the feature. The TP_Complex may be subcomplex within a larger TP_Complex that represents multiple features.</li> <li>2) TP_Complexes may be used to represent attributes of features in order to represent connectivity that is independent of the geometric configuration of the features. In this case, the TP_Complex is not a member of a REALIZATION ASSOCIATION.</li> </ol>
----------------------------------	--

**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”.

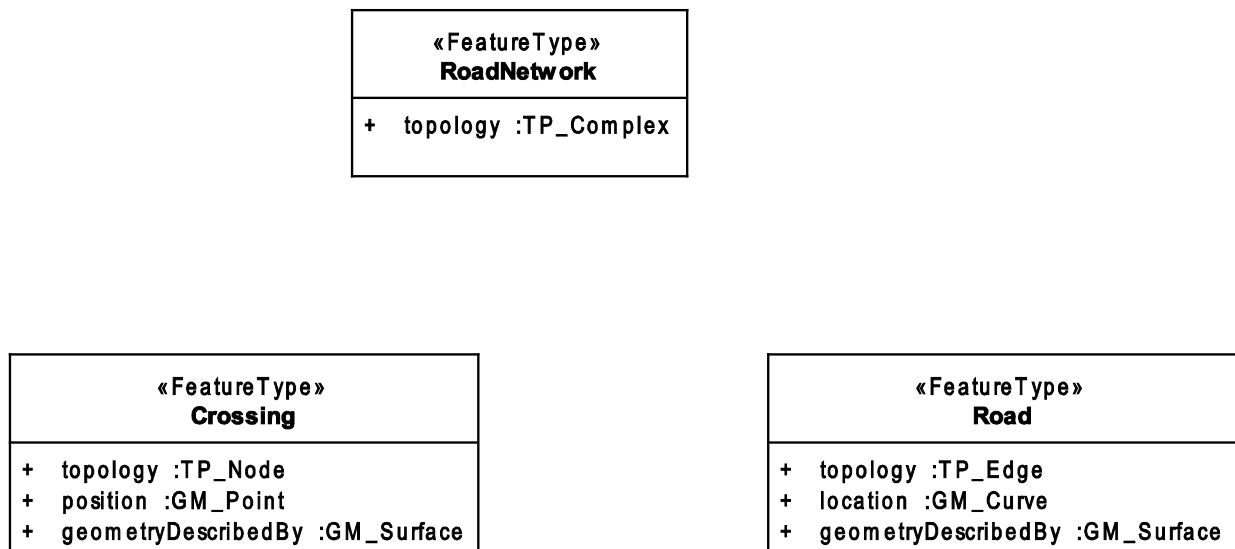
### 8.7.5 Spatial associations between features

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

**NOTE** ISO 19107:2003 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.

<b>/req/spatial/association</b>	<ol style="list-style-type: none"> <li>1) A spatial association between features may be described implicitly by representing the features as geometric or topological objects that are associated with each other in a geometric or topological complex.</li> <li>2) A spatial association that is not described by an ASSOCIATION between the underlying geometric or topological primitives shall be defined in an application schema as an ASSOCIATION between the features.</li> </ol>
---------------------------------	--

**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.

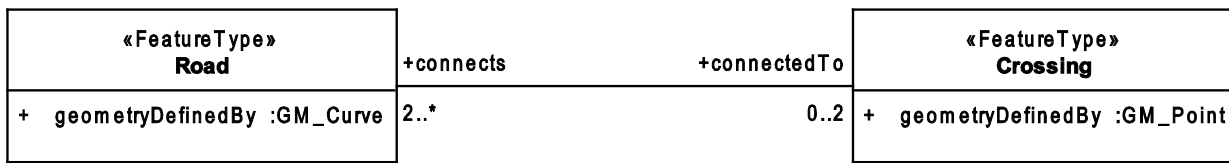


Figure 29 — Example of spatial associations implemented as associations in the application schema

### 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.

<b>/req/spatial/shared</b>	<ol style="list-style-type: none"> <li>1) An application schema may require instances of two or more feature types to share their geometry completely by including a constraint that the GM_Objects representing the features be equal.</li> <li>2) A spatial association that is not described by an ASSOCIATION between the underlying geometric or topological primitives shall be defined in an application schema as an ASSOCIATION between the features.</li> </ol>
----------------------------	---

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 30a shows a transformer mounted on a pole whose geometry is the same GM\_Point that describes the geometry of the pole. Note that this application schema requires every transformer to share the geometry of a pole, but a pole does not have to share the geometry of a transformer.

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

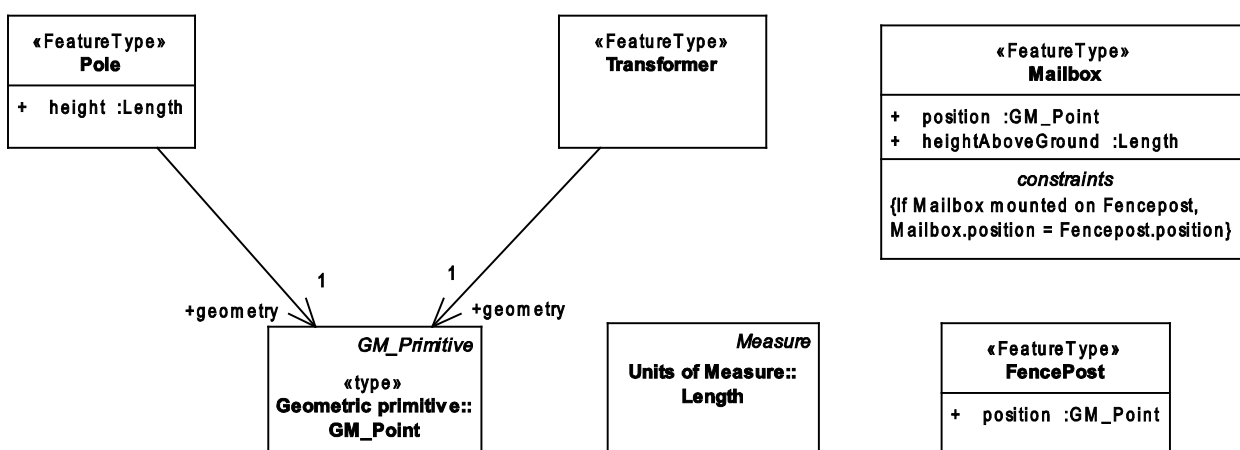


Figure 30 — Examples of features sharing geometry

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.

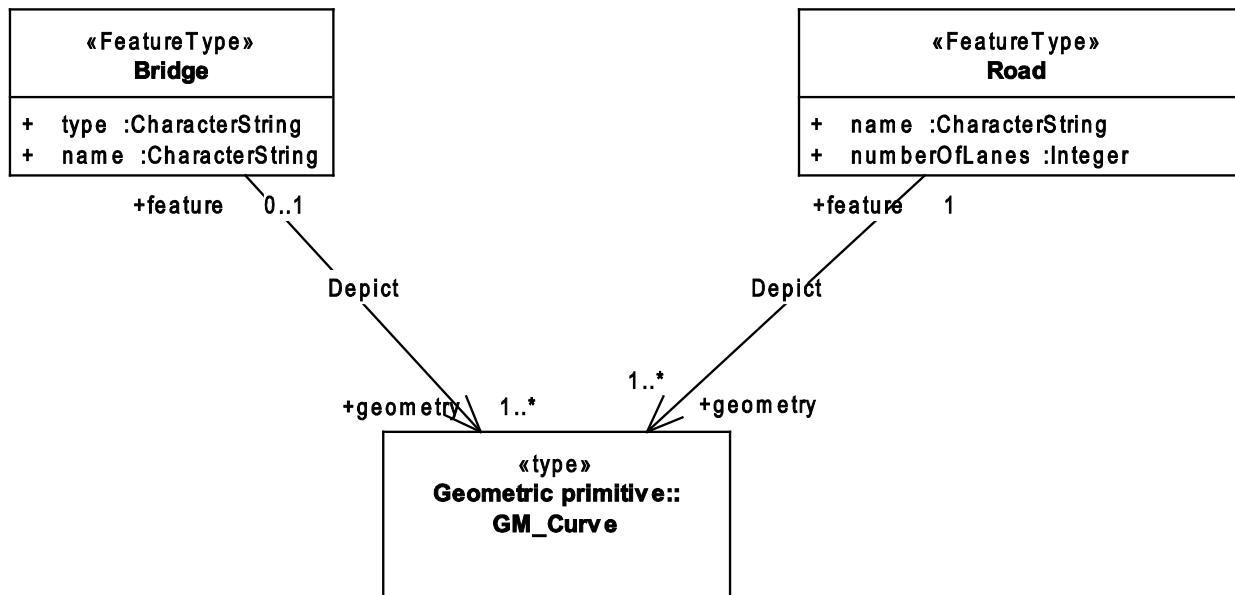


Figure 31 — Example of partial sharing of geometry between features

### 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**

- 1) A point feature shall take a GM\_Point as the value of its spatial attribute.
- 2) A line feature shall take a GM\_Curve or a GM\_CompositeCurve as the value of its spatial attribute.
- 3) An area feature shall take a GM\_Surface or a GM\_CompositeSurface as the value of its spatial attribute.
- 4) 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 19107:2003, a GM\_Curve can contain any number of segments where every segment is one of the subtypes of GM\_CurveSegment. Observe that different segments in one curve do not have to be of the same type. A GM\_Surface can contain any number of patches where every patch is one of the subtypes of GM\_SurfacePatch. Observe that different patches in one surface do not have to be of the same type. If, and only if, the existing set of subtypes of GM\_CurveSegment and GM\_SurfacePatch defined in Spatial Schema

**ISO/DIS 19109**

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**
- 1) If any existing subtype of GM\_CurveSegment or GM\_SurfacePatch fulfils the requirements of interpolation method, this subtype shall be used in the application schema.
  - 2) If, and only if, none of the existing classes fulfils the requirements of the application schema, the application schema shall subtype GM\_CurveSegment or GM\_SurfacePatch (or one of the subtypes) with a CLASS that encapsulates the appropriate data and behaviour for the application-specific interpolation method.

**8.7.9 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**
- 1) Different spatial representations of the same feature are allowed, but they must belong to different complexes.
  - 2) Topologically independent sets of spatial objects must belong to different TP\_Complexes.

**EXAMPLE 1** Consider an urban area with an underground rail system, such as that represented by Figure 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.



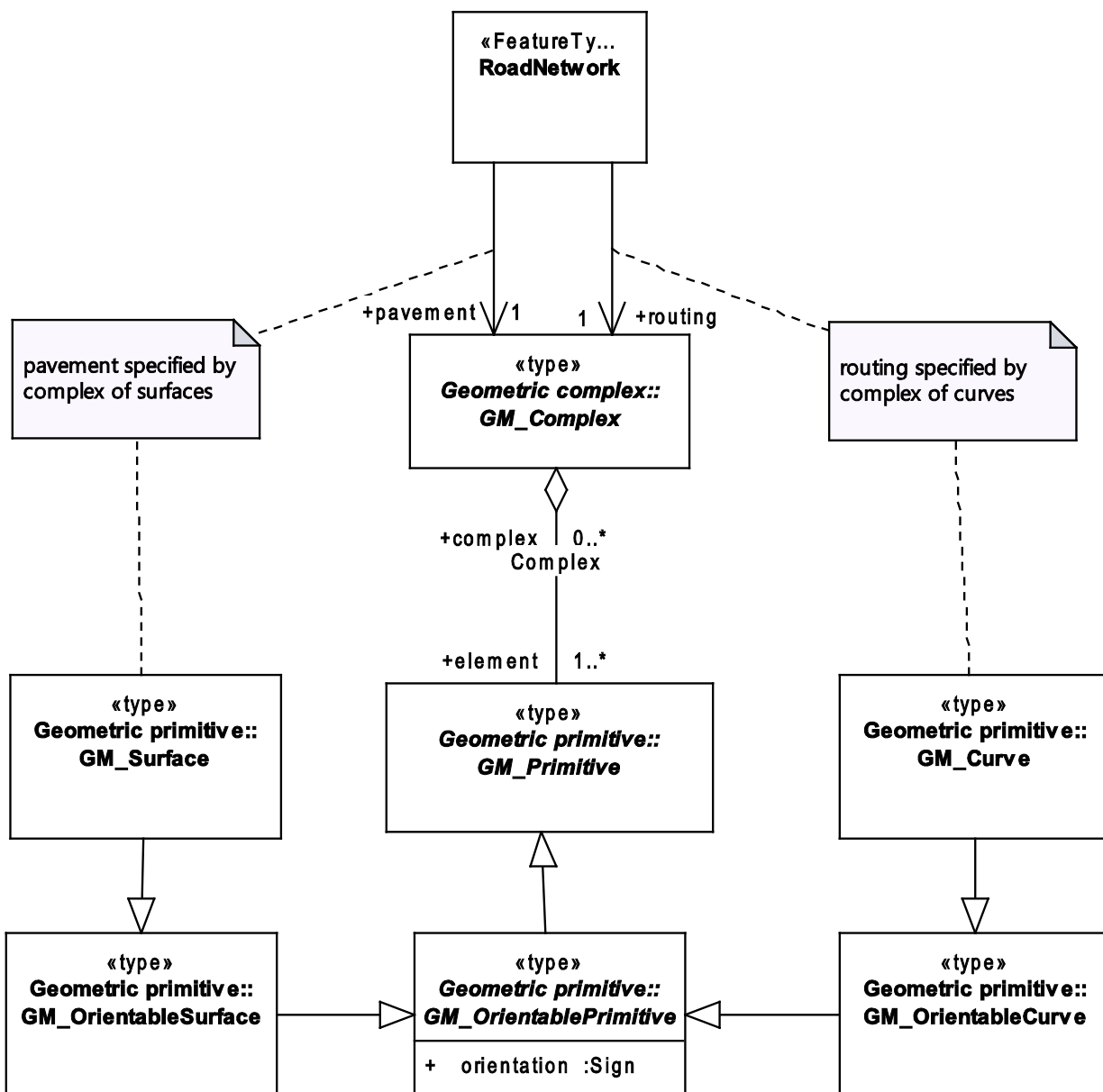


Figure 33 — Example of independent spatial complexes

## 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, precipitation, imagery. 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.

This sub-clause describes the requirements for an application schema with feature types that have coverage-valued properties. 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

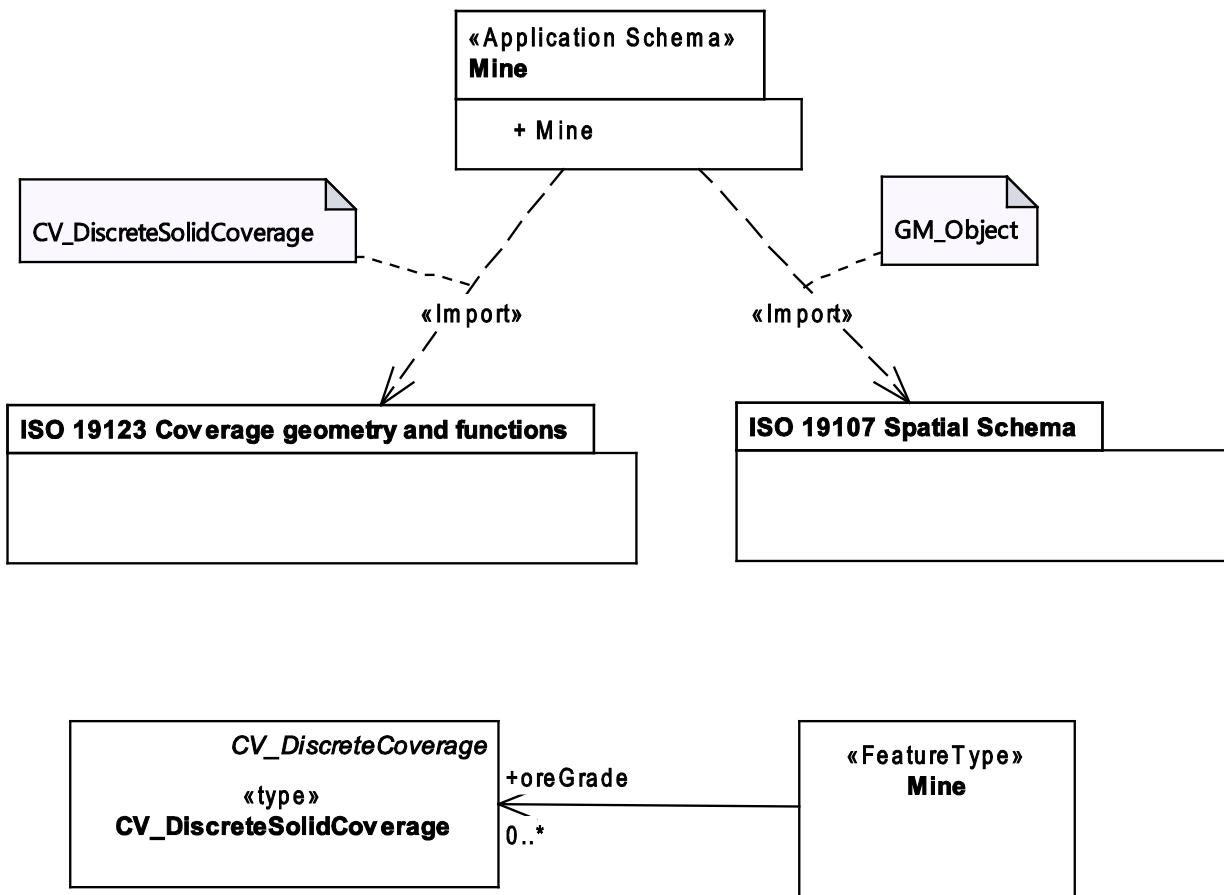
<b>/req/coverage/schema</b>	<ol style="list-style-type: none"> <li>1) Any specification of a coverage function in an application schema shall be in accordance with ISO 19123:2005 and 7.5.8.</li> <li>2) An application schema package that uses coverage functions shall import the coverage schema specified by ISO 19123:2005.</li> <li>3) 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.</li> </ol>
-----------------------------	---

NOTE This approach creates an implementation profile of the relevant ISO 19123:2005 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_ContinuousQuadrilateralGridCoverage 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.

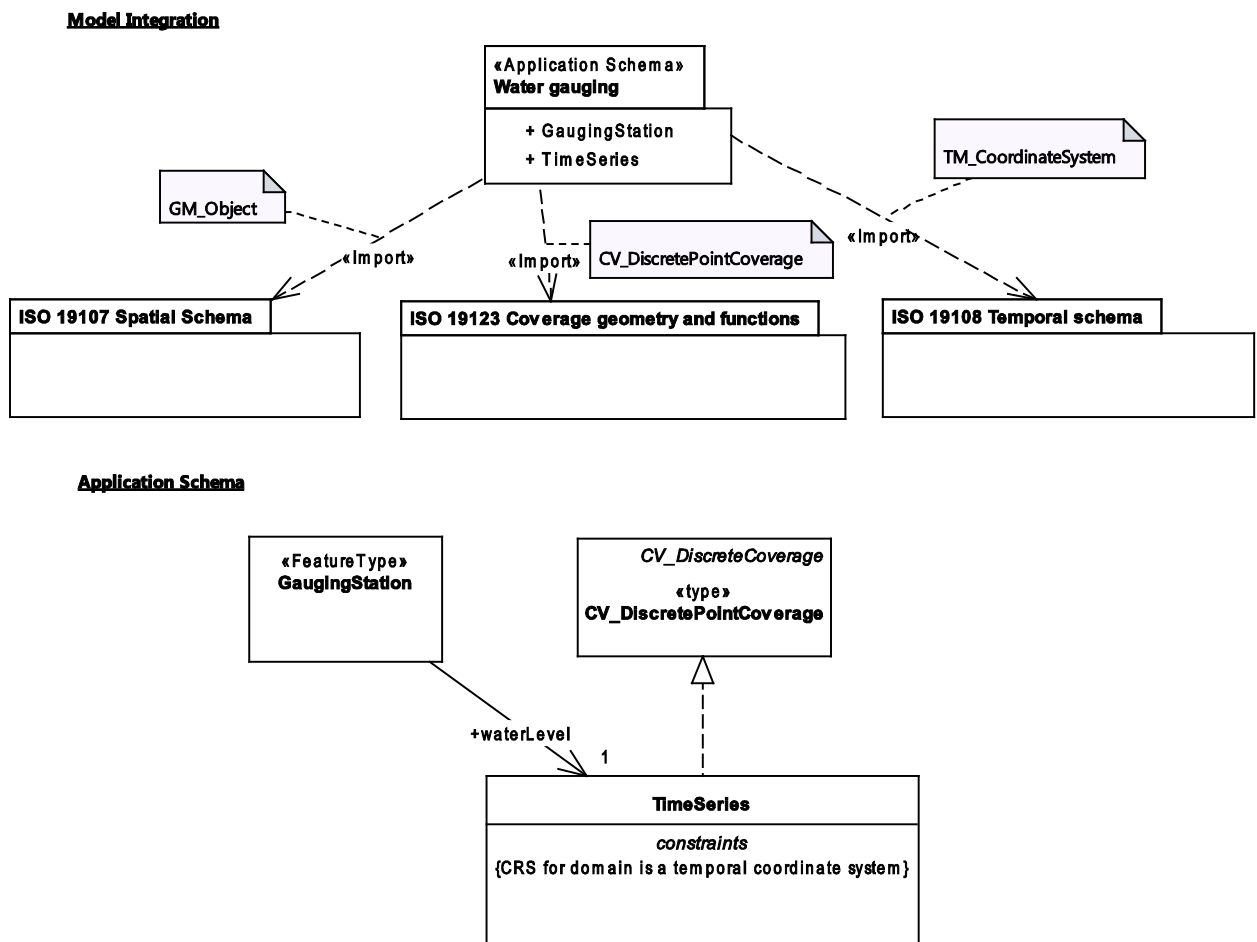
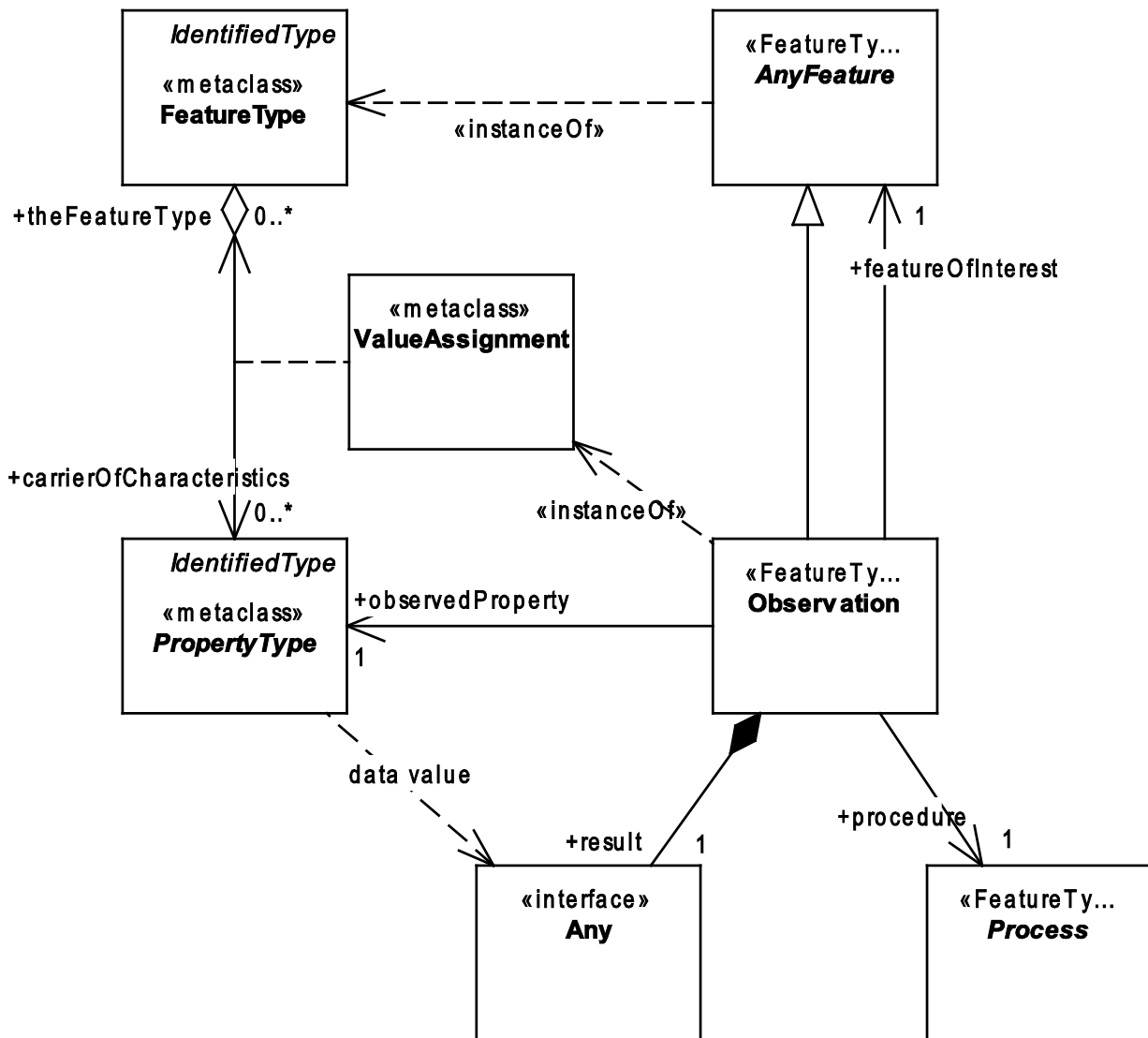


Figure 35 — Example of a coverage-valued property using a specialized coverage

## 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 (7.4.9). 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 19156:2011) 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 19156:2011 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 19156:2011. If necessary, specific observation types may be derived by specialization of one of the types provided in ISO 19156:2011.

This sub-clause describes the requirements for an application schema with feature types that involve value assignment through observation. 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	/req/observation/estimated

ISO 19156:2011 provides a model for observations.

<b>/req/observation/model</b>	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.

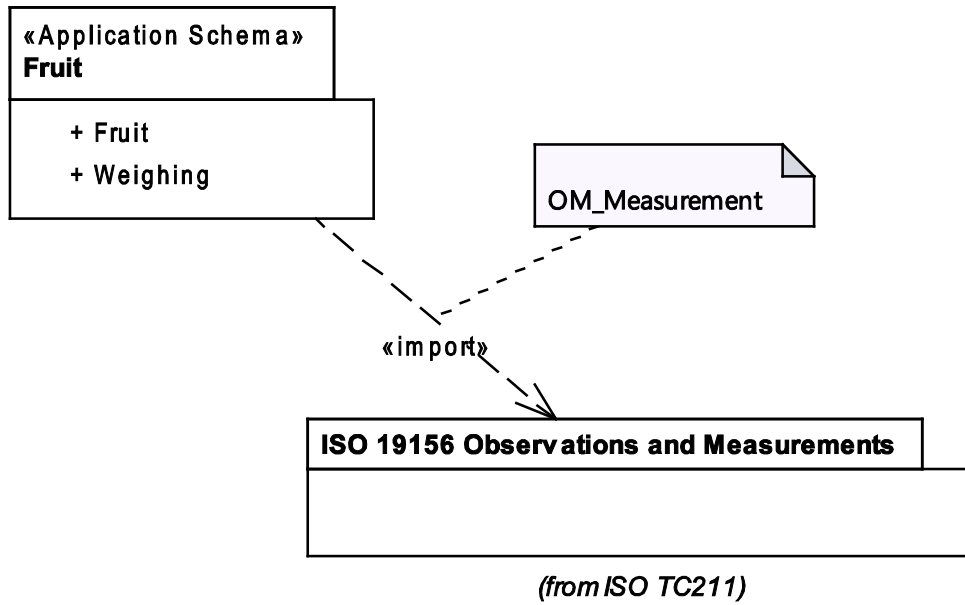
<b>/req/observation/property</b>	The value of observed-property in data conforming to the application schema shall be constrained to be a property-type from the feature catalogue for the application schema, and shall be a characteristic property-type for the feature-type of the feature-of-interest.
----------------------------------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

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

<b>/req/observation/estimated</b>	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** Figure 37 illustrates 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.

**Model integration**



**Application schema**

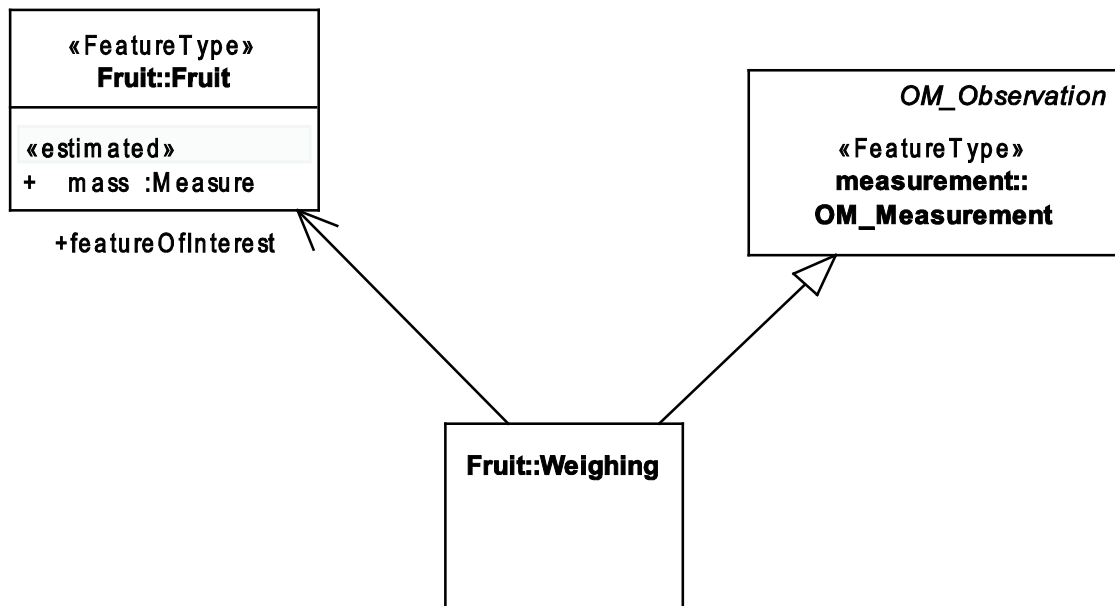


Figure 37 — Example of links between properties and evidence for the assignment of values (1 of 2)

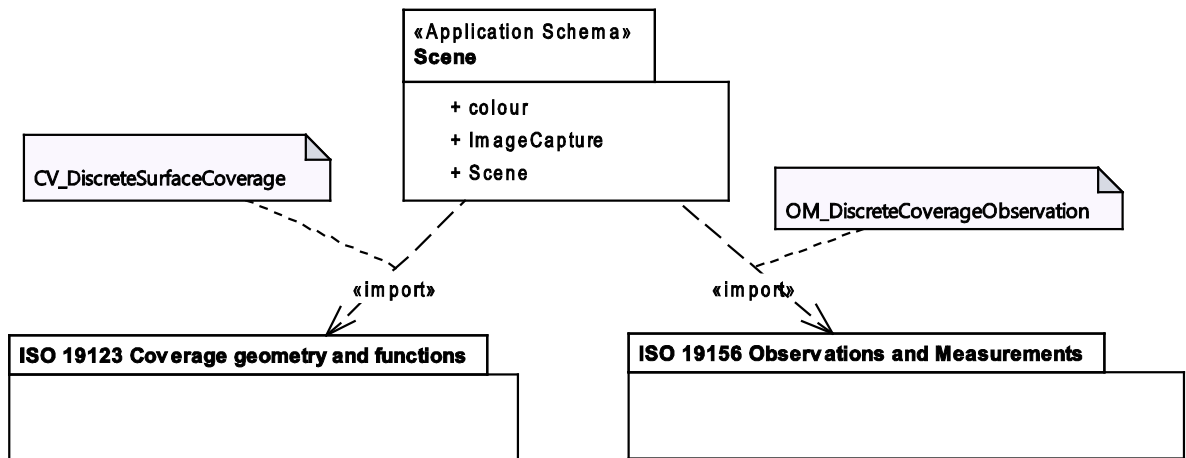
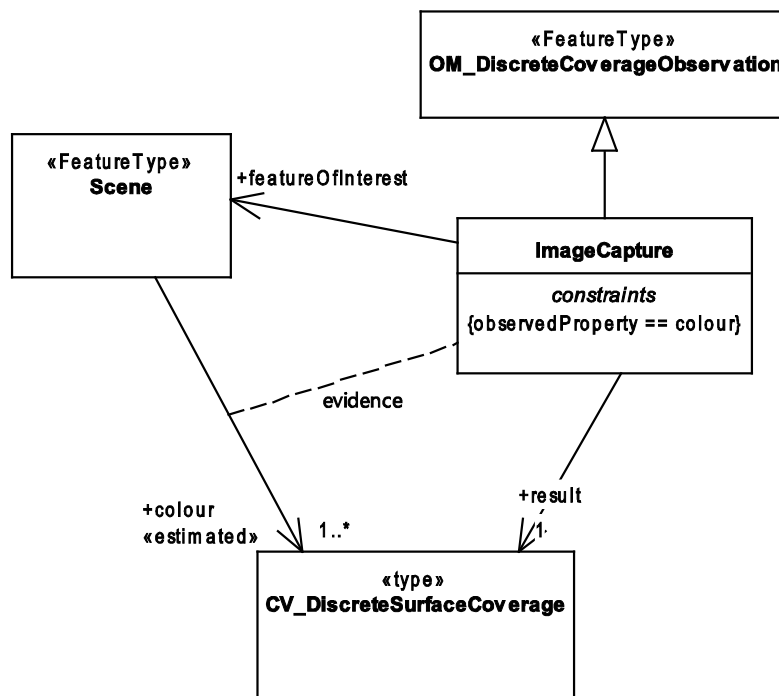
**Model integration****Application schema**

Figure 37 — Example of links between properties and evidence for the assignment of values (2 of 2)

## 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 38 shows the concept of spatial referencing using geographic identifiers where coordinate information is provided.

## ISO/DIS 19109

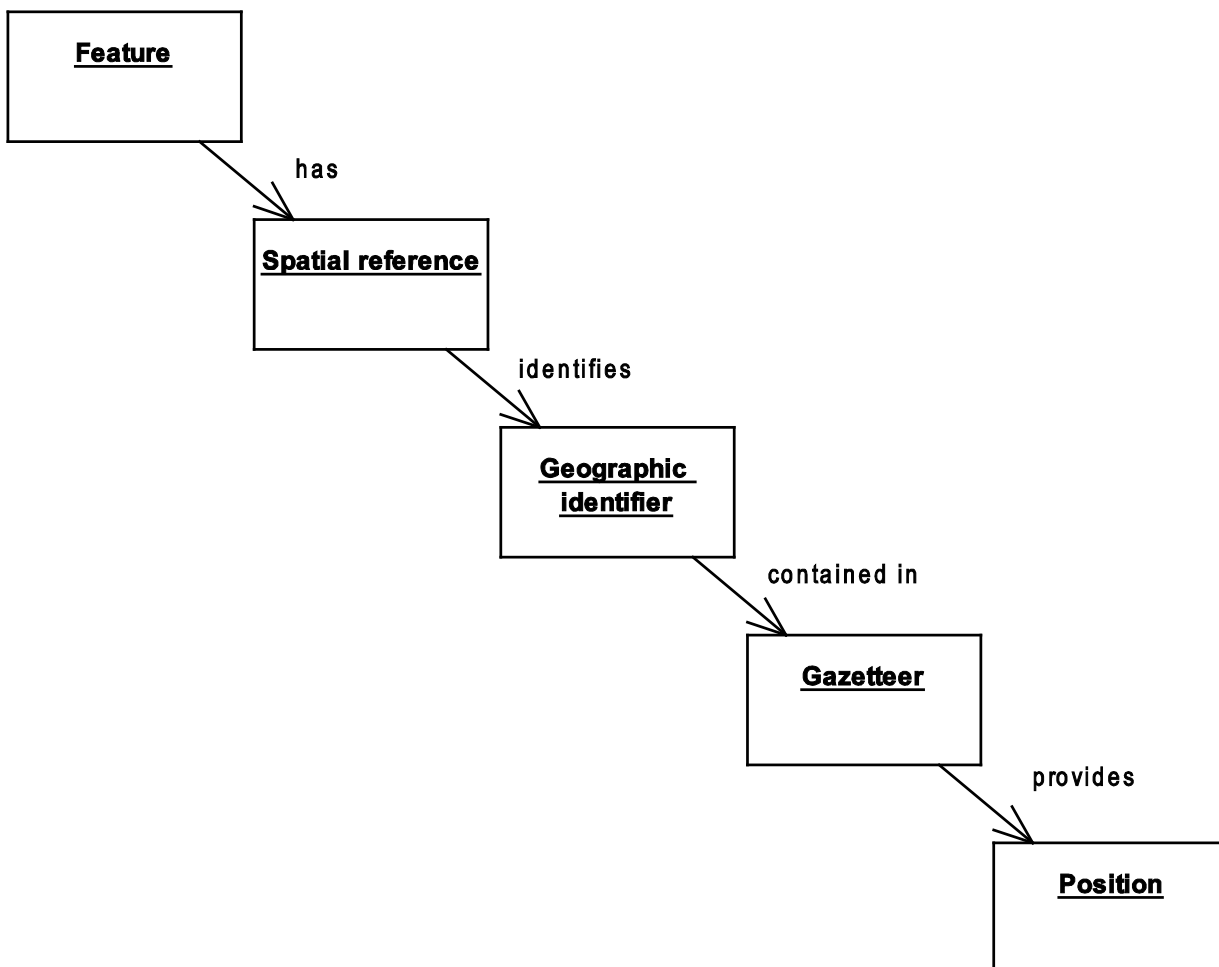
Multiple gazetteers may exist for a spatial referencing system, containing different coordinate representations of the locations. The choice of gazetteer to be used will be application dependent. For example, for a census application, in one case a gazetteer of census area centroids may be sufficient,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

This sub-clause describes the requirements for an application schema with feature types that use geographic identifiers. 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.

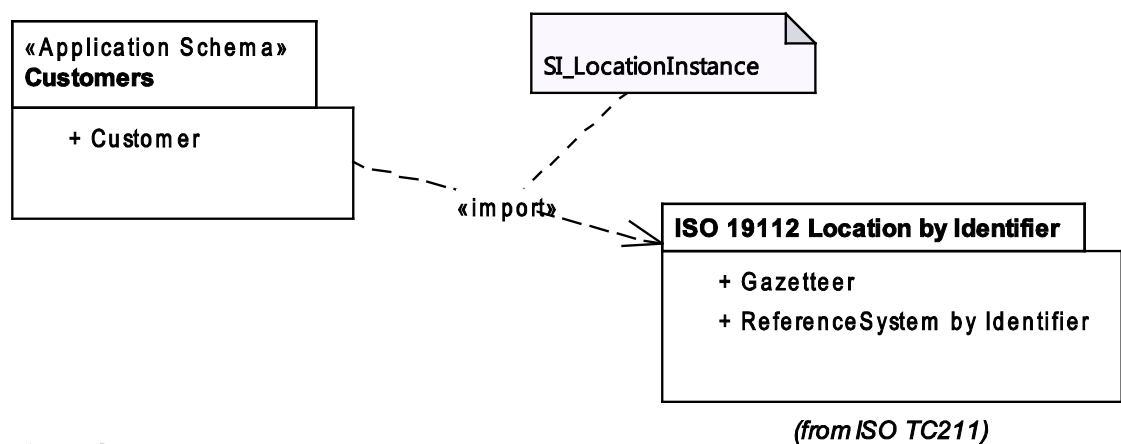


**Figure 38 — Spatial referencing by geographic identifiers**

<b>/req/identifier/general</b>	<ol style="list-style-type: none"> <li>1) The value domain of attributes using spatial referencing by geographic identifiers shall identify objects that are in accordance with ISO 19112:2003.</li> <li>2) 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.</li> </ol>
--------------------------------	---

EXAMPLE Figure 39 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.

### Model Integration



### Application Schema

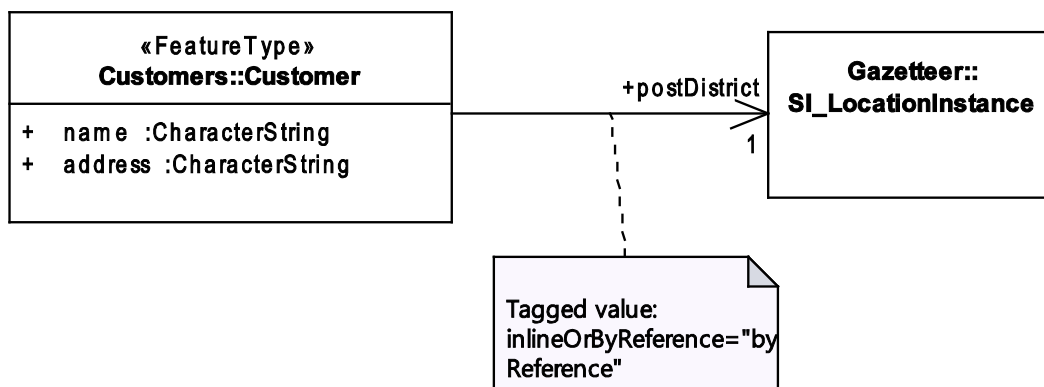


Figure 39 — Example of spatial referencing by geographic identifier

## 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.

This sub-clause describes the requirements for the use of code-lists in application schema. These are formalized as a single requirements class, summarized in Table 31.

**Table 31 — Requirements class for applications schemas using code-lists**

Requirements class	/req/codeList
Target type	UML application schema
Dependency	ISO 19103:201X
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.

<b>/req/codeList/management</b>	<ol style="list-style-type: none"> <li>1) A code-list shall be managed by a competent organisation;</li> <li>2) The code-list and each of its values shall be denoted by a persistent identifier, such as a URI;</li> <li>3) All values of a code-list shall be available, including those that have been deprecated, retired or superseded.</li> </ol>
---------------------------------	---

The code-list formalization defined in ISO 19103:201X is used in application schemas.

<b>/req/codeList/formalization</b>	Where a class attribute takes a value from a code-list, the code-list shall be formalized in the application schema according to ISO 19103:201X.
------------------------------------	--

## 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. This sub-clause describes the requirements for annotating a UML application schema for use in multi-lingual environments. 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

<b>/req/multi-lingual/package</b>	<ol style="list-style-type: none"> <li>1) 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.</li> <li>2) 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:  <div style="text-align: center;"> <p>“{Name}”@{language}</p> </div> <p>Where {Name} is the name in a desired language, and {language} is its language code, taken from IETF RFC 5646.</p> </li> </ol>
-----------------------------------	---

EXAMPLE      “Observations et Mesures”@fr

NOTE        This follows the compact syntax for localized strings used in the W3C N-Triples RDF encoding [2].

<b>/req/multi-lingual/feature</b>	<ol style="list-style-type: none"> <li>1) A FeatureType or PropertyType shall use a TAGGED VALUE "designation" to record its name in an alternative language to the primary language.</li> <li>2) A FeatureType or PropertyType shall use a TAGGED VALUE "definition" to record its textual definition in an alternative language to the primary language.</li> <li>3) The TAGGED VALUES “designation”, “definition” and “description” may be repeated for multiple languages.</li> <li>4) The value of these TAGGED VALUES shall be formatted according to the pattern:  <div style="text-align: center;"> <p>“{Name}”@{language}</p> </div> <p>Where {Name} is the name in a desired language, and {language} is its language code, taken from IETF RFC 5646.</p> </li> </ol>
-----------------------------------	---

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, 7.4.6, 7.4.7, 7.4.8 and 7.4.9;
- d) reference: /req/general/feature, /req/general/property-name, /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.

## ISO/DIS 19109

### 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: /req/general/constraint;
- 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/req/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/req/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 other schemas according to rules specified in this International Standard;
- c) test method: Inspect the application schema to verify that (i) integration with existing schemas, including standard schemas from the ISO geographic information standards, are recorded as explicit dependencies between packages, (ii) packages with the stereotype «ApplicationSchema» do not contain other packages with the stereotype «ApplicationSchema», (iii) dependencies are only between schemas with the stereotype «ApplicationSchema» except for dependencies on standard schemas, and (iv) the dependency graph does not contain mutual dependencies or cycles;
- d) reference: /req/uml/integration;
- e) test type: capability.

### A.3.5 Modelling structure

The test for modelling structures in an application schema is as follows:

- a) test identifier: /conf/uml/structure
- b) test purpose: Verify that the structures of the application schema have been modelled as instantiable classes;
- c) test method: Inspect the application schema to verify that all structures required for the application are represented as instantiable classes or abstract classes with concrete specializations;
- d) reference: /req/uml/structure;
- e) test type: capability.

### A.3.6 Modelling features

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

- a) test identifier: /conf/uml/feature
- b) test purpose: Verify that the structures of the application schema have been modelled as instantiable classes;
- c) test method: Inspect the application schema to verify that all structures required for the application are represented as instantiable classes or abstract classes with concrete specializations, that these classes

**ISO/DIS 19109**

have attributes and association roles representing the basic properties, and associations and aggregations representing relationships between features, and that the classes and named uniquely and well documented;

- d) reference: /req/uml/feature, /req/uml/association, /req/uml/aggregation, /req/uml/attribute, /req/uml/role;
- e) test type: capability.

**A.3.7 Property names**

The test for naming properties in an application schema is as follows:

- a) test identifier: /conf/uml/propertyName
- b) test purpose: Verify that the names of feature properties are unique within application schemas;
- c) test method: Inspect the application schema to verify that all attribute- and role-names are unique within a package, or if names are re-used in the scope of different classes, they represent the same kind of relationship;
- d) reference: /req/uml/propertyName;
- e) test type: capability.

**A.3.8 Value assignment**

The test for value assignment metadata in an application schema is as follows:

- a) test identifier: /conf/uml/valueAssignment
- b) test purpose: Verify that value assignment metadata is modelled as a class;
- c) test method: Inspect the application schema to verify metadata related to the assignment of property values is modelled as an instance of the ValueAssignment class;
- d) reference: /req/uml/valueAssignment;
- e) test type: capability.

**A.3.9 Attribute of attribute**

The test for modelling attributes of attributes in an application schema is as follows:

- a) test identifier: /conf/uml/attributeOfAttribute
- b) test purpose: Verify that attributes of attributes are modelled according to the model;
- c) test method: Inspect the application schema to verify that attributes of attributes follow requirement /req/uml/attributeOfAttribute
- d) reference: /req/uml/attributeOfAttribute;
- e) test type: capability.

**A.3.10 Feature operation**

The test for modelling feature operations in an application schema is as follows:

- a) test identifier: /conf/uml/operation
- b) test purpose: Verify that feature operations are modelled using UML;
- c) test method: Inspect the application schema to verify feature operations are modelled as UML Operations;
- d) reference: /req/uml/operation;
- e) test type: capability.

### **A.3.11 Feature inheritance**

The test for modelling feature inheritance in an application schema is as follows:

- a) test identifier: /conf/uml/inheritance
- b) test purpose: Verify that feature inheritance is modelled as UML generalization;
- c) test method: Inspect the application schema to verify that feature inheritance relationships are modelled as UML generalization.
- d) reference: /req/uml/inheritance;
- e) test type: capability.

### **A.3.12 Constraints**

The test for modelling constraints in an application schema is as follows:

- a) test identifier: /conf/uml/constraint
- b) test purpose: Verify that constraints are expressed using standard UML mechanisms;
- c) test method: Inspect the application schema to verify that constraints are encoded as UML constraints, using either plain language or OCL;
- d) reference: /req/uml/constraint;
- e) test type: capability.

## **A.4 Profiling standard schemas**

### **A.4.1 Extension**

The test for extending the definitions in a standard schema is as follows:

- a) test identifier: /conf/profile/extend
- b) test purpose: Verify that extensions to standard schemas follow the rules;
- c) test method: Inspect the application schema to verify that extensions to the definitions in a standard schema are implemented in a new UML package, as specializations of classes from the standard schema;
- d) reference: /req/profile/extend;

**ISO/DIS 19109**

- e) test type: capability.

**A.4.2 Restriction**

The test for restricting the use of elements from a standard schema is as follows:

- a) test identifier: /conf/profile/restrict
- b) test purpose: Verify that restrictions to standard schemas follow the rule;
- c) test method: Inspect the application schema to verify that restrictions to the definitions in a standard schema are implemented in a new UML package, as new classes with either
  - realization dependencies on the standard classes
  - specialization relationships with the standard classes, with constraints to restrict cardinality or type;
- d) reference: /req/profile/restrict;
- e) test type: capability.

**A.5 Use of metadata schema**

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

- a) test identifier: /conf/metadata/feature
- b) test purpose: Verify that, if metadata concepts appear in the application, these are implemented according to rules specified in this International Standard;
- c) test method: Inspect the application schema to verify that requirement /req/metadata/feature has been obeyed for implementing the metadata schemas;
- d) reference: /req/metadata/feature;
- e) test type: capability.

**A.6 Use of quality schema****A.6.1 Use of the quality schema in an application schema**

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

- a) test identifier: /conf/quality/attribute
- b) test purpose: Verify that, if quality concepts appear in the application schema, these are implemented according to rules specified in this International Standard.
- c) test method: Inspect the application schema to verify that requirement /req/quality/attribute has been obeyed for implementing the quality schemas;
- d) reference: /req/quality/attribute;
- e) test type: capability.

### A.6.2 Additional quality metadata in an application schema

The test for extension of the standard quality schema in an application schema is as follows:

- a) test identifier: /conf/quality/additional-quality
- b) test purpose: Verify that, if elements from the quality conceptual schema are extended in the application, this has been implemented according to rules specified in this International Standard.
- c) test method: Inspect the application schema to verify requirement /req/quality/additional-quality has been obeyed for extending the quality schemas;
- d) reference: /req/quality/additional-quality;
- e) test type: capability.

### A.6.3 Instance quality metadata in an application schema

The test for use of standard quality conceptual schema for feature instances is as follows:

- a) test identifier: /conf/quality/attribute-quality
- b) test purpose: Verify that, if a feature instance quality information appears in an application, it is according to rules specified in this International Standard.
- c) test method: Inspect the application schema to verify that requirement /req/quality/attribute-quality has been obeyed for using the quality schemas;
- d) reference: /req/quality/attribute-quality;
- e) test type: capability.

## A.7 Temporal elements

### A.7.1 Use of the temporal conceptual schema in an application schema

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

- a) test identifier: /conf/temporal/schema
- b) test purpose: Verify that, if temporal concepts are used in the application, these have been implemented according to rules specified in this International Standard;
- c) test method: Inspect the application schema to verify that temporal concepts use the appropriate elements from ISO 19108:2002;
- d) reference: /req/temporal/schema;
- e) test type: basic.

### A.7.2 Temporal attributes

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

- a) test identifier: /conf/temporal/attribute

## ISO/DIS 19109

- b) test purpose: Verify that temporal attributes have been implemented according to rules specified in this International Standard;
- c) test method: Inspect the application schema to verify that temporal attributes are implemented following requirement /req/temporal/attribute;
- d) reference: /req/temporal/attribute;
- e) test type: capability.

### A.7.3 Temporal associations

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

- a) test identifier: /conf/temporal/association
- b) test purpose: Verify that temporal associations have been implemented according to rules specified in this International Standard;
- c) test method: Inspect the application schema to verify that temporal associations are implemented following requirement /req/temporal/association;
- d) reference: /req/temporal/association;
- e) test type: capability.

### A.7.4 Feature succession

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

- a) test identifier: /conf/temporal/succession
- b) test purpose: Verify that feature succession is implemented according to rules specified in this International Standard;
- c) test method: Inspect the application schema to verify that feature succession is implemented following requirement /req/temporal/succession;
- d) reference: /req/temporal/succession;
- e) test type: capability.

## A.8 Spatial elements

### A.8.1 Use of the spatial conceptual schema in an application schema

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

- a) test identifier: /conf/spatial/object
- b) test purpose: Verify that, if spatial concepts are used in the application, these have been implemented according to rules specified in this International Standard;
- c) test method: Inspect the application schema to verify that spatial concepts use the appropriate elements from ISO 19107:2003, and that geometry elements are not used either directly or as the basis of feature-type definitions;

- d) reference: /req/spatial/object, /req/spatial/schema, /req/spatial/aggregate, /req/spatial/complex, /req/spatial/composite;
- e) test type: basic.

### **A.8.2 Spatial attributes**

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

- a) test identifier: /conf/spatial/attribute
- b) test purpose: Verify that spatial attributes have been implemented according to rules specified in this International Standard;
- c) test method: Inspect the application schema to verify that spatial attributes are implemented following requirement /req/spatial/attribute;
- d) reference: /req/spatial/attribute;
- e) test type: capability.

### **A.8.3 Spatial complexes**

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

- a) test identifier: /conf/spatial/complex
- b) test purpose: Verify that spatial complexes have been implemented according to rules specified in this International Standard;
- c) test method: Inspect the application schema to verify that spatial complexes are implemented following requirements /req/spatial/geom-complex and /req/spatial/topo-complex;
- d) reference: /req/spatial/geom-complex , /req/spatial/topo-complex;
- e) test type: capability.

### **A.8.4 Spatial associations**

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

- a) test identifier: /conf/spatial/association
- b) test purpose: Verify that spatial associations have been implemented according to rules specified in this International Standard;
- c) test method: Inspect the application schema to verify that spatial associations are implemented following requirement /req/spatial/association;
- d) reference: /req/spatial/association;
- e) test type: capability.

### **A.8.5 Shared geometry**

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

## ISO/DIS 19109

- a) test identifier: /conf/spatial/shared
- b) test purpose: Verify that shared geometry has been implemented according to rules specified in this International Standard;
- c) test method: Inspect the application schema to verify that constraints on shared geometry are implemented following requirement /req/spatial/shared;
- d) reference: /req/spatial/shared;
- e) test type: capability.

### A.8.6 Single geometry

The test for implementation of features with a single, simple geometry in an application schema is as follows:

- a) test identifier: /conf/spatial/single
- b) test purpose: Verify that features with a single geometry have been implemented according to rules specified in this International Standard;
- c) test method: Inspect the application schema to verify that features characterized by a single geometric primitive (point, line or area) are implemented following requirement /req/spatial/single;
- d) reference: /req/spatial/single;
- e) test type: capability.

### A.8.7 Interpolation method

The test for implementation of spatial interpolation methods in an application schema is as follows:

- a) test identifier: /conf/spatial/interpolation
- b) test purpose: Verify that spatial interpolation methods are specified according to rules specified in this International Standard;
- c) test method: Inspect the application schema to verify that spatial interpolation methods are implemented following requirement /req/spatial/interpolation;
- d) reference: /req/spatial/interpolation;
- e) test type: capability.

### A.8.8 Independent spatial complexes

The test for inclusion of independent spatial complexes in an application schema is as follows:

- a) test identifier: /conf/spatial/independent-complex
- b) test purpose: Verify that independent spatial complexes are specified according to rules specified in this International Standard;
- c) test method: Inspect the application schema to verify that independent spatial complexes are implemented following requirement /req/spatial/independent-complex;
- d) reference: /req/spatial/independent-complex;

- e) test type: capability.

## A.9 Coverages

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

- a) test identifier: /conf/coverage/schema
- b) test purpose: Verify that, if coverage functions are used in the application, these have been implemented and used according to requirement /req/coverage/schema;
- c) test method: Inspect the application schema to verify that coverage functions are implemented and used following requirement /req/coverage/schema;
- d) reference: /req/coverage/schema;
- e) test type: basic.

## A.10 Observations

The test for use of observations in an application schema is as follows:

- a) test identifier: /conf/observation/model
- b) test purpose: Verify that, if observations are used in the application, these have been implemented and used according to rules specified in this International Standard;
- c) test method: Inspect the application schema to verify that observations are implemented following the model in ISO 19156:2011, and that the value of any observation type is constrained to be a property-type taken from the feature catalogue for the application schema;
- d) reference: /req/observation/model /req/observation/property;
- e) test type: capability.

## A.11 Spatial referencing using geographic identifiers

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

- a) test identifier: /conf/identifier/general
- b) test purpose: Verify that, if referencing by geographic identifiers has been used in the application schema, it has been implemented according to rules specified in this International Standard;
- c) test method: Inspect the application schema to verify that requirement /req/identifier/general has been obeyed for implementing the gazetteer schemas;
- d) reference: /req/identifier/general;
- e) test type: capability.

## ISO/DIS 19109

### A.12 Code-lists and vocabularies

#### A.12.1 Management of code-lists

The test for management of code-lists used in an application schema is as follows:

- a) test identifier: /conf/codeList/management
- b) test purpose: Verify that, if a code-list has been used in the application schema, it is managed according to rules specified in this International Standard;
- c) test method: Inspect the application schema to verify that requirement /req/codeList/management has been obeyed for implementing the gazetteer schemas;
- d) reference: /req/codeList/management;
- e) test type: capability.

#### A.12.2 Formalization of code-lists

The test for formalization of code-lists used in an application schema is as follows:

- a) test identifier: /conf/codeList/formalization
- b) test purpose: Verify that, if a code-list has been used in the application schema, it is formalized according to the rules in ISO 19103:201X;
- c) test method: Inspect the application schema to verify that requirement /req/codeList/formalization has been obeyed for implementing any code-lists;
- d) reference: /req/codeList/formalization;
- e) test type: capability.

### A.13 Multi-lingual support

#### A.13.1 Application schema

The test for linguistic and cultural adaptation of an application schema is as follows:

- a) test identifier: /conf/multi-lingual/package
- b) test purpose: Verify that the package containing an application schema contains the information required for linguistic and cultural adaptation;
- c) test method: Inspect the application schema to verify that the tagged values 'language' and 'designation' are included, and used according to the requirement /req/multi-lingual/package;
- d) reference: /req/multi-lingual/package;
- e) test type: capability.

#### A.13.2 Feature and property types

The test for multi-lingual labelling and description of features and properties an application schema is as follows:

- a) test identifier: /conf/multi-lingual/feature
- b) test purpose: Verify that features and properties in an application schema are labelled and described as required for linguistic and cultural adaptation;
- c) test method: Inspect the application schema to verify that features and properties carry the tagged values 'designation', 'definition' and 'description' and used according to the requirement /req/multi-lingual/feature;
- d) reference: /req/multi-lingual/feature;
- e) test type: capability.

## Annex B (informative)

### The modelling approach and the General Feature Model

#### B.1 The four-layer architecture

##### B.1.1 Introduction

This approach to modelling geographic information conforms with a four-layer architecture. Figure B.1 describes the relationships between the General Feature Model, the application schema and the four-layer architecture.

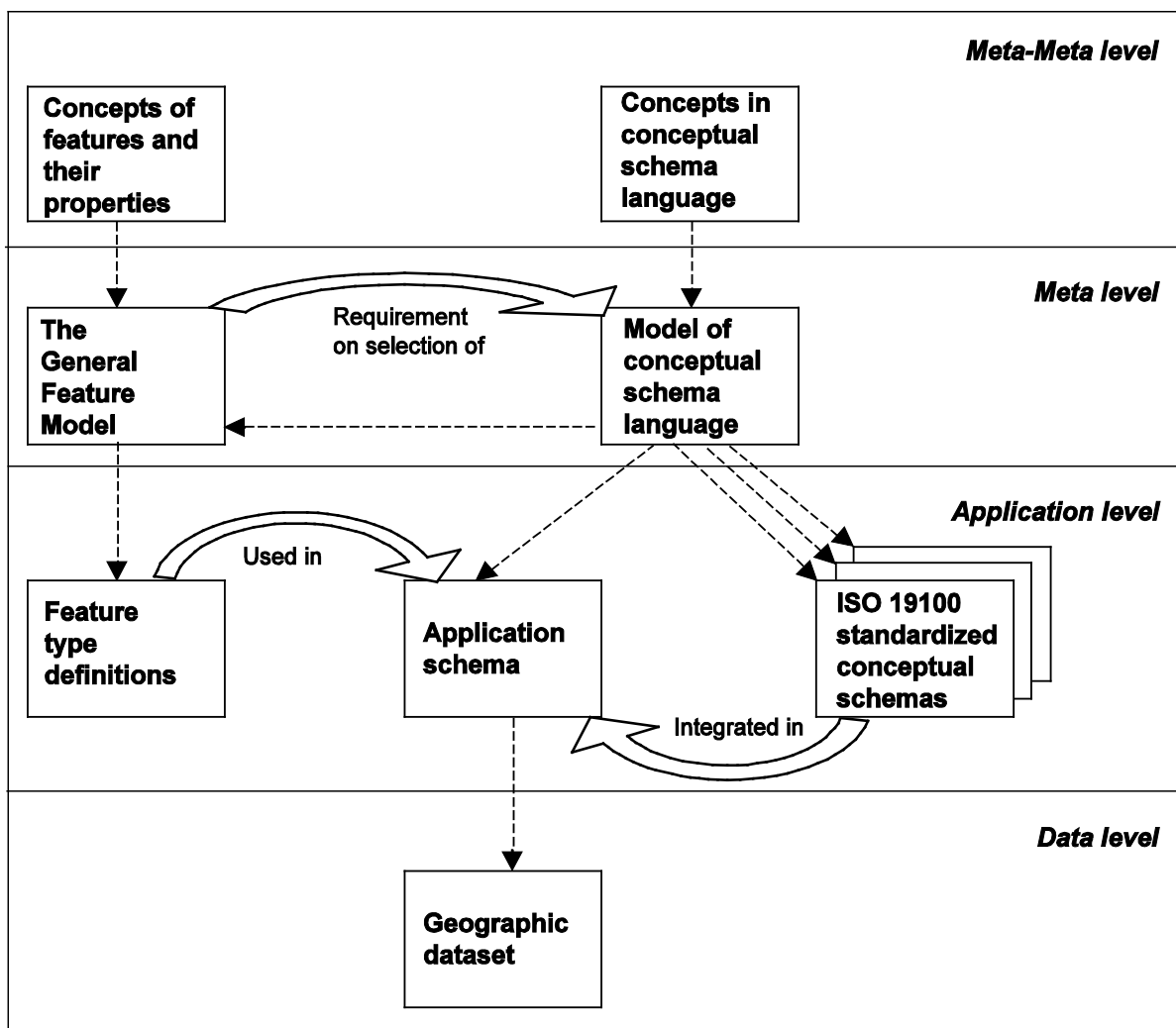


Figure B.1 — The four-layer architecture

### B.1.2 The Meta Meta level and the Meta level

In the four-layer architecture, the Meta Meta level specifies concepts, terms, and formalism. It is usually expressed in natural language.

The Meta level uses definitions from the Meta Meta level. In the Meta level, a syntax (different from natural language syntax) is added to the semantics of the concepts defined in the Meta Meta level.

In the Meta Meta level, we find the concepts of defining features. These concepts are used and structured in a conceptual model, the General Feature Model, which is on the Meta level.

In the Meta Meta level, we also find the concepts of defining conceptual schema languages. Those concepts are used and structured in a conceptual model, Model of a conceptual language, which is on the Meta level. This model is expressed in its own terms, e.g. the UML has the UML-metamodel expressed in UML.

Here is a difference between the General Feature Model and a model of a conceptual schema language. The General Feature Model is not a model expressed in its own terms, it is expressed in a conceptual schema language, that is chosen to be UML.

NOTE In this respect the General Feature Model could be placed in the Application level, but this is not the purpose of the General Feature Model; see also B.1.4.

### B.1.3 The Meta level and the Application level

In the Meta level, we have the General Feature Model. The General Feature Model is used to describe different feature types that are in the Application level.

A conceptual schema language defined in the Meta level is used to express schemas in the Application level. In the Application level, we find the application schema. Some of the other standardized conceptual schemas from the ISO geographic information standards can be used as resources while building an application schema, and are integrated in an application schema when necessary. While many of the standard schemas model the key concepts as abstract types (<<type>>, <<interface>>), these may be used in applications with the understanding that a concrete class that realizes the type is implied.

The standardized conceptual schemas in the ISO geographic information standards are expressed in the conceptual schema language UML. Because of this, it is pertinent to choose UML as the language of the application schema, but this is not a standardized rule.

### B.1.4 The Application level and the Data level

An application schema is in the Application level. It expresses the structure, content and possible behaviour of data which are in the Data level.

### B.1.5 The General Feature Model and the conceptual schema language of the application schema

Feature type definitions are implemented in the application schema in the Application level. This brings forth requirements on the conceptual schema language. There has to be mapping from the General Feature Model into the chosen conceptual schema language. Within the ISO geographic information standards, the chosen conceptual schema language is UML. This mapping from the General Feature Model concepts to UML concepts is described in the main rules in Clause 8.

The General Feature Model serves two purposes:

- gives requirements on the choice of conceptual schema language;
- serves as conceptual model for the classification defining feature types of a feature catalogue according to ISO 19110:2005.

## ISO/DIS 19109

The General Feature Model defines concepts needed within the geographic information domain. There is no conceptual schema language corresponding exactly to these specific needs. There must be a mapping between the concepts of the General Feature Model and the chosen conceptual schema language.

NOTE 1 If this mapping is done one to one, the feature types could be expressed in the terms of the conceptual schema language and not in terms of the General Feature Model.

A feature catalogue developed in accordance with ISO 19110:2005 uses the concepts of the General Feature Model. This means that the feature type definitions used from a feature catalogue have to be mapped into the concepts of the chosen conceptual schema language. On the other hand, the catalogue is not dependent on any specific conceptual schema language.

NOTE 2 An application schema may be documented in parts in a feature catalogue.

### B.2 The term “feature”

In the ISO geographic information standards the term “feature” is used in descriptions of all levels of the four-layer architecture. This is described in Table B.1.

**Table B.1 — Feature used in different levels**

Level in architecture	Use of the term “feature”
Meta Meta level	“feature” as the general concept, not specified being type or instance
Meta level	a CLASS in the UML-expressed General Feature Model, with the CLASS-name “FeatureType”
Application level	a specific feature type representing a class of real-world phenomena, e.g. “Road” an instance of the CLASS “FeatureType” of the General Feature Model expressed in an application schema. The result in an application schema in UML is a CLASS called “Road”.
Data level	feature instance representing a set of data for an instance of the feature type, e.g. the road “Route 66”

NOTE Generally, the reader will understand from the context from which level the term “feature” is used. The term “type” or “instance” is used in addition to “feature” when only one is meant.

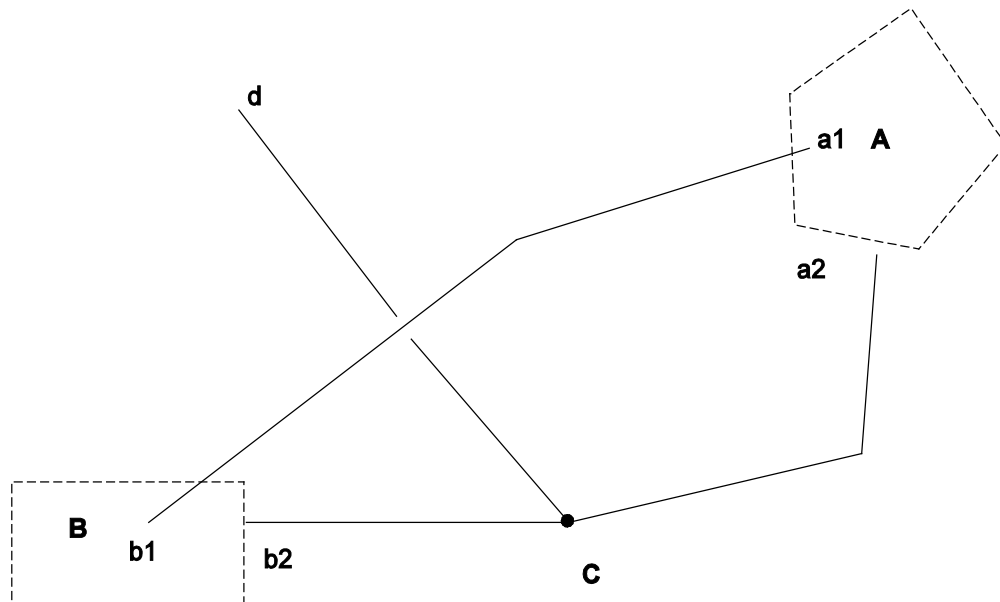
## Annex C (informative)

### Application schema examples

#### C.1 Utility network

##### C.1.1 Introduction

The following example of a high voltage network system is a simplified description of a utility system data model. This example is specially designed to show a separate usage of topology and geometry.



#### Key

a1-b1, b2-C, C-a2 and C-d	four transmission lines
A and B	two main substations
C	a degenerated substation, in fact a tower

**Figure C.1 — Example of utility network**

Three lines are connected to the tower. In this case, the geometry shows the connections. Two lines are connected to each main substation, but here the geometry does not show the connections. There is a crossing without intersection between lines C-d and a1-b1. It is possible because the two transmission lines are not on the same level.

ISO/DIS 19109

C.1.2 The application schema

The data model of the utility application described in UML could be as shown in Figure C.2.

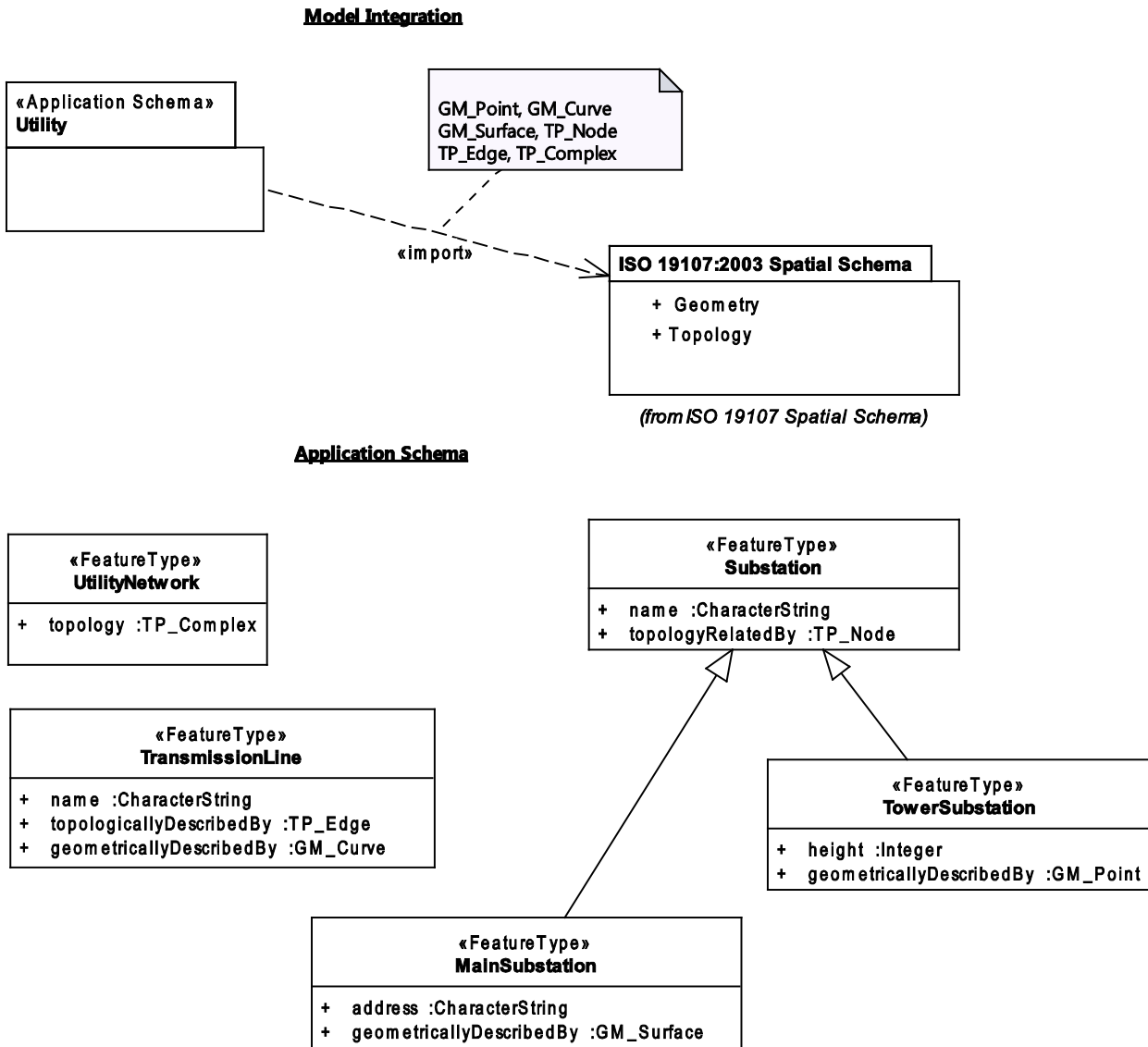


Figure C.2 — Example of application schema for utility network

### C.1.3 An example of the document on Utility Network

#### C.1.3.1 Substation

The substation is described as follows.

Feature Type	Name	Substation
	Definition	a subsidiary station in which electric current is transformed
	Attribute Names	name, topologyRelatedBy
Feature Attribute	Name	name
	Definition	unique name of the substation
	Data Type	CharacterString
Feature Attribute	Name	topologyRelatedBy
	Definition	The substation is a node of transmission network. Constraints: The node shall connect more than one of the transmission lines represented by TM_Edges. This cannot be an isolated node.
	Data Type	TM_Node

#### C.1.3.2 Main Substation

The main substation is described as follows.

Feature Type	Name	Main Substation
	Definition	substation having a function to change voltage
	Attribute Names	address, geometricallyDescribedBy
	Subtype of	Substation
Feature Attribute	Name	address
	Definition	geographic identifier to indicated the location of the main substation
	Data Type	CharacterString
Feature Attribute	Name	geometricallyDescribedBy
	Definition	the geographic extent of the main substation represented as a surface
	Data Type	GM_Surface

**ISO/DIS 19109****C.1.3.3 Tower Substation**

The tower substation is described as follows.

Feature Type	Name	Tower Substation
	Definition	electric tower to distribute electric power
	Attribute Names	height, geometricallyDescribedBy
	Subtype of	Substation
Feature Attribute	Name	height
	Definition	the height of the tower measured from the ground to the top of it. The unit of height is metre
	Data Type	Integer
	Measurement Unit	metre
	Domain Type	1
	Domain of values	height > 0
Feature Attribute	Name	geometricallyDescribedBy
	Definition	The position of the tower represented by a point
	Data Type	GM_Point

**C.1.3.4 Transmission Line**

The transmission line is described as follows.

Feature Type	Name	Transmission Line
	Definition	the passage of electric power between substations
	Attribute Names	name, topologicallyDescribedBy, geometricallyDescribedBy
Feature Attribute	Name	name
	Definition	the unique identifier of the transmission line
	Data Type	CharacterString
Feature Attribute	Name	topologicallyDescribedBy
	Definition	topological relationship between substations. TP_Edge represents this relation.
	Data Type	TP_Edge
Feature Attribute	Name	geometricallyDescribedBy
	Definition	geometric shape of the transmission line. GM_Curve represents the shape.  Constraints: 1. It shall be a realization of the corresponding edge. 2. The boundaries of the curve abut on the boundary of the main substation or the position of the tower.
	Data Type	GM_Curve

### C.1.3.5 Utility Network

The utility network is described as follows.

Feature Type	Name	Utility Network
	Definition	a network supplying the community with electricity
	Attribute Names	topology
Feature Attribute	Name	topology
	Definition	a network topology, which consists of transmission lines as edges and substations as nodes
	Data Type	TP_Complex

## C.2 Administrative units

### C.2.1 Introduction

The following example is based on the Seamless Administrative Boundaries of Europe (SABE) dataset containing data about administrative units of the European countries. The dataset contains the following:

- low-level administrative units and their boundaries which are represented by the feature *administrative area*;
- the *administrative unit hierarchy*, which is a complex feature type carrying the information about each administrative unit and also the hierarchy where an administrative unit on a higher level consists of one or many administrative units on a lower level. Only administrative units on the lowest level are associated with spatial properties (boundary/area).

C.2.2 The application schema in UML

Figure C.3 illustrates the UML application schema of SABE data set.

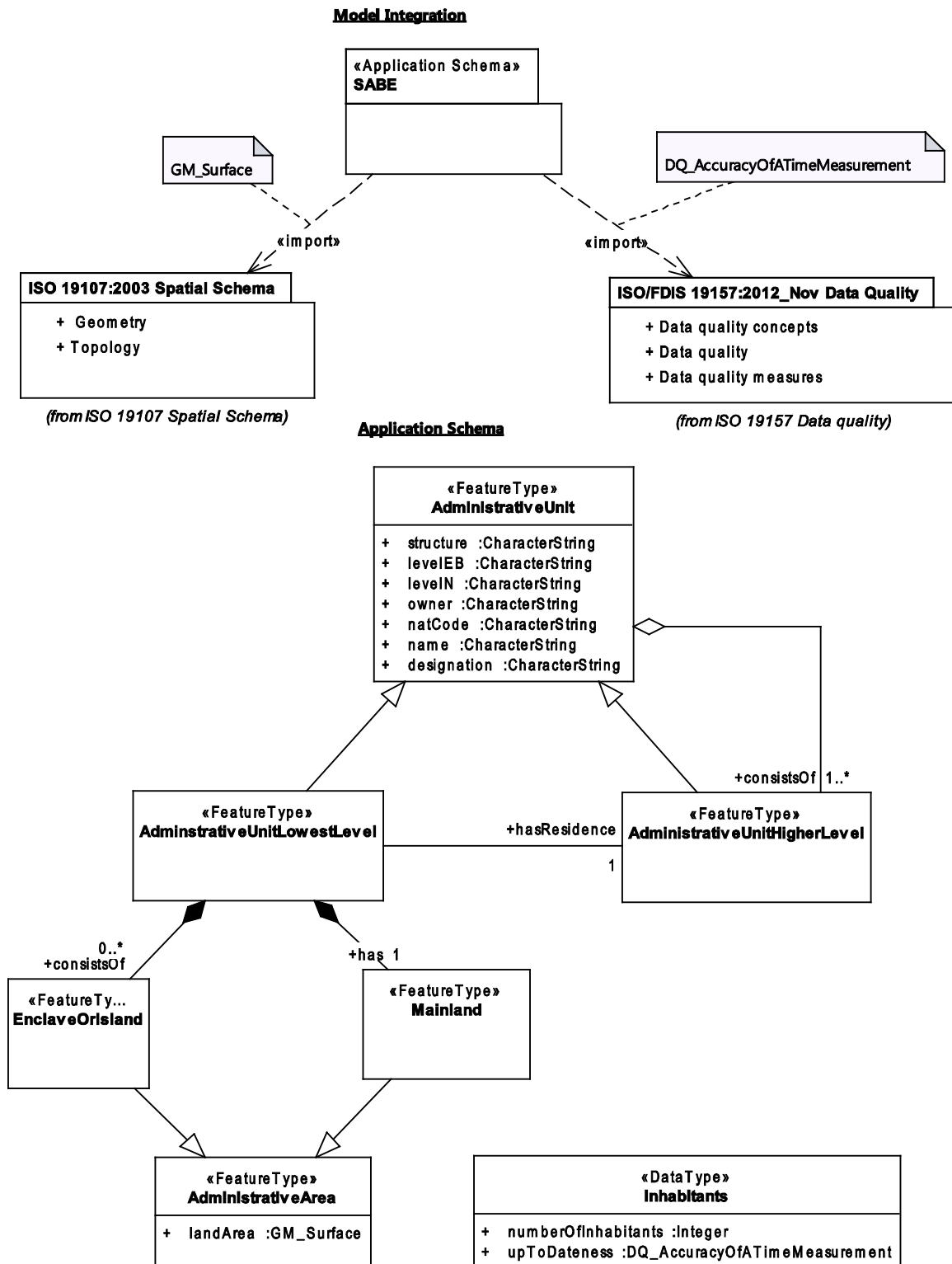


Figure C.3 — UML application schema

## Bibliography

- [1] Booch, G., Jacobsson I., and Rumbaugh J., *UML User Guide*, 1999, Addison-Wesley ISBN 0 201 57168-4
- [2] Grant, J. and Beckett, D. (Editors), *RDF Test Cases*, W3C Recommendation, 10 February 2004. <http://www.w3.org/TR/rdf-testcases> (See N-Triples section)
- [3] Miles, A. and Bechhofer, S. (Editors), *SKOS Simple Knowledge Organization System*, W3C Recommendation, 18 August 2009. <http://www.w3.org/TR/skos-reference/>
- [4] ISO 19101-1:201X, *Geographic information — Reference model — Part 1: Fundamentals*
- [5] ISO 19104:2008, *Geographic information — Terminology*
- [6] ISO 19105:2000, *Geographic information — Conformance and testing*
- [7] ISO 19110:2005, *Geographic information — Methodology for feature cataloguing*
- [8] ISO 19114:2003, *Geographic information — Quality evaluation procedure*
- [9] ISO 19118:2005, *Geographic information — Encoding*
- [10] ISO 19119:2005, *Geographic information — Services*
- [11] ISO 19136:2007, *Geographic information — Geography Markup Language (GML)*
- [12] ISO/TS 19139:2007, *Geographic information — Metadata — XML schema implementation*
- [13] ISO 19143:2010, *Geographic Information — Filter encoding*
- [14] ISO 19150-2:201X, *Geographic information — Ontology — Part 2: Rules for developing ontologies in the Web Ontology Language (OWL)*
- [15] OGC, *Geographic Information — Simple feature access — Part 1: Common Architecture*, OpenGIS Implementation Specification, 2006
- [16] *OMG Object Constraint Language (OCL)*. Object Management Group, January 2012
- [17] Rumbaugh, J., Booch, G., and Jacobsson, I., *UML Reference Manual*, 1999, Addison-Wesley ISBN 0 201-57168-X
- [18] Woolf, A., Cox, S.J.D., Portele, C., *Data Harmonization*. GIGAS Contribution to GEOSS AIP-3 40pp, available at <<http://www.thegigasforum.eu/cgi-bin/download.pl?f=545.pdf>>