Open Geospatial Consortium

Saved 2018-08-15 10:08

External identifier of this OGC® document: http://www.opengis.net/doc/AS//FG/P1/FM/1.0

Internal identifier of this OGC® document:
https://opengeospatial.org/as/17-087r8.html

Logical identifier of standard
https://opengeospatial.org/as/FG/P1/FM/1.0

Internal reference number of this OGC® document: 17-087r8

Version: 1.0

Category: OGC® Abstract Specification

Editor: John R. Herring

Features and geometry – Part 1: Feature models

Information géographique —
Entités géographiques et géométrie —
Partie 1 : Modèles de entités géographiques

Table of contents

List of Figures

List of Tables

i. Abstract

The Implementation Standard "Features and geometry – Part 1: Feature models" describes how geographic information in datasets and databases using a "feature model" are structured, created, stored, queried and manipulated.

URLs in this document are identifiers of structural elements of a standard, such as requirements, tests, requirement classes, and conformances test suites.

ii. Keywords

The following are keywords to be used by search engines and document catalogues.

ogcdoc	schemata
OGC document	SQL
cartography	geographic
controlled vocabulary	geospatial
geospatial information	databases
geography	information management
taxonomy	unstructured data
big data	structured data
ontology	semi-structured data
semantic web	key-value pairs

iii. Preface

This document "Features and geometry – Part 1: Feature models" covers:

1. Feature models describe the digital entities in which information is represented and its various logical structures for the representation of real world phenomena.
2. Schematic, ontological and taxonomic definitions and representations of features and their properties and relations.
3. Definitions of the structures and operation associated to these digital entities to represent, manipulate and query feature data based on those models. This includes organization for data and database structures such as Relation, Object and NoSQL databases, for object systems either static or dynamic class-based systems.
4. Additional parts will define coordinate reference and geometric systems consistent with geodesy to insure the metric and semantic accuracy of results of the analysis of such data in such models.

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

iv. Submitting organizations

The following organizations, members of the OGC Simple Features SWG (standards working group) submitted this Document to the Open Geospatial Consortium (OGC):

v. Submitters

All questions regarding this submission should be directed to the editor and the submitters:

Name	Affiliation
John Herring (ed.)	Oracle USA
Paul Birkel	Geosemantic Resources, LLC
Hideki Hayashi	Hitachi, Ltd., Central Research Laboratory
David Valentine	University of California, San Diego Supercomputer Center
Clemens Portele	interactive instruments GmbH
Satish Sankaran	Esri
Keith Ryden	Esri
Carl Stephen Smyth	Open Site Plan

Comments on the content of this standard were given both informally and formally by members of the OGC. The OGC OAB (OGC Architecture Board) reviewed this document and made both types of comments. Written comments were submitted by the following individual members curing the public review:

Name	Affiliation
Jean Brodeur	GéoSémantic Recherche
Dimitri Sarafinof	Institut National de l'information géographique et forestière (IGN France)

This series of documents is intended to replace the Simple Feature standard, which has been under discussion since at least as early as 1997. Thus, the discussions creating this document has been going on for 21 years.

Features and geometries – Part 1: Feature models

https://opengeospatial.org/as/FG/P1/FM/1.0/[1]

1 Scope

1.1 The first part: feature models

This standard enumerates the requirements for defining and representing geographic features in geographic information systems, applications and transmittable data formats.

This "Part 1: Feature models" defines a new General Feature Model with various representations, and how it determines the basic form of geographic information. This is derived from and extends **ISO 19109,** which, in its current form, only addresses schema-based data storage defined in UML, but leaves possibilities open for other non-schematic design systems as extensions, such as dynamic object systems, as JavaScript and other formats using key-value pairs.

The strict schematic formulations such as might be used in a complied object language or in query languages requiring a predictable format such as SQL in relational databases or any Object-Oriented Databases which depend on a compiled set of object types, tend towards uniformity in the structure of any feature class. This schema approach matches formal abstractions and is useful in controlled applications that depend on the consistency of a strict relational or object-class structures. In this sense, strict schemata are an interoperability barrier between different applications which may share semantic structure but not implementation structures.

The more flexible feature data structures described in this document, will use semantics, taxonomies and ontologies, to interpret a variable structure that match a common taxonomy and constraints but may vary in their structures because two instances in the real world might not be fully describable by a single "attribute template". The approach follows the ideas of a controlled vocabulary that strictly defines terms such as, in this document, the names of the features' types, relations and properties but includes restriction in a class only if demanded by the semantics of reality. Such approaches have proven workable in medical, legal and cultural record keeping, see [15] and [17], and its use is not new to geographic information. Feature and attribute catalogs (which are controlled vocabularies) have been in use in digital mapping since the 1980’s, see [7].

Semantically structured data sets aim to represent reality, not the specific needs of a single application. This leads to application independent data stores that can support all applications because they support a data structure that is both flexible and extendable. This shifts the support of interoperability from creating transfer formats, to creating inclusive, flexible, and thereby interoperable data stores usable by a wide range of application.

1.2 Other parts of this standard

"Part 2: Metrics for geometry" will describe the distinctions between "Euclidean geometry" in a plane such as might be appropriate for a large-scale map and "geodetic, ellipsoidal and non-Euclidean geometry" in general on a curved surface such that of the Earth. The most common implementation is ellipsoidal geometry using the datum surface. Spherical geometry is sufficient for a small area using a local best fit spherical approximation consistent with local curvature. Global use of a single sphere is problematic; for example, geodesics on a sphere close in a single arc (great circles) but on more general ellipsoids, only the meridians and the equator do so. The other issue is that the ellipsoidal coordinates used are based not on the central angle of the radial line to the position, but the tangential angles of the surface, which introduces other subtle and non-uniform errors in "spherical approximations".

"Part 3: Simple geometry" will describe the simplest mathematical geometry definitions, based on linear interpolation in the **CRS**. The first 3 parts are a replacement for **ISO 19125-1:2004, Geographic information – Simple feature access – Part 1: Common architecture**. For completeness, this part will also include definitions for geodesic and rhumb "line" interpolations which complete the concept of "line" from Euclidean geometry. Linear equations in the CRS match the Cartesian coordinate equations for lines. The concept of rhumb lines match the constant bearing concept in navigation, which are lines in maps using a Mercator projection. While linear curves depend on the coordinate applied to the geography, geodesic and rhumb lines are controlled by the geometry of the reference surface (usually an ellipsoid of the CRS). Geodesics

[1] URLs inserted into the text are identifiers for sections (which will end in "/"), or paragraphs (which will not).

match the concept of the shortest distance curve. What is a single curve type in flat Euclidian spaces requires 3 different curve types in non-Euclidian, non-flat, ellipsoidal geometries.

Further parts will describe other geometry and topology structures and a set of parallel standards will define WKT (well-known text) and JSON (JavaScript Object Notation) for both features and geometry. It should be noted that WKT and JSON are dependent on key-value encodings, which can use taxonomies to organize the keys, and in some cases, the values as in code lists.

1.3 Geometry as feature spatial extent

A feature is a representation of a real-world object. For spatial applications, the most important property of a feature is location which will be dealt with in later parts of this standard on geometry, which will be derived from and extend **ISO 19107 Geographic information — Spatial schema** and will contain implementation guidance and suggestions. The corresponding SQL syntax will be defined in **ISO/IEC 13249-3: Information technology — SQL Multimedia and Application Packages - Part 3: Spatial**. Where possible, the SQL/MM approaches to geometry will be paralleled in this series of standards, except where an equivalent approach is more consistent with the common mathematical approached used in CAD/CAM applications, modified to fit non-Euclidean reference surfaces. The only example of this divergence currently-known is the use of projective space in the common mathematical derivations for B-spline curves. The current SQL/MM approach uses a slightly different approach to including weights for points, which would have to be translated to projective space for the common B-spline implementations. In SQL, a purely schema form is used, and dynamic data approaches belong to the looser database models such as NoSQL which depends less on transaction and more of data size and simplicity (big data approaches, see [5]).

1.4 Normative language

All normative language (statements that directly affect the implementation of items compliant to the standard, called "standardization targets") is restricted to **requirements**, **recommendations** and **permissions**. The word "**shall**" implies that conformance requires the statement to apply in the conditions cited in that statement; these are referred to as "**requirements**" and are numbered with "**Req#**".

The word "**should**" implies that the use of best practice recommends the statement to apply in the conditions cited, and these are referred to as "**recommendations**" and are numbered with "**Rec#**".

The word "**may**" implies that conformance and practice allow the statement to apply in the conditions cited, and these are referred to as "**permissions**" and are numbered with "**Per#**". Permissions are not necessarily complete, in the sense that everything not specifically blocked by a requirement is permissible. Permissions are often used to prevent the over-interpretation of requirements which might block a valid implementation of the requirement as stated. This is akin to the open-world assumption, e.g. all approaches are permitted unless explicitly and unambiguously forbidden by a requirement.

Any similar words used the text body ("must", "might", "will", etc.) are not official normative statements and are not marked as such; they usually refer to a logical result of compliance to one or more normative statements, and often appear in the discussion of normative implications. No unmarked statement is normative although it may be truly implied by the marked normative statements. The statements (requirements, recommendations and permissions) marked and numbered are normative even if the "normative language" is inappropriately used.

Requirements ("**Req#**") are collected in requirements classes. Corresponding conformance classes, in Annex A, collate all requirements associated to each conformance class which outline or suggest tests that may be use in to prove conformance with the associated requirements class. Each requirement class and conformance class have a URL label in the namespace of this document. Conformance classes often require conformances to a previously defined class.

Recommendations ("**Rec#**") and Permissions ("**Per#**") can be anywhere in the document because they are logically not testable and are not addressed in a Conformance Class "testable items" in any sense. Any test of a requirement that blocks the usage of either any recommendation or permission is invalid. All normative statements, requirements, recommendations, permissions and tests are labelled statements a having a URL in the namespace of the clause, which in turn is in the namespace of this document defining the standard. Any test suite that references this document should use the URL’s to validate its procedures. All such names are listed immediately after each named item.

2 Conformance

2.1 Conformance classes: https://opengeospatial.org/as/FG/P1/FM/1.0/general/

A conformance class tests the ability of a system to conform to the mandatory requirements expressed in this document. Recommendations and Permissions are not tested. The Conformance Clauses are in Annex A where requirements are associated to required tests. There are three conformance classes in Annex A which will gather the Requirements of the corresponding requirement classes, clauses 6.2, 6.3, 6.4 for representing feature models types and suppling suggestion to how the tests may be formulated:

- A.1 Feature Taxonomy Conformance
- A.2 Feature Ontology Conformance
- A.3 Feature Schema Conformance

In a Feature Taxonomy or Controlled Vocabulary, two lists of "well-known" terms are created to act as named keys and searchable indexes for both:

- Feature Types, and
- Property Types (which include association and its roles)

The abstract root of the Feature Types is always "Feature" defined in 4.7, Properties are attributes that describe feature instances usually represented by data types or relationship roles that represent associations to other features. The abstract root of all properties is always feature property defined in 4.15. Subclassed under properties is the feature relation defined in 4.16 which is a collection of feature relation roles defined in 4.17.

Depending on the flexibility of the system being uses, a Feature Association can be implemented as a Feature where its roles viewed as "properties" which have a type of Feature Identity. For example, we often give names to these relations, as in "roads cross at an intersection" where intersection becomes a relationship between the roads that meet there. People often do this by naming these place by their properties, such as Five-Points (as in Manhattan and Huntsville, Alabama) or Three-Rivers (as in Pittsburg (Allegheny, Monongahela, and Ohio Rivers) and California, a braided river section on the Kaweah River north of Lake Kaweah, in the Sequoia forests, a Shangri-La for whitewater).

**Rec1 Each feature or property type should be accompanied by a semantically valid definition stating which real-world object might be represented or described by this type.
https://opengeospatial.org/as/FG/P1/FM/1.0/general/Rec1**

**Per1 Each feature or property type may be defined by a reference to a standard language-specific dictionary.
https://opengeospatial.org/as/FG/P1/FM/1.0/general/Per1**

Note to Per1: In English, the Oxford and Webster's dictionary may be the default for "British" and/or "American" English. More recent Oxford dictionaries carry extensive variations in spelling that can cover several dialects and local idioms.

**Per2 Each feature instance may be indexed by any number of semantically consistent feature types and be associated to and indexed by any number of properties or association roles.
https://opengeospatial.org/as/FG/P1/FM/1.0/general/Per2**

**Per3 In a Feature Ontology, further restrictions may be included to enforce semantic constraints on the data sets. These constraints may include information about valid property and association roles for individual feature types.
https://opengeospatial.org/as/FG/P1/FM/1.0/general/Per3**

**Per4 Any feature instance may be associated to multiple, semantically consistent feature types.
https://opengeospatial.org/as/FG/P1/FM/1.0/general/Per4**

Example: A bridge may carry a road way, a railway, a canal or a walking path, so that the bridge's geometry is likely to be part of several feature types even though it is one physical object (real-world phenomena).

In a Feature Ontology as defined in this document, definitions for features, properties and relationships specify the possibilities for the data representations; i.e. the form of the digital objects each representing one possible view of a feature – see [4], [8], [9], [10], [12], [16] and [20]. An ontology must first contain a taxonomy but can go further to include constraints consistent with those real-world features that can enforce semantic consistency. A feature data set conformant to an ontology is one in which the feature, properties and relationship are semantically consistent with their definitions in the specified taxonomy and other ontological constraints. Since these constraints should reflect "real-world" truths, so in a sense, a valid taxonomy-structured data set which matches the real-world should pass any valid ontology constraints specified by the ontology. In other words, those constraints are tests for the quality of data collection in its representation of reality. If a collected data set is known to be validly consistent with reality, then a "violation" of an ontological constraint may be a flaw in the constraint, not necessarily in the data collection.

Note: In this document, the terms "feature", "class", "category" and "taxon" may be used interchangeably. They are essentially different metaphorical views of the same concepts. In a web view, instances of a feature class are feature "resources" and will often be associated with URI identifiers.

"Whereas current Web content has implemented the separation of content from presentation to a large extent, the Semantic Web aims to externalize the inherent semantics from syntax, structure and other considerations. This has led to a layered characterization of metadata that have been used to capture these various aspects of information", see [16], p 24. This separation of concern leads to two meta-requirements (see[16], p 26):

- "To enable the abstraction of representation details such as the format and organization of data and capture the information content of the underlying data independent of representational details"[2].
- "To enable the representation of domain knowledge describing the information domain to which the underlying data belongs"[3].

The solution for this separation of semantics and structure is found in the use of both schema and ontologies for the definition of data, see [16], Chapters 5, 6 and 7. Classical structured data formats such as relational or XML data require schemata, but the merging of that data requires the understanding of the underlying semantics. Fluid data structures such as JSON (JavaScript Object Notation) or WKT (well-known text) are based on key-value structures which are dependent on data definition and mostly independent of data structure. The merging of these flexible data formats is dependent on a common set of keys, and a common set of definition for those keys (a common set or mergeable sets of taxonomic metadata).

If the data set is controlled by a taxonomy, each feature instance can incorporate any classes, attributes or associations that are consistent with the semantics of the definitions involved. In this case, each feature instance is essentially its own class, an aggregation of the properties defined by its accumulated edits. "Dynamic" programming languages such as JavaScript (objects represented by JSON) and various forms of LISP, see [18], can support this approach. Even a strict object language can be used to build a dynamic class system using associations in lieu of compile-time linkages, see Figure B.1. Ontology languages begin with a taxonomy but can also add other semantic restrictions like those in an object model. Object models can be mapped to ontologies, but not all ontologies are as restrictive as a non-dynamic UML model. Even in this case, it would be possible to include a feature relation that interlinks object representing the same physical object.

Rec2 In a non-dynamic object model, if two separate class instances represent the same physical object, then there should be an "alternate representation relation" that links those instances based on that equality. https://opengeospatial.org/as/FG/P1/FM/1.0/general/Rec2

A feature schema implementation contains definitions for each feature type including the properties that might or must be contained and the relationship in which it might or must participate. Since the items in a schema are the same as the items in

[2] This implies that constraints in a formal ontology should be held submissive to reality, so that a single valid counter-example for a constraint, should imply that the ontology is in question, not the reality.

[3] In the case of this specification, the domain knowledge is mainly in the taxonomy and ontology associate to the data. In geographic information the domain is the "feature representing a real-world phenomenon" See 4.6.

an ontology, a feature schema should contain a feature ontology and taxonomy (or a reference to a default dictionary). This recommendation for an ontology to be associated to each schema is new but has been a best practice for decades. As mentioned, it is the integration of information management in the semantic web that is driving this movement to a more formal semantic approach to data handling.

In cases where the "dictionary definitions" for the names of the features, properties or relations is sufficient to meet the ontology's requirements, then these definitions will be sufficient. In cases where multiple languages are used in the application community for the ontology, a well-structured multi-lingual cross walk will be necessary to alleviate issues of interoperability within and between multi-lingual environments.

2.2 Conformance targets

This standard defines conformance for combinations of associated instances as follows:

1 A taxonomy and the features and properties it allows consistent with the taxonomy's definitions,

2 An ontology is a taxonomy where the entities are consistent with the ontology's restrictions and

3 A schema is an ontology which fully defines the structures of each feature and property type allowed.

A taxonomy lists all feature and property types. A single feature instance can have any number of feature type "tags" and any number of property-value pairs if the semantics of the combination are valid, which will be true if the feature instance reflects the existence and state of the real phenomena that the feature represents, see 4.7.

An ontology consistent with this standard will contain a taxonomy, see **Req25**, and include constraints, based on the semantics of the interrelationships of the features and properties. In the best circumstances, the features possible in an ontology will be the same as would be collected by the taxonomy alone which, implying that the listed constraints reflect reality.

A schema consistent with this standard will contain a taxonomy, see **Req27**, but many schemata consistent with previous standards will not. In most of those cases, it defines feature-type and property-type names consistent with geographic tradition and are consistent with common dictionary entries (e.g. Oxford or Webster's). A schema is more likely to be associated to a specialized application which requires consistent object structures. Schemata tend to segregate feature types, possibly by theme, however the case where a real-world feature/phenomenon is represented by two separate feature types, depending on the system, the digital representation of this feature may be separated into multiple entity/object classes based on applications implying that the real-world feature is instantiated as a digital entity multiple times. If the system recognizes this case, it has several mechanisms to represent the reality:

1 A single feature-identity is used for multiple digital-entities which have separate software instantiations, having different schema representations.

2 In a dynamic object model where a single object/entity can instantiate multiple schema classes, the data instances can be structurally isomorphic to the standard taxonomic model.

2.3 Existing Implementations

Some cloud computer implementations use taxonomies, or their near equivalents to structure, index and search for data, see [21]. Big Data application literature similarly discusses the use of taxonomies to control the semantic of tags and the integration of structured, semi-structured and unstructured data, see [1] and [10].

In the literature, the "taxonomy to ontology to schema" spectrum discussed in this standard is usually referred to as "unstructured to semi-structured to structured" data. The elements of the feature taxonomy are feature class names, the elements of the property taxonomy are property or association role names. Ontologies may add constraints on data element instances and schemata may add constraints which will include a full object model. These constraints are often most useful in testing data for consistency. Each instance of feature shall have a feature identity to facilitate association specification (see **Req18**).

3 Normative references

The following publications contain normative information on the topics in this document.

ISO 19101-1:	**Geographic information — Reference model**
ISO 19103:	**Geographic information — Conceptual schema language**

ISO 19109:	**Geographic information — Rules for application schema**
ISO 19111:	**Geographic information — Spatial referencing by coordinates**
ISO 19126:	**Geographic information — Feature concept dictionaries and registers**
ISO 19135-1:	**Geographic information — Procedures for item registration -- Part 1: Fundamentals**
ISO/IEC 13249-3:	**Information technology — SQL Multimedia and Application Packages - Part 3: Spatial**
ISO 19162:2015:	**Geographic information — Well-known text for coordinate reference systems https://www.iso.org/standard/63094.html OGC 12-063r3 Well-known text for CRS https://portal.opengeospatial.org/files/?artifact_id=57255&version=1**
SKOS:	**Simple Knowledge Organization System Reference. https://www.w3.org/TR/skos-reference.**
Mod Spec:	**OGC 08-131r3 The Specification Model - A Standard for Modular specifications. https://portal.opengeospatial.org/files/?artifact_id=34762**

4 Definitions

In addition to the list below, any definition in any normative reference in Clause 3 will be acceptable. All Standard English words in either in the Oxford or Webster's dictionary or both (sometimes with spelling variations) apply.

4.1 coercion, type coercion, implicit type conversion

⟨programming⟩ automatic conversion of a value in one type to another based on the equivalence of values

Note 1 to entry: Coercion will usually be implemented as a conversion of the value of the expressed type to an expression of a related type (that shares its semantic value but not its structure). It is sufficient that there exists a conversion function taking the original value that would create an equivalent instance of the target class. If a function protocol requires a real number, then the integer (a subclass of real) value of "1" will be coerced into a real number of equivalent value, i.e. the real number format "1.0". Coercion is an automated ability of all popular programming language.

4.2 composite feature

⟨modeling⟩ feature with an extent consisting of a set of contained features

Note 1 to entry: The spatial extent of a composite feature is the spatial union of the spatial extents of all its subfeatures.

4.3 connected

⟨geometry⟩ inability to be divided into two disjoint closed subset geometries

Note 1 to entry: The usual mathematical definition of connected is "set which cannot be partitioned into two non-empty subsets so that each subset has no point in common with the set closure of the other", see [6]. Geometries are assumed to be closed and finitely represented, and hence most of the "geometric paradoxes" between connected and path connected do not apply.

4.4 container, container object

⟨programming⟩ object which contains or is associated by some form of aggregation to other objects

Note 1 to entry: If a contained object does not specify optional attributes, then it may inherit the value of those attributes from its container. If its containers are not consistent, then the contained object should contain values for all ambiguous attribute values.

4.5 controlled vocabulary
controlled taxonomy

⟨taxonomy⟩ established list of standardized terminology (names, words or phrases) with associated definitions for use to identify, describe, index or retrieve information

Note 1 to entry: The vocabulary is "controlled" in its use in the sense that only terms from the list may be used for the intended purpose. In this document, the principal vocabulary terms are the "feature", "property", "relation" and "relation roles" in the taxonomy that controls items in a conformant feature data set. In its earliest uses, legal and medical communities used a well-defined list of terms to be used; in the legal community in the descriptions of legal opinions, reports, records and laws, and in the medical community to similarly control medical records and reports of diseases, symptoms and treatments.

Note 2 to entry: Another view of the idea of controlled may also apply in the sense that in an information community using a controlled vocabulary would also have a control procedure or mechanism for its use and its evolution in the addition or modification of the terms. Hence both the vocabulary and the items built from that vocabulary are semantically controlled. For the scheme to work, all terms must be defined. The term "vocabulary" is often replaced by "thesaurus" or "taxonomy" indicating the importance of consistency of the formal, lexical and conceptual semantics in the vocabulary's use. In its simplest use, vocabulary may connote a set of words, but the use to which a "controlled vocabulary" is put, the importance of the implied meaning is self-evident. It may be that "lexicon" (vocabulary of a branch of knowledge) might be more precise which is implied in using "terminology" (terms used with a technical application in a subject of study, theory, profession, etc.) in the definition in [17].

Note 3 to entry: Vocabularies can and often include a referential index based on the use of terminology for subsequent query, locate and retrieve processes. All professions tend to have a social contract that controls the terms and symbols used in their literature, enforce by peer-review journals, and in geographic information by standards.

[Modified from [13], [14], [15] and [17]]

4.6 dynamic object

⟨programming⟩ object instance which is, as necessary, assumed to support any operation, property or relation

Note 1 to entry: Each dynamic object can be modified at run-time by the program adding or modifying an existing object. Once an object is dynamic, it becomes its own class, which can be "cloned" to create identical class instances. Metaphorically, the object can "learn" anything the program can "teach" it.

Note 2 to entry: Dynamic objects are or can be supported in several programming languages such as C# ("C-sharp"), JavaScript, Visual Basic, Python and even C. Layering a block style of primitive, such as in Figure B.1, can produce similar objects in a complied language environment. The programming language cannot constrict the inventiveness of the programmers.

4.7 feature

⟨geographic information⟩ abstraction of real-world phenomena or entity

[ISO 19101 and expanded in ISO 19109]

Note 1 to entry: Feature instances are data objects, and as such are distinguished by their object identity. The entity or phenomena that the feature represents are real-world objects. All feature instances that reference the same real-world phenomena should have the same feature identity but do not need to be stored together. Storage mechanisms are the purview of the applications. If the same feature has alternate representation, and are thus identified as distinct, this may lead to misinterpretation of the data. In this concept, the feature identity is distinct from the digital object identity which are always distinct identities if they are not stored as a single digital object.

4.8 feature association
⟨geographic information⟩ relationship that links one feature with another feature

[modified from ISO 19110 and ISO 19109]

Note 1 to entry: The original definition of feature association is "relationship that links instances of one feature type with instances of the same or a different feature type" and assumed a schema constrained feature model which is validated by feature types and object instances. An ontology-constrained feature model depends on comparisons of feature definitions with association definitions for validation. If conformance is by schema-constraints, the original definition is still valid.

4.9 feature catalog
⟨geographic information⟩ catalogue containing definitions and descriptions of the feature types, feature attributes and feature associations occurring in one or more sets of geographic data, together with any feature operations that may be applied

Note 1 to entry: This definition of feature catalogue was originally used for the feature models defined by schemata, but is general and thus sufficient to cover taxonomic or ontological definitions of feature models.

[ISO 19110]

4.10 feature class
feature type
⟨geographic information⟩ reusable model of a features with common aspects

Note 1 to entry: A class for this interpretation is "intentional" in the sense that it is defined by general characteristics that must be shared by all its members. "Extensional" classes are defined by which objects are in them, which may have no common characteristics.

4.11 feature component
primitive feature
⟨geographic information⟩ feature without any contained smaller features with a connected geometric extent defined by a single closed and connected geometry of single dimension

Note 1 to entry: Feature components are often labeled by the type of geometric primitive used to express its extent; giving point feature, line feature, area feature, solid feature (depending on the dimension of the geometric primitive being used.

4.12 feature concept
⟨geographic information⟩ concept that may be specified in detail as one or more feature types
EXAMPLE The feature concept "road" may be used to specify several different feature types, each with a different set of properties appropriate for an application. For a travel planning application, it might have a limited set of attributes such as name, route number, location and number of lanes, while for a maintenance application it might have an extensive set of attributes detailing the structure and composition of each of the layers of material for which it is composed.

Note 1 to entry: A feature concept can be thought of as an "abstract root" or "superclass" of a related set of feature types. In an ontology specification, this should be instantiated as a "is a" relationship at the definition level. Similarly, in a schema, this should be instantiated in the subclassing hierarchy.

[ISO 19126]

4.13 feature concept dictionary
⟨geographic information⟩ dictionary that contains definitions of and related descriptive information about concepts that may be specified in detail in a feature catalogue

Note 1 to entry: This is essentially the root rationale for using an "ontology" which merges the concept of "definition" and "class".

[ISO 19126]

4.14 feature identity

⟨programming⟩ unique identifier associated to a real-world phenomenon,

Note 1 to entry: This is usually a semantic construct such as a name or a real-world formal identity. This identity links the reality of the thing being described by a feature object. This is distinct from the object identity of a feature data instance which identifies the digital object for the data manager. Compare "feature" and "feature identity" with "object" and "object identity". In general, supporting the distinction between object and feature is an application decision, but it can lead to semantics issues with query results. Most query languages search for object or data instances of which a feature may associated to many, but more semantically correct statements are about features.

4.15 feature property
feature attribute

⟨geographic information⟩ potential attribute, quality, or characteristic of a feature

[modified from ISO 19101-1]

Note 1 to entry: Dynamic and static object systems vary on which feature is associated to which attribute. In a static system, the class determines the relation between an object and its properties. In a dynamic system, the connection of an attribute type to an object instance is based on the semantics and the particular instance of the real world "phenomena". In a schematic system, the schema defines this relation, in an ontology system, the constraints define the options. In a simple taxonomic system, the data collection and editing processes makes that decision based on the interpretation of the semantics of the feature and the property in the instance. This shows up even in schematic systems in instances of feature classes that do not match the template in the schema, e.g. "unnamed road" is not uncommon on maps of remote or rural areas where a GPS navigation system may issue an instruction such as "turn right onto unnamed road".

Note 2 to entry: The original definition of feature attribute used "characteristic" instead of "property". In general semantics, the two definitions are identical in interpretation, but since this document uses the more generic term "property", the definition has been adjusted to coincide with terminology both in ontology-constraints and schema-constraints.

Note 3 to entry: This definition should not be construed to indicate how a feature property is implemented. Any containment, or logical reference can be used in the various implementations of property.

Note 4 to entry: If the dataset is associated to a static schema, the "potential" of the attribute has already been decided. In a dynamic object model, a data editor would be able to add any added property to any feature if the semantics of the two do not directly collide semantically. A "cat" cannot have a "wingspan" but a "flying boat" or "flying fox" may. The difference between a static object model and a dynamic object model is the latter follows the behavior of reality, and the former follows a static view of the world in which a mistaken assumption can cause difficulties because things and outlooks are always changing over time.

> *"You could not step twice into the same river; for other waters are ever flowing on to you".*
> *— Heraclitus of Ephesus (c.535 BC - 475 BC).*

4.16 feature relation
feature association

⟨geographic information⟩ relationship that links instances of features

Note 1 to entry: In some discussions, it is implied that relations are binary relating one instance to another constrained by class. This is not a limitation. In database terminology, a relation can reference any number of entities. A relation on sets S_1, S_2, S_3, S_4, … S_n can be defined as a subset of the cross-product of the n sets based on feature classes (represented by an n-

tuples, usually containing feature identities or keys). In a dynamic system and some static system when subtyping structures can broaden the possibilities, the relation can be independent or quasi-independent of feature type. Spatial relations which are used in spatial queries are determined by the interactions of feature geometries, are oblivious to feature types, unless addition property restrictions are included in the query.

[19110], [6]

4.17 feature relation role
⟨geographic information⟩ name or semantic purpose of an offset in a feature relationship

Note 1 to entry: Roles in a relation can be implemented as references or pointers in any number of the objects in the relation instance.

4.18 feature schema
⟨geographic information⟩ collection of definitions and declarations for feature classes, properties and associations

Note 1 to entry: In a static schema, all properties and associations are associated to classes. In a dynamic schema, any property or association can be used for any object upon the action of that object. Such operations to change the structure of an object in a static schema requires require changes to code off-line.

4.19 feature subclass
feature subtype
⟨geographic information⟩ feature class or type, whose members are also members of a superclass

Note 1 to entry: This implies that "broader" or "narrower" feature classes are defined in a transitive manner. This means subclass is "transitively narrower" than its superclasses and a superclass is a "transitively broader" than its subclass.

Example: The abstract root class of "feature" contains all "features"

4.20 hierarchical taxonomy
⟨taxonomy⟩ taxonomy such that categories are arranged in a tree structure built to represent the context of a category, using parent-child "is-a" relationships where the parent is broader conceptually than the child

Note 1 to term: Some sources require that the hierarchy be single inheritance (only one superclass per category), but this is not required in general and is inconsistent with dynamic class models.

Note 2 to term: Some taxonomies also use a containment view of the terms broader and narrower to include containment hierarchy so that a "door" is an element of a building means the "door" is a narrow concept than building. In a hierarchical content, door and building are not in the "is-a" hierarchy and cannot be used to mimic object inheritance as can be done in a hierarchical "is-a" taxonomy.

4.21 identifier
⟨programming, web⟩ linguistically independent sequence of characters capable of uniquely and permanently identifying that with which it is associated

[ISO 19135-1]

4.22 key value pairs
⟨programming⟩ ordered pair of two data items; first, the key (identifier) identifying the semantics of the associated value, and second, the value, either the identified data or a reference (pointer or identity) to that data

Note 1 to entry: In this document the common expression will be formatted as: "⟨key⟩: ⟨value⟩".

4.23 name

compact and human-readable designator that is used to denote and identify a data item

Note 1 to term: Aliases may be defined for use within the scope of an information system, but data sets used in information exchange use standard specified name for all data items.

4.24 object

⟨programming⟩ identifiable component of a software system or design, now more commonly applied to a component that is in some sense self-contained, having an identifiable representation, and sets of operations and relations

Note 1 to entry: An object is an instance of a class in object-oriented programming, but the term may be used generically in computer programming to indicate an instance of any software component. This standard is concerned with the "objects" that contain information about real world phenomena or features. A feature instance is an object, and as such will be supplied with an object identity by the programming system (often related to physical addresses in a program's image.

[modified from A Dictionary of Computer Science,[3]]

4.25 object identity

⟨programming⟩ identifier associated to an instance of an digital object to distinguish it from all other objects

Note 1 to entry: The object identity of a feature instance is not the same as the feature identity. The object identity names a block of data, the feature identity names the real-world phenomena that the object is representing. The object identity is unique to a block of data. The feature identity should be unique to a block of reality. The two concepts are often confused and the moniker "feature identity" is often used to describe the object identity of a feature class instance.

4.26 ontology

⟨semantics⟩ formal naming and definition of types, properties and relationships of entities that exist in a domain of discourse.

Note 1 to entry: The definitions in a feature ontology include features and feature properties (attributes, associations and roles). An ontology defines a set of representational primitives with which to model a domain of knowledge or discourse and can be viewed as a level of abstraction of data models, analogous to hierarchical and relational models. See [14].

4.27 ontological schema

⟨programming⟩ taxonomic schema with additional ontological constraints

Note 1 to entry: Any object schema is an ontological schema that included all appropriate object-oriented constraints for the appropriate OOPL targeted.

4.28 property

⟨modeling⟩ named descriptive value associated to or contained in an entity (object or feature type)

Note 1 to entry: A property's name normally reflects the semantics of the property's value, which may be a value or the identity of an associated entity. Properties linking entity to one another are relations. Properties associating descriptive data values to an entity are attributes.

4.29 reach

⟨geography, hydrography⟩ feature component represented as a curve between two intersections of features of the same type.

Note 1 to entry: Reach is commonly used in hydrography to describe sections of waterways that intersect others at its end points. In braided streams, each "parallel" stream is a single reach. In a road system that is used for navigation, most *navigation actions accrue at the end of a reach where decisions on directions can be made, as along a reach a change of direction can be difficult.*

4.30 relation
association (of dimension n)
⟨mathematics, database⟩ set of ordered sets (call "n-tuples") of a fixed number "n" of entities that have a common interrelation status

Note 1 to entry: Geometric relations may be defined. Other relations that cannot be calculated from raw data may have to be supported by lists of n-tuples such as exist in a relational database. As discussed in this text, each n-tuple can be considered as "a feature-instance" of the "relation class".

4.31 semantic class
⟨ontology, taxonomy⟩ set of entities consistent with a taxonomy definition consisting of a necessary and sufficient set of criteria for an entity being an element of the class.

Note 1 to entry: The set of criteria of a semantic class will be necessary but not sufficient for any of its subclasses. A subclass is necessarily more restrictive than any of its superclasses as they inherit their superclass's criteria but must distinguish themselves as more specific and non-redundant.

4.32 semantic class hierarchy
⟨ontology, taxonomy⟩ hierarchy of semantic classes based on subseting

Note 1 to item Any logical subset (B) of a semantic class(A), based on the a more specific definition is a logical subclass of that class (A).

4.33 sequence
⟨mathematics⟩ ordered set of items in which the order has semantic importance

4.34 schema
⟨modeling⟩ representation of a plan or theory in the form of an outline or model.

*Note 1 to item A schema can be expressed as a plan or theory in the form of an outline or model. The realization can be based on definition of entities in the model (**ontology**) or as structures representing those entities (**schema**), or as both.*

4.35 set
⟨mathematics⟩ unordered collection of related items with no repetition in which the order is of no importance nor significance

4.36 taxon
class
category
type
⟨taxonomy⟩ classifying entity in a taxonomy consisting of a name, a definition and relations to other such entities in the taxonomy

*Note 1 to term: The terms are interchangeable, in the sense that a "taxon" in a taxonomy will become a "class" in an object schema, or a type in a "query language", as appropriate. The most common type of relationship is a hierarchical one listing classes as broader (superclass) or narrower (subclass). See **hierarchical taxonomy**.*

4.37 taxonomy
⟨taxonomy⟩ system for classifying entities by logical definitions (criteria) and a hierarchy subtype relationship base on logical subsets of instances in a class

Note 1 to term: Taxonomies are often used as indexes to support and optimize searches based either on entity types, property types or values. In most of the instances in this document, a taxonomy will tend towards being a controlled vocabulary, basically since the more controlled across users and user communities the better the affect toward interoperability of datasets.

4.38 theme
layer
thematic layer
⟨geographic information⟩ abstract root classification for a feature hierarchy included feature which contribute to a single geographic subject

Note 1 to term: Themes might include both geological and sociological theme such as geology, hydrological, political, cadaster, political and sociological themes. Theme will often parallel a type of geographical related area of study. See [19]

5 Symbols (and abbreviated terms)

The following abbreviations, acronyms, symbols and names are important key to understanding the normative or informational content to follow. They are in general known publicly, and readers of this document may need to understand their meaning.

« »	limits of a UML stereotype or keyword, either in text or a diagram; the symbols are "guillemets"
API	Application Programming Interface
ATS	Abstract Test Suite
C++, C#	Programming languages based on C with object-oriented extensions, "C plus plus", "C sharp"
CAD/CAM	Computer Aided Design, Drafting or Manufacture, Mapping (usually geometry-based design systems)
CRS	Coordinate Reference System, usually a reference to an instance defined in ISO 19111
GIS	Geographic Information System/Science
JavaScript	prototype-based scripting language with objects, dynamic typing and first-class functions
JSON	JavaScript Object Notation, alpha-numeric format for JavaScript objects
LISP	Programming language based on LISt Processing. The standard is Common LISP.
MBR	Minimum Bounding Region (sometimes Rectangle)
OOPL	Object Oriented Programming Languages
RSID	Reference System Identifier
SQL 3	Common name for SQL 99 during its development
SQL 99	SQL language adopted in 1999, which includes object-oriented data types
SQL/MM	SQL Multi Media extensions for SQL, see ISO/IEC 13249-3
URI	Uniform Resource Identifier (IETF RFC 3986)
UML	Unified Modeling Language (from OMG, Object Management Group)
WKB	Well-Known Binary (data represented as standardized structured binary)
WKT	Well-Known Text (data represented as standardized structured text)
$\varnothing$	the empty set, i.e. containing no elements

6 Feature models

6.1 Introduction to feature taxonomy

A feature is a representation of a real-world entity/object. This idea can trace its definition back to the 1960’s in the definition of Simula (a simulation language), the first object-oriented programming language. In Simula, "digital objects" represented "real-world objects"[4]. Since the 1980’s, spatial or geographic information systems have paralleled this same idea, at about the same time as Simula and OOPL were first being used. Much of the early standardization of GIS followed the same object-oriented paradigm and defined feature classes as object classes (often in UML or it predecessors, such as OMT, object modeling technique).

The problem with digital objects representing real world objects is that, in most schematic modeling, all objects of the same class have essentially the same representation. The "real-world" is not as neat and clean. Any two real-world objects are distinct, often at a fundamental level and, for that reason, each may be a unique instance of their own "class", e.g. must be described in different manners with different properties, not just different values for the same property. While feature schemata are the core of many systems and their associated standards, there is another approach that supports schema but allows more flexibility as needed. This approach steps back from structural formalism and uses semantic **taxonomy** and **ontology**, to define feature types and prescribes descriptive properties of features as related parts of a **taxonomy**. In contrast with a schema, where a class is defined by its properties, a taxonomy-based system allows any feature to be associated with any property where the feature and property definitions are consistent with one another according to a given taxonomy.

A geographic feature taxonomy, ontology or schema will define a namespace that contains definitions of the keys that will be used to identify the semantics of key-value pairs associated to a feature:

- a set of feature types (a type name and a definition)
    - including a "is a type of" relation where a definition may reference a more general type,
        - e.g. "weir <u>is a type of</u> dam such that"…[5]
- a set of feature properties:
    - a set of feature attributes (an attribute name, a definition and a data structure or object to represent its value), and
    - a set of feature relations that associate features to one another (a name, a definition, a list of rolenames and pointer or reference type to identify the feature in each role).

A feature instance contains:

- a unique local identity, (possibly part of a universal identity scheme such as URN)
- one or more feature type names,
- a list of properties with attribute names and associated values,
- a list of feature relationship names each with a list of roles and the target feature identities for those roles.

A geographic feature schema is an ontology that further defines which properties and relations that can be (optional) or that must be (mandatory) associated to instances of which feature types. All properties not specifically mentioned in this list cannot be part of the structure of a feature of that type. Table 1 below shows how a set of features could be represented and gives some potential examples of a taxonomy for feature type definition that are used in the example.

Classical schema formalism arises from application requirements. For example, a radar-based application would concentrate on the attributes of a feature that affect its radar reflectivity, but a visual-based application would concentrate on the visual

[4] The GIS definition used "real-world phenomena" and Simula used "real world objects". This is an extension needed for GIS since many "features" displayed in data sets are "conceptual" like political boundaries, buffer zones, voting districts and others whose existing in the real world has real world implications, but is not always a real-world physical but a conceptual "object".

[5] The USSD (US Society of Dams, https://www.ussdams.org/) recognizes 12 basic types of dams; a weir is an "overflow dam".

attributes of a feature such as color. Radar is not good at detecting color (wrong part of the spectrum), but human eyes are. A radar system would not care what color the Golden Gate Bridge is painted, but a system for visual identification would know "international orange" is a good visual recognition clue; in fact, the "color is used in the aerospace industry to set things apart from their surroundings". See [11]. The use of "international orange" is of interest to a "visual-based" application, but not so much to a "radar reflectivity" application. The "interoperability" advantages of a single database for multiple application are obvious, especially when a single operation uses detection approaching the "DC to Daylight"[6] spectrum range.

As different applications require integrated formats and information, it is easier to integrate taxonomies than schemata. In fact, integration of schemata often requires that a taxonomy be constructed to map the semantics of attributes and entity names used in the labels of the various schema. These taxonomies can then be compared to determine how the various application schemata can be integrated into a generic "application-independent" format.

Different applications will often have different schemata for the same types of objects, and different data collection rules driven by these differences. If diverse applications are required for a single operation or project, the interoperability advantages inherent in a single datastore, and a single set of "facts" is obvious and usually essential.

Once the decision to create this "common database" the use of a "taxonomy merge" is likely unavoidable. Some features will be only one source dataset, but the majority of the "real-world entities" will have multiple instance features to merge. The first step is the merging of the taxonomies of the disparate datasets, then followed by identifying which of those entities are associated to multiple feature instances, which requires the merging of different object instances from different applications. Easy to do in a taxonomy, difficult to do in schema. Over time, the holes may be filled, and the merged dataset will eventually be an application-independent store that supports all participants in the collections of the data; and the synergistic interoperability of the applications.

The driving purpose of OGC is and has always been making applications interoperable. A taxonomy paradigm for datastores seems to be a rational route to creating synergistic applications that have a single view of the world because they can depend on a merged, complete and flexible taxonomic datastore.

The values of attribute properties will have to be represented in a format consistent with the applications supported. In the textual examples in this standard those types may be in text, such as a JSON, YAML or WKT.

We can represent Wheeler Dam as in Table 1; the "URL #" expressions would be system generated entity ids, which can take the real-world name "Wheeler Dam" into consideration.

In a taxonomy-controlled feature data set, the features and their properties are defined. The association of feature instances to properties only require that the definitions be consistent with data instances. Feature instances would be associated to attributes and association roles defined in the taxonomy as needed. Some programming languages (e.g. JavaScript and LISP) can handle this directly, and non-dynamic object-oriented languages that can be expressed in UML can be programmed to do so also (see Annex B.1). Taxonomy instances for feature types would specify: a feature type name and definition, feature type inheritance, and mandatory properties.

In a schema-controlled feature data set, the properties are defined by the controlling schema. For example, the "logical default" for a dam does not have the properties of a bridge, but in the real world several dams have been designed or modified to include a bridge in its superstructure (such as "Hoover Dam" on the Colorado, and "Wheeler Dam" on the Tennessee). In a parallel case, some but not all dams have hydroelectric properties or flood control mechanism, again optional descriptions are needed to create a complete description dependent on the applications using the data. In a schema-controlled data set, the class for "dam" would have to be able to coexist with "bridge" when it is needed. Considering the complexity of reality and difficulty of an *a priori* schema that could adapt to all possibilities, the ontological approach is more likely to be usable without continuous redefinition. A schema would require potentially dynamic multiple inheritance or instantiation in a dynamic object-oriented language.

Each of the "dams" serves as part of a water control system (possibly the main purpose of a "dam"), a link in the road system across a waterway (the main purpose of a "bridge") and a source of hydroelectrical power (the main purpose of a "power plant"). This feature-type combination in a single feature is not universal. Some bridges are not associated to neither flood control nor hydroelectric generation. In New York state, the Robert Moses Niagara Hydroelectric Power Station diverts water from above the Niagara Falls and routes it below the falls while generating power. This hydroelectric power plant is not

[6] Euphonious and alliterative jargon for the "entire useful electromagnetic spectrum" ordered by frequency, direct current (0Hz) to light (400-770THz) and now maybe up to gamma rays (30EHz).

associated to a dam and occupies shoreline only on the New York side of the Niagara River. It also has a Canadian near-twin in the Sir Adam Beck Hydroelectric Power Stations in Ontario, Canada. To minimize erosion at the fall's face, both Niagara facilities are used to affect the amount of water going over the falls at Niagara. The Wheeler Dam on the Tennessee is also used for mosquito control; Wheeler Lake's water level is periodically adjusted up or down to disrupt the mosquito life-cycle either drying out or washing out mosquito larvae. This extra-model information is important to only some applications, but to those applications they are often essential. In the case of the TVA dams mentioned here, mosquito control is as important as the hydroelectric processes. The Tennessee Valley in the last half of the 19th century was affected by yellow fever sometimes yearly, spread by mosquitos.

Table 1 – Example feature entities in a key-value format

```
Object-1:
      {
      Feature:     "URL 1"                        ‼ [some URL]
      FeatureType: "Dam"
      FeatureType: "Bridge"
      Name:        "Wheeler Dam"
      On:          "URL 3"                        ‼ Tennessee River
      CenterPoint  34°48'14.99" N, -87°22'32.99" W
      Geometry:    Location                       ‼ [LineString expression]
      }
Object-2:
      {
      Feature:     "URL 2"                        ‼ [some URL]
      FeatureType: "Lock"
      On:          "URL 3"                        ‼ Tennessee River
      In:          "URL 1"                        ‼ Wheeler Dam
      }
Object-3:
      {
      Feature:     "URL 3"                        ‼ [some URL]
      FeatureType: River
      Name:        Tennessee River
      }
```

The "use of taxonomy" versus the "use of schema" is a shift of emphasis from syntax (the form or structure of the information) to semantics (the meaning of the information). This transition should not be a surprise since from the beginning the great barrier to the flow of information has been format i.e. schema, and so the biggest boon to interoperability is to step over that barrier and shift our application in the same direction that the web is moving, towards semantic solutions. This makes the data easier to understand and thus easier to move, and finally easier to interoperate between functionality.

6.2 Requirements Class: Feature Taxonomy and Data: https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/

6.2.1 Feature types and thematic layering

A feature taxonomy will always contain at least two collections of definitions (controlled taxonomies, or controlled vocabularies, see [2], [14], and [15]) that separately define and describe "real-world" features and properties. See Figure 1. A feature data set consistent with this feature taxonomy.

Req1 **A taxonomy shall be associated to a unique namespace. Any feature data set consistent with this taxonomy shall reference that namespace.**
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req1

Req2 **A taxonomy shall contain a complete semantically valid definition for each feature type, for each property type, and for each association and role associated to the taxonomy by its namespace.**
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req2

Req3 The feature type component of a taxonomy shall include a "is a" type-hierarchy for features (e.g. a hierarchical taxonomy, engendering a multiple inheritance of feature type).
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req3

Note: A hierarchical taxonomy allows the category/class structure to be inheritable like an object classes inheritance structure. If a taxonomy is used in a class model schema, the taxonomy hierarchy would be identical to the class hierarchy. In a class hierarchy, a superclass is more general than its subclasses, and subclass is more specific than its superclasses.

Req4 In any search or query, a reference to a feature-type shall include all members of the corresponding feature type and all its subtypes below it in the type-hierarchy; i.e. all such instances of those taxons which match the criteria stated in the query via inheritance.
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req4

Req5 The taxonomy entry for a feature type shall contain a name, a definition, a list of any direct superclasses and any required properties.
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req5

Note: Any other property type consistent with the feature definition would be optional in this case. In an ontology extension, constraints could be used to forbid specific properties.

Per5 The taxonomy entry for a feature type may contain common optional properties for entities of the feature type. Any feature instance may reference any number of semantically consistent feature types.
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Per5

Per6 Any taxonomy item may be associated to searching for instances of that type of item.
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Per6

Per7 Any property instance may be used to describe any feature instance if the semantic of the feature and property is logically consistent.
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Per7

Note: Ontology extensions would be able to pre-specify consistent property-feature pairs.

Req6 The taxonomy entry for a property type shall contain a name, a definition, a default data type able to represent the value of the property.
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req6

Per8 In an instance of a property, the value may be any datatype coercible into the default datatype.
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Per8

Note: Type coercion is defined in 4.1, page 12.

Note: Linkages could be accomplished by URI associations, but other pointer types may be more appropriate for the representation format used for the taxonomy.

Req7 Each data entity consistent with a taxonomy shall be a dynamic object, contain a local identity associated to its name space and type names used to link it to semantically appropriate feature, property or association names and roles.
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req7

Table 2 – Example feature-type taxonomy entries in a key-value format

```
Taxonomy Definitions:
       {
       FeatureType: Feature
       Definition:  abstraction of real-world phenomena or entity
       Property:    location:    Geometry
       Property:    coordinateSystem:  CRS
       }
       {
       Relation:    Subset
       SourceRole   superset|covers:x
       TargetRole   subset|covered|in|on:y
       Type         geometric relation (derived)
       Definition:  y.geometry is subset of x.geometry
       }
       {
       FeatureType: Dam
       Definition:  barrier to stop or control the flow of water
       Property:    location: Geometry
       ProperyRole: on: waterway
       Property:    name: Text
       ProperyRole: Contains:    Sluice[0..*]
       }
       {
       FeatureType: Weir
       Defintion:   barrier built across a waterway to make the water
                    approaching it deeper, usually by making the water
                    flow over a part of the barrier at a controlled height.
       IsTypeOf:    Dam
       }
       {
       FeatureType: Lock
       Definition:  section of a waterway, enclosed by gates, in which the
                    amount of water can be modified to raise or lower a vessel
       IsTypeOf:    Sluice
       }
       {
       FeatureType: Sluice
       Definition:  sliding gate or other device for controlling the flow of water
       Property:    on:Waterway
       Property:    in:Dam(0,1)
       }
       {
       FeatureType: Watercourse
       Defintion:   natural or artificial water channel
       }
       {
       FeatureType: Waterway
       Defintion:   Watercourse usable for travel
       Property:    navigability:NavigabilityParameters
       IsTypeOf:    Watercourse
       }
       {
       FeatureType: Canal
       Defintion:   artificial Waterway
       IsTypeOf:    Waterway
       }
```

Per9 **If an implementation does not support dynamic objects, then any number of static objects may be associated to the same feature identity.**
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/0

Req8 **A feature taxonomy shall contain a non-empty set of definitions of real-world phenomena "features" that may be represented as features in datasets consistent with this taxonomy.**
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req8

Req9 **Any feature entity shall be associated to at least one feature definition in the feature taxonomy and any number of properties or associations consistent with its definition.**
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req9

Per10 **Theme or thematic layer definitions may be root classifiers for feature type definitions.**
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Per10

See definition 4.38

Per11 **https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Per5**

Per12 **Root classes in a feature taxonomy may be themes or thematic layers.**
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Per12

Per13 **Any feature type definition may contain any applicable subtype of ("is-a") relationship (subclass) to other feature definitions.**
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Per13

Req10 **Every feature instance shall have a feature identifier that represents the real-world thing being described by the feature.**
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req10

Req11 **Any feature instance shall be consistent with the corresponding feature type and property definitions in the associated taxonomy with which it is associated.**
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req11

Rec3 **Any feature instance may contain properties or be involved in associations consistent with its definition.**
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Rec3

Rec4 **Any feature instances that reference the same real-world phenomena should have the same feature identifier.**
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Rec4

Req12 **A feature access process shall be able to collect all feature instances that refer to the same real-world entity.**
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req12

Req13 **A feature access process shall be able to locate feature instances based on their feature type, the inclusion of properties, the value of those properties and spatial conditions or any valid Boolean combination of the same.**
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req13

6.2.2 Feature properties

A feature model will always contain a non-empty collection of definitions (taxonomy) that either define or describe the *potential* properties of "real-world" features. The use of any property type to describe any feature type would be the purview of the collection mechanism and what situational peculiarities may require or preclude the use of the properties in the situation.

Req14 **A feature taxonomy will contain a non-empty set of definitions of properties of real-world phenomena that may be represented as properties associated to features in data sets consistent with this ontology.**
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req14

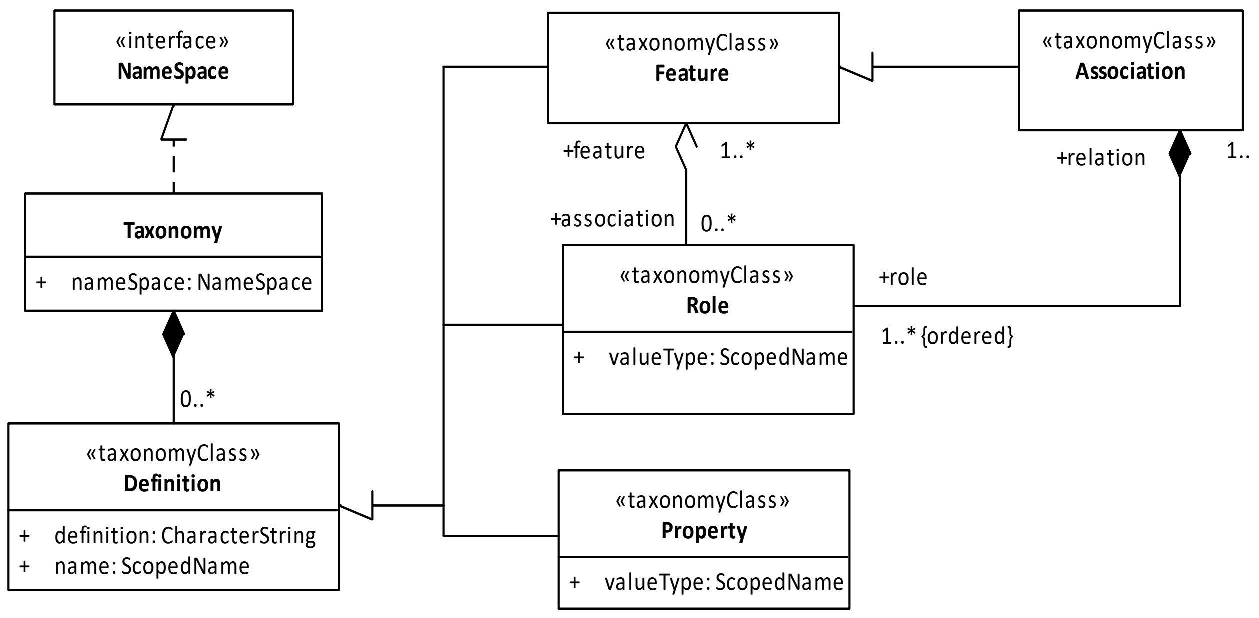

Figure 1 — Basic Feature Taxonomy Structure

Req15 Properties and association roles shall have values that are datatypes, including references to other features by either object or feature identities.
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req15

Req16 A feature property definition shall include a unique property name and datatype.
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req16

Req17 Each instance of feature shall have a unique object identity.
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req17

Note: The object identity is identity the digital representation of the feature.

Req18 Each instance of feature shall have a feature identity.
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req18

Note: The feature identity is a name or identity the real-world entity that the feature references.

Per14 Each real-world phenomenon may be referenced by multiple feature instances.
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Per14

Req19 Any two features labeled with the same feature identity shall reference the same real-world phenomenon.
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req19

Rec5 Any two features instances that reference the same real-world phenomenon should where feasible have the same feature identity.
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Rec5

This last recommendation may be difficult, especially across data sets. Object identities are easier to keep unique, if a hierarchical identity namespace system such as URN is used. Global grid systems may be useful in defining feature identity system based on location, but dynamic datums, which recognize a dynamic location issue, may be an encumbrance. Unfortunately, the planet is not as stable as we once thought, and we now know things move even without our permission nor intervention.

**Req20 Each feature shall have at least one feature type.
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req20**

It will of course be possible to have a feature type called "Unknown" but knowing that we do not know is knowledge worth having (akin to Socratic Ignorance, "I do not think I know what I do not know,").

**Per15 Each feature instance may have a property "Type" which contains all feature types which this feature instantiates.
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Per15**

**Req21 Each feature will be considered a member of each feature type in its type references and any of their supertypes.
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req21**

**Req22 Each feature instance shall have complete ownership of all its property instances values.
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req22**

**Rec6 If a feature is in a containment hierarchy, each property should be present in the largest element consistent with the semantics.
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Rec6**

Example: Rivers are often broken into reaches (segments of river with no junctions with other waterway excepts at it endpoints). The name of the river, which applies to all its reaches, should normally be associated with the largest segment who shares that name.

**Per16 A feature property's definition may contain or be augmented by restrictions on which feature types it describes.
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Per16**

6.2.3 Feature associations

A feature taxonomy or ontology will contain a collection of definitions that either define or describe the potential associations between "real-world" features.

**Req23 A feature property taxonomy shall contain a set of definitions of associations of real-world phenomena that may be used to express relationships between feature instances in data sets consistent with the ontology.
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req23**

**Req24 Binary association shall be implemented as symmetric roles linking each feature in the relation to the other.
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req24**

**Per17 A "n-ary" relation may be implemented as a "relation feature" with n binary relations named for the roles in the original relation.
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Per17**

6.2.4 Feature Component

A feature component is feature whose geometry is a geometric primitive. In 2D or 2½ data, the primitives would be points, curves and areas (surfaces). In 3D, solids would be included.

**Rec7 A feature containing multiple types of feature primitive should be decomposed into feature components each containing a single geometric primitive. The original feature should then be represented as an aggregation of its components.
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Rec7**

Example: Roads are built in segments, usually intersection to intersection, but highways can then be described as a weak aggregation of road segments, some of which carry multiple "route marker" enumerating the shared highway numbers which only change at segment boundaries.

6.2.5 Composite feature and their hierarchies

A composite feature is an aggregation of smaller, often simpler, features.

Rec8 A feature hierarchy should consist of shared aggregations of composite features and feature components, terminating at the bottom as feature components.
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Rec8

6.3 Requirements Class: Ontology: https://opengeospatial.org/as/FG/P1/FM/1.0/ontology/

An ontology includes a taxonomy for feature and property. It may add restrictions, for example, limiting which properties are associated to which features. Some describe taxonomy-controlled data as "unstructured" and describe schema-controlled data as "structured". Having labeled the extremes, they declare the choice as binary, missing the very broad middle ground, often called "semi-structured".

Req25 A feature ontology shall be fully compliant as a feature taxonomy.
https://opengeospatial.org/as/FG/P1/FM/1.0/ontology/Req25

Req26 A feature associated to an ontology shall be fully compliant with the constraints in the associated feature ontology.
https://opengeospatial.org/as/FG/P1/FM/1.0/ontology/Req26

Per18 An ontology may add constraints on feature types and properties.
https://opengeospatial.org/as/FG/P1/FM/1.0/ontology/Per18

6.4 Requirements Class: Feature schema: https://opengeospatial.org/as/FG/P1/FM/1.0/schema/

A feature schema is a feature ontology that has a more complete set of constraints consistent with a classical object model.

Req27 A feature schema shall be fully compliant with a feature ontology.
https://opengeospatial.org/as/FG/P1/FM/1.0/schema/Req27

Req28 A feature schema's ontology shall contain a complete object model for all associations, association roles, features, properties and data types required by the definitions in the ontology.
https://opengeospatial.org/as/FG/P1/FM/1.0/schema/Req28

Annex A Conformance Classes

A.1 Test Suite for Taxonomy https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/

An implementation of the Conformance class Taxonomy shall contain valid definitions for feature and property types, meeting the requirements in Clause **6.2**. Each clause below shall associate the numbered requirements to a generic method of verification. Each test is associated to a single requirement defined in the requirements class.

A.1.1 A taxonomy shall be associated to a unique namespace. Any feature data set consistent with this taxonomy shall reference that namespace. https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req1/test

Test to identify the namespace of the taxonomy.

A.1.2 A taxonomy shall contain a complete semantically valid definition for each feature type, for each property type, and for each association and role associated to the taxonomy by its namespace. https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req2/test

Test to assure definitions are complete and non-ambiguous (without contradictory descriptions). Property types will also contain a feature type, an object or a data type to represent the value of the defined property. Feature type valued properties are association roles.

A.1.3 The feature type component of a taxonomy shall include a "is a" type-hierarchy for features (e.g. a hierarchical taxonomy, engendering a multiple inheritance of feature type). https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req3/test

Test to verify that the feature type and property hierarchical taxonomies support a “is-a" or "is a subtype of" so that object inheritance is mimicked. All feature types should be transitively a subtype of the abstract root "feature".

A.1.4 In any search or query, a reference to a feature-type shall include all members of the corresponding feature type and all its subtypes below it in the type-hierarchy; i.e. all such instances of those taxons which match the criteria stated in the query via inheritance. https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req4/test

Test to verify that results from a query statement include all viable subtypes of the feature types named.

A.1.5 The taxonomy entry for a feature type shall contain a name, a definition, a list of any direct superclasses and any required properties. https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req5/test

Test to verify that each feature type definition is complete, including superclass and required property list. There will always be supertypes or be directly inherited from the abstract root "feature".

A.1.6 The taxonomy entry for a property type shall contain a name, a definition, a default data type able to represent the value of the property. https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req6/test

Test to verify that each property type definition is complete.

A.1.7 Each data entity consistent with a taxonomy shall be a dynamic object, contain a local identity associated to its name space and type names used to link it to semantically appropriate feature, property or association names and roles.
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req7/test

Test to verify that each object instance can modify its properties and feature type list within the constraints of the taxonomy.

A.1.8 A feature taxonomy shall contain a non-empty set of definitions of real-world phenomena "features" that may be represented as features in datasets consistent with this taxonomy.
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req8/test

Test to verify that the manipulation of features does not cause any constraints of the ontology to be violated.

A.1.9 Any feature entity shall be associated to at least one feature definition in the feature taxonomy and any number of properties or associations consistent with its definition.
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req9/test

Test to verify that all feature instances are associated to at least one feature definition in the feature taxonomy.

A.1.10 Every feature instance shall have a feature identifier that represents the real-world thing being described by the feature.
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req10/test

Test to verify that all feature instances have a feature identifier, which is consistent with all other feature instances associated to the same real-world feature.

A.1.11 Any feature instance shall be consistent with the corresponding feature type and property definitions in the associated taxonomy with which it is associated.
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req11/test

Test to verify that the definition of each feature type is semantically consistent with the feature instances, its properties and associations.

A.1.12 A feature access process shall be able to collect all feature instances that refer to the same real-world entity.
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req12/test

Test to see that all feature instances of the same real-world entity have the same feature identity.

A.1.13 A feature access process shall be able to locate feature instances based on their feature type, the inclusion of properties, the value of those properties and spatial conditions or any valid Boolean combination of the same.
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req13/test

Test to verify that all feature instances of the same feature type are locatable by type, included properties and the values of properties. The usual implementation is indexed feature types and property values.

A.1.14 A feature taxonomy will contain a non-empty set of definitions of properties of real-world phenomena that may be represented as properties associated to features in data sets consistent with this ontology. https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req14/test

Test to verify that properties and association roles are defined and can be associated to features. Test to verify association roles can link to features using either the feature or object identity value, or equivalent functionality.

A.1.15 Properties and association roles shall have values that are datatypes, including references to other features by either object or feature identities. https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req15/test

Test to make sure that all property definitions include a datatype to represent its value. This may include pointers to other features, e.g. the feature identity of a targeted feature.

A.1.16 A feature property definition shall include a unique property name and datatype. https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req16/test

Test by inspection that each property has a unique name, and an appropriate value consistent with its taxon definition.

A.1.17 Each instance of feature shall have a unique object identity. https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req17/test

Test that the feature identity of each feature is unique and can be used as a reference to implement property roles as pointers.

A.1.18 Each instance of feature shall have a feature identity. https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req18/test

Test that each feature has a feature identity.

A.1.19 Any two features labeled with the same feature identity shall reference the same real-world phenomenon. https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req19/test

Test that equal feature identities always reference the same real-world entity. This does not mean that there is only one object (identity) for a feature.

A.1.20 Each feature shall have at least one feature type. https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req20/test

Test to ensure that feature type is mandatory for each feature instance.

A.1.21 Each feature will be considered a member of each feature type in its type references and any of their supertypes. https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req21/test

Test that any retrieval of feature by feature type include all objects by that feature type and any of its subtypes as defined by the taxonomy hierarchy.

A.1.22 Each feature instance shall have complete ownership of all its property instances values.
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req22/test

Test to ensure that each property instance is associated to a single feature object instance.

A.1.23 A feature property taxonomy shall contain a set of definitions of associations of real-world phenomena that may be used to express relationships between feature instances in data sets consistent with the ontology.
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req23/test

Test to ensure that feature associations are defined by sets of association roles, e.g. by properties whose values are references to features, either by feature identities by object identities.

A.1.24 Binary association shall be implemented as symmetric roles linking each feature in the relation to the other.
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req24/test

Test to ensure that property entities are not shared to the extent that no property value of any feature can be modified except by navigation through the owning feature instance.

A.2 Test Suite for Ontology https://opengeospatial.org/as/FG/P1/FM/1.0/ontology/

An implementation the Conformance class Ontology shall contain ontology valid definitions and constraints meeting the requirements in Clause **6.3**.

A.2.1 A feature ontology shall be fully compliant as a feature taxonomy.
https://opengeospatial.org/as/FG/P1/FM/1.0/ontology/Req25/test

Test to ensure that the ontology is associated to a taxonomy consistent with the test suite for taxonomy

A.2.2 A feature associated to an ontology shall be fully compliant with the constraints in the associated feature ontology.
https://opengeospatial.org/as/FG/P1/FM/1.0/ontology/Req26/test

Test to ensure that constraints in the ontology can be tested and enforced in the feature instances. This included testing that the constraints in the ontology are not contradictory and thus preventing the creation of valid feature instances.

A.3 Test Suite for Schema https://opengeospatial.org/as/FG/P1/FM/1.0/schema/

An implementation the Conformance class Schema shall contain schematic valid definitions and constraints meeting the requirements in Clause **6.4**

A.3.1 A feature schema shall be fully compliant with a feature ontology.
https://opengeospatial.org/as/FG/P1/FM/1.0/schema/Req27/test

Test to ensure that the schema is associated to an ontology consistent with the test suite for ontology.

Note: Most schema languages (e.g. UML) are easily converted to ontology languages (e.g. OWL). Since most schema do not require definitions, a taxonomy may have to be created, unless it can be covered by ***Per1.***

A.3.2 A feature schema’s ontology shall contain a complete object model for all associations, association roles, features, properties and data types required by the definitions in the ontology. https://opengeospatial.org/as/FG/P1/FM/1.0/schema/Req28/test

Test to ensure there is a complete and valid object model.

Annex B Example implementation

B.1 Object implementation of taxonomy-controlled data

The discussion above implies that the most natural object language to use to implement taxonomic controlled feature data would have to have a dynamic schema/object model. Since most compiled languages (C++, C#) do not work quite that way, this design model assumes a non-dynamic schema by using inheritance and associations.

The classical object implementation of schema-controlled data uses a separate class for each feature type, with embedded attributes for simple properties and associations and roles. The alternative model in Figure B.1 breaks the feature classes into separate components allowing a statically structured language to be dynamic.

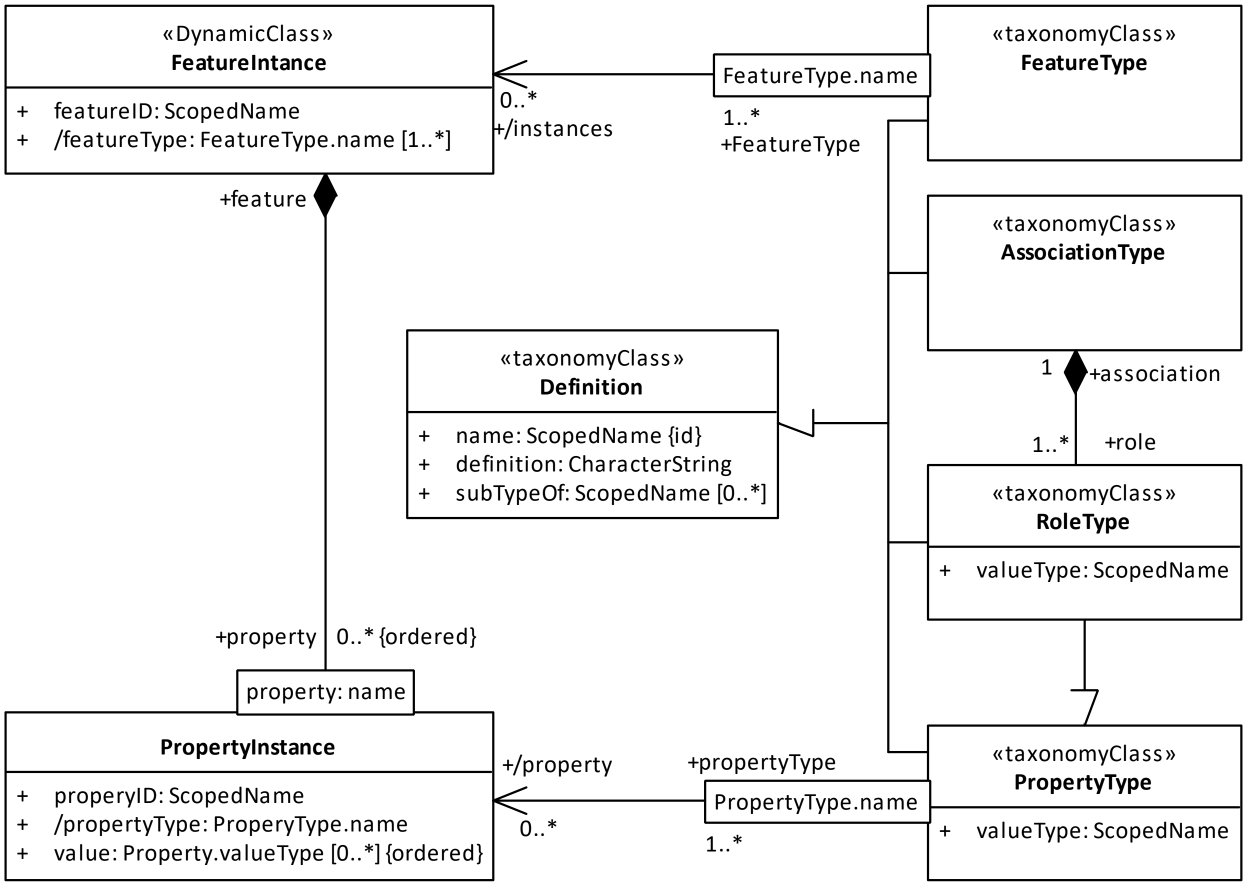

Figure B.1 – A Taxonomy-based Instantiation in Classical OOPL

B.2 Relational implementation of taxonomy-controlled data

In a strong feature schema, where all feature classes with their properties and associations are fully defined, as in the original "Simple Features for SQL" model, can used a single table for each feature class. Extending SQL for geographic information only requires the definition of a SQL-datatype for geometries, see ISO/IEC 13249-3.

If we loosen schema constraints, properties will have to be split off into separate tables to be shared by different feature types through foreign key links to these property tables.

To define the idea, the table design below assumes the other extreme where each property and association role can be associated to any feature type, i.e. where only the feature and property taxonomies are used, and no additional constraints are given. Just to be consistent, the taxonomy in the example is presented as a set of relation tables.

Table B.1 — Taxonomy Instantiation in a Classical RDBMS

```
Feature Taxonomy
featureType: ScopedName {id}              ‼ primary key
featureDefinition: CharacterString        ‼ text definition

Feature Inheritance
superType: ScopedName                     ‼ foreign key to Feature Taxonomy
subType: ScopedName                       ‼ foreign key to Feature Taxonomy

Association Taxonomy
associationType: ScopedName {id}          ‼ primary key
associationDefinition: CharacterString    ‼ text definition

Association Inheritance
superType: ScopedName                     ‼ foreign key to Association Taxonomy
subType: ScopedName                       ‼ foreign key to Association Taxonomy

Role Taxonomy
roleType: ScopedName                      ‼ primary key
roleDefinition: CharacterString           ‼ text definition
association: ScopedName                   ‼ foreign key to Association Taxonomy

Property Taxonomy
propertyType: ScopedName                  ‼ primary key
propertyDefinition: CharacterString       ‼ text definition
propertyValueType: ScopedName             ‼ type of property value (datatype)

Property Inheritance
superType: ScopedName                     ‼ foreign key to Property Taxonomy
subType: ScopedName                       ‼ foreign key to Property Taxonomy
```

Table B.2 — Taxonomy-base RDBMS feature implementation

```
Feature Table
featureType: ScopedName                   ‼ foreign key to feature taxonomy
featureID: ScopedName {id}                ‼ primary key
required property: propertyType

Property Table for propertyID {one for each property in property taxonomy}
featureID: ScopedName                     ‼ foreign key to property taxonomy
value: propertyID.valueType               ‼ type derived from property taxonomy

Association Table, columns
      ‼one for each role in the Role Taxonomy
      ‼linked to the association}
      ‼For each role, roleID where association is the same associationType.
featureID: ScopedName                     ‼ foreign key to feature table
…
featureID: ScopedName                     ‼ foreign key to feature table
```

Annex C Bibliography

Additional information and definitions were taken from the following sources as indicated, with some edits for clarity or brevity in the special context of this International Standard. Normative References are in Clause 2.2.

[1] Andritsos, Periklis, and Keilty, Patrick. "Level-Wise Exploration of Linked and Big Data Guided by Controlled Vocabularies and Folksonomies". Advances in Classification Research 24 (2012). https://doi.org/10.7152/acro.v24i1.14670.

[2] ANSI/NISO (R2010. "Guidelines for the Construction, Format and Management of Monolingual Controlled Vocabularies". An American National Standard developed by the National Information Standards Organization, Baltimore, Maryland, USA. Approved July 5, 2006 by the American National Standards Institute, Reaffirmed May 13, 2010. https://www.niso.org/apps/group_public/download.php/12591/z39-19-2005r2010.pdf.

[3] Butterfield, Andrew, Gerard Ekembe Ngondi, and Anne Kerr, eds. 2016. "A Dictionary of Computer Science". 7th Edition. Oxford, United Kingdom; New York, NY, United States of America: Oxford University Press.

[4] Chung, TongLee, Bin Xu, Peng Zhang, Yuanhua Tan, Ping Zhu, and Adeli Wubulihasimu. "Constructing City Ontology from Expert for Smart City Management". Semantic Technology, 187–94. Lecture Notes in Computer Science. Springer, Cham, 2013. doi:10.1007/978-3-319-06826-8_15.

[5] Ciancarini, P., Poggi, F., Russo, D.: Big data quality: a roadmap for open data. In: Big Data Computing Service and Applications (Big Data Service), 2016 IEEE Second International Conference on. pp. 210–215. IEEE (2016).

[6] Clapham, Christopher, "Concise Dictionary of Mathematics", Second Edition, Oxford University Press, 1996

[7] Digital Geographic Information Exchange Standard (DIGEST) Part 4 FEATURE and ATTRIBUTE CODING CATALOGUE (FACC), 2000. https://www.dgiwg.org/digest/html/DIGEST_2-1_Part4.pdf.

[8] Fonseca, F., M. Egenhofer, et al. (2002). “Semantic Granularity in Ontology-Driven Geographic Information Systems.” AMAI Annals of Mathematics and Artificial Intelligence - Special Issue on Spatial and Temporal Granularity: 36.1-2 (2002): 121-51. Semantic

[9] Frank, Andrew U. "Tiers of Ontology and Consistency Constraints in Geographical Information Systems". International Journal of Geographical Information Science, IJGIS Special Issue (EURESCO Conference), 15, no. 7 (2001): 667–78.

[10] Ghawi, Raji, and Nadine Cullot. "Database-to-Ontology Mapping Generation for Semantic Interoperability". in Third International Workshop on Database Interoperability (InterDB 2007), in Conjunction with VLDB 2007, 2007.

[11] Golden Gate Bridge. Accessed December 18, 2017. http://goldengatebridge.org/research/factsGGBIntOrngColor.php.

[12] Guttman, A. (1984). "R-Trees: A Dynamic Index Structure for Spatial Searching", Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data – SIGMOD '84. p. 47. doi:10.1145/602259.602266

[13] Harpring, Patricia. Introduction to Controlled Vocabularies: Terminology for Art, Architecture, and Other Cultural Works. Getty Publications, 2010.

[14] Hedden, Heather. "The Accidental Taxonomist". 2nd edition. Information Today, Inc., 2016.

[15] Introduction to Controlled Vocabularies. Getty Research Institute. Accessed October 19, 2017. https://www.getty.edu/research/publications/electronic_publications/intro_controlled_vocab/.

[16] Kashyap, Vipul, Christoph Bussler, and Matthew Moran. The Semantic Web: Semantics for Data and Services on the Web. 2008 edition. Berlin: Springer, 2008.

[17] OECD Glossary of Statistical Terms, "Controlled Vocabulary Definition". Accessed September 19, 2017. https://stats.oecd.org/glossary/detail.asp?ID=6260.

[18] The Common Lisp Foundation. "Welcome to Common-Lisp.net!" Accessed August 9, 2017. https://common-lisp.net/.

[19] Topological and Thematic Layering of Geological Map Information. Accessed October 27, 2017. https://pubs.usgs.gov/of/1997/of97-269/viljoen.html.

[20] Uschold, Michael. "Ontologies and Database Schema: What’s the Difference?" Accessed August 9, 2017. https://pdfs.semanticscholar.org/b44f/a4592b69183c1965d0075dea1a3bc58dfbfe.pdf.

[21] Using Oracle Data Cloud. Oracle Help Center. Accessed November 2, 2017. https://docs.oracle.com/en/cloud/saas/data-cloud/dsmkt/preface.html and https://docs.oracle.com/en/cloud/saas/data-cloud/dsmkt/managing-your-taxonomy.html.